



Référence de l'API Objective-C



# Table des matières

<b>1. Introduction</b>	<b>1</b>
<b>2. Utilisation du Yocto-Demo en Objective-C</b>	<b>3</b>
2.1. Contrôle de la fonction Led	3
2.2. Contrôle de la partie module	5
2.3. Gestion des erreurs	7
Blueprint	10
<b>3. Reference</b>	<b>10</b>
3.1. Fonctions générales	11
3.2. Interface de la fonction Accelerometer	37
3.3. Interface de la fonction AnButton	83
3.4. Interface de la fonction CarbonDioxide	125
3.5. Interface de la fonction ColorLed	168
3.6. Interface de la fonction Compass	201
3.7. Interface de la fonction Current	245
3.8. Interface de la fonction DataLogger	288
3.9. Séquence de données mise en forme	323
3.10. Séquence de données enregistrées	333
3.11. Séquence de données enregistrées brute	346
3.12. Interface de la fonction DigitalIO	361
3.13. Interface de la fonction Display	409
3.14. Interface des objets DisplayLayer	460
3.15. Interface de contrôle de l'alimentation	492
3.16. Interface de la fonction Files	521
3.17. Interface de la fonction GenericSensor	554
3.18. Interface de la fonction Gyro	604
3.19. Interface d'un port de Yocto-hub	659
3.20. Interface de la fonction Humidity	688
3.21. Interface de la fonction Led	731
3.22. Interface de la fonction LightSensor	762
3.23. Interface de la fonction Magnetometer	806
3.24. Valeur mesurée	852
3.25. Interface de contrôle du module	858
3.26. Interface de la fonction Network	904

3.27. contrôle d'OS .....	965
3.28. Interface de la fonction Power .....	992
3.29. Interface de la fonction Pressure .....	1039
3.30. Interface de la fonction Pwm .....	1082
3.31. Interface de la fonction PwmPowerSource .....	1124
3.32. Interface du quaternion .....	1151
3.33. Interface de la fonction Horloge Temps Real .....	1194
3.34. Configuration du référentiel .....	1225
3.35. Interface de la fonction Relay .....	1265
3.36. Interface des fonctions de type senseur .....	1305
3.37. Interface de la fonction Servo .....	1348
3.38. Interface de la fonction Temperature .....	1387
3.39. Interface de la fonction Tilt .....	1432
3.40. Interface de la fonction Voc .....	1475
3.41. Interface de la fonction Voltage .....	1518
3.42. Interface de la fonction Source de tension .....	1561
3.43. Interface de la fonction WakeUpMonitor .....	1597
3.44. Interface de la fonction WakeUpSchedule .....	1636
3.45. Interface de la fonction Watchdog .....	1677
3.46. Interface de la fonction Wireless .....	1726

<b>Index .....</b>	<b>1759</b>
--------------------	-------------

# 1. Introduction

Ce manuel est votre référence pour l'utilisation de la librairie Objective-C de Yoctopuce pour interfacier vos senseurs et contrôleurs USB.

Le chapitre suivant reprend un chapitre du manuel du module USB gratuit Yocto-Demo, afin d'illustrer l'utilisation de la librairie sur des exemples concrets.

Le reste du manuel documente chaque fonction, classe et méthode de l'API. La première section décrit les fonctions globales d'ordre général, et les sections décrivent les différentes classes, utiles selon le module Yoctopuce utilisé. Pour plus d'informations sur la signification et l'utilisation d'un attribut particulier d'un module, il est recommandé de se référer à la documentation spécifique du module, qui contient plus de détails.



## 2. Utilisation du Yocto-Demo en Objective-C

Objective-C est le langage de prédilection pour programmer sous Mac OS X, en raison de son intégration avec le générateur d'interfaces Cocoa. Pour pouvoir utiliser la librairie Objective-C vous aurez impérativement besoin de XCode 4.2, qui est disponible gratuitement sous Lion. Si vous êtes encore sous Snow Leopard il vous faudra être enregistré comme développeur auprès d'Apple pour pouvoir télécharger XCode 4.2. La librairie Yoctopuce est compatible ARC. Il vous sera donc possible de coder vos projet soit en utilisant la traditionnelle méthode de *retain / release*, soit en activant l'*Automatic Reference Counting*.

Les librairies Yoctopuce<sup>1</sup> pour Objective-C vous sont fournies au format source dans leur intégralité. Une partie de la librairie de bas-niveau est écrite en C pur sucre, mais vous n'aurez à priori pas besoin d'interagir directement avec elle: tout a été fait pour que l'interaction soit le plus simple possible depuis Objective-C.

Vous allez rapidement vous rendre compte que l'API Objective-C définit beaucoup de fonctions qui retournent des objets. Vous ne devez jamais désallouer ces objets vous-même. Ils seront désalloués automatiquement par l'API à la fin de l'application.

Afin des les garder simples, tous les exemples fournis dans cette documentation sont des applications consoles. Il va de soit que que les fonctionnement des librairies est strictement identiques si vous les intégrez dans une application dotée d'une interface graphique. Vous trouverez sur le blog de Yoctopuce un exemple détaillé<sup>2</sup> avec des séquences vidéo montrant comment intégrer les fichiers de la librairie à vos projets.

### 2.1. Contrôle de la fonction Led

Lancez Xcode 4.2 et ouvrez le projet exemple correspondant, fourni dans le répertoire **Examples/Doc-GettingStarted-Yocto-Demo** de la librairie Yoctopuce.

```
#import <Foundation/Foundation.h>
#import "yocto_api.h"
#import "yocto_led.h"

static void usage(void)
{
    NSLog(@"usage: demo <serial_number> [ on | off ]");
    NSLog(@"      demo <logical_name> [ on | off ]");
    NSLog(@"      demo any [ on | off ]          (use any discovered device)");
    exit(1);
}
```

<sup>1</sup> [www.yoctopuce.com/FR/libraries.php](http://www.yoctopuce.com/FR/libraries.php)

<sup>2</sup> [www.yoctopuce.com/FR/article/nouvelle-librairie-objective-c-pour-mac-os-x](http://www.yoctopuce.com/FR/article/nouvelle-librairie-objective-c-pour-mac-os-x)

```

}

int main(int argc, const char * argv[])
{
    NSError *error;
    if(argc < 3) {
        usage();
    }

    @autoreleasepool {
        NSString *target = [NSString stringWithUTF8String:argv[1]];
        NSString *on_off = [NSString stringWithUTF8String:argv[2]];
        YLed *led;

        if([YAPI RegisterHub:@"usb": &error] != YAPI_SUCCESS) {
            NSLog(@"RegisterHub error: %@", [error localizedDescription]);
            return 1;
        }
        if([target isEqualToString:@"any"]){
            led = [YLed FirstLed];
        }else{
            led = [YLed FindLed:[target stringByAppendingString:@".led"]];
        }
        if ([led isOnline]) {
            if ([on_off isEqualToString:@"on"])
                [led set_power:Y_POWER_ON];
            else
                [led set_power:Y_POWER_OFF];
        } else {
            NSLog(@"Module not connected (check identification and USB cable)\n");
        }
    }
    return 0;
}

```

Il n'y a que peu de lignes véritablement importantes dans le code précédent. Nous allons les expliquer en détail.

## yocto\_api.h et yocto\_led.h

Ces deux fichiers importés permettent d'avoir accès aux fonctions permettant de gérer les modules Yoctopuce. `yocto_api.h` doit toujours être utilisé, `yocto_led.h` est nécessaire pour gérer les modules contenant une led, comme le Yocto-Demo.

## yRegisterHub

La fonction `yRegisterHub` initialise l'API de Yoctopuce en indiquant où les modules doivent être recherchés. Utilisée avec le paramètre `@"usb"`, elle permet de travailler avec les modules connectés localement à la machine. Si l'initialisation se passe mal, cette fonction renverra une valeur différente de `YAPI_SUCCESS`, et retournera via le paramètre `errmsg` une explication du problème.

## yFindLed

La fonction `yFindLed`, permet de retrouver une led en fonction du numéro de série de son module hôte et de son nom de fonction. Mais vous pouvez tout aussi bien utiliser des noms logiques que vous auriez préalablement configurés. Imaginons un module Yocto-Demo avec le numéros de série `YCTOPOC1-123456` que vous auriez appelé *"MonModule"* et dont vous auriez nommé la fonction led *"MaFonction"*, les cinq appels suivants seront strictement équivalents (pour autant que *MaFonction* ne soit définie qu'une fois, pour éviter toute ambiguïté):

```

YLed *led = yFindLed(@"YCTOPOC1-123456.led");
YLed *led = yFindLed(@"YCTOPOC1-123456.MaFonction");
YLed *led = yFindLed(@"MonModule.led");
YLed *led = yFindLed(@"MonModule.MaFonction");
YLed *led = yFindLed(@"MaFonction");

```

`yFindLed` renvoie un objet que vous pouvez ensuite utiliser à loisir pour contrôler la led.



## isOnline

La méthode `isOnline()` de l'objet renvoyé par `yFindLed` permet de savoir si le module correspondant est présent et en état de marche.

## set\_power

La fonction `set_power()` de l'objet renvoyé par `yFindLed` permet d'allumer et d'éteindre la led. L'argument est `Y_POWER_ON` ou `Y_POWER_OFF`. Vous trouverez dans la référence de l'interface de programmation d'autres méthodes permettant de contrôler précisément la luminosité et de faire clignoter automatiquement la led.

## 2.2. Contrôle de la partie module

Chaque module peut-être contrôlé d'une manière similaire, vous trouverez ci dessous un simple programme d'exemple affichant les principaux paramètres d'un module et permettant d'activer la balise de localisation.

```
#import <Foundation/Foundation.h>
#import "yocto_api.h"

static void usage(const char *exe)
{
    NSLog(@"usage: %s <serial or logical name> [ON/OFF]\n",exe);
    exit(1);
}

int main (int argc, const char * argv[])
{
    NSError *error;

    @autoreleasepool {
        // Setup the API to use local USB devices
        if([YAPI RegisterHub:@"usb": &error] != YAPI_SUCCESS) {
            NSLog(@"RegisterHub error: %@", [error localizedDescription]);
            return 1;
        }
        if(argc < 2)
            usage(argv[0]);
        NSString *serial_or_name = [NSString stringWithUTF8String:argv[1]];
        YModule *module = [YModule FindModule:serial_or_name]; // use serial or logical
name
        if ([module isOnline]) {
            if (argc > 2) {
                if (strcmp(argv[2], "ON")==0)
                    [module setBeacon:Y_BEACON_ON];
                else
                    [module setBeacon:Y_BEACON_OFF];
            }
            NSLog(@"serial:      %@\n", [module serialNumber]);
            NSLog(@"logical name: %@\n", [module logicalName]);
            NSLog(@"luminosity:   %d\n", [module luminosity]);
            NSLog(@"beacon:      ");
            if ([module beacon] == Y_BEACON_ON)
                NSLog(@"ON\n");
            else
                NSLog(@"OFF\n");
            NSLog(@"upTime:      %ld sec\n", [module upTime]/1000);
            NSLog(@"USB current:  %d mA\n", [module usbCurrent]);
            NSLog(@"logs:        %@\n", [module get_lastLogs]);
        } else {
            NSLog(@"%@ not connected (check identification and USB cable)\n",serial_or_name);
        }
    }
    return 0;
}
```

Chaque propriété `xxx` du module peut être lue grâce à une méthode du type `get_xxxx`, et les propriétés qui se sont pas en lecture seule peuvent être modifiées à l'aide de la méthode `set_xxx`. Pour plus de détails concernant ces fonctions utilisées, reportez-vous aux chapitre API

## Modifications des réglages du module

Lorsque que vous souhaitez modifier les réglages d'un module, il suffit d'appeler la fonction `set_xxx`: correspondante, cependant cette modification n'a lieu que dans la mémoire vive du module: si le module redémarre, les modifications seront perdues. Pour qu'elle soient mémorisées de manière persistante, il est nécessaire de demander au module de sauvegarder sa configuration courante dans sa mémoire non volatile. Pour cela il faut utiliser la méthode `saveToFlash`. Inversement il est possible de forcer le module à oublier ses réglages courants en utilisant la méthode `revertFromFlash`. Ce petit exemple ci-dessous vous permet changer le nom logique d'un module.

```
#import <Foundation/Foundation.h>
#import "yocto_api.h"

static void usage(const char *exe)
{
    NSLog(@"usage: %s <serial> <newLogicalName>\n",exe);
    exit(1);
}

int main (int argc, const char * argv[])
{
    NSError *error;

    @autoreleasepool {
        // Setup the API to use local USB devices
        if([YAPI RegisterHub:@"usb" :&error] != YAPI_SUCCESS) {
            NSLog(@"RegisterHub error: %@", [error localizedDescription]);
            return 1;
        }

        if(argc < 2)
            usage(argv[0]);

        NSString *serial_or_name = [NSString stringWithUTF8String:argv[1]];
        YModule *module = [YModule FindModule:serial_or_name]; // use serial or logical
name

        if (module.isOnline) {
            if (argc >= 3){
                NSString *newname = [NSString stringWithUTF8String:argv[2]];
                if (![YAPI CheckLogicalName:newname]){
                    NSLog(@"Invalid name (%@)\n", newname);
                    usage(argv[0]);
                }
                module.logicalName = newname;
                [module saveToFlash];
            }
            NSLog(@"Current name: %@\n", module.logicalName);
        } else {
            NSLog(@"%% not connected (check identification and USB cable)\n",serial_or_name);
        }
    }
    return 0;
}
```

Attention, le nombre de cycles d'écriture de la mémoire non volatile du module est limité. Passé cette limite plus rien ne garantit que la sauvegarde des réglages se passera correctement. Cette limite, liée à la technologie employée par le micro-processeur du module se situe aux alentours de 100000 cycles. Pour résumer vous ne pouvez employer la fonction `saveToFlash` que 100000 fois au cours de la vie du module. Veillez donc à ne pas appeler cette fonction depuis l'intérieur d'une boucle.

## Enumeration des modules

Obtenir la liste des modules connectés se fait à l'aide de la fonction `yFirstModule()` qui renvoie le premier module trouvé, il suffit ensuite d'appeler la fonction `nextModule()` de cet objet pour trouver les modules suivants, et ce tant que la réponse n'est pas un `NULL`. Ci-dessous un petit exemple listant les module connectés

```

#import <Foundation/Foundation.h>
#import "yocto_api.h"

int main (int argc, const char * argv[])
{
    NSError *error;

    @autoreleasepool {
        // Setup the API to use local USB devices
        if([YAPI RegisterHub:@"usb" :&error] != YAPI_SUCCESS) {
            NSLog(@"RegisterHub error: %@\n", [error localizedDescription]);
            return 1;
        }

        NSLog(@"Device list:\n");

        YModule *module = [YModule FirstModule];
        while (module != nil) {
            NSLog(@"%@ %@", module.serialNumber, module.productName);
            module = [module nextModule];
        }
        return 0;
    }
}

```

## 2.3. Gestion des erreurs

Lorsque vous implémentez un programme qui doit interagir avec des modules USB, vous ne pouvez pas faire abstraction de la gestion des erreurs. Il y aura forcément une occasion où un utilisateur aura débranché le périphérique, soit avant de lancer le programme, soit même en pleine opération. La librairie Yoctopuce est prévue pour vous aider à supporter ce genre de comportements, mais votre code doit néanmoins être fait pour se comporter au mieux pour interpréter les erreurs signalées par la librairie.

La manière la plus simple de contourner le problème est celle que nous avons employé pour les petits exemples précédents de ce chapitre: avant d'accéder à un module, on vérifie qu'il est en ligne avec la méthode `isOnline()` et on suppose ensuite qu'il va y rester pendant la fraction de seconde nécessaire à exécuter les lignes de code suivantes. Ce n'est pas parfait, mais ça peut suffire dans certains cas. Il faut toutefois être conscient qu'on ne peut pas totalement exclure une erreur se produisant après le `isOnline()`, qui pourrait faire planter le programme. La seule manière de l'éviter est d'implémenter une des deux techniques de gestion des erreurs décrites ci-dessous.

La méthode recommandée par la plupart des langages de programmation pour la gestion des erreurs imprévisibles est l'utilisation d'exceptions. C'est le comportement par défaut de la librairie Yoctopuce. Si une erreur se produit alors qu'on essaie d'accéder à un module, la librairie va lancer une exception. Dans ce cas, de trois choses l'une:

- Si votre code attrape l'exception au vol et la gère, et tout se passe bien.
- Si votre programme tourne dans le debugger, vous pourrez relativement facilement déterminer où le problème s'est produit, et voir le message explicatif lié à l'exception.
- Sinon... l'exception va crasher votre programme, boum!

Comme cette dernière situation n'est pas la plus souhaitable, la librairie Yoctopuce offre une autre alternative pour la gestion des erreurs, permettant de faire un programme robuste sans devoir attraper les exceptions à chaque ligne de code. Il suffit d'appeler la fonction `yDisableExceptions()` pour commuter la librairie dans un mode où les exceptions de chaque fonction sont systématiquement remplacées par des valeurs de retour particulières, qui peuvent être testées par l'appelant lorsque c'est pertinent. Le nom de la valeur de retour en cas d'erreur pour chaque fonction est systématiquement documenté dans la référence de la librairie. Il suit toujours la même logique: une méthode `get_state()` retournera une valeur `Y_STATE_INVALID`, une méthode `get_currentValue` retournera une valeur `Y_CURRENTVALUE_INVALID`, etc. Dans tous les cas, la valeur retournée sera du type attendu, et ne sera pas un pointeur nul qui risquerait de faire crasher votre programme. Au pire, si vous affichez la valeur sans la tester, elle sera hors du

cadre attendu pour la valeur retournée. Dans le cas de fonctions qui ne retournent à priori pas d'information, la valeur de retour sera `YAPI_SUCCESS` si tout va bien, et un code d'erreur différent en cas d'échec.

Quand vous travaillez sans les exceptions, il est possible d'obtenir un code d'erreur et un message expliquant l'origine de l'erreur en le demandant à l'objet qui a retourné une erreur à l'aide des méthodes `errType()` et `errMessage()`. Ce sont les même informations qui auraient été associées à l'exception si elles avaient été actives.



### **3. Reference**

## 3.1. Fonctions générales

Ces quelques fonctions générales permettent l'initialisation et la configuration de la librairie Yoctopuce. Dans la plupart des cas, un appel à `yRegisterHub()` suffira en tout et pour tout. Ensuite, vous pourrez appeler la fonction globale `yFind...()` ou `yFirst...()` correspondant à votre module pour pouvoir interagir avec lui.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_api.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YAPI = yoctolib.YAPI;</code> <code>var YModule = yoctolib.YModule;</code>
php	<code>require_once('yocto_api.php');</code>
c++	<code>#include "yocto_api.h"</code>
m	<code>#import "yocto_api.h"</code>
pas	<code>uses yocto_api;</code>
vb	<code>yocto_api.vb</code>
cs	<code>yocto_api.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YModule;</code>
py	<code>from yocto_api import *</code>

### Fonction globales

#### **yCheckLogicalName(name)**

Vérifie si un nom donné est valide comme nom logique pour un module ou une fonction.

#### **yDisableExceptions()**

Désactive l'utilisation d'exceptions pour la gestion des erreurs.

#### **yEnableExceptions()**

Réactive l'utilisation d'exceptions pour la gestion des erreurs.

#### **yEnableUSBHost(osContext)**

Cette fonction est utilisée uniquement sous Android.

#### **yFreeAPI()**

Libère la mémoire dynamique utilisée par la librairie Yoctopuce.

#### **yGetAPIVersion()**

Retourne la version de la librairie Yoctopuce utilisée.

#### **yGetTickCount()**

Retourne la valeur du compteur monotone de temps (en millisecondes).

#### **yHandleEvents(errmsg)**

Maintient la communication de la librairie avec les modules Yoctopuce.

#### **yInitAPI(mode, errmsg)**

Initialise la librairie de programmation de Yoctopuce explicitement.

#### **yPreregisterHub(url, errmsg)**

Alternative plus tolérante à `RegisterHub()`.

#### **yRegisterDeviceArrivalCallback(arrivalCallback)**

Enregistre une fonction de callback qui sera appelée à chaque fois qu'un module est branché.

#### **yRegisterDeviceRemovalCallback(removalCallback)**

Enregistre une fonction de callback qui sera appelée à chaque fois qu'un module est débranché.

#### **yRegisterHub(url, errmsg)**

Configure la librairie Yoctopuce pour utiliser les modules connectés sur une machine donnée.

#### **yRegisterHubDiscoveryCallback(hubDiscoveryCallback)**

### 3. Reference

Enregistre une fonction de callback qui est appelée chaque fois qu'un hub réseau s'annonce avec un message SSDP.

#### **yRegisterLogFunction(logfun)**

Enregistre une fonction de callback qui sera appelée à chaque fois que l'API a quelque chose à dire.

#### **ySelectArchitecture(arch)**

Sélectionne manuellement l'architecture de la librairie dynamique à utiliser pour accéder à USB.

#### **ySetDelegate(object)**

(Objective-C uniquement) Enregistre un objet délégué qui doit se conformer au protocole YDeviceHotPlug.

#### **ySetTimeout(callback, ms\_timeout, arguments)**

Appelle le callback spécifié après un temps d'attente spécifié.

#### **ySleep(ms\_duration, errmsg)**

Effectue une pause dans l'exécution du programme pour une durée spécifiée.

#### **yTriggerHubDiscovery(errmsg)**

Relance une détection des hubs réseau.

#### **yUnregisterHub(url)**

Configure la librairie Yoctopuce pour ne plus utiliser les modules connectés sur une machine préalablement enregistrer avec RegisterHub.

#### **yUpdateDeviceList(errmsg)**

Force une mise-à-jour de la liste des modules Yoctopuce connectés.

#### **yUpdateDeviceList\_async(callback, context)**

Force une mise-à-jour de la liste des modules Yoctopuce connectés.



## YAPI.CheckLogicalName() yCheckLogicalName()yCheckLogicalName()

YAPI

Vérifie si un nom donné est valide comme nom logique pour un module ou une fonction.

js	function <b>yCheckLogicalName</b> ( <b>name</b> )
nodejs	function <b>CheckLogicalName</b> ( <b>name</b> )
php	function <b>yCheckLogicalName</b> ( <b>\$name</b> )
cpp	bool <b>yCheckLogicalName</b> ( const string& <b>name</b> )
m	BOOL <b>yCheckLogicalName</b> ( NSString * <b>name</b> )
pas	function <b>yCheckLogicalName</b> ( <b>name</b> : string): boolean
vb	function <b>yCheckLogicalName</b> ( ByVal <b>name</b> As String) As Boolean
cs	bool <b>CheckLogicalName</b> ( string <b>name</b> )
java	boolean <b>CheckLogicalName</b> ( String <b>name</b> )
py	def <b>CheckLogicalName</b> ( <b>name</b> )

Un nom logique valide est formé de 19 caractères au maximum, choisis parmi A . . Z, a . . z, 0 . . 9, \_ et -. Lorsqu'on configure un nom logique avec une chaîne incorrecte, les caractères invalides sont ignorés.

### Paramètres :

**name** une chaîne de caractères contenant le nom vérifier.

### Retourne :

true si le nom est valide, false dans le cas contraire.

**YAPI.DisableExceptions()****YAPI****yDisableExceptions()****yDisableExceptions()**

Désactive l'utilisation d'exceptions pour la gestion des erreurs.

js	function <b>yDisableExceptions</b> ( )
nodejs	function <b>DisableExceptions</b> ( )
php	function <b>yDisableExceptions</b> ( )
cpp	void <b>yDisableExceptions</b> ( )
m	void <b>yDisableExceptions</b> ( )
pas	procedure <b>yDisableExceptions</b> ( )
vb	procedure <b>yDisableExceptions</b> ( )
cs	void <b>DisableExceptions</b> ( )
py	def <b>DisableExceptions</b> ( )

Lorsque les exceptions sont désactivées, chaque fonction retourne une valeur d'erreur spécifique selon son type, documentée dans ce manuel de référence.

## YAPI.EnableExceptions() yEnableExceptions()yEnableExceptions()

YAPI

Réactive l'utilisation d'exceptions pour la gestion des erreurs.

js	function <b>yEnableExceptions</b> ( )
nodejs	function <b>EnableExceptions</b> ( )
php	function <b>yEnableExceptions</b> ( )
cpp	void <b>yEnableExceptions</b> ( )
m	void <b>yEnableExceptions</b> ( )
pas	procedure <b>yEnableExceptions</b> ( )
vb	procedure <b>yEnableExceptions</b> ( )
cs	void <b>EnableExceptions</b> ( )
py	def <b>EnableExceptions</b> ( )

Attention, lorsque les exceptions sont activées, tout appel à une fonction de la librairie qui échoue déclenche une exception. Dans le cas où celle-ci n'est pas interceptée correctement par le code appelant, soit le debugger se lance, soit le programme de l'utilisateur est immédiatement stoppé (crash).

## YAPI.EnableUSBHost() yEnableUSBHost()

YAPI

Cette fonction est utilisée uniquement sous Android.

```
java void EnableUSBHost( Object osContext)
```

Avant d'appeler `yRegisterHub( "usb" )` il faut activer le port USB host du systeme. Cette fonction prend en argument un objet de la classe `android.content.Context` (ou d'une sous-classe). Il n'est pas nécessaire d'appeler cette fonction pour accéder au modules à travers le réseau.

### Paramètres :

**osContext** un objet de classe `android.content.Context` (ou une sous-classe)

**YAPI.FreeAPI()****YAPI****yFreeAPI()****yFreeAPI()**

---

Libère la mémoire dynamique utilisée par la librairie Yoctopuce.

js	function <b>yFreeAPI</b> ( )
nodejs	function <b>FreeAPI</b> ( )
php	function <b>yFreeAPI</b> ( )
cpp	void <b>yFreeAPI</b> ( )
m	void <b>yFreeAPI</b> ( )
pas	procedure <b>yFreeAPI</b> ( )
vb	procedure <b>yFreeAPI</b> ( )
cs	void <b>FreeAPI</b> ( )
java	void <b>FreeAPI</b> ( )
py	def <b>FreeAPI</b> ( )

Il n'est en général pas nécessaire d'appeler cette fonction, sauf si vous désirez libérer tous les blocs de mémoire alloués dynamiquement dans le but d'identifier une source de blocs perdus par exemple. Vous ne devez plus appeler aucune fonction de la librairie après avoir appelé `yFreeAPI( )`, sous peine de crash.

## YAPI.GetAPIVersion() yGetAPIVersion()yGetAPIVersion()

YAPI

Retourne la version de la librairie Yoctopuce utilisée.

js	function <b>yGetAPIVersion</b> ( )
nodejs	function <b>GetAPIVersion</b> ( )
php	function <b>yGetAPIVersion</b> ( )
cpp	string <b>yGetAPIVersion</b> ( )
m	NSString* <b>yGetAPIVersion</b> ( )
pas	function <b>yGetAPIVersion</b> ( ): string
vb	function <b>yGetAPIVersion</b> ( ) As String
cs	String <b>GetAPIVersion</b> ( )
java	String <b>GetAPIVersion</b> ( )
py	def <b>GetAPIVersion</b> ( )

La version est retournée sous forme d'une chaîne de caractères au format "Majeure.Mineure.NoBuild", par exemple "1.01.5535". Pour les langages utilisant une DLL externe (par exemple C#, VisualBasic ou Delphi), la chaîne contient en outre la version de la DLL au même format, par exemple "1.01.5535 (1.01.5439)".

Si vous désirez vérifier dans votre code que la version de la librairie est compatible avec celle que vous avez utilisé durant le développement, vérifiez que le numéro majeur soit strictement égal et que le numéro mineur soit égal ou supérieur. Le numéro de build n'est pas significatif par rapport à la compatibilité de la librairie.

**Retourne :**

une chaîne de caractères décrivant la version de la librairie.

## YAPI.GetTickCount() yGetTickCount()yGetTickCount()

YAPI

Retourne la valeur du compteur monotone de temps (en millisecondes).

js	function <b>yGetTickCount</b> ( )
nodejs	function <b>GetTickCount</b> ( )
php	function <b>yGetTickCount</b> ( )
cpp	u64 <b>yGetTickCount</b> ( )
m	u64 <b>yGetTickCount</b> ( )
pas	function <b>yGetTickCount</b> ( ): u64
vb	function <b>yGetTickCount</b> ( ) As Long
cs	ulong <b>GetTickCount</b> ( )
java	long <b>GetTickCount</b> ( )
py	def <b>GetTickCount</b> ( )

Ce compteur peut être utilisé pour calculer des délais en rapport avec les modules Yoctopuce, dont la base de temps est aussi la milliseconde.

**Retourne :**

un long entier contenant la valeur du compteur de millisecondes.

## YAPI.HandleEvents() yHandleEvents()yHandleEvents()

Maintient la communication de la librairie avec les modules Yoctopuce.

js	function <b>yHandleEvents</b> ( <b>errmsg</b> )
nodejs	function <b>HandleEvents</b> ( <b>errmsg</b> )
php	function <b>yHandleEvents</b> ( <b>&amp;\$errmsg</b> )
cpp	YRETCODE <b>yHandleEvents</b> ( string& <b>errmsg</b> )
m	YRETCODE <b>yHandleEvents</b> ( NSError** <b>errmsg</b> )
pas	function <b>yHandleEvents</b> ( var <b>errmsg</b> : string): integer
vb	function <b>yHandleEvents</b> ( ByRef <b>errmsg</b> As String) As YRETCODE
cs	YRETCODE <b>HandleEvents</b> ( ref string <b>errmsg</b> )
java	int <b>HandleEvents</b> ( )
py	def <b>HandleEvents</b> ( <b>errmsg</b> =None)

Si votre programme inclut des longues boucles d'attente, vous pouvez y inclure un appel à cette fonction pour que la librairie prenne en charge les informations mise en attente par les modules sur les canaux de communication. Ce n'est pas strictement indispensable mais cela peut améliorer la réactivité des la librairie pour les commandes suivantes.

Cette fonction peut signaler une erreur au cas à la communication avec un module Yoctopuce ne se passerait pas comme attendu.

### Paramètres :

**errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## YAPI.InitAPI() ylnitAPI()ylnitAPI()

## YAPI

Initialise la librairie de programmation de Yoctopuce explicitement.

js	function <b>ylnitAPI</b> ( <b>mode</b> , <b>errmsg</b> )
nodejs	function <b>InitAPI</b> ( <b>mode</b> , <b>errmsg</b> )
php	function <b>ylnitAPI</b> ( <b>\$mode</b> , <b>&amp;\$errmsg</b> )
cpp	YRETCODE <b>ylnitAPI</b> ( int <b>mode</b> , string& <b>errmsg</b> )
m	YRETCODE <b>ylnitAPI</b> ( int <b>mode</b> , NSError** <b>errmsg</b> )
pas	function <b>ylnitAPI</b> ( <b>mode</b> : integer, var <b>errmsg</b> : string): integer
vb	function <b>ylnitAPI</b> ( ByVal <b>mode</b> As Integer, ByRef <b>errmsg</b> As String) As Integer
cs	int <b>InitAPI</b> ( int <b>mode</b> , ref string <b>errmsg</b> )
java	int <b>InitAPI</b> ( int <b>mode</b> )
py	def <b>InitAPI</b> ( <b>mode</b> , <b>errmsg=None</b> )

Il n'est pas indispensable d'appeler `ylnitAPI()`, la librairie sera automatiquement initialisée de toute manière au premier appel à `yRegisterHub()`.

Lorsque cette fonction est utilisée avec comme `mode` la valeur `Y_DETECT_NONE`, il faut explicitement appeler `yRegisterHub()` pour indiquer à la librairie sur quel VirtualHub les modules sont connectés, avant d'essayer d'y accéder.

### Paramètres :

- mode** un entier spécifiant le type de détection automatique de modules à utiliser. Les valeurs possibles sont `Y_DETECT_NONE`, `Y_DETECT_USB`, `Y_DETECT_NET` et `Y_DETECT_ALL`.
- errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.PreregisterHub() yPreregisterHub()yPreregisterHub()

YAPI

Alternative plus tolérante à RegisterHub().

js	function <b>yPreregisterHub</b> ( url, errmsg)
nodejs	function <b>PreregisterHub</b> ( url, errmsg)
php	function <b>yPreregisterHub</b> ( \$url, &\$errmsg)
cpp	YRETCODE <b>yPreregisterHub</b> ( const string& url, string& errmsg)
m	YRETCODE <b>yPreregisterHub</b> ( NSString * url, NSError** errmsg)
pas	function <b>yPreregisterHub</b> ( url: string, var errmsg: string): integer
vb	function <b>yPreregisterHub</b> ( ByVal url As String, ByRef errmsg As String) As Integer
cs	int <b>PreregisterHub</b> ( string url, ref string errmsg)
java	int <b>PreregisterHub</b> ( String url)
py	def <b>PreregisterHub</b> ( url, errmsg=None)

Cette fonction a le même but et la même paramètre que la fonction RegisterHub, mais contrairement à celle-ci PreregisterHub( ) ne déclenche pas d'erreur si le hub choisi n'est pas joignable au moment de l'appel. Il est ainsi possible d'enregistrer un hub réseau indépendamment de la connectivité, afin de tenter de ne le contacter que lorsqu'on cherche réellement un module.

### Paramètres :

- url** une chaîne de caractères contenant "usb", "callback", ou l'URL racine du VirtualHub à utiliser.
- errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**YAPI.RegisterDeviceArrivalCallback()****YAPI****yRegisterDeviceArrivalCallback()****yRegisterDeviceArrivalCallback()**

Enregistre une fonction de callback qui sera appelée à chaque fois qu'un module est branché.

js	function <b>yRegisterDeviceArrivalCallback</b> ( <b>arrivalCallback</b> )
nodejs	function <b>RegisterDeviceArrivalCallback</b> ( <b>arrivalCallback</b> )
php	function <b>yRegisterDeviceArrivalCallback</b> ( <b>\$arrivalCallback</b> )
cpp	void <b>yRegisterDeviceArrivalCallback</b> ( yDeviceUpdateCallback <b>arrivalCallback</b> )
m	void <b>yRegisterDeviceArrivalCallback</b> ( yDeviceUpdateCallback <b>arrivalCallback</b> )
pas	procedure <b>yRegisterDeviceArrivalCallback</b> ( <b>arrivalCallback</b> : yDeviceUpdateFunc)
vb	procedure <b>yRegisterDeviceArrivalCallback</b> ( ByVal <b>arrivalCallback</b> As yDeviceUpdateFunc)
cs	void <b>RegisterDeviceArrivalCallback</b> ( yDeviceUpdateFunc <b>arrivalCallback</b> )
java	void <b>RegisterDeviceArrivalCallback</b> ( DeviceArrivalCallback <b>arrivalCallback</b> )
py	def <b>RegisterDeviceArrivalCallback</b> ( <b>arrivalCallback</b> )

Le callback sera appelé pendant l'exécution de la fonction `yHandleDeviceList`, que vous devrez appeler régulièrement.

**Paramètres :**

**arrivalCallback** une procédure qui prend un `YModule` en paramètre, ou `null`

**YAPI.RegisterDeviceRemovalCallback()****yRegisterDeviceRemovalCallback()****yRegisterDeviceRemovalCallback()**

Enregistre une fonction de callback qui sera appelée à chaque fois qu'un module est débranché.

js	function <b>yRegisterDeviceRemovalCallback</b> ( <b>removalCallback</b> )
nodejs	function <b>RegisterDeviceRemovalCallback</b> ( <b>removalCallback</b> )
php	function <b>yRegisterDeviceRemovalCallback</b> ( <b>\$removalCallback</b> )
cpp	void <b>yRegisterDeviceRemovalCallback</b> ( yDeviceUpdateCallback <b>removalCallback</b> )
m	void <b>yRegisterDeviceRemovalCallback</b> ( yDeviceUpdateCallback <b>removalCallback</b> )
pas	procedure <b>yRegisterDeviceRemovalCallback</b> ( <b>removalCallback</b> : yDeviceUpdateFunc)
vb	procedure <b>yRegisterDeviceRemovalCallback</b> ( ByVal <b>removalCallback</b> As yDeviceUpdateFunc)
cs	void <b>RegisterDeviceRemovalCallback</b> ( yDeviceUpdateFunc <b>removalCallback</b> )
java	void <b>RegisterDeviceRemovalCallback</b> ( DeviceRemovalCallback <b>removalCallback</b> )
py	def <b>RegisterDeviceRemovalCallback</b> ( <b>removalCallback</b> )

Le callback sera appelé pendant l'exécution de la fonction `yHandleDeviceList`, que vous devrez appeler régulièrement.

**Paramètres :**

**removalCallback** une procédure qui prend un `YModule` en paramètre, ou `null`

## YAPI.RegisterHub() yRegisterHub()yRegisterHub()

YAPI

Configure la librairie Yoctopuce pour utiliser les modules connectés sur une machine donnée.

```

js function yRegisterHub( url, errmsg)
nodejs function RegisterHub( url, errmsg)
php function yRegisterHub( $url, &$errmsg)
cpp YRETCODE yRegisterHub( const string& url, string& errmsg)
m YRETCODE yRegisterHub( NSString * url, NSError** errmsg)
pas function yRegisterHub( url: string, var errmsg: string): integer
vb function yRegisterHub( ByVal url As String,
                        ByRef errmsg As String) As Integer
cs int RegisterHub( string url, ref string errmsg)
java int RegisterHub( String url)
py def RegisterHub( url, errmsg=None)

```

Le premier paramètre détermine le fonctionnement de l'API, il peut prendre les valeurs suivantes:

**usb**: Si vous utilisez le mot-clé **usb**, l'API utilise les modules Yoctopuce connectés directement par USB. Certains langages comme PHP, Javascript et Java ne permettent pas un accès direct aux couches matérielles, **usb** ne marchera donc pas avec ces langages. Dans ce cas, utilisez un VirtualHub ou un YoctoHub réseau (voir ci-dessous).

**x.x.x.x** ou **hostname**: L'API utilise les modules connectés à la machine dont l'adresse IP est x.x.x.x, ou dont le nom d'hôte DNS est *hostname*. Cette machine peut être un ordinateur classique faisant tourner un VirtualHub, ou un YoctoHub avec réseau (YoctoHub-Ethernet / YoctoHub-Wireless). Si vous désirez utiliser le VirtualHub tournant sur votre machine locale, utilisez l'adresse IP 127.0.0.1.

**callback** Le mot-clé **callback** permet de faire fonctionner l'API dans un mode appelé "*callback HTTP*". C'est un mode spécial permettant, entre autres, de prendre le contrôle de modules Yoctopuce à travers un filtre NAT par l'intermédiaire d'un VirtualHub ou d'un Hub Yoctopuce. Il vous suffit de configurer le hub pour qu'il appelle votre script à intervalle régulier. Ce mode de fonctionnement n'est disponible actuellement qu'en PHP et en Node.JS.

Attention, seule une application peut fonctionner à la fois sur une machine donnée en accès direct à USB, sinon il y aurait un conflit d'accès aux modules. Cela signifie en particulier que vous devez stopper le VirtualHub avant de lancer une application utilisant l'accès direct à USB. Cette limitation peut être contournée en passant par un VirtualHub plutôt que d'utiliser directement USB.

Si vous désirez vous connecter à un Hub, virtuel ou non, sur lequel le contrôle d'accès a été activé, vous devez donner le paramètre url sous la forme:

```
http://nom:mot_de_passe@adresse:port
```

Vous pouvez appeler *RegisterHub* plusieurs fois pour vous connecter à plusieurs machines différentes.

### Paramètres :

**url** une chaîne de caractères contenant "**usb**", "**callback**", ou l'URL racine du VirtualHub à utiliser.  
**errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

### 3. Reference

---

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.RegisterHubDiscoveryCallback()

### yRegisterHubDiscoveryCallback()

### yRegisterHubDiscoveryCallback()

YAPI

Enregistre une fonction de callback qui est appelée chaque fois qu'un hub réseau s'annonce avec un message SSDP.

cpp	void <b>yRegisterHubDiscoveryCallback</b> ( YHubDiscoveryCallback <b>hubDiscoveryCallback</b> )
m	+(void) <b>yRegisterHubDiscoveryCallback</b> : (YHubDiscoveryCallback) <b>hubDiscoveryCallback</b>
pas	procedure <b>yRegisterHubDiscoveryCallback</b> ( <b>hubDiscoveryCallback</b> : YHubDiscoveryCallback)
vb	procedure <b>yRegisterHubDiscoveryCallback</b> ( ByVal <b>hubDiscoveryCallback</b> As YHubDiscoveryCallback)
cs	void <b>RegisterHubDiscoveryCallback</b> ( YHubDiscoveryCallback <b>hubDiscoveryCallback</b> )
java	void <b>RegisterHubDiscoveryCallback</b> ( HubDiscoveryCallback <b>hubDiscoveryCallback</b> )
py	def <b>RegisterHubDiscoveryCallback</b> ( <b>hubDiscoveryCallback</b> )

la fonction de callback reçoit deux chaînes de caractères en paramètre La première chaîne contient le numéro de série du hub réseau et la deuxième chaîne contient l'URL du hub. L'URL peut être passée directement en argument à la fonction `yRegisterHub`. Le callback sera appelé pendant l'exécution de la fonction `yHandleDeviceList`, que vous devrez appeler régulièrement.

#### Paramètres :

**hubDiscoveryCallback** une procédure qui prend deux chaîne de caractères en paramètre, ou `null`

**YAPI.RegisterLogFunction()****YAPI****yRegisterLogFunction()****yRegisterLogFunction()**

Enregistre une fonction de callback qui sera appelée à chaque fois que l'API a quelque chose à dire.

cpp	void <b>yRegisterLogFunction</b> ( yLogFunction <b>logfun</b> )
m	void <b>yRegisterLogFunction</b> ( yLogCallback <b>logfun</b> )
pas	procedure <b>yRegisterLogFunction</b> ( <b>logfun</b> : yLogFunc)
vb	procedure <b>yRegisterLogFunction</b> ( ByVal <b>logfun</b> As yLogFunc)
cs	void <b>RegisterLogFunction</b> ( yLogFunc <b>logfun</b> )
java	void <b>RegisterLogFunction</b> ( LogCallback <b>logfun</b> )
py	def <b>RegisterLogFunction</b> ( <b>logfun</b> )

Utile pour déboguer le fonctionnement de l'API.

**Paramètres :**

**logfun** une procedure qui prend une chaîne de caractère en paramètre,



## YAPI.SelectArchitecture() ySelectArchitecture()

YAPI

Sélectionne manuellement l'architecture de la librairie dynamique à utiliser pour accéder à USB.

```
py def SelectArchitecture( arch)
```

Par défaut, la librairie Python détecte automatiquement la version de la librairie dynamique à utiliser pour accéder au port USB. Sous Linux ARM il n'est pas possible de détecter de manière fiable si il s'agit d'une installation Soft float (armel) ou Hard float (armhf). Dans ce cas, il est donc recommandé d'appeler `SelectArchitecture()` avant tout autre appel à la librairie pour forcer l'utilisation d'une architecture spécifiée.

### Paramètres :

**arch** une chaîne de caractère spécifiant l'architecture à utiliser. Les valeurs possibles sont "armhf", "armel", "i386", "x86\_64", "32bit", "64bit"

### Retourne :

rien. En cas d'erreur, déclenche une exception.

**YAPI.SetDelegate()****YAPI****ySetDelegate()ySetDelegate()**

(Objective-C uniquement) Enregistre un objet délégué qui doit se conformer au protocole YDeviceHotPlug.

```
m void ySetDelegate( id object)
```

Les méthodes `yDeviceArrival` et `yDeviceRemoval` seront appelées pendant l'exécution de la fonction `yHandleDeviceList`, que vous devrez appeler régulièrement.

**Paramètres :**

**object** un objet qui soit se conformer au protocole `YAPIDelegate`, ou `nil`

## YAPI.SetTimeout() ySetTimeout()

## YAPI

Appelle le callback spécifié après un temps d'attente spécifié.

js	function <b>ySetTimeout</b> ( <b>callback</b> , <b>ms_timeout</b> , <b>arguments</b> )
nodejs	function <b>SetTimeout</b> ( <b>callback</b> , <b>ms_timeout</b> , <b>arguments</b> )

Cette fonction se comporte plus ou moins comme la fonction Javascript `setTimeout`, mais durant le temps d'attente, elle va appeler `yHandleEvents` et `yUpdateDeviceList` périodiquement pour maintenir l'API à jour avec les modules connectés.

### Paramètres :

- callback** la fonction à appeler lorsque le temps d'attente est écoulé. Sous Microsoft Internet Explorer, le callback doit être spécifié sous forme d'une string à évaluer.
- ms\_timeout** un entier correspondant à la durée de l'attente, en millisecondes
- arguments** des arguments supplémentaires peuvent être fournis, pour être passés à la fonction de callback si nécessaire (pas supporté sous Microsoft Internet Explorer).

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.Sleep() ySleep()ySleep()

Effectue une pause dans l'exécution du programme pour une durée spécifiée.

js	function <b>ySleep</b> ( <b>ms_duration</b> , <b>errmsg</b> )
nodejs	function <b>Sleep</b> ( <b>ms_duration</b> , <b>errmsg</b> )
php	function <b>ySleep</b> ( <b>\$ms_duration</b> , <b>&amp;\$errmsg</b> )
c++	YRETCODE <b>ySleep</b> ( unsigned <b>ms_duration</b> , string& <b>errmsg</b> )
m	YRETCODE <b>ySleep</b> ( unsigned <b>ms_duration</b> , NSError ** <b>errmsg</b> )
pas	function <b>ySleep</b> ( <b>ms_duration</b> : integer, var <b>errmsg</b> : string): integer
vb	function <b>ySleep</b> ( ByVal <b>ms_duration</b> As Integer, ByRef <b>errmsg</b> As String) As Integer
cs	int <b>Sleep</b> ( int <b>ms_duration</b> , ref string <b>errmsg</b> )
java	int <b>Sleep</b> ( long <b>ms_duration</b> )
py	def <b>Sleep</b> ( <b>ms_duration</b> , <b>errmsg=None</b> )

L'attente est passive, c'est-à-dire qu'elle n'occupe pas significativement le processeur, de sorte à le laisser disponible pour les autres processus fonctionnant sur la machine. Durant l'attente, la librairie va néanmoins continuer à lire périodiquement les informations en provenance des modules Yoctopuce en appelant la fonction `yHandleEvents()` afin de se maintenir à jour.

Cette fonction peut signaler une erreur au cas à la communication avec un module Yoctopuce ne se passerait pas comme attendu.

### Paramètres :

**ms\_duration** un entier correspondant à la durée de la pause, en millisecondes

**errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.TriggerHubDiscovery() yTriggerHubDiscovery()yTriggerHubDiscovery()

YAPI

Relance une détection des hubs réseau.

cpp	YRETCODE yTriggerHubDiscovery( string& errmsg)
m	+(YRETCODE) yTriggerHubDiscovery : (NSError**) errmsg
pas	function yTriggerHubDiscovery( var errmsg: string): integer
vb	function yTriggerHubDiscovery( ByRef errmsg As String) As Integer
cs	int TriggerHubDiscovery( ref string errmsg)
java	int TriggerHubDiscovery( )
py	def TriggerHubDiscovery( errmsg=None)

Si une fonction de callback est enregistrée avec yRegisterDeviceRemovalCallback elle sera appelée à chaque hub réseau qui répondra à la détection SSDP.

### Paramètres :

**errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.UnregisterHub() yUnregisterHub()yUnregisterHub()

YAPI

Configure la librairie Yoctopuce pour ne plus utiliser les modules connectés sur une machine préalablement enregistrer avec RegisterHub.

js	function <b>yUnregisterHub</b> ( <b>url</b> )
nodejs	function <b>UnregisterHub</b> ( <b>url</b> )
php	function <b>yUnregisterHub</b> ( <b>\$url</b> )
cpp	void <b>yUnregisterHub</b> ( const string& <b>url</b> )
m	void <b>yUnregisterHub</b> ( NSString * <b>url</b> )
pas	procedure <b>yUnregisterHub</b> ( <b>url</b> : string)
vb	procedure <b>yUnregisterHub</b> ( ByVal <b>url</b> As String)
cs	void <b>UnregisterHub</b> ( string <b>url</b> )
java	void <b>UnregisterHub</b> ( String <b>url</b> )
py	def <b>UnregisterHub</b> ( <b>url</b> )

### Paramètres :

**url** une chaîne de caractères contenant "usb" ou

## YAPI.UpdateDeviceList() yUpdateDeviceList()yUpdateDeviceList()

YAPI

Force une mise-à-jour de la liste des modules Yoctopuce connectés.

js	function <b>yUpdateDeviceList</b> ( <b>errmsg</b> )
nodejs	function <b>UpdateDeviceList</b> ( <b>errmsg</b> )
php	function <b>yUpdateDeviceList</b> ( <b>&amp;\$errmsg</b> )
cpp	YRETCODE <b>yUpdateDeviceList</b> ( string& <b>errmsg</b> )
m	YRETCODE <b>yUpdateDeviceList</b> ( NSError** <b>errmsg</b> )
pas	function <b>yUpdateDeviceList</b> ( var <b>errmsg</b> : string): integer
vb	function <b>yUpdateDeviceList</b> ( ByRef <b>errmsg</b> As String) As YRETCODE
cs	YRETCODE <b>UpdateDeviceList</b> ( ref string <b>errmsg</b> )
java	int <b>UpdateDeviceList</b> ( )
py	def <b>UpdateDeviceList</b> ( <b>errmsg</b> =None)

La librairie va vérifier sur les machines ou ports USB précédemment enregistrés en utilisant la fonction `yRegisterHub` si un module a été connecté ou déconnecté, et le cas échéant appeler les fonctions de callback définies par l'utilisateur.

Cette fonction peut être appelée aussi souvent que désiré, afin de rendre l'application réactive aux événements de hot-plug.

### Paramètres :

**errmsg** une chaîne de caractères passée par référence, dans laquelle sera stocké un éventuel message d'erreur.

### Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## YAPI.UpdateDeviceList\_async() yUpdateDeviceList\_async()

YAPI

Force une mise-à-jour de la liste des modules Yoctopuce connectés.

```
js function yUpdateDeviceList_async( callback, context)
nodejs function UpdateDeviceList_async( callback, context)
```

La librairie va vérifier sur les machines ou ports USB précédemment enregistrés en utilisant la fonction `yRegisterHub` si un module a été connecté ou déconnecté, et le cas échéant appeler les fonctions de callback définies par l'utilisateur.

Cette fonction peut être appelée aussi souvent que désiré, afin de rendre l'application réactive aux événements de hot-plug.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

### Paramètres :

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et le code de retour (`YAPI_SUCCESS` si l'opération se déroule sans erreur).
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



## 3.2. Interface de la fonction Accelerometer

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_accelerometer.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YAccelerometer = yoctolib.YAccelerometer;</code>
php	<code>require_once('yocto_accelerometer.php');</code>
c++	<code>#include "yocto_accelerometer.h"</code>
m	<code>#import "yocto_accelerometer.h"</code>
pas	<code>uses yocto_accelerometer;</code>
vb	<code>yocto_accelerometer.vb</code>
cs	<code>yocto_accelerometer.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YAccelerometer;</code>
py	<code>from yocto_accelerometer import *</code>

### Fonction globales

#### **yFindAccelerometer(func)**

Permet de retrouver un accéléromètre d'après un identifiant donné.

#### **yFirstAccelerometer()**

Commence l'énumération des accéléromètres accessibles par la librairie.

### Méthodes des objets YAccelerometer

#### **accelerometer→calibrateFromPoints(rawValues, refValues)**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### **accelerometer→describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'accéléromètre au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### **accelerometer→get\_advertisedValue()**

Retourne la valeur courante de l'accéléromètre (pas plus de 6 caractères).

#### **accelerometer→get\_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### **accelerometer→get\_currentValue()**

Retourne la valeur actuelle de l'accélération.

#### **accelerometer→get\_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'accéléromètre.

#### **accelerometer→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'accéléromètre.

#### **accelerometer→get\_friendlyName()**

Retourne un identifiant global de l'accéléromètre au format `NOM_MODULE . NOM_FONCTION`.

#### **accelerometer→get\_functionDescriptor()**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **accelerometer→get\_functionId()**

Retourne l'identifiant matériel de l'accéléromètre, sans référence au module.

#### **accelerometer→get\_hardwareId()**

Retourne l'identifiant matériel unique de l'accéléromètre au format `SERIAL . FUNCTIONID`.

**accelerometer→get\_highestValue()**

Retourne la valeur maximale observée pour l'accélération depuis le démarrage du module.

**accelerometer→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**accelerometer→get\_logicalName()**

Retourne le nom logique de l'accéléromètre.

**accelerometer→get\_lowestValue()**

Retourne la valeur minimale observée pour l'accélération depuis le démarrage du module.

**accelerometer→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**accelerometer→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**accelerometer→get\_recordedData(startTime, endTime)**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

**accelerometer→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**accelerometer→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**accelerometer→get\_unit()**

Retourne l'unité dans laquelle l'accélération est exprimée.

**accelerometer→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

**accelerometer→get\_xValue()**

Retourne la composante X de l'accélération, sous forme de nombre à virgule.

**accelerometer→get\_yValue()**

Retourne la composante Y de l'accélération, sous forme de nombre à virgule.

**accelerometer→get\_zValue()**

Retourne la composante Z de l'accélération, sous forme de nombre à virgule.

**accelerometer→isOnline()**

Vérifie si le module hébergeant l'accéléromètre est joignable, sans déclencher d'erreur.

**accelerometer→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'accéléromètre est joignable, sans déclencher d'erreur.

**accelerometer→load(msValidity)**

Met en cache les valeurs courantes de l'accéléromètre, avec une durée de validité spécifiée.

**accelerometer→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.

**accelerometer→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'accéléromètre, avec une durée de validité spécifiée.

**accelerometer→nextAccelerometer()**

Continue l'énumération des accéléromètres commencée à l'aide de yFirstAccelerometer().

**accelerometer→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**accelerometer→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**accelerometer→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

**accelerometer→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**accelerometer→set\_logicalName(newval)**

Modifie le nom logique de l'accéléromètre.

**accelerometer→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

**accelerometer→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**accelerometer→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**accelerometer→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**accelerometer→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YAccelerometer.FindAccelerometer() yFindAccelerometer()yFindAccelerometer()

YAccelerometer

Permet de retrouver un accéléromètre d'après un identifiant donné.

js	function <b>yFindAccelerometer</b> ( <b>func</b> )
nodejs	function <b>FindAccelerometer</b> ( <b>func</b> )
php	function <b>yFindAccelerometer</b> ( <b>\$func</b> )
cpp	YAccelerometer* <b>yFindAccelerometer</b> ( const string& <b>func</b> )
m	YAccelerometer* <b>yFindAccelerometer</b> ( NSString* <b>func</b> )
pas	function <b>yFindAccelerometer</b> ( <b>func</b> : string): TYAccelerometer
vb	function <b>yFindAccelerometer</b> ( ByVal <b>func</b> As String) As YAccelerometer
cs	YAccelerometer <b>FindAccelerometer</b> ( string <b>func</b> )
java	YAccelerometer <b>FindAccelerometer</b> ( String <b>func</b> )
py	def <b>FindAccelerometer</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'accéléromètre soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YAccelerometer.isOnline()` pour tester si l'accéléromètre est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'accéléromètre sans ambiguïté

### Retourne :

un objet de classe `YAccelerometer` qui permet ensuite de contrôler l'accéléromètre.

## YAccelerometer.FirstAccelerometer() yFirstAccelerometer()yFirstAccelerometer()

## YAccelerometer

Commence l'énumération des accéléromètres accessibles par la librairie.

js	function <b>yFirstAccelerometer</b> ( )
nodejs	function <b>FirstAccelerometer</b> ( )
php	function <b>yFirstAccelerometer</b> ( )
cpp	YAccelerometer* <b>yFirstAccelerometer</b> ( )
m	YAccelerometer* <b>yFirstAccelerometer</b> ( )
pas	function <b>yFirstAccelerometer</b> ( ): TYAccelerometer
vb	function <b>yFirstAccelerometer</b> ( ) As YAccelerometer
cs	YAccelerometer <b>FirstAccelerometer</b> ( )
java	YAccelerometer <b>FirstAccelerometer</b> ( )
py	def <b>FirstAccelerometer</b> ( )

Utiliser la fonction `YAccelerometer.nextAccelerometer()` pour itérer sur les autres accéléromètres.

### Retourne :

un pointeur sur un objet `YAccelerometer`, correspondant à le premier accéléromètre accessible en ligne, ou `null` si il n'y a pas de accéléromètres disponibles.

## accelerometer→calibrateFromPoints()[accelerometer calibrateFromPoints: ]

YAccelerometer

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YAccelerometer target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→describe()[accelerometer describe]****YAccelerometer**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'accéléromètre au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'accéléromètre (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**accelerometer**→**get\_advertisedValue()****YAccelerometer****accelerometer**→**advertisedValue()[accelerometer  
advertisedValue]**

Retourne la valeur courante de l'accéléromètre (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'accéléromètre (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.



**accelerometer→get\_currentRawValue()****YAccelerometer****accelerometer→currentRawValue()[accelerometer  
currentRawValue]**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**accelerometer**→**get\_currentValue()****YAccelerometer****accelerometer**→**currentValue()**[**accelerometer**  
**currentValue**]

Retourne la valeur actuelle de l'accélération.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de l'accélération

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**accelerometer**→**get\_errorMessage()****YAccelerometer****accelerometer**→**errorMessage()**[**accelerometer**  
**errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'accéléromètre.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'accéléromètre.

## **accelerometer→get\_errorType()** **accelerometer→errorType()**

**YAccelerometer**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'accéléromètre.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

### **Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'accéléromètre.

**accelerometer**→**get\_friendlyName()**  
**accelerometer**→**friendlyName()**[**accelerometer**  
**friendlyName**]

**YAccelerometer**

Retourne un identifiant global de l'accéléromètre au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'accéléromètre si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'accéléromètre (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'accéléromètre en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**accelerometer→get\_functionDescriptor()****YAccelerometer****accelerometer→functionDescriptor()[accelerometer  
functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**accelerometer→get\_functionId()****YAccelerometer****accelerometer→functionId()[accelerometer  
functionId]**

Retourne l'identifiant matériel de l'accéléromètre, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'accéléromètre (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**accelerometer**→**get\_hardwareId()**

**YAccelerometer**

**accelerometer**→**hardwareId()**[**accelerometer**  
**hardwareId**]

Retourne l'identifiant matériel unique de l'accéléromètre au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'accéléromètre (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'accéléromètre (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**accelerometer**→**get\_highestValue()****YAccelerometer****accelerometer**→**highestValue()**[**accelerometer**  
**highestValue**]

Retourne la valeur maximale observée pour l'accélération depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
node.js	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour l'accélération depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**accelerometer**→**get\_logFrequency()****YAccelerometer****accelerometer**→**logFrequency()**[**accelerometer**  
**logFrequency**]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**accelerometer**→**get\_logicalName()****YAccelerometer****accelerometer**→**logicalName()**[**accelerometer**  
**logicalName**]

Retourne le nom logique de l'accéléromètre.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'accéléromètre. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**accelerometer**→**get\_lowestValue()****YAccelerometer****accelerometer**→**lowestValue()**[**accelerometer**  
**lowestValue**]

Retourne la valeur minimale observée pour l'accélération depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour l'accélération depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**accelerometer→get\_module()****YAccelerometer****accelerometer→module()[accelerometer module]**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**accelerometer**→**get\_module\_async()**  
**accelerometer**→**module\_async()**

**YAccelerometer**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**accelerometer**→**get\_recordedData()****YAccelerometer****accelerometer**→**recordedData()[accelerometer  
recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YAccelerometer <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**accelerometer→get\_reportFrequency()****YAccelerometer****accelerometer→reportFrequency()[accelerometer  
reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.



**accelerometer→get\_resolution()**  
**accelerometer→resolution()[accelerometer resolution]**

**YAccelerometer**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**accelerometer**→**get\_unit()****YAccelerometer****accelerometer**→**unit()**[**accelerometer unit**]

Retourne l'unité dans laquelle l'accélération est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle l'accélération est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**accelerometer→get\_userData()****YAccelerometer****accelerometer→userData()[accelerometer userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
c++	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**accelerometer→get\_xValue()**

**YAccelerometer**

**accelerometer→xValue()[accelerometer xValue]**

Retourne la composante X de l'accélération, sous forme de nombre à virgule.

js	function <b>get_xValue</b> ( )
nodejs	function <b>get_xValue</b> ( )
php	function <b>get_xValue</b> ( )
cpp	double <b>get_xValue</b> ( )
m	-(double) xValue
pas	function <b>get_xValue</b> ( ): double
vb	function <b>get_xValue</b> ( ) As Double
cs	double <b>get_xValue</b> ( )
java	double <b>get_xValue</b> ( )
py	def <b>get_xValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_xValue</b>

**Retourne :**

une valeur numérique représentant la composante X de l'accélération, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_XVALUE\_INVALID.

**accelerometer→get\_yValue()****YAccelerometer****accelerometer→yValue()[accelerometer yValue]**

Retourne la composante Y de l'accélération, sous forme de nombre à virgule.

js	function <b>get_yValue</b> ( )
nodejs	function <b>get_yValue</b> ( )
php	function <b>get_yValue</b> ( )
cpp	double <b>get_yValue</b> ( )
m	-(double) yValue
pas	function <b>get_yValue</b> ( ): double
vb	function <b>get_yValue</b> ( ) As Double
cs	double <b>get_yValue</b> ( )
java	double <b>get_yValue</b> ( )
py	def <b>get_yValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_yValue</b>

**Retourne :**

une valeur numérique représentant la composante Y de l'accélération, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_YVALUE\_INVALID.

**accelerometer→get\_zValue()**

**YAccelerometer**

**accelerometer→zValue()[accelerometer zValue]**

Retourne la composante Z de l'accélération, sous forme de nombre à virgule.

js	function <b>get_zValue</b> ( )
nodejs	function <b>get_zValue</b> ( )
php	function <b>get_zValue</b> ( )
cpp	double <b>get_zValue</b> ( )
m	-(double) zValue
pas	function <b>get_zValue</b> ( ): double
vb	function <b>get_zValue</b> ( ) As Double
cs	double <b>get_zValue</b> ( )
java	double <b>get_zValue</b> ( )
py	def <b>get_zValue</b> ( )
cmd	YAccelerometer <b>target</b> <b>get_zValue</b>

**Retourne :**

une valeur numérique représentant la composante Z de l'accélération, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_ZVALUE\_INVALID.

**accelerometer→isOnline()[accelerometer isOnline]****YAccelerometer**

Vérifie si le module hébergeant l'accéléromètre est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'accéléromètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'accéléromètre est joignable, `false` sinon

**accelerometer**→**isOnline\_async()****YAccelerometer**

Vérifie si le module hébergeant l'accéléromètre est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'accéléromètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**accelerometer→load()[accelerometer load: ]****YAccelerometer**

Met en cache les valeurs courantes de l'accéléromètre, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→loadCalibrationPoints()****YAccelerometer****[accelerometer loadCalibrationPoints: ]**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

js	<code>function loadCalibrationPoints( rawValues, refValues)</code>
nodejs	<code>function loadCalibrationPoints( rawValues, refValues)</code>
php	<code>function loadCalibrationPoints( &amp;\$rawValues, &amp;\$refValues)</code>
cpp	<code>int loadCalibrationPoints( vector&lt;double&gt;&amp; rawValues, vector&lt;double&gt;&amp; refValues)</code>
m	<code>-(int) loadCalibrationPoints : (NSMutableArray*) rawValues : (NSMutableArray*) refValues</code>
pas	<code>function loadCalibrationPoints( var rawValues: TDoubleArray, var refValues: TDoubleArray): LongInt</code>
vb	<code>procedure loadCalibrationPoints( )</code>
cs	<code>int loadCalibrationPoints( List&lt;double&gt; rawValues, List&lt;double&gt; refValues)</code>
java	<code>int loadCalibrationPoints( ArrayList&lt;Double&gt; rawValues, ArrayList&lt;Double&gt; refValues)</code>
py	<code>def loadCalibrationPoints( rawValues, refValues)</code>
cmd	<code>YAccelerometer target loadCalibrationPoints rawValues refValues</code>

**Paramètres :**

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

**Retourne :**

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→load\_async()****YAccelerometer**

Met en cache les valeurs courantes de l'accéléromètre, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**accelerometer**→**nextAccelerometer()**[**accelerometer**  
**nextAccelerometer**]**YAccelerometer**

Continue l'énumération des accéléromètres commencée à l'aide de `yFirstAccelerometer()`.

js	function <b>nextAccelerometer</b> ( )
nodejs	function <b>nextAccelerometer</b> ( )
php	function <b>nextAccelerometer</b> ( )
cpp	YAccelerometer * <b>nextAccelerometer</b> ( )
m	-(YAccelerometer*) <b>nextAccelerometer</b>
pas	function <b>nextAccelerometer</b> ( ): TYAccelerometer
vb	function <b>nextAccelerometer</b> ( ) As YAccelerometer
cs	YAccelerometer <b>nextAccelerometer</b> ( )
java	YAccelerometer <b>nextAccelerometer</b> ( )
py	def <b>nextAccelerometer</b> ( )

**Retourne :**

un pointeur sur un objet `YAccelerometer` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**accelerometer→registerTimedReportCallback()****YAccelerometer****[accelerometer registerTimedReportCallback: ]**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YAccelerometerTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YAccelerometerTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYAccelerometerTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## accelerometer→registerValueCallback() [accelerometer registerValueCallback: ]

YAccelerometer

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YAccelerometerValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YAccelerometerValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYAccelerometerValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**accelerometer→set\_highestValue()****YAccelerometer****accelerometer→setHighestValue()[accelerometer  
setHighestValue: ]**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target</b> <b>set_highestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→set\_logFrequency()****YAccelerometer****accelerometer→setLogFrequency()[accelerometer  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**accelerometer→set\_logicalName()****YAccelerometer****accelerometer→setLogicalName()[accelerometer  
setLogicalName: ]**

Modifie le nom logique de l'accéléromètre.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'accéléromètre.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer**→**set\_lowestValue()****YAccelerometer****accelerometer**→**setLowestValue()**[**accelerometer**  
**setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→set\_reportFrequency()****YAccelerometer****accelerometer→setReportFrequency()[accelerometer  
setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer**→**set\_resolution()****YAccelerometer****accelerometer**→**setResolution()**[**accelerometer**  
**setResolution:** ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YAccelerometer <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**accelerometer→set\_userdata()****YAccelerometer****accelerometer→setUserData()[accelerometer  
setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## accelerometer→wait\_async()

## YAccelerometer

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :

### 3.3. Interface de la fonction AnButton

La librairie de programmation Yoctopuce permet aussi bien de mesurer l'état d'un simple bouton que de lire un potentiomètre analogique (résistance variable), comme par exemple bouton rotatif continue, une poignée de commande de gaz ou un joystick. Le module est capable de se calibrer sur les valeurs minimales et maximales du potentiomètre, et de restituer une valeur calibrée variant proportionnellement avec la position du potentiomètre, indépendant de sa résistance totale.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_anbutton.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YAnButton = yoctolib.YAnButton;</code>
php	<code>require_once('yocto_anbutton.php');</code>
c++	<code>#include "yocto_anbutton.h"</code>
m	<code>#import "yocto_anbutton.h"</code>
pas	<code>uses yocto_anbutton;</code>
vb	<code>yocto_anbutton.vb</code>
cs	<code>yocto_anbutton.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YAnButton;</code>
py	<code>from yocto_anbutton import *</code>

#### Fonction globales

##### **yFindAnButton(func)**

Permet de retrouver une entrée analogique d'après un identifiant donné.

##### **yFirstAnButton()**

Commence l'énumération des entrées analogiques accessibles par la librairie.

#### Méthodes des objets YAnButton

##### **anbutton→describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'entrée analogique au format `TYPE ( NAME ) =SERIAL . FUNCTIONID`.

##### **anbutton→get\_advertisedValue()**

Retourne la valeur courante de l'entrée analogique (pas plus de 6 caractères).

##### **anbutton→get\_analogCalibration()**

Permet de savoir si une procédure de calibration est actuellement en cours.

##### **anbutton→get\_calibratedValue()**

Retourne la valeur calibrée de l'entrée (entre 0 et 1000 inclus).

##### **anbutton→get\_calibrationMax()**

Retourne la valeur maximale observée durant la calibration (entre 0 et 4095 inclus).

##### **anbutton→get\_calibrationMin()**

Retourne la valeur minimale observée durant la calibration (entre 0 et 4095 inclus).

##### **anbutton→get\_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'entrée analogique.

##### **anbutton→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'entrée analogique.

##### **anbutton→get\_friendlyName()**

Retourne un identifiant global de l'entrée analogique au format `NOM_MODULE . NOM_FONCTION`.

##### **anbutton→get\_functionDescriptor()**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

#### **anbutton→get\_functionId()**

Retourne l'identifiant matériel de l'entrée analogique, sans référence au module.

#### **anbutton→get\_hardwareId()**

Retourne l'identifiant matériel unique de l'entrée analogique au format SERIAL . FUNCTIONID.

#### **anbutton→get\_isPressed()**

Retourne vrai si l'entrée (considérée comme binaire) est active (contact fermé), et faux sinon.

#### **anbutton→get\_lastTimePressed()**

Retourne le temps absolu (nombre de millisecondes) entre la mise sous tension du module et la dernière pression observée du bouton à l'entrée (transition du contact de ouvert à fermé).

#### **anbutton→get\_lastTimeReleased()**

Retourne le temps absolu (nombre de millisecondes) entre la mise sous tension du module et le dernier relâchement observée du bouton à l'entrée (transition du contact de fermé à ouvert).

#### **anbutton→get\_logicalName()**

Retourne le nom logique de l'entrée analogique.

#### **anbutton→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

#### **anbutton→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

#### **anbutton→get\_pulseCounter()**

Retourne la valeur du compteur d'impulsions.

#### **anbutton→get\_pulseTimer()**

Retourne le timer du compteur d'impulsions (ms)

#### **anbutton→get\_rawValue()**

Retourne la valeur mesurée de l'entrée telle-quelle (entre 0 et 4095 inclus).

#### **anbutton→get\_sensitivity()**

Retourne la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks.

#### **anbutton→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

#### **anbutton→isOnline()**

Vérifie si le module hébergeant l'entrée analogique est joignable, sans déclencher d'erreur.

#### **anbutton→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'entrée analogique est joignable, sans déclencher d'erreur.

#### **anbutton→load(msValidity)**

Met en cache les valeurs courantes de l'entrée analogique, avec une durée de validité spécifiée.

#### **anbutton→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'entrée analogique, avec une durée de validité spécifiée.

#### **anbutton→nextAnButton()**

Continue l'énumération des entrées analogiques commencée à l'aide de yFirstAnButton( ).

#### **anbutton→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **anbutton→resetCounter()**

réinitialise le compteur d'impulsions et son timer

#### **anbutton→set\_analogCalibration(newval)**

Enclenche ou déclenche le procédure de calibration.

#### **anbutton→set\_calibrationMax(newval)**



Modifie la valeur maximale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique.

**anbutton→set\_calibrationMin(newval)**

Modifie la valeur minimale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique.

**anbutton→set\_logicalName(newval)**

Modifie le nom logique de l'entrée analogique.

**anbutton→set\_sensitivity(newval)**

Modifie la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks.

**anbutton→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userData.

**anbutton→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YAnButton.FindAnButton() yFindAnButton()yFindAnButton()

YAnButton

Permet de retrouver une entrée analogique d'après un identifiant donné.

js	function <b>yFindAnButton</b> ( <b>func</b> )
nodejs	function <b>FindAnButton</b> ( <b>func</b> )
php	function <b>yFindAnButton</b> ( <b>\$func</b> )
c++	YAnButton* <b>yFindAnButton</b> ( const string& <b>func</b> )
m	YAnButton* <b>yFindAnButton</b> ( NSString* <b>func</b> )
pas	function <b>yFindAnButton</b> ( <b>func</b> : string): TYAnButton
vb	function <b>yFindAnButton</b> ( ByVal <b>func</b> As String) As YAnButton
cs	YAnButton <b>FindAnButton</b> ( string <b>func</b> )
java	YAnButton <b>FindAnButton</b> ( String <b>func</b> )
py	def <b>FindAnButton</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'entrée analogique soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YAnButton.isOnline()` pour tester si l'entrée analogique est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'entrée analogique sans ambiguïté

### Retourne :

un objet de classe `YAnButton` qui permet ensuite de contrôler l'entrée analogique.

## YAnButton.FirstAnButton() yFirstAnButton()yFirstAnButton()

## YAnButton

Commence l'énumération des entrées analogiques accessibles par la librairie.

js	function <b>yFirstAnButton</b> ( )
nodejs	function <b>FirstAnButton</b> ( )
php	function <b>yFirstAnButton</b> ( )
cpp	YAnButton* <b>yFirstAnButton</b> ( )
m	YAnButton* <b>yFirstAnButton</b> ( )
pas	function <b>yFirstAnButton</b> ( ): TYAnButton
vb	function <b>yFirstAnButton</b> ( ) As YAnButton
cs	YAnButton <b>FirstAnButton</b> ( )
java	YAnButton <b>FirstAnButton</b> ( )
py	def <b>FirstAnButton</b> ( )

Utiliser la fonction `YAnButton.nextAnButton( )` pour itérer sur les autres entrées analogiques.

### Retourne :

un pointeur sur un objet `YAnButton`, correspondant à la première entrée analogique accessible en ligne, ou `null` si il n'y a pas de entrées analogiques disponibles.

**anbutton→describe()[anbutton describe]****YAnButton**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'entrée analogique au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'entrée analogique (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**anbutton→get\_advertisedValue()**  
**anbutton→advertisedValue()[anbutton**  
**advertisedValue]**

**YAnButton**

Retourne la valeur courante de l'entrée analogique (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YAnButton <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'entrée analogique (pas plus de 6 caractères).  
 En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

## **anbutton→get\_analogCalibration()** **anbutton→analogCalibration()[anbutton** **analogCalibration]**

**YAnButton**

Permet de savoir si une procédure de calibration est actuellement en cours.

js	function <b>get_analogCalibration</b> ( )
nodejs	function <b>get_analogCalibration</b> ( )
php	function <b>get_analogCalibration</b> ( )
cpp	Y_ANALOGCALIBRATION_enum <b>get_analogCalibration</b> ( )
m	-(Y_ANALOGCALIBRATION_enum) analogCalibration
pas	function <b>get_analogCalibration</b> ( ): Integer
vb	function <b>get_analogCalibration</b> ( ) As Integer
cs	int <b>get_analogCalibration</b> ( )
java	int <b>get_analogCalibration</b> ( )
py	def <b>get_analogCalibration</b> ( )
cmd	YAnButton <b>target</b> <b>get_analogCalibration</b>

**Retourne :**

soit Y\_ANALOGCALIBRATION\_OFF, soit Y\_ANALOGCALIBRATION\_ON

En cas d'erreur, déclenche une exception ou retourne Y\_ANALOGCALIBRATION\_INVALID.

**anbutton→get\_calibratedValue()**  
**anbutton→calibratedValue()[anbutton**  
**calibratedValue]**

**YAnButton**

Retourne la valeur calibrée de l'entrée (entre 0 et 1000 inclus).

js	function <b>get_calibratedValue</b> ( )
nodejs	function <b>get_calibratedValue</b> ( )
php	function <b>get_calibratedValue</b> ( )
cpp	int <b>get_calibratedValue</b> ( )
m	-(int) calibratedValue
pas	function <b>get_calibratedValue</b> ( ): LongInt
vb	function <b>get_calibratedValue</b> ( ) As Integer
cs	int <b>get_calibratedValue</b> ( )
java	int <b>get_calibratedValue</b> ( )
py	def <b>get_calibratedValue</b> ( )
cmd	YAnButton <b>target</b> <b>get_calibratedValue</b>

**Retourne :**

un entier représentant la valeur calibrée de l'entrée (entre 0 et 1000 inclus)

En cas d'erreur, déclenche une exception ou retourne Y\_CALIBRATEDVALUE\_INVALID.

**anbutton→get\_calibrationMax()****YAnButton****anbutton→calibrationMax()[anbutton calibrationMax]**

Retourne la valeur maximale observée durant la calibration (entre 0 et 4095 inclus).

js	function <b>get_calibrationMax</b> ( )
nodejs	function <b>get_calibrationMax</b> ( )
php	function <b>get_calibrationMax</b> ( )
cpp	int <b>get_calibrationMax</b> ( )
m	-(int) calibrationMax
pas	function <b>get_calibrationMax</b> ( ): LongInt
vb	function <b>get_calibrationMax</b> ( ) As Integer
cs	int <b>get_calibrationMax</b> ( )
java	int <b>get_calibrationMax</b> ( )
py	def <b>get_calibrationMax</b> ( )
cmd	YAnButton <b>target</b> <b>get_calibrationMax</b>

**Retourne :**

un entier représentant la valeur maximale observée durant la calibration (entre 0 et 4095 inclus)

En cas d'erreur, déclenche une exception ou retourne Y\_CALIBRATIONMAX\_INVALID.



**anbutton→get\_calibrationMin()****YAnButton****anbutton→calibrationMin()[anbutton calibrationMin]**

Retourne la valeur minimale observée durant la calibration (entre 0 et 4095 inclus).

js	function <b>get_calibrationMin</b> ( )
nodejs	function <b>get_calibrationMin</b> ( )
php	function <b>get_calibrationMin</b> ( )
cpp	int <b>get_calibrationMin</b> ( )
m	-(int) calibrationMin
pas	function <b>get_calibrationMin</b> ( ): LongInt
vb	function <b>get_calibrationMin</b> ( ) As Integer
cs	int <b>get_calibrationMin</b> ( )
java	int <b>get_calibrationMin</b> ( )
py	def <b>get_calibrationMin</b> ( )
cmd	YAnButton <b>target</b> <b>get_calibrationMin</b>

**Retourne :**

un entier représentant la valeur minimale observée durant la calibration (entre 0 et 4095 inclus)

En cas d'erreur, déclenche une exception ou retourne Y\_CALIBRATIONMIN\_INVALID.

**anbutton→get\_errorMessage()****YAnButton****anbutton→errorMessage()[anbutton errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'entrée analogique.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'entrée analogique.

**anbutton→get\_errorType()**  
**anbutton→errorType()****YAnButton**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'entrée analogique.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'entrée analogique.

**anbutton**→**get\_friendlyName()****YAnButton****anbutton**→**friendlyName()**[anbutton friendlyName]

Retourne un identifiant global de l'entrée analogique au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'entrée analogique si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'entrée analogique (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'entrée analogique en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**anbutton→get\_functionDescriptor()**  
**anbutton→functionDescriptor()[anbutton**  
**functionDescriptor]**

**YAnButton**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**anbutton**→**get\_functionId()****YAnButton****anbutton**→**functionId()**[anbutton functionId]

Retourne l'identifiant matériel de l'entrée analogique, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'entrée analogique (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**anbutton→get\_hardwareId()****YAnButton****anbutton→hardwareId()[anbutton hardwareId]**

Retourne l'identifiant matériel unique de l'entrée analogique au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'entrée analogique (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'entrée analogique (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**anbutton**→**get\_isPressed()****YAnButton****anbutton**→**isPressed()**[**anbutton isPressed**]

Retourne vrai si l'entrée (considérée comme binaire) est active (contact fermé), et faux sinon.

js	function <b>get_isPressed</b> ( )
nodejs	function <b>get_isPressed</b> ( )
php	function <b>get_isPressed</b> ( )
cpp	Y_ISPRESSED_enum <b>get_isPressed</b> ( )
m	-(Y_ISPRESSED_enum) isPressed
pas	function <b>get_isPressed</b> ( ): Integer
vb	function <b>get_isPressed</b> ( ) As Integer
cs	int <b>get_isPressed</b> ( )
java	int <b>get_isPressed</b> ( )
py	def <b>get_isPressed</b> ( )
cmd	YAnButton <b>target get_isPressed</b>

**Retourne :**

soit Y\_ISPRESSED\_FALSE, soit Y\_ISPRESSED\_TRUE, selon vrai si l'entrée (considérée comme binaire) est active (contact fermé), et faux sinon

En cas d'erreur, déclenche une exception ou retourne Y\_ISPRESSED\_INVALID.



## anbutton→get\_lastTimePressed() anbutton→lastTimePressed()[anbutton lastTimePressed]

YAnButton

Retourne le temps absolu (nombre de millisecondes) entre la mise sous tension du module et la dernière pression observée du bouton à l'entrée (transition du contact de ouvert à fermé).

js	function <b>get_lastTimePressed</b> ( )
nodejs	function <b>get_lastTimePressed</b> ( )
php	function <b>get_lastTimePressed</b> ( )
c++	s64 <b>get_lastTimePressed</b> ( )
m	-(s64) lastTimePressed
pas	function <b>get_lastTimePressed</b> ( ): int64
vb	function <b>get_lastTimePressed</b> ( ) As Long
cs	long <b>get_lastTimePressed</b> ( )
java	long <b>get_lastTimePressed</b> ( )
py	def <b>get_lastTimePressed</b> ( )
cmd	YAnButton <b>target</b> <b>get_lastTimePressed</b>

### Retourne :

un entier représentant le temps absolu (nombre de millisecondes) entre la mise sous tension du module et la dernière pression observée du bouton à l'entrée (transition du contact de ouvert à fermé)

En cas d'erreur, déclenche une exception ou retourne Y\_LASTTIMEPRESSED\_INVALID.

**anbutton**→**get\_lastTimeReleased()**  
**anbutton**→**lastTimeReleased()**[**anbutton**  
**lastTimeReleased**]

**YAnButton**

Retourne le temps absolu (nombre de millisecondes) entre la mise sous tension du module et le dernier relâchement observée du bouton à l'entrée (transition du contact de fermé à ouvert).

js	function <b>get_lastTimeReleased</b> ( )
nodejs	function <b>get_lastTimeReleased</b> ( )
php	function <b>get_lastTimeReleased</b> ( )
cpp	s64 <b>get_lastTimeReleased</b> ( )
m	-(s64) lastTimeReleased
pas	function <b>get_lastTimeReleased</b> ( ): int64
vb	function <b>get_lastTimeReleased</b> ( ) As Long
cs	long <b>get_lastTimeReleased</b> ( )
java	long <b>get_lastTimeReleased</b> ( )
py	def <b>get_lastTimeReleased</b> ( )
cmd	YAnButton <b>target</b> <b>get_lastTimeReleased</b>

**Retourne :**

un entier représentant le temps absolu (nombre de millisecondes) entre la mise sous tension du module et le dernier relâchement observée du bouton à l'entrée (transition du contact de fermé à ouvert)

En cas d'erreur, déclenche une exception ou retourne Y\_LASTTIMERELEASED\_INVALID.

**anbutton→get\_logicalName()****YAnButton****anbutton→logicalName()[anbutton logicalName]**

Retourne le nom logique de l'entrée analogique.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YAnButton <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'entrée analogique. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**anbutton**→**get\_module()****YAnButton****anbutton**→**module()**[anbutton module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	<code>YModule *</code> <b>get_module</b> ( )
m	-( <code>YModule*</code> ) module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**anbutton**→**get\_module\_async()****YAnButton****anbutton**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**anbutton**→**get\_pulseCounter()****YAnButton****anbutton**→**pulseCounter()**[anbutton pulseCounter]

Retourne la valeur du compteur d'impulsions.

js	function <b>get_pulseCounter</b> ( )
nodejs	function <b>get_pulseCounter</b> ( )
php	function <b>get_pulseCounter</b> ( )
cpp	s64 <b>get_pulseCounter</b> ( )
m	-(s64) pulseCounter
pas	function <b>get_pulseCounter</b> ( ): int64
vb	function <b>get_pulseCounter</b> ( ) As Long
cs	long <b>get_pulseCounter</b> ( )
java	long <b>get_pulseCounter</b> ( )
py	def <b>get_pulseCounter</b> ( )

**Retourne :**

un entier représentant la valeur du compteur d'impulsions

En cas d'erreur, déclenche une exception ou retourne Y\_PULSECOUNTER\_INVALID.

**anbutton→get\_pulseTimer()****YAnButton****anbutton→pulseTimer()[anbutton pulseTimer]**

Retourne le timer du compteur d'impulsions (ms)

js	function <b>get_pulseTimer</b> ( )
nodejs	function <b>get_pulseTimer</b> ( )
php	function <b>get_pulseTimer</b> ( )
cpp	s64 <b>get_pulseTimer</b> ( )
m	-(s64) pulseTimer
pas	function <b>get_pulseTimer</b> ( ): int64
vb	function <b>get_pulseTimer</b> ( ) As Long
cs	long <b>get_pulseTimer</b> ( )
java	long <b>get_pulseTimer</b> ( )
py	def <b>get_pulseTimer</b> ( )

**Retourne :**

un entier représentant le timer du compteur d'impulsions (ms)

En cas d'erreur, déclenche une exception ou retourne Y\_PULSETIMER\_INVALID.

**anbutton**→**get\_rawValue()****YAnButton****anbutton**→**rawValue()**[**anbutton rawValue**]

Retourne la valeur mesurée de l'entrée telle-quelle (entre 0 et 4095 inclus).

js	function <b>get_rawValue</b> ( )
nodejs	function <b>get_rawValue</b> ( )
php	function <b>get_rawValue</b> ( )
cpp	int <b>get_rawValue</b> ( )
m	-(int) rawValue
pas	function <b>get_rawValue</b> ( ): LongInt
vb	function <b>get_rawValue</b> ( ) As Integer
cs	int <b>get_rawValue</b> ( )
java	int <b>get_rawValue</b> ( )
py	def <b>get_rawValue</b> ( )
cmd	YAnButton <b>target</b> <b>get_rawValue</b>

**Retourne :**

un entier représentant la valeur mesurée de l'entrée telle-quelle (entre 0 et 4095 inclus)

En cas d'erreur, déclenche une exception ou retourne Y\_RAWVALUE\_INVALID.



**anbutton→get\_sensitivity()****YAnButton****anbutton→sensitivity()[anbutton sensitivity]**

Retourne la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks.

js	function <b>get_sensitivity</b> ( )
nodejs	function <b>get_sensitivity</b> ( )
php	function <b>get_sensitivity</b> ( )
cpp	int <b>get_sensitivity</b> ( )
m	-(int) sensitivity
pas	function <b>get_sensitivity</b> ( ): LongInt
vb	function <b>get_sensitivity</b> ( ) As Integer
cs	int <b>get_sensitivity</b> ( )
java	int <b>get_sensitivity</b> ( )
py	def <b>get_sensitivity</b> ( )
cmd	YAnButton <b>target</b> <b>get_sensitivity</b>

**Retourne :**

un entier représentant la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks

En cas d'erreur, déclenche une exception ou retourne Y\_SENSITIVITY\_INVALID.

**anbutton**→**get\_userData()**

**YAnButton**

**anbutton**→**userData()**[anbutton userData]

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**anbutton→isOnline()[anbutton isOnline]****YAnButton**

Vérifie si le module hébergeant l'entrée analogique est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'entrée analogique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si l'entrée analogique est joignable, false sinon

**anbutton→isOnline\_async()****YAnButton**

Vérifie si le module hébergeant l'entrée analogique est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'entrée analogique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**anbutton→load()**[anbutton load: ]**YAnButton**

Met en cache les valeurs courantes de l'entrée analogique, avec une durée de validité spécifiée.

js	function load( msValidity)
nodejs	function load( msValidity)
php	function load( \$msValidity)
cpp	YRETCODE load( int msValidity)
m	-(YRETCODE) load : (int) msValidity
pas	function load( msValidity: integer): YRETCODE
vb	function load( ByVal msValidity As Integer) As YRETCODE
cs	YRETCODE load( int msValidity)
java	int load( long msValidity)
py	def load( msValidity)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**anbutton→load\_async()****YAnButton**

Met en cache les valeurs courantes de l'entrée analogique, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**anbutton→nextAnButton()[anbutton nextAnButton]****YAnButton**

Continue l'énumération des entrées analogiques commencée à l'aide de `yFirstAnButton()`.

js	function <b>nextAnButton</b> ( )
nodejs	function <b>nextAnButton</b> ( )
php	function <b>nextAnButton</b> ( )
cpp	YAnButton * <b>nextAnButton</b> ( )
m	-(YAnButton*) <b>nextAnButton</b>
pas	function <b>nextAnButton</b> ( ): TYAnButton
vb	function <b>nextAnButton</b> ( ) As YAnButton
cs	YAnButton <b>nextAnButton</b> ( )
java	YAnButton <b>nextAnButton</b> ( )
py	def <b>nextAnButton</b> ( )

**Retourne :**

un pointeur sur un objet `YAnButton` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## anbutton→registerValueCallback()[anbutton registerValueCallback: ]

YAnButton

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
c++	int <b>registerValueCallback</b> ( YAnButtonValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YAnButtonValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYAnButtonValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.



**anbutton→resetCounter()[anbutton resetCounter]****YAnButton**

réinitialise le compteur d'impulsions et son timer

js	function <b>resetCounter</b> ( )
nodejs	function <b>resetCounter</b> ( )
php	function <b>resetCounter</b> ( )
cpp	int <b>resetCounter</b> ( )
m	-(int) <b>resetCounter</b>
pas	function <b>resetCounter</b> ( ): LongInt
vb	function <b>resetCounter</b> ( ) As Integer
cs	int <b>resetCounter</b> ( )
java	int <b>resetCounter</b> ( )
py	def <b>resetCounter</b> ( )

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

anbutton→set\_analogCalibration()

YAnButton

anbutton→setAnalogCalibration()[anbutton  
setAnalogCalibration: ]

Enclenche ou déclenche le procédure de calibration.

js	function <b>set_analogCalibration</b> ( <b>newval</b> )
nodejs	function <b>set_analogCalibration</b> ( <b>newval</b> )
php	function <b>set_analogCalibration</b> ( <b>\$newval</b> )
cpp	int <b>set_analogCalibration</b> ( Y_ANALOGCALIBRATION_enum <b>newval</b> )
m	-(int) setAnalogCalibration : (Y_ANALOGCALIBRATION_enum) <b>newval</b>
pas	function <b>set_analogCalibration</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_analogCalibration</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_analogCalibration</b> ( int <b>newval</b> )
java	int <b>set_analogCalibration</b> ( int <b>newval</b> )
py	def <b>set_analogCalibration</b> ( <b>newval</b> )
cmd	YAnButton <b>target set_analogCalibration newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module à la fin de la calibration si le réglage doit être préservé.

#### Paramètres :

**newval** soit Y\_ANALOGCALIBRATION\_OFF, soit Y\_ANALOGCALIBRATION\_ON

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**anbutton→set\_calibrationMax()****YAnButton****anbutton→setCalibrationMax()[anbutton  
setCalibrationMax: ]**

Modifie la valeur maximale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique.

js	function <b>set_calibrationMax</b> ( <b>newval</b> )
nodejs	function <b>set_calibrationMax</b> ( <b>newval</b> )
php	function <b>set_calibrationMax</b> ( <b>\$newval</b> )
cpp	int <b>set_calibrationMax</b> ( int <b>newval</b> )
m	-(int) setCalibrationMax : (int) <b>newval</b>
pas	function <b>set_calibrationMax</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_calibrationMax</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_calibrationMax</b> ( int <b>newval</b> )
java	int <b>set_calibrationMax</b> ( int <b>newval</b> )
py	def <b>set_calibrationMax</b> ( <b>newval</b> )
cmd	YAnButton <b>target</b> <b>set_calibrationMax</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant la valeur maximale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**anbutton→set\_calibrationMin()****YAnButton****anbutton→setCalibrationMin()[anbutton  
setCalibrationMin: ]**

Modifie la valeur minimale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique.

js	function <b>set_calibrationMin</b> ( <b>newval</b> )
nodejs	function <b>set_calibrationMin</b> ( <b>newval</b> )
php	function <b>set_calibrationMin</b> ( <b>\$newval</b> )
cpp	int <b>set_calibrationMin</b> ( int <b>newval</b> )
m	-(int) setCalibrationMin : (int) <b>newval</b>
pas	function <b>set_calibrationMin</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_calibrationMin</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_calibrationMin</b> ( int <b>newval</b> )
java	int <b>set_calibrationMin</b> ( int <b>newval</b> )
py	def <b>set_calibrationMin</b> ( <b>newval</b> )
cmd	YAnButton <b>target set_calibrationMin newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant la valeur minimale de calibration pour l'entrée (entre 0 et 4095 inclus), sans lancer la calibration automatique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## anbutton→set\_logicalName() anbutton→setLogicalName()[anbutton setLogicalName: ]

YAnButton

Modifie le nom logique de l'entrée analogique.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YAnButton <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le nom logique de l'entrée analogique.

### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**anbutton→set\_sensitivity()****YAnButton****anbutton→setSensitivity()[anbutton setSensitivity: ]**

Modifie la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks.

js	function <b>set_sensitivity</b> ( <b>newval</b> )
nodejs	function <b>set_sensitivity</b> ( <b>newval</b> )
php	function <b>set_sensitivity</b> ( <b>\$newval</b> )
cpp	int <b>set_sensitivity</b> ( int <b>newval</b> )
m	-(int) setSensitivity : (int) <b>newval</b>
pas	function <b>set_sensitivity</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_sensitivity</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_sensitivity</b> ( int <b>newval</b> )
java	int <b>set_sensitivity</b> ( int <b>newval</b> )
py	def <b>set_sensitivity</b> ( <b>newval</b> )
cmd	YAnButton <b>target set_sensitivity newval</b>

La sensibilité sert à filtrer les variations autour d'une valeur fixe, mais ne préterite pas la transmission d'événements lorsque la valeur d'entrée évolue constamment dans la même direction. Cas particulier: lorsque la valeur 1000 est utilisée, seuls les valeurs déclenchant une commutation d'état pressé/non-pressé sont transmises. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant la sensibilité pour l'entrée (entre 1 et 1000) pour le déclenchement de callbacks

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**anbutton→set\_userdata()****YAnButton****anbutton→setUserData()[anbutton setUserData: ]**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## anbutton→wait\_async()

YAnButton

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :



## 3.4. Interface de la fonction CarbonDioxide

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_carbondioxide.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YCarbonDioxide = yoctolib.YCarbonDioxide;
php	require_once('yocto_carbondioxide.php');
c++	#include "yocto_carbondioxide.h"
m	#import "yocto_carbondioxide.h"
pas	uses yocto_carbondioxide;
vb	yocto_carbondioxide.vb
cs	yocto_carbondioxide.cs
java	import com.yoctopuce.YoctoAPI.YCarbonDioxide;
py	from yocto_carbondioxide import *

### Fonction globales

#### yFindCarbonDioxide(func)

Permet de retrouver un capteur de CO2 d'après un identifiant donné.

#### yFirstCarbonDioxide()

Commence l'énumération des capteurs de CO2 accessibles par la librairie.

### Méthodes des objets YCarbonDioxide

#### carbondioxide→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### carbondioxide→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de CO2 au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### carbondioxide→get\_advertisedValue()

Retourne la valeur courante du capteur de CO2 (pas plus de 6 caractères).

#### carbondioxide→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### carbondioxide→get\_currentValue()

Retourne la valeur actuelle du taux de CO2.

#### carbondioxide→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de CO2.

#### carbondioxide→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de CO2.

#### carbondioxide→get\_friendlyName()

Retourne un identifiant global du capteur de CO2 au format `NOM_MODULE . NOM_FONCTION`.

#### carbondioxide→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### carbondioxide→get\_functionId()

Retourne l'identifiant matériel du capteur de CO2, sans référence au module.

#### carbondioxide→get\_hardwareId()

Retourne l'identifiant matériel unique du capteur de CO2 au format `SERIAL . FUNCTIONID`.

**carbondioxide→get\_highestValue()**

Retourne la valeur maximale observée pour le taux de CO2 depuis le démarrage du module.

**carbondioxide→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**carbondioxide→get\_logicalName()**

Retourne le nom logique du capteur de CO2.

**carbondioxide→get\_lowestValue()**

Retourne la valeur minimale observée pour le taux de CO2 depuis le démarrage du module.

**carbondioxide→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**carbondioxide→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**carbondioxide→get\_recordedData(startTime, endTime)**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

**carbondioxide→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**carbondioxide→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**carbondioxide→get\_unit()**

Retourne l'unité dans laquelle le taux de CO2 est exprimée.

**carbondioxide→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

**carbondioxide→isOnline()**

Vérifie si le module hébergeant le capteur de CO2 est joignable, sans déclencher d'erreur.

**carbondioxide→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le capteur de CO2 est joignable, sans déclencher d'erreur.

**carbondioxide→load(msValidity)**

Met en cache les valeurs courantes du capteur de CO2, avec une durée de validité spécifiée.

**carbondioxide→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.

**carbondioxide→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du capteur de CO2, avec une durée de validité spécifiée.

**carbondioxide→nextCarbonDioxide()**

Continue l'énumération des capteurs de CO2 commencée à l'aide de yFirstCarbonDioxide().

**carbondioxide→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**carbondioxide→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**carbondioxide→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

**carbondioxide→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**carbondioxide→set\_logicalName(newval)**

Modifie le nom logique du capteur de CO2.

**carbondioxide→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

**carbondioxide→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**carbondioxide→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**carbondioxide→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userData.

**carbondioxide→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YCarbonDioxide.FindCarbonDioxide() yFindCarbonDioxide()yFindCarbonDioxide()

## YCarbonDioxide

Permet de retrouver un capteur de CO2 d'après un identifiant donné.

js	function <b>yFindCarbonDioxide</b> ( <b>func</b> )
nodejs	function <b>FindCarbonDioxide</b> ( <b>func</b> )
php	function <b>yFindCarbonDioxide</b> ( <b>\$func</b> )
cpp	YCarbonDioxide* <b>yFindCarbonDioxide</b> ( const string& <b>func</b> )
m	YCarbonDioxide* <b>yFindCarbonDioxide</b> ( NSString* <b>func</b> )
pas	function <b>yFindCarbonDioxide</b> ( <b>func</b> : string): TYCarbonDioxide
vb	function <b>yFindCarbonDioxide</b> ( ByVal <b>func</b> As String) As YCarbonDioxide
cs	YCarbonDioxide <b>FindCarbonDioxide</b> ( string <b>func</b> )
java	YCarbonDioxide <b>FindCarbonDioxide</b> ( String <b>func</b> )
py	def <b>FindCarbonDioxide</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de CO2 soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YCarbonDioxide.isOnline()` pour tester si le capteur de CO2 est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de CO2 sans ambiguïté

### Retourne :

un objet de classe `YCarbonDioxide` qui permet ensuite de contrôler le capteur de CO2.

## YCarbonDioxide.FirstCarbonDioxide() yFirstCarbonDioxide()yFirstCarbonDioxide()

## YCarbonDioxide

Commence l'énumération des capteurs de CO2 accessibles par la librairie.

js	function <b>yFirstCarbonDioxide</b> ( )
nodejs	function <b>FirstCarbonDioxide</b> ( )
php	function <b>yFirstCarbonDioxide</b> ( )
cpp	YCarbonDioxide* <b>yFirstCarbonDioxide</b> ( )
m	YCarbonDioxide* <b>yFirstCarbonDioxide</b> ( )
pas	function <b>yFirstCarbonDioxide</b> ( ): TYCarbonDioxide
vb	function <b>yFirstCarbonDioxide</b> ( ) As YCarbonDioxide
cs	YCarbonDioxide <b>FirstCarbonDioxide</b> ( )
java	YCarbonDioxide <b>FirstCarbonDioxide</b> ( )
py	def <b>FirstCarbonDioxide</b> ( )

Utiliser la fonction `YCarbonDioxide.nextCarbonDioxide( )` pour itérer sur les autres capteurs de CO2.

### Retourne :

un pointeur sur un objet `YCarbonDioxide`, correspondant à le premier capteur de CO2 accessible en ligne, ou `null` si il n'y a pas de capteurs de CO2 disponibles.

## carbondioxide→calibrateFromPoints()[carbondioxide calibrateFromPoints: ]

YCarbonDioxide

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YCarbonDioxide target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide→describe()[carbondioxide describe]****YCarbonDioxide**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de CO2 au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de CO2 (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**carbondioxide→get\_advertisedValue()****YCarbonDioxide****carbondioxide→advertisedValue()[carbondioxide  
advertisedValue]**

Retourne la valeur courante du capteur de CO2 (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de CO2 (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.



**carbondioxide→get\_currentRawValue()**  
**carbondioxide→currentRawValue()[carbondioxide**  
**currentRawValue]**

**YCarbonDioxide**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

carbondioxide→**get\_currentValue()**

YCarbonDioxide

carbondioxide→**currentValue()**[carbondioxide  
**currentValue]**

Retourne la valeur actuelle du taux de CO2.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_currentValue</b>

#### Retourne :

une valeur numérique représentant la valeur actuelle du taux de CO2

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**carbondioxide**→**get\_errorMessage()**  
**carbondioxide**→**errorMessage()**[**carbondioxide**  
**errorMessage**]

**YCarbonDioxide**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de CO2.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de CO2.

## **carbondioxide→get\_errorType() carbondioxide→errorType()**

**YCarbonDioxide**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de CO2.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

### **Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de CO2.

**carbondioxide→get\_friendlyName()**  
**carbondioxide→friendlyName()[carbondioxide**  
**friendlyName]**

**YCarbonDioxide**

Retourne un identifiant global du capteur de CO2 au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de CO2 si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de CO2 (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de CO2 en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**carbondioxide→get\_functionDescriptor()****YCarbonDioxide****carbondioxide→functionDescriptor()[carbondioxide  
functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**carbondioxide**→**get\_functionId()**  
**carbondioxide**→**functionId()**[**carbondioxide**  
**functionId**]

**YCarbonDioxide**

Retourne l'identifiant matériel du capteur de CO2, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de CO2 (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**carbondioxide**→**get\_hardwareId()**  
**carbondioxide**→**hardwareId()**[**carbondioxide**  
**hardwareId**]

**YCarbonDioxide**

Retourne l'identifiant matériel unique du capteur de CO2 au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de CO2 (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de CO2 (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**carbondioxide→get\_highestValue()**  
**carbondioxide→highestValue()[carbondioxide**  
**highestValue]**

**YCarbonDioxide**

Retourne la valeur maximale observée pour le taux de CO2 depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour le taux de CO2 depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

## carbondioxide→get\_logFrequency() carbondioxide→logFrequency()[carbondioxide logFrequency]

YCarbonDioxide

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**carbondioxide→get\_logicalName()**  
**carbondioxide→logicalName()[carbondioxide**  
**logicalName]**

**YCarbonDioxide**

Retourne le nom logique du capteur de CO2.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de CO2. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**carbondioxide**→**get\_lowestValue()****YCarbonDioxide****carbondioxide**→**lowestValue()**[**carbondioxide**  
**lowestValue**]

Retourne la valeur minimale observée pour le taux de CO2 depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour le taux de CO2 depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**carbondioxide**→**get\_module()****YCarbonDioxide****carbondioxide**→**module()**[carbondioxide module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**carbondioxide→get\_module\_async()**  
**carbondioxide→module\_async()**

**YCarbonDioxide**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**carbondioxide**→**get\_recordedData()**  
**carbondioxide**→**recordedData()**[**carbondioxide**  
**recordedData:** ]

**YCarbonDioxide**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YCarbonDioxide <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

#### Paramètres :

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

#### Retourne :

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**carbondioxide→get\_reportFrequency()**  
**carbondioxide→reportFrequency()[carbondioxide**  
**reportFrequency]**

**YCarbonDioxide**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.



**carbondioxide**→**get\_resolution()**  
**carbondioxide**→**resolution()**[**carbondioxide**  
**resolution**]

**YCarbonDioxide**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YCarbonDioxide <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**carbondioxide**→**get\_unit()**

**YCarbonDioxide**

**carbondioxide**→**unit()**[carbondioxide unit]

Retourne l'unité dans laquelle le taux de CO2 est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YCarbonDioxide <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle le taux de CO2 est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**carbondioxide→get\_userData()****YCarbonDioxide****carbondioxide→userData()[carbondioxide userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

Vérifie si le module hébergeant le capteur de CO2 est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de CO2 sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de CO2 est joignable, false sinon

**carbondioxide→isOnline\_async()****YCarbonDioxide**

Vérifie si le module hébergeant le capteur de CO2 est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de CO2 sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**carbondioxide→load()[carbondioxide load: ]****YCarbonDioxide**

Met en cache les valeurs courantes du capteur de CO2, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## carbondioxide→loadCalibrationPoints() [carbondioxide loadCalibrationPoints: ]

YCarbonDioxide

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YCarbonDioxide target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide→load\_async()****YCarbonDioxide**

Met en cache les valeurs courantes du capteur de CO2, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**carbondioxide→nextCarbonDioxide()**[carbondioxide  
**nextCarbonDioxide]**

**YCarbonDioxide**

Continue l'énumération des capteurs de CO2 commencée à l'aide de `yFirstCarbonDioxide()`.

js	function <b>nextCarbonDioxide</b> ( )
nodejs	function <b>nextCarbonDioxide</b> ( )
php	function <b>nextCarbonDioxide</b> ( )
cpp	YCarbonDioxide * <b>nextCarbonDioxide</b> ( )
m	-(YCarbonDioxide*) <b>nextCarbonDioxide</b>
pas	function <b>nextCarbonDioxide</b> ( ): TYCarbonDioxide
vb	function <b>nextCarbonDioxide</b> ( ) As YCarbonDioxide
cs	YCarbonDioxide <b>nextCarbonDioxide</b> ( )
java	YCarbonDioxide <b>nextCarbonDioxide</b> ( )
py	def <b>nextCarbonDioxide</b> ( )

**Retourne :**

un pointeur sur un objet `YCarbonDioxide` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## carbondioxide→registerTimedReportCallback() [carbondioxide registerTimedReportCallback: ]

YCarbonDioxide

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YCarbonDioxideTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YCarbonDioxideTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYCarbonDioxideTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## carbondioxide→registerValueCallback() [carbondioxide registerValueCallback: ]

## YCarbonDioxide

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YCarbonDioxideValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YCarbonDioxideValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYCarbonDioxideValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**carbondioxide→set\_highestValue()**  
**carbondioxide→setHighestValue()[carbondioxide**  
**setHighestValue: ]**

**YCarbonDioxide**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YCarbonDioxide <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide→set\_logFrequency()****YCarbonDioxide****carbondioxide→setLogFrequency()[carbondioxide  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YCarbonDioxide <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide→set\_logicalName()****YCarbonDioxide****carbondioxide→setLogicalName()[carbondioxide  
setLogicalName: ]**

Modifie le nom logique du capteur de CO2.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YCarbonDioxide <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de CO2.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide→set\_lowestValue()****YCarbonDioxide****carbondioxide→setLowestValue()[carbondioxide  
setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YCarbonDioxide <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

carbondioxide→set\_reportFrequency()

YCarbonDioxide

carbondioxide→setReportFrequency()[carbondioxide  
setReportFrequency: ]

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function set_reportFrequency( newval)
nodejs	function set_reportFrequency( newval)
php	function set_reportFrequency( \$newval)
cpp	int set_reportFrequency( const string& newval)
m	-(int) setReportFrequency : (NSString*) newval
pas	function set_reportFrequency( newval: string): integer
vb	function set_reportFrequency( ByVal newval As String) As Integer
cs	int set_reportFrequency( string newval)
java	int set_reportFrequency( String newval)
py	def set_reportFrequency( newval)
cmd	YCarbonDioxide target set_reportFrequency newval

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## carbondioxide→set\_resolution() carbondioxide→setResolution()[carbondioxide setResolution: ]

YCarbonDioxide

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YCarbonDioxide <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

### Paramètres :

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**carbondioxide**→**set\_userdata()****YCarbonDioxide****carbondioxide**→**setUserData()**[**carbondioxide**  
**setUserData:** ]

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**carbondioxide→wait\_async()****YCarbonDioxide**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.5. Interface de la fonction ColorLed

La librairie de programmation Yoctopuce permet de piloter une led couleur aussi bien en coordonnées RGB qu'en coordonnées HSL, les conversions RGB vers HSL étant faites automatiquement par le module. Ceci permet aisément d'allumer la led avec une certaine teinte et d'en faire progressivement varier la saturation ou la luminosité. Si nécessaire, vous trouverez plus d'information sur la différence entre RGB et HSL dans la section suivante.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_colorled.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YColorLed = yoctolib.YColorLed;</code>
php	<code>require_once('yocto_colorled.php');</code>
c++	<code>#include "yocto_colorled.h"</code>
m	<code>#import "yocto_colorled.h"</code>
pas	<code>uses yocto_colorled;</code>
vb	<code>yocto_colorled.vb</code>
cs	<code>yocto_colorled.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YColorLed;</code>
py	<code>from yocto_colorled import *</code>

### Fonction globales

#### yFindColorLed(func)

Permet de retrouver une led RGB d'après un identifiant donné.

#### yFirstColorLed()

Commence l'énumération des leds RGB accessibles par la librairie.

### Méthodes des objets YColorLed

#### colorled→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de la led RGB au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### colorled→get\_advertisedValue()

Retourne la valeur courante de la led RGB (pas plus de 6 caractères).

#### colorled→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la led RGB.

#### colorled→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la led RGB.

#### colorled→get\_friendlyName()

Retourne un identifiant global de la led RGB au format `NOM_MODULE . NOM_FONCTION`.

#### colorled→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### colorled→get\_functionId()

Retourne l'identifiant matériel de la led RGB, sans référence au module.

#### colorled→get\_hardwareId()

Retourne l'identifiant matériel unique de la led RGB au format `SERIAL . FUNCTIONID`.

#### colorled→get\_hslColor()

Retourne la couleur HSL courante de la led.

#### colorled→get\_logicalName()

Retourne le nom logique de la led RGB.

**colorled→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**colorled→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**colorled→get\_rgbColor()**

Retourne la couleur RGB courante de la led.

**colorled→get\_rgbColorAtPowerOn()**

Retourne la couleur configurée pour être affichée à l'allumage du module.

**colorled→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**colorled→hslMove(hsl\_target, ms\_duration)**

Effectue une transition continue dans l'espace HSL entre la couleur courante et une nouvelle couleur.

**colorled→isOnline()**

Vérifie si le module hébergeant la led RGB est joignable, sans déclencher d'erreur.

**colorled→isOnline\_async(callback, context)**

Vérifie si le module hébergeant la led RGB est joignable, sans déclencher d'erreur.

**colorled→load(msValidity)**

Met en cache les valeurs courantes de la led RGB, avec une durée de validité spécifiée.

**colorled→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de la led RGB, avec une durée de validité spécifiée.

**colorled→nextColorLed()**

Continue l'énumération des leds RGB commencée à l'aide de `yFirstColorLed()`.

**colorled→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**colorled→rgbMove(rgb\_target, ms\_duration)**

Effectue une transition continue dans l'espace RGB entre la couleur courante et une nouvelle couleur.

**colorled→set\_hslColor(newval)**

Modifie la couleur courante de la led, en utilisant une couleur HSL spécifiée.

**colorled→set\_logicalName(newval)**

Modifie le nom logique de la led RGB.

**colorled→set\_rgbColor(newval)**

Modifie la couleur courante de la led, en utilisant une couleur RGB (Rouge Vert Bleu).

**colorled→set\_rgbColorAtPowerOn(newval)**

Modifie la couleur que la led va afficher spontanément à l'allumage du module.

**colorled→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**colorled→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YColorLed.FindColorLed() yFindColorLed()yFindColorLed()

YColorLed

Permet de retrouver une led RGB d'après un identifiant donné.

js	function <b>yFindColorLed</b> ( <b>func</b> )
nodejs	function <b>FindColorLed</b> ( <b>func</b> )
php	function <b>yFindColorLed</b> ( <b>\$func</b> )
cpp	YColorLed* <b>yFindColorLed</b> ( const string& <b>func</b> )
m	YColorLed* <b>yFindColorLed</b> ( NSString* <b>func</b> )
pas	function <b>yFindColorLed</b> ( <b>func</b> : string): TYColorLed
vb	function <b>yFindColorLed</b> ( ByVal <b>func</b> As String) As YColorLed
cs	YColorLed <b>FindColorLed</b> ( string <b>func</b> )
java	YColorLed <b>FindColorLed</b> ( String <b>func</b> )
py	def <b>FindColorLed</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que la led RGB soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YColorLed.isOnline()` pour tester si la led RGB est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence la led RGB sans ambiguïté

### Retourne :

un objet de classe `YColorLed` qui permet ensuite de contrôler la led RGB.

## YColorLed.FirstColorLed() yFirstColorLed()yFirstColorLed()

## YColorLed

Commence l'énumération des leds RGB accessibles par la librairie.

js	function <b>yFirstColorLed</b> ( )
nodejs	function <b>FirstColorLed</b> ( )
php	function <b>yFirstColorLed</b> ( )
cpp	YColorLed* <b>yFirstColorLed</b> ( )
m	YColorLed* <b>yFirstColorLed</b> ( )
pas	function <b>yFirstColorLed</b> ( ): TYColorLed
vb	function <b>yFirstColorLed</b> ( ) As YColorLed
cs	YColorLed <b>FirstColorLed</b> ( )
java	YColorLed <b>FirstColorLed</b> ( )
py	def <b>FirstColorLed</b> ( )

Utiliser la fonction `YColorLed.nextColorLed( )` pour itérer sur les autres leds RGB.

### Retourne :

un pointeur sur un objet `YColorLed`, correspondant à la première led RGB accessible en ligne, ou `null` si il n'y a pas de leds RGB disponibles.

**colored→describe()[colored describe]****YColorLed**

Retourne un court texte décrivant de manière non-ambigüe l'instance de la led RGB au format  
 TYPE (NAME) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant la led RGB (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)



**colorled**→**get\_advertisedValue()**  
**colorled**→**advertisedValue()[colorled**  
**advertisedValue]**

**YColorLed**

Retourne la valeur courante de la led RGB (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YColorLed <b>target</b> <b>get_advertisedValue</b>

#### Retourne :

une chaîne de caractères représentant la valeur courante de la led RGB (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**colored**→**get\_errorMessage()****YColorLed****colored**→**errorMessage()**[**colored errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la led RGB.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de la led RGB.

**colorled**→**get\_errorType()**  
**colorled**→**errorType()**

**YColorLed**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la led RGB.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de la led RGB.

**colorled**→**get\_friendlyName()****YColorLed****colorled**→**friendlyName()**[colorled friendlyName]

Retourne un identifiant global de la led RGB au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de la led RGB si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de la led RGB (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant la led RGB en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**colorled**→**get\_functionDescriptor()**  
**colorled**→**functionDescriptor()[colorled**  
**functionDescriptor]**

**YColorLed**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**colored**→**get\_functionId()****YColorLed****colored**→**functionId()**[**colored functionId**]

Retourne l'identifiant matériel de la led RGB, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant la led RGB (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**colorled**→**get\_hardwareId()****YColorLed****colorled**→**hardwareId()**[**colorled hardwareId**]

Retourne l'identifiant matériel unique de la led RGB au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la led RGB (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant la led RGB (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**colorled**→**get\_hslColor()**

**YColorLed**

**colorled**→**hslColor()**[colorled hslColor]

Retourne la couleur HSL courante de la led.

js	function <b>get_hslColor</b> ( )
nodejs	function <b>get_hslColor</b> ( )
php	function <b>get_hslColor</b> ( )
cpp	int <b>get_hslColor</b> ( )
m	-(int) hslColor
pas	function <b>get_hslColor</b> ( ): LongInt
vb	function <b>get_hslColor</b> ( ) As Integer
cs	int <b>get_hslColor</b> ( )
java	int <b>get_hslColor</b> ( )
py	def <b>get_hslColor</b> ( )
cmd	YColorLed <b>target</b> <b>get_hslColor</b>

**Retourne :**

un entier représentant la couleur HSL courante de la led

En cas d'erreur, déclenche une exception ou retourne Y\_HSLCOLOR\_INVALID.



**colorled**→**get\_logicalName()****YColorLed****colorled**→**logicalName()**[colorled logicalName]

Retourne le nom logique de la led RGB.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YColorLed <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de la led RGB. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**colored**→**get\_module()****YColorLed****colored**→**module()**[**colored module**]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**colorled**→**get\_module\_async()****YColorLed****colorled**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**colorled**→**get\_rgbColor()****YColorLed****colorled**→**rgbColor()**[colorled rgbColor]

Retourne la couleur RGB courante de la led.

js	function <b>get_rgbColor</b> ( )
nodejs	function <b>get_rgbColor</b> ( )
php	function <b>get_rgbColor</b> ( )
cpp	int <b>get_rgbColor</b> ( )
m	-(int) rgbColor
pas	function <b>get_rgbColor</b> ( ): LongInt
vb	function <b>get_rgbColor</b> ( ) As Integer
cs	int <b>get_rgbColor</b> ( )
java	int <b>get_rgbColor</b> ( )
py	def <b>get_rgbColor</b> ( )
cmd	YColorLed <b>target</b> <b>get_rgbColor</b>

**Retourne :**

un entier représentant la couleur RGB courante de la led

En cas d'erreur, déclenche une exception ou retourne Y\_RGBCOLOR\_INVALID.

**colorled**→**get\_rgbColorAtPowerOn()**  
**colorled**→**rgbColorAtPowerOn()**[**colorled**  
**rgbColorAtPowerOn**]

**YColorLed**

Retourne la couleur configurée pour être affichage à l'allumage du module.

js	function <b>get_rgbColorAtPowerOn</b> ( )
nodejs	function <b>get_rgbColorAtPowerOn</b> ( )
php	function <b>get_rgbColorAtPowerOn</b> ( )
cpp	int <b>get_rgbColorAtPowerOn</b> ( )
m	-(int) rgbColorAtPowerOn
pas	function <b>get_rgbColorAtPowerOn</b> ( ): LongInt
vb	function <b>get_rgbColorAtPowerOn</b> ( ) As Integer
cs	int <b>get_rgbColorAtPowerOn</b> ( )
java	int <b>get_rgbColorAtPowerOn</b> ( )
py	def <b>get_rgbColorAtPowerOn</b> ( )
cmd	YColorLed <b>target</b> <b>get_rgbColorAtPowerOn</b>

**Retourne :**

un entier représentant la couleur configurée pour être affichage à l'allumage du module

En cas d'erreur, déclenche une exception ou retourne Y\_RGBCOLORATPOWERON\_INVALID.

**colorled**→**get\_userdata()**

**YColorLed**

**colorled**→**userData()**[colorled userData]

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**colorled→hslMove()[colorled hslMove: ]****YColorLed**

Effectue une transition continue dans l'espace HSL entre la couleur courante et une nouvelle couleur.

js	function <b>hslMove</b> ( <b>hsl_target</b> , <b>ms_duration</b> )
nodejs	function <b>hslMove</b> ( <b>hsl_target</b> , <b>ms_duration</b> )
php	function <b>hslMove</b> ( <b>\$hsl_target</b> , <b>\$ms_duration</b> )
cpp	int <b>hslMove</b> ( int <b>hsl_target</b> , int <b>ms_duration</b> )
m	-(int) <b>hslMove</b> : (int) <b>hsl_target</b> : (int) <b>ms_duration</b>
pas	function <b>hslMove</b> ( <b>hsl_target</b> : LongInt, <b>ms_duration</b> : LongInt): integer
vb	function <b>hslMove</b> ( ByVal <b>hsl_target</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>hslMove</b> ( int <b>hsl_target</b> , int <b>ms_duration</b> )
java	int <b>hslMove</b> ( int <b>hsl_target</b> , int <b>ms_duration</b> )
py	def <b>hslMove</b> ( <b>hsl_target</b> , <b>ms_duration</b> )
cmd	YColorLed <b>target</b> <b>hslMove</b> <b>hsl_target</b> <b>ms_duration</b>

**Paramètres :**

**hsl\_target**    couleur HSL désirée à la fin de la transition  
**ms\_duration** durée de la transition, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled**→**isOnline()**[colorled isOnline]**YColorLed**

Vérifie si le module hébergeant la led RGB est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de la led RGB sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si la led RGB est joignable, false sinon



**colorled**→**isOnline\_async()****YColorLed**

Vérifie si le module hébergeant la led RGB est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de la led RGB sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**colored→load()[colored load: ]****YColorLed**

Met en cache les valeurs courantes de la led RGB, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled**→**load\_async()****YColorLed**

Met en cache les valeurs courantes de la led RGB, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**colorled**→**nextColorLed()**[**colorled** **nextColorLed**]**YColorLed**

Continue l'énumération des leds RGB commencée à l'aide de `yFirstColorLed()`.

js	function <b>nextColorLed</b> ( )
nodejs	function <b>nextColorLed</b> ( )
php	function <b>nextColorLed</b> ( )
cpp	YColorLed * <b>nextColorLed</b> ( )
m	-(YColorLed*) <b>nextColorLed</b>
pas	function <b>nextColorLed</b> ( ): TYColorLed
vb	function <b>nextColorLed</b> ( ) As YColorLed
cs	YColorLed <b>nextColorLed</b> ( )
java	YColorLed <b>nextColorLed</b> ( )
py	def <b>nextColorLed</b> ( )

**Retourne :**

un pointeur sur un objet `YColorLed` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## colorled→registerValueCallback()[colorled registerValueCallback: ]

## YColorLed

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YColorLedValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YColorLedValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYColorLedValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**colorled→rgbMove()[colorled rgbMove: ]****YColorLed**

Effectue une transition continue dans l'espace RGB entre la couleur courante et une nouvelle couleur.

js	function <b>rgbMove</b> ( <b>rgb_target</b> , <b>ms_duration</b> )
nodejs	function <b>rgbMove</b> ( <b>rgb_target</b> , <b>ms_duration</b> )
php	function <b>rgbMove</b> ( <b>\$rgb_target</b> , <b>\$ms_duration</b> )
cpp	int <b>rgbMove</b> ( int <b>rgb_target</b> , int <b>ms_duration</b> )
m	-(int) <b>rgbMove</b> : (int) <b>rgb_target</b> : (int) <b>ms_duration</b>
pas	function <b>rgbMove</b> ( <b>rgb_target</b> : LongInt, <b>ms_duration</b> : LongInt): integer
vb	function <b>rgbMove</b> ( ByVal <b>rgb_target</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>rgbMove</b> ( int <b>rgb_target</b> , int <b>ms_duration</b> )
java	int <b>rgbMove</b> ( int <b>rgb_target</b> , int <b>ms_duration</b> )
py	def <b>rgbMove</b> ( <b>rgb_target</b> , <b>ms_duration</b> )
cmd	YColorLed <b>target</b> <b>rgbMove</b> <b>rgb_target</b> <b>ms_duration</b>

**Paramètres :**

**rgb\_target** couleur RGB désirée à la fin de la transition  
**ms\_duration** durée de la transition, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled**→**set\_hslColor()****YColorLed****colorled**→**setHslColor()**[**colorled setHslColor:** ]

Modifie la couleur courante de la led, en utilisant une couleur HSL spécifiée.

js	function <b>set_hslColor</b> ( <b>newval</b> )
nodejs	function <b>set_hslColor</b> ( <b>newval</b> )
php	function <b>set_hslColor</b> ( <b>\$newval</b> )
cpp	int <b>set_hslColor</b> ( int <b>newval</b> )
m	-(int) setHslColor : (int) <b>newval</b>
pas	function <b>set_hslColor</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_hslColor</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_hslColor</b> ( int <b>newval</b> )
java	int <b>set_hslColor</b> ( int <b>newval</b> )
py	def <b>set_hslColor</b> ( <b>newval</b> )
cmd	YColorLed <b>target set_hslColor newval</b>

L'encodage est réalisé de la manière suivante: 0xHHSSL.

**Paramètres :**

**newval** un entier représentant la couleur courante de la led, en utilisant une couleur HSL spécifiée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled**→**set\_logicalName()****YColorLed****colorled**→**setLogicalName()**[**colorled**  
**setLogicalName:** ]

Modifie le nom logique de la led RGB.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YColorLed <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de la led RGB.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**colorled**→**set\_rgbColor()****YColorLed****colorled**→**setRgbColor()**[**colorled setRgbColor:** ]

Modifie la couleur courante de la led, en utilisant une couleur RGB (Rouge Vert Bleu).

js	function <b>set_rgbColor</b> ( <b>newval</b> )
nodejs	function <b>set_rgbColor</b> ( <b>newval</b> )
php	function <b>set_rgbColor</b> ( <b>\$newval</b> )
cpp	int <b>set_rgbColor</b> ( int <b>newval</b> )
m	-(int) setRgbColor : (int) <b>newval</b>
pas	function <b>set_rgbColor</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_rgbColor</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_rgbColor</b> ( int <b>newval</b> )
java	int <b>set_rgbColor</b> ( int <b>newval</b> )
py	def <b>set_rgbColor</b> ( <b>newval</b> )
cmd	YColorLed <b>target set_rgbColor newval</b>

L'encodage est réalisé de la manière suivante: 0xRRGGBB.

**Paramètres :**

**newval** un entier représentant la couleur courante de la led, en utilisant une couleur RGB (Rouge Vert Bleu)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled→set\_rgbColorAtPowerOn()****YColorLed****colorled→setRgbColorAtPowerOn()[colorled  
setRgbColorAtPowerOn: ]**

Modifie la couleur que la led va afficher spontanément à l'allumage du module.

js	function <b>set_rgbColorAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_rgbColorAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_rgbColorAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_rgbColorAtPowerOn</b> ( int <b>newval</b> )
m	-(int) setRgbColorAtPowerOn : (int) <b>newval</b>
pas	function <b>set_rgbColorAtPowerOn</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_rgbColorAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_rgbColorAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_rgbColorAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_rgbColorAtPowerOn</b> ( <b>newval</b> )
cmd	YColorLed <b>target</b> <b>set_rgbColorAtPowerOn</b> <b>newval</b>

Cette couleur sera affichée dès que le module sera sous tension. Ne pas oublier d'appeler la fonction `saveToFlash( )` du module correspondant pour que ce paramètre soit mémorisé.

**Paramètres :**

**newval** un entier représentant la couleur que la led va afficher spontanément à l'allumage du module

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**colorled**→**set\_userdata()****YColorLed****colorled**→**setUserData()**[**colorled setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## colorled→wait\_async()

YColorLed

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :

## 3.6. Interface de la fonction Compass

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_compass.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YCompass = yoctolib.YCompass;
php	require_once('yocto_compass.php');
c++	#include "yocto_compass.h"
m	#import "yocto_compass.h"
pas	uses yocto_compass;
vb	yocto_compass.vb
cs	yocto_compass.cs
java	import com.yoctopuce.YoctoAPI.YCompass;
py	from yocto_compass import *

### Fonction globales

#### yFindCompass(func)

Permet de retrouver un compas d'après un identifiant donné.

#### yFirstCompass()

Commence l'énumération des compas accessibles par la librairie.

### Méthodes des objets YCompass

#### compass→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### compass→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du compas au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### compass→get\_advertisedValue()

Retourne la valeur courante du compas (pas plus de 6 caractères).

#### compass→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### compass→get\_currentValue()

Retourne la valeur actuelle du cap relatif.

#### compass→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du compas.

#### compass→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du compas.

#### compass→get\_friendlyName()

Retourne un identifiant global du compas au format `NOM_MODULE . NOM_FONCTION`.

#### compass→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### compass→get\_functionId()

Retourne l'identifiant matériel du compas, sans référence au module.

#### compass→get\_hardwareId()

Retourne l'identifiant matériel unique du compas au format `SERIAL . FUNCTIONID`.

**compass→get\_highestValue()**

Retourne la valeur maximale observée pour le cap relatif depuis le démarrage du module.

**compass→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**compass→get\_logicalName()**

Retourne le nom logique du compas.

**compass→get\_lowestValue()**

Retourne la valeur minimale observée pour le cap relatif depuis le démarrage du module.

**compass→get\_magneticHeading()**

Retourne la direction du nord magnétique, indépendamment du cap configuré.

**compass→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**compass→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**compass→get\_recordedData(startTime, endTime)**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

**compass→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**compass→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**compass→get\_unit()**

Retourne l'unité dans laquelle le cap relatif est exprimée.

**compass→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

**compass→isOnline()**

Vérifie si le module hébergeant le compas est joignable, sans déclencher d'erreur.

**compass→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le compas est joignable, sans déclencher d'erreur.

**compass→load(msValidity)**

Met en cache les valeurs courantes du compas, avec une durée de validité spécifiée.

**compass→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.

**compass→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du compas, avec une durée de validité spécifiée.

**compass→nextCompass()**

Continue l'énumération des compas commencée à l'aide de yFirstCompass().

**compass→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**compass→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**compass→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

**compass→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**compass→set\_logicalName(newval)**

Modifie le nom logique du compas.

**compass→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

**compass→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**compass→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**compass→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userData.

**compass→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YCompass.FindCompass() yFindCompass()yFindCompass()

YCompass

Permet de retrouver un compas d'après un identifiant donné.

js	function <b>yFindCompass</b> ( <b>func</b> )
nodejs	function <b>FindCompass</b> ( <b>func</b> )
php	function <b>yFindCompass</b> ( <b>\$func</b> )
cpp	YCompass* <b>yFindCompass</b> ( const string& <b>func</b> )
m	YCompass* <b>yFindCompass</b> ( NSString* <b>func</b> )
pas	function <b>yFindCompass</b> ( <b>func</b> : string): TYCompass
vb	function <b>yFindCompass</b> ( ByVal <b>func</b> As String) As YCompass
cs	YCompass <b>FindCompass</b> ( string <b>func</b> )
java	YCompass <b>FindCompass</b> ( String <b>func</b> )
py	def <b>FindCompass</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le compas soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YCompass.isOnline()` pour tester si le compas est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le compas sans ambiguïté

### Retourne :

un objet de classe `YCompass` qui permet ensuite de contrôler le compas.



## YCompass.FirstCompass() yFirstCompass()yFirstCompass()

## YCompass

Commence l'énumération des compas accessibles par la librairie.

js	function <b>yFirstCompass</b> ( )
nodejs	function <b>FirstCompass</b> ( )
php	function <b>yFirstCompass</b> ( )
cpp	YCompass* <b>yFirstCompass</b> ( )
m	YCompass* <b>yFirstCompass</b> ( )
pas	function <b>yFirstCompass</b> ( ): TYCompass
vb	function <b>yFirstCompass</b> ( ) As YCompass
cs	YCompass <b>FirstCompass</b> ( )
java	YCompass <b>FirstCompass</b> ( )
py	def <b>FirstCompass</b> ( )

Utiliser la fonction `YCompass.nextCompass( )` pour itérer sur les autres compas.

### Retourne :

un pointeur sur un objet `YCompass`, correspondant à le premier compas accessible en ligne, ou `null` si il n'y a pas de compas disponibles.

## compass→calibrateFromPoints()[compass calibrateFromPoints: ]

YCompass

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YCompass target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass→describe()[compass describe]****YCompass**

Retourne un court texte décrivant de manière non-ambigüe l'instance du compas au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le compas (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**compass**→**get\_advertisedValue()**  
**compass**→**advertisedValue()**[**compass**  
**advertisedValue**]

**YCompass**

Retourne la valeur courante du compas (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YCompass <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du compas (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**compass**→**get\_currentRawValue()**  
**compass**→**currentRawValue()**[**compass**  
**currentRawValue**]

**YCompass**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YCompass <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne `Y_CURRENTRAWVALUE_INVALID`.

**compass**→**get\_currentValue()****YCompass****compass**→**currentValue()**[compass currentValue]

Retourne la valeur actuelle du cap relatif.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YCompass <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle du cap relatif

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**compass**→**get\_errorMessage()****YCompass****compass**→**errorMessage()**[**compass errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du compas.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du compas.

**compass**→**get\_errorType()****YCompass****compass**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du compas.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du compas.



**compass**→**get\_friendlyName()****YCompass****compass**→**friendlyName()**[compass friendlyName]

Retourne un identifiant global du compas au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du compas si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du compas (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le compas en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **compass→get\_functionDescriptor()** **compass→functionDescriptor()[compass** **functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**compass**→**get\_functionId()****YCompass****compass**→**functionId()**[**compass functionId**]

Retourne l'identifiant matériel du compas, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le compas (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**compass**→**get\_hardwareId()****YCompass****compass**→**hardwareId()**[**compass hardwareId**]

Retourne l'identifiant matériel unique du compas au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du compas (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le compas (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**compass→get\_highestValue()****YCompass****compass→highestValue()[compass highestValue]**

Retourne la valeur maximale observée pour le cap relatif depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YCompass <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour le cap relatif depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**compass→get\_logFrequency()****YCompass****compass→logFrequency()[compass logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YCompass <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

---

**compass**→**get\_logicalName()****YCompass****compass**→**logicalName()**[compass logicalName]

---

Retourne le nom logique du compas.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YCompass <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du compas. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**compass**→**get\_lowestValue()****YCompass****compass**→**lowestValue()**[compass lowestValue]

Retourne la valeur minimale observée pour le cap relatif depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YCompass <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour le cap relatif depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.



**compass**→**get\_magneticHeading()**  
**compass**→**magneticHeading()**[**compass**  
**magneticHeading**]

**YCompass**

Retourne la direction du nord magnétique, indépendamment du cap configuré.

js	function <b>get_magneticHeading</b> ( )
nodejs	function <b>get_magneticHeading</b> ( )
php	function <b>get_magneticHeading</b> ( )
cpp	double <b>get_magneticHeading</b> ( )
m	-(double) magneticHeading
pas	function <b>get_magneticHeading</b> ( ): double
vb	function <b>get_magneticHeading</b> ( ) As Double
cs	double <b>get_magneticHeading</b> ( )
java	double <b>get_magneticHeading</b> ( )
py	def <b>get_magneticHeading</b> ( )
cmd	YCompass <b>target</b> <b>get_magneticHeading</b>

**Retourne :**

une valeur numérique représentant la direction du nord magnétique, indépendamment du cap configuré

En cas d'erreur, déclenche une exception ou retourne Y\_MAGNETICHEADING\_INVALID.

**compass**→**get\_module()****compass**→**module()**[compass module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

---

**compass**→**get\_module\_async()****YCompass****compass**→**module\_async()**

---

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**compass→get\_recordedData()****YCompass****compass→recordedData()[compass recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
c++	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YCompass <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**compass→get\_reportFrequency()**  
**compass→reportFrequency()[compass**  
**reportFrequency]**

**YCompass**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YCompass <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**compass**→**get\_resolution()****YCompass****compass**→**resolution()**[**compass resolution**]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YCompass <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**compass**→**get\_unit()****YCompass****compass**→**unit()**[compass unit]

Retourne l'unité dans laquelle le cap relatif est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YCompass <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle le cap relatif est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**compass**→**get\_userData()****compass**→**userData()**[compass userData]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.



**compass**→**isOnline()**[**compass isOnline**]**YCompass**

Vérifie si le module hébergeant le compas est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du compas sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le compas est joignable, `false` sinon

Vérifie si le module hébergeant le compas est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du compas sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**compass→load()[compass load: ]****YCompass**

Met en cache les valeurs courantes du compas, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## compass→loadCalibrationPoints()[compass loadCalibrationPoints: ]

YCompass

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)
java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)
py def loadCalibrationPoints( rawValues, refValues)
cmd YCompass target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass→load\_async()****YCompass**

Met en cache les valeurs courantes du compas, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**compass**→**nextCompass()**[**compass nextCompass**]**YCompass**

Continue l'énumération des compas commencée à l'aide de `yFirstCompass()`.

js	function <b>nextCompass</b> ( )
nodejs	function <b>nextCompass</b> ( )
php	function <b>nextCompass</b> ( )
cpp	YCompass * <b>nextCompass</b> ( )
m	-(YCompass*) <b>nextCompass</b>
pas	function <b>nextCompass</b> ( ): TYCompass
vb	function <b>nextCompass</b> ( ) As YCompass
cs	YCompass <b>nextCompass</b> ( )
java	YCompass <b>nextCompass</b> ( )
py	def <b>nextCompass</b> ( )

**Retourne :**

un pointeur sur un objet `YCompass` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## compass→registerTimedReportCallback()[compass registerTimedReportCallback: ]

## YCompass

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YCompassTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YCompassTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYCompassTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## compass→registerValueCallback()[compass registerValueCallback: ]

YCompass

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YCompassValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YCompassValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYCompassValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.



**compass→set\_highestValue()**  
**compass→setHighestValue()[compass**  
**setHighestValue: ]**

**YCompass**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YCompass <b>target</b> <b>set_highestValue</b> <b>newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## compass→set\_logFrequency() compass→setLogFrequency()[compass setLogFrequency: ]

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YCompass <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass**→**set\_logicalName()****YCompass****compass**→**setLogicalName()**[**compass**  
**setLogicalName:** ]

Modifie le nom logique du compas.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YCompass <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du compas.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass**→**set\_lowestValue()**  
**compass**→**setLowestValue()**[**compass**  
**setLowestValue:** ]

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YCompass <b>target set_lowestValue newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass→set\_reportFrequency()**  
**compass→setReportFrequency()[compass**  
**setReportFrequency: ]**

**YCompass**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YCompass <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass**→**set\_resolution()****compass**→**setResolution()**[**compass** **setResolution:** ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YCompass <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**compass**→**set\_userdata()****YCompass****compass**→**setUserData()**[**compass setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**compass**→**wait\_async()****YCompass**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :



## 3.7. Interface de la fonction Current

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_current.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YCurrent = yoctolib.YCurrent;
php	require_once('yocto_current.php');
c++	#include "yocto_current.h"
m	#import "yocto_current.h"
pas	uses yocto_current;
vb	yocto_current.vb
cs	yocto_current.cs
java	import com.yoctopuce.YoctoAPI.YCurrent;
py	from yocto_current import *

### Fonction globales

#### yFindCurrent(func)

Permet de retrouver un capteur de courant d'après un identifiant donné.

#### yFirstCurrent()

Commence l'énumération des capteurs de courant accessibles par la librairie.

### Méthodes des objets YCurrent

#### current→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### current→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de courant au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### current→get\_advertisedValue()

Retourne la valeur courante du capteur de courant (pas plus de 6 caractères).

#### current→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### current→get\_currentValue()

Retourne la valeur instantanée du courant.

#### current→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de courant.

#### current→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de courant.

#### current→get\_friendlyName()

Retourne un identifiant global du capteur de courant au format `NOM_MODULE . NOM_FONCTION`.

#### current→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### current→get\_functionId()

Retourne l'identifiant matériel du capteur de courant, sans référence au module.

#### current→get\_hardwareId()

	Retourne l'identifiant matériel unique du capteur de courant au format <code>SERIAL.FUNCTIONID</code> .
<b>current</b> → <b>get_highestValue()</b>	Retourne la valeur maximale observée pour le courant.
<b>current</b> → <b>get_logFrequency()</b>	Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.
<b>current</b> → <b>get_logicalName()</b>	Retourne le nom logique du capteur de courant.
<b>current</b> → <b>get_lowestValue()</b>	Retourne la valeur minimale observée pour le courant.
<b>current</b> → <b>get_module()</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>current</b> → <b>get_module_async(callback, context)</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>current</b> → <b>get_recordedData(startTime, endTime)</b>	Retourne un objet <code>DataSet</code> représentant des mesures de ce capteur précédemment enregistrées à l'aide du <code>DataLogger</code> , pour l'intervalle de temps spécifié.
<b>current</b> → <b>get_reportFrequency()</b>	Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.
<b>current</b> → <b>get_resolution()</b>	Retourne la résolution des valeurs mesurées.
<b>current</b> → <b>get_unit()</b>	Retourne l'unité dans laquelle le courant est exprimée.
<b>current</b> → <b>get_userData()</b>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<b>current</b> → <b>isOnline()</b>	Vérifie si le module hébergeant le capteur de courant est joignable, sans déclencher d'erreur.
<b>current</b> → <b>isOnline_async(callback, context)</b>	Vérifie si le module hébergeant le capteur de courant est joignable, sans déclencher d'erreur.
<b>current</b> → <b>load(msValidity)</b>	Met en cache les valeurs courantes du capteur de courant, avec une durée de validité spécifiée.
<b>current</b> → <b>loadCalibrationPoints(rawValues, refValues)</b>	Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode <code>calibrateFromPoints</code> .
<b>current</b> → <b>load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du capteur de courant, avec une durée de validité spécifiée.
<b>current</b> → <b>nextCurrent()</b>	Continue l'énumération des capteurs de courant commencée à l'aide de <code>yFirstCurrent()</code> .
<b>current</b> → <b>registerTimedReportCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque notification périodique.
<b>current</b> → <b>registerValueCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b>current</b> → <b>set_highestValue(newval)</b>	Modifie la mémoire de valeur maximale observée pour le courant.
<b>current</b> → <b>set_logFrequency(newval)</b>	

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**current**→**set\_logicalName(newval)**

Modifie le nom logique du capteur de courant.

**current**→**set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée pour le courant.

**current**→**set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**current**→**set\_resolution(newval)**

Modifie la résolution des valeurs mesurées.

**current**→**set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**current**→**wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YCurrent.FindCurrent() yFindCurrent()yFindCurrent()

YCurrent

Permet de retrouver un capteur de courant d'après un identifiant donné.

js	function <b>yFindCurrent</b> ( <b>func</b> )
nodejs	function <b>FindCurrent</b> ( <b>func</b> )
php	function <b>yFindCurrent</b> ( <b>\$func</b> )
cpp	YCurrent* <b>yFindCurrent</b> ( const string& <b>func</b> )
m	YCurrent* <b>yFindCurrent</b> ( NSString* <b>func</b> )
pas	function <b>yFindCurrent</b> ( <b>func</b> : string): TYCurrent
vb	function <b>yFindCurrent</b> ( ByVal <b>func</b> As String) As YCurrent
cs	YCurrent <b>FindCurrent</b> ( string <b>func</b> )
java	YCurrent <b>FindCurrent</b> ( String <b>func</b> )
py	def <b>FindCurrent</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de courant soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YCurrent.isOnline()` pour tester si le capteur de courant est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de courant sans ambiguïté

### Retourne :

un objet de classe `YCurrent` qui permet ensuite de contrôler le capteur de courant.

## YCurrent.FirstCurrent() yFirstCurrent()yFirstCurrent()

## YCurrent

Commence l'énumération des capteurs de courant accessibles par la librairie.

js	function <b>yFirstCurrent</b> ( )
nodejs	function <b>FirstCurrent</b> ( )
php	function <b>yFirstCurrent</b> ( )
cpp	YCurrent* <b>yFirstCurrent</b> ( )
m	YCurrent* <b>yFirstCurrent</b> ( )
pas	function <b>yFirstCurrent</b> ( ): TYCurrent
vb	function <b>yFirstCurrent</b> ( ) As YCurrent
cs	YCurrent <b>FirstCurrent</b> ( )
java	YCurrent <b>FirstCurrent</b> ( )
py	def <b>FirstCurrent</b> ( )

Utiliser la fonction `YCurrent.nextCurrent( )` pour itérer sur les autres capteurs de courant.

### Retourne :

un pointeur sur un objet `YCurrent`, correspondant à le premier capteur de courant accessible en ligne, ou `null` si il n'y a pas de capteurs de courant disponibles.

## current→calibrateFromPoints()[current calibrateFromPoints: ]

YCurrent

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                   refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YCurrent target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current→describe()[current describe]****YCurrent**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de courant au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de courant (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**current**→**get\_advertisedValue()****YCurrent****current**→**advertisedValue()[current advertisedValue]**

Retourne la valeur courante du capteur de courant (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YCurrent <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de courant (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.



**current**→**get\_currentRawValue()**  
**current**→**currentRawValue()[current**  
**currentRawValue]**

**YCurrent**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YCurrent <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**current**→**get\_currentValue()****YCurrent****current**→**currentValue()**[**current currentValue**]

Retourne la valeur instantanée du courant.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YCurrent <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur instantanée du courant

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**current**→**get\_errorMessage()****YCurrent****current**→**errorMessage()**[**current errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de courant.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de courant.

**current**→**get\_errorType()**

**YCurrent**

**current**→**errorType()**

---

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de courant.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de courant.

**current→get\_friendlyName()****YCurrent****current→friendlyName()[current friendlyName]**

Retourne un identifiant global du capteur de courant au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de courant si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de courant (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de courant en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**current**→**get\_functionDescriptor()**  
**current**→**functionDescriptor()[current**  
**functionDescriptor]**

**YCurrent**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**current**→**get\_functionId()****YCurrent****current**→**functionId()**[**current functionId**]

Retourne l'identifiant matériel du capteur de courant, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de courant (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**current→get\_hardwareId()****YCurrent****current→hardwareId()[current hardwareId]**

Retourne l'identifiant matériel unique du capteur de courant au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de courant (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de courant (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**current**→**get\_highestValue()****YCurrent****current**→**highestValue()**[**current highestValue**]

Retourne la valeur maximale observée pour le courant.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YCurrent <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour le courant

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**current**→**get\_logFrequency()****YCurrent****current**→**logFrequency()**[**current logFrequency**]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YCurrent <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**current**→**get\_logicalName()****YCurrent****current**→**logicalName()**[current logicalName]

Retourne le nom logique du capteur de courant.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YCurrent <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de courant. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**current**→**get\_lowestValue()****YCurrent****current**→**lowestValue()**[**current** **lowestValue**]

Retourne la valeur minimale observée pour le courant.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YCurrent <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour le courant

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**current→get\_module()****YCurrent****current→module()[current module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**current**→**get\_module\_async()****YCurrent****current**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**current**→**get\_recordedData()****YCurrent****current**→**recordedData()**[**current recordedData:** ]

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YCurrent <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**current**→**get\_reportFrequency()****YCurrent****current**→**reportFrequency()**[current reportFrequency]

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YCurrent <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.



**current**→**get\_resolution()**  
**current**→**resolution()**[current resolution]

YCurrent

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YCurrent <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**current**→**get\_unit()****YCurrent****current**→**unit()**[current unit]

Retourne l'unité dans laquelle le courant est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YCurrent <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle le courant est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**current**→**get\_userData()**  
**current**→**userData()[current userData]**

**YCurrent**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
c++	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**current→isOnline()[current isOnline]****YCurrent**

Vérifie si le module hébergeant le capteur de courant est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de courant sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de courant est joignable, false sinon

**current→isOnline\_async()****YCurrent**

Vérifie si le module hébergeant le capteur de courant est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de courant sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**current→load()[current load: ]****YCurrent**

Met en cache les valeurs courantes du capteur de courant, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## current→loadCalibrationPoints()[current loadCalibrationPoints: ]

YCurrent

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
                             : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YCurrent target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current→load\_async()****YCurrent**

Met en cache les valeurs courantes du capteur de courant, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**current→nextCurrent()[current nextCurrent]****YCurrent**

Continue l'énumération des capteurs de courant commencée à l'aide de `yFirstCurrent()`.

<code>js</code>	<code>function nextCurrent( )</code>
<code>nodejs</code>	<code>function nextCurrent( )</code>
<code>php</code>	<code>function nextCurrent( )</code>
<code>cpp</code>	<code>YCurrent * nextCurrent( )</code>
<code>m</code>	<code>-(YCurrent*) nextCurrent</code>
<code>pas</code>	<code>function nextCurrent( ): TYCurrent</code>
<code>vb</code>	<code>function nextCurrent( ) As YCurrent</code>
<code>cs</code>	<code>YCurrent nextCurrent( )</code>
<code>java</code>	<code>YCurrent nextCurrent( )</code>
<code>py</code>	<code>def nextCurrent( )</code>

**Retourne :**

un pointeur sur un objet `YCurrent` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## current→registerTimedReportCallback()[current registerTimedReportCallback: ]

YCurrent

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YCurrentTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YCurrentTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYCurrentTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## current→registerValueCallback()[current registerValueCallback: ]

YCurrent

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YCurrentValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YCurrentValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYCurrentValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**current**→**set\_highestValue()****YCurrent****current**→**setHighestValue()**[**current setHighestValue:**  
**]**

Modifie la mémoire de valeur maximale observée pour le courant.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YCurrent <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour le courant

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current**→**set\_logFrequency()**  
**current**→**setLogFrequency()**[**current**  
**setLogFrequency:** ]

YCurrent

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
c++	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YCurrent <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current**→**set\_logicalName()****YCurrent****current**→**setLogicalName()**[**current setLogicalName:**  
**]**

Modifie le nom logique du capteur de courant.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YCurrent <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de courant.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current**→**set\_lowestValue()****YCurrent****current**→**setLowestValue()**[**current setLowestValue:** ]

Modifie la mémoire de valeur minimale observée pour le courant.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YCurrent <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour le courant

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current→set\_reportFrequency()**  
**current→setReportFrequency()[current**  
**setReportFrequency: ]**

**YCurrent**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YCurrent <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**current**→**set\_resolution()****YCurrent****current**→**setResolution()**[**current setResolution:** ]

Modifie la résolution des valeurs mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YCurrent <b>target set_resolution newval</b>

La résolution correspond à la précision de la représentation numérique des mesures. Changer la résolution ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**current**→**set\_userData()****YCurrent****current**→**setUserData()**[**current setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function <b>set_userData</b> ( <b>data</b> )
nodejs	function <b>set_userData</b> ( <b>data</b> )
php	function <b>set_userData</b> ( <b>\$data</b> )
cpp	void <b>set_userData</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userData</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userData</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userData</b> ( object <b>data</b> )
java	void <b>set_userData</b> ( Object <b>data</b> )
py	def <b>set_userData</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**current→wait\_async()****YCurrent**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.8. Interface de la fonction DataLogger

Les capteurs de Yoctopuce sont équipés d'une mémoire non-volatile permettant de mémoriser les données mesurées d'une manière autonome, sans nécessiter le suivi permanent d'un ordinateur. La fonction DataLogger contrôle les paramètres globaux de cet enregistreur de données.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_datalogger.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YDataLogger = yoctolib.YDataLogger;
php	require_once('yocto_datalogger.php');
c++	#include "yocto_datalogger.h"
m	#import "yocto_datalogger.h"
pas	uses yocto_datalogger;
vb	yocto_datalogger.vb
cs	yocto_datalogger.cs
java	import com.yoctopuce.YoctoAPI.YDataLogger;
py	from yocto_datalogger import *

### Fonction globales

#### yFindDataLogger(func)

Permet de retrouver un enregistreur de données d'après un identifiant donné.

#### yFirstDataLogger()

Commence l'énumération des enregistreurs de données accessibles par la librairie.

### Méthodes des objets YDataLogger

#### datalogger→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'enregistreur de données au format  
TYPE ( NAME ) = SERIAL . FUNCTIONID.

#### datalogger→forgetAllDataStreams()

Efface tout l'historique des mesures de l'enregistreur de données.

#### datalogger→get\_advertisedValue()

Retourne la valeur courante de l'enregistreur de données (pas plus de 6 caractères).

#### datalogger→get\_autoStart()

Retourne le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

#### datalogger→get\_currentRunIndex()

Retourne le numéro du Run actuel, correspondant au nombre de fois que le module a été mis sous tension avec la fonction d'enregistreur de données active.

#### datalogger→get\_dataSets()

Retourne une liste d'objets YDataSet permettant de récupérer toutes les mesures stockées par l'enregistreur de données.

#### datalogger→get\_dataStreams(v)

Construit une liste de toutes les séquences de mesures mémorisées par l'enregistreur (ancienne méthode).

#### datalogger→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'enregistreur de données.

#### datalogger→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'enregistreur de données.

#### datalogger→get\_friendlyName()

Retourne un identifiant global de l'enregistreur de données au format `NOM_MODULE . NOM_FONCTION`.

**`datalogger→get_functionDescriptor()`**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

**`datalogger→get_functionId()`**

Retourne l'identifiant matériel de l'enregistreur de données, sans référence au module.

**`datalogger→get_hardwareId()`**

Retourne l'identifiant matériel unique de l'enregistreur de données au format `SERIAL . FUNCTIONID`.

**`datalogger→get_logicalName()`**

Retourne le nom logique de l'enregistreur de données.

**`datalogger→get_module()`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`datalogger→get_module_async(callback, context)`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`datalogger→get_recording()`**

Retourne l'état d'activation de l'enregistreur de données.

**`datalogger→get_timeUTC()`**

Retourne le timestamp Unix de l'heure UTC actuelle, lorsqu'elle est connue.

**`datalogger→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`datalogger→isOnline()`**

Vérifie si le module hébergeant l'enregistreur de données est joignable, sans déclencher d'erreur.

**`datalogger→isOnline_async(callback, context)`**

Vérifie si le module hébergeant l'enregistreur de données est joignable, sans déclencher d'erreur.

**`datalogger→load(msValidity)`**

Met en cache les valeurs courantes de l'enregistreur de données, avec une durée de validité spécifiée.

**`datalogger→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes de l'enregistreur de données, avec une durée de validité spécifiée.

**`datalogger→nextDataLogger()`**

Continue l'énumération des enregistreurs de données commencée à l'aide de `yFirstDataLogger()`.

**`datalogger→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`datalogger→set_autoStart(newval)`**

Modifie le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

**`datalogger→set_logicalName(newval)`**

Modifie le nom logique de l'enregistreur de données.

**`datalogger→set_recording(newval)`**

Modifie l'état d'activation de l'enregistreur de données.

**`datalogger→set_timeUTC(newval)`**

Modifie la référence de temps UTC, afin de l'attacher aux données enregistrées.

**`datalogger→set_userData(data)`**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**`datalogger→wait_async(callback, context)`**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YDataLogger.FindDataLogger() yFindDataLogger()yFindDataLogger()

YDataLogger

Permet de retrouver un enregistreur de données d'après un identifiant donné.

js	function <b>yFindDataLogger</b> ( <b>func</b> )
nodejs	function <b>FindDataLogger</b> ( <b>func</b> )
php	function <b>yFindDataLogger</b> ( <b>\$func</b> )
cpp	YDataLogger* <b>yFindDataLogger</b> ( string <b>func</b> )
m	+(YDataLogger*) <b>yFindDataLogger</b> : (NSString*) <b>func</b>
pas	function <b>yFindDataLogger</b> ( <b>func</b> : string): TYDataLogger
vb	function <b>yFindDataLogger</b> ( ByVal <b>func</b> As String) As YDataLogger
cs	YDataLogger <b>FindDataLogger</b> ( string <b>func</b> )
java	YDataLogger <b>FindDataLogger</b> ( String <b>func</b> )
py	def <b>FindDataLogger</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'enregistreur de données soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YDataLogger.isOnline()` pour tester si l'enregistreur de données est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'enregistreur de données sans ambiguïté

### Retourne :

un objet de classe `YDataLogger` qui permet ensuite de contrôler l'enregistreur de données.

## YDataLogger.FirstDataLogger() yFirstDataLogger()yFirstDataLogger()

## YDataLogger

Commence l'énumération des enregistreurs de données accessibles par la librairie.

js	function <b>yFirstDataLogger</b> ( )
nodejs	function <b>FirstDataLogger</b> ( )
php	function <b>yFirstDataLogger</b> ( )
cpp	YDataLogger* <b>yFirstDataLogger</b> ( )
m	YDataLogger* <b>yFirstDataLogger</b> ( )
pas	function <b>yFirstDataLogger</b> ( ): TYDataLogger
vb	function <b>yFirstDataLogger</b> ( ) As YDataLogger
cs	YDataLogger <b>FirstDataLogger</b> ( )
java	YDataLogger <b>FirstDataLogger</b> ( )
py	def <b>FirstDataLogger</b> ( )

Utiliser la fonction `YDataLogger.nextDataLogger( )` pour itérer sur les autres enregistreurs de données.

### Retourne :

un pointeur sur un objet `YDataLogger`, correspondant à le premier enregistreur de données accessible en ligne, ou `null` si il n'y a pas de enregistreurs de données disponibles.

**datalogger**→**describe()**[**datalogger describe**]**YDataLogger**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'enregistreur de données au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'enregistreur de données (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)



## **datalogger**→**forgetAllDataStreams()**[datalogger **forgetAllDataStreams**]

## **YDataLogger**

Efface tout l'historique des mesures de l'enregistreur de données.

js	function <b>forgetAllDataStreams</b> ( )
nodejs	function <b>forgetAllDataStreams</b> ( )
php	function <b>forgetAllDataStreams</b> ( )
cpp	int <b>forgetAllDataStreams</b> ( )
m	-(int) <b>forgetAllDataStreams</b>
pas	function <b>forgetAllDataStreams</b> ( ): LongInt
vb	function <b>forgetAllDataStreams</b> ( ) As Integer
cs	int <b>forgetAllDataStreams</b> ( )
java	int <b>forgetAllDataStreams</b> ( )
py	def <b>forgetAllDataStreams</b> ( )
cmd	YDataLogger <b>target</b> <b>forgetAllDataStreams</b>

Cette méthode remet aussi à zéro le compteur de Runs.

### **Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## **datalogger→get\_advertisedValue() datalogger→advertisedValue()[datalogger advertisedValue]**

YDataLogger

Retourne la valeur courante de l'enregistreur de données (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YDataLogger <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'enregistreur de données (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**datalogger→get\_autoStart()****YDataLogger****datalogger→autoStart()[datalogger autoStart]**

Retourne le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

js	function <b>get_autoStart</b> ( )
nodejs	function <b>get_autoStart</b> ( )
php	function <b>get_autoStart</b> ( )
cpp	Y_AUTOSTART_enum <b>get_autoStart</b> ( )
m	-(Y_AUTOSTART_enum) autoStart
pas	function <b>get_autoStart</b> ( ): Integer
vb	function <b>get_autoStart</b> ( ) As Integer
cs	int <b>get_autoStart</b> ( )
java	int <b>get_autoStart</b> ( )
py	def <b>get_autoStart</b> ( )
cmd	YDataLogger <b>target get_autoStart</b>

**Retourne :**

soit Y\_AUTOSTART\_OFF, soit Y\_AUTOSTART\_ON, selon le mode d'activation automatique de l'enregistreur de données à la mise sous tension

En cas d'erreur, déclenche une exception ou retourne Y\_AUTOSTART\_INVALID.

## **datalogger→get\_currentRunIndex() datalogger→currentRunIndex()[datalogger currentRunIndex]**

**YDataLogger**

Retourne le numéro du Run actuel, correspondant au nombre de fois que le module a été mis sous tension avec la fonction d'enregistreur de données active.

js	function <b>get_currentRunIndex</b> ( )
nodejs	function <b>get_currentRunIndex</b> ( )
php	function <b>get_currentRunIndex</b> ( )
cpp	int <b>get_currentRunIndex</b> ( )
m	-(int) currentRunIndex
pas	function <b>get_currentRunIndex</b> ( ): LongInt
vb	function <b>get_currentRunIndex</b> ( ) As Integer
cs	int <b>get_currentRunIndex</b> ( )
java	int <b>get_currentRunIndex</b> ( )
py	def <b>get_currentRunIndex</b> ( )
cmd	YDataLogger <b>target</b> <b>get_currentRunIndex</b>

**Retourne :**

un entier représentant le numéro du Run actuel, correspondant au nombre de fois que le module a été mis sous tension avec la fonction d'enregistreur de données active

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRUNINDEX\_INVALID.

**datalogger→get\_dataSets()****YDataLogger****datalogger→dataSets()[datalogger dataSets]**

Retourne une liste d'objets YDataSet permettant de récupérer toutes les mesures stockées par l'enregistreur de données.

js	function <b>get_dataSets</b> ( )
nodejs	function <b>get_dataSets</b> ( )
php	function <b>get_dataSets</b> ( )
cpp	vector<YDataSet> <b>get_dataSets</b> ( )
m	-(NSMutableArray*) dataSets
pas	function <b>get_dataSets</b> ( ): TYDataSetArray
vb	function <b>get_dataSets</b> ( ) As List
cs	List<YDataSet> <b>get_dataSets</b> ( )
java	ArrayList<YDataSet> <b>get_dataSets</b> ( )
py	def <b>get_dataSets</b> ( )
cmd	YDataLogger <b>target</b> <b>get_dataSets</b>

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets YDataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Retourne :**

une liste d'objets YDataSet

En cas d'erreur, déclenche une exception ou retourne une liste vide.

**datalogger→get\_dataStreams()****YDataLogger****datalogger→dataStreams()[datalogger dataStreams:  
]**

Construit une liste de toutes les séquences de mesures mémorisées par l'enregistreur (ancienne méthode).

js	function <b>get_dataStreams</b> ( <b>v</b> )
nodejs	function <b>get_dataStreams</b> ( <b>v</b> )
php	function <b>get_dataStreams</b> ( <b>&amp;\$v</b> )
cpp	int <b>get_dataStreams</b> ( )
m	-(int) dataStreams : (NSArray**) <b>v</b>
pas	function <b>get_dataStreams</b> ( <b>v</b> : Tlist): integer
vb	procedure <b>get_dataStreams</b> ( ByVal <b>v</b> As List)
cs	int <b>get_dataStreams</b> ( List<YDataStream> <b>v</b> )
java	int <b>get_dataStreams</b> ( ArrayList<YDataStream> <b>v</b> )
py	def <b>get_dataStreams</b> ( <b>v</b> )

L'appelant doit passer par référence un tableau vide pour stocker les objets YDataStream, et la méthode va les remplir avec des objets décrivant les séquences de données disponibles.

Cette méthode est préservée pour maintenir la compatibilité avec les applications existantes. Pour les nouvelles applications, il est préférable d'utiliser la méthode `get_dataSets()` ou d'appeler directement la méthode `get_recordedData()` sur l'objet représentant le capteur désiré.

**Paramètres :**

**v** un tableau de YDataStreams qui sera rempli avec les séquences trouvées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## **datalogger→get\_errorMessage()** **datalogger→errorMessage()[datalogger** **errorMessage]**

## **YDataLogger**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'enregistreur de données.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

### **Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'enregistreur de données.

**datalogger→get\_errorType()****YDataLogger****datalogger→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'enregistreur de données.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'enregistreur de données.



**datalogger→get\_friendlyName()**  
**datalogger→friendlyName()[datalogger  
friendlyName]**

**YDataLogger**

Retourne un identifiant global de l'enregistreur de données au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'enregistreur de données si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'enregistreur de données (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'enregistreur de données en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **datalogger→get\_functionDescriptor() datalogger→functionDescriptor()[datalogger functionDescriptor]**

YDataLogger

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**datalogger→get\_functionId()****YDataLogger****datalogger→functionId()[datalogger functionId]**

Retourne l'identifiant matériel de l'enregistreur de données, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'enregistreur de données (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**datalogger→get\_hardwareId()****YDataLogger****datalogger→hardwareId()[datalogger hardwareId]**

Retourne l'identifiant matériel unique de l'enregistreur de données au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'enregistreur de données (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'enregistreur de données (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**datalogger→get\_logicalName()****YDataLogger****datalogger→logicalName()[datalogger logicalName]**

Retourne le nom logique de l'enregistreur de données.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YDataLogger <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'enregistreur de données. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**datalogger**→**get\_module()****YDataLogger****datalogger**→**module()**[**datalogger module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

## **datalogger→get\_module\_async()** **datalogger→module\_async()**

## **YDataLogger**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

### **Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

### **Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**datalogger**→**get\_recording()****YDataLogger****datalogger**→**recording()**[**datalogger recording**]

Retourne l'état d'activation de l'enregistreur de données.

js	function <b>get_recording</b> ( )
nodejs	function <b>get_recording</b> ( )
php	function <b>get_recording</b> ( )
cpp	Y_RECORDING_enum <b>get_recording</b> ( )
m	-(Y_RECORDING_enum) recording
pas	function <b>get_recording</b> ( ): Integer
vb	function <b>get_recording</b> ( ) As Integer
cs	int <b>get_recording</b> ( )
java	int <b>get_recording</b> ( )
py	def <b>get_recording</b> ( )
cmd	YDataLogger <b>target</b> <b>get_recording</b>

**Retourne :**

soit Y\_RECORDING\_OFF, soit Y\_RECORDING\_ON, selon l'état d'activation de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_RECORDING\_INVALID.



**datalogger→get\_timeUTC()****YDataLogger****datalogger→timeUTC()[datalogger timeUTC]**

Retourne le timestamp Unix de l'heure UTC actuelle, lorsqu'elle est connue.

js	function <b>get_timeUTC</b> ( )
nodejs	function <b>get_timeUTC</b> ( )
php	function <b>get_timeUTC</b> ( )
cpp	s64 <b>get_timeUTC</b> ( )
m	-(s64) timeUTC
pas	function <b>get_timeUTC</b> ( ): int64
vb	function <b>get_timeUTC</b> ( ) As Long
cs	long <b>get_timeUTC</b> ( )
java	long <b>get_timeUTC</b> ( )
py	def <b>get_timeUTC</b> ( )
cmd	YDataLogger <b>target get_timeUTC</b>

**Retourne :**

un entier représentant le timestamp Unix de l'heure UTC actuelle, lorsqu'elle est connue

En cas d'erreur, déclenche une exception ou retourne Y\_TIMEUTC\_INVALID.

**datalogger**→**get\_userdata()****YDataLogger****datalogger**→**userData()**[**datalogger** **userData**]

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) <b>userData</b>
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**datalogger→isOnline()[datalogger isOnline]****YDataLogger**

Vérifie si le module hébergeant l'enregistreur de données est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'enregistreur de données sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'enregistreur de données est joignable, `false` sinon

Vérifie si le module hébergeant l'enregistreur de données est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'enregistreur de données sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**datalogger→load()[datalogger load: ]****YDataLogger**

Met en cache les valeurs courantes de l'enregistreur de données, avec une durée de validité spécifiée.

js	function load( msValidity)
nodejs	function load( msValidity)
php	function load( \$msValidity)
cpp	YRETCODE load( int msValidity)
m	-(YRETCODE) load : (int) msValidity
pas	function load( msValidity: integer): YRETCODE
vb	function load( ByVal msValidity As Integer) As YRETCODE
cs	YRETCODE load( int msValidity)
java	int load( long msValidity)
py	def load( msValidity)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**datalogger**→**load\_async()****YDataLogger**

Met en cache les valeurs courantes de l'enregistreur de données, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## **datalogger→nextDataLogger() [datalogger nextDataLogger]**

## **YDataLogger**

Continue l'énumération des enregistreurs de données commencée à l'aide de `yFirstDataLogger()`.

js	function <b>nextDataLogger</b> ( )
nodejs	function <b>nextDataLogger</b> ( )
php	function <b>nextDataLogger</b> ( )
cpp	YDataLogger * <b>nextDataLogger</b> ( )
m	-(YDataLogger*) <b>nextDataLogger</b>
pas	function <b>nextDataLogger</b> ( ): TYDataLogger
vb	function <b>nextDataLogger</b> ( ) As YDataLogger
cs	YDataLogger <b>nextDataLogger</b> ( )
java	YDataLogger <b>nextDataLogger</b> ( )
py	def <b>nextDataLogger</b> ( )

### **Retourne :**

un pointeur sur un objet `YDataLogger` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## `datalogger`→`registerValueCallback()`[`datalogger` `registerValueCallback:` ]

YDataLogger

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YDataLoggerValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YDataLoggerValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYDataLoggerValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.



**datalogger→set\_autoStart()****YDataLogger****datalogger→setAutoStart()[datalogger setAutoStart:  
]**

Modifie le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

js	function <b>set_autoStart</b> ( <b>newval</b> )
nodejs	function <b>set_autoStart</b> ( <b>newval</b> )
php	function <b>set_autoStart</b> ( <b>\$newval</b> )
cpp	int <b>set_autoStart</b> ( Y_AUTOSTART_enum <b>newval</b> )
m	-(int) setAutoStart : (Y_AUTOSTART_enum) <b>newval</b>
pas	function <b>set_autoStart</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_autoStart</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_autoStart</b> ( int <b>newval</b> )
java	int <b>set_autoStart</b> ( int <b>newval</b> )
py	def <b>set_autoStart</b> ( <b>newval</b> )
cmd	YDataLogger <b>target</b> <b>set_autoStart</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** soit Y\_AUTOSTART\_OFF, soit Y\_AUTOSTART\_ON, selon le mode d'activation automatique de l'enregistreur de données à la mise sous tension

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**datalogger→set\_logicalName()****YDataLogger****datalogger→setLogicalName()[datalogger  
setLogicalName: ]**

Modifie le nom logique de l'enregistreur de données.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YDataLogger <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'enregistreur de données.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## **datalogger→set\_recording()** **datalogger→setRecording() [datalogger** **setRecording: ]**

YDataLogger

Modifie l'état d'activation de l'enregistreur de données.

js	function <b>set_recording</b> ( <b>newval</b> )
nodejs	function <b>set_recording</b> ( <b>newval</b> )
php	function <b>set_recording</b> ( <b>\$newval</b> )
cpp	int <b>set_recording</b> ( Y_RECORDING_enum <b>newval</b> )
m	-(int) setRecording : (Y_RECORDING_enum) <b>newval</b>
pas	function <b>set_recording</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_recording</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_recording</b> ( int <b>newval</b> )
java	int <b>set_recording</b> ( int <b>newval</b> )
py	def <b>set_recording</b> ( <b>newval</b> )
cmd	YDataLogger <b>target</b> <b>set_recording newval</b>

### Paramètres :

**newval** soit Y\_RECORDING\_OFF, soit Y\_RECORDING\_ON, selon l'état d'activation de l'enregistreur de données

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**datalogger**→**set\_timeUTC()****YDataLogger****datalogger**→**setTimeUTC()**[**datalogger setTimeUTC:** ]

Modifie la référence de temps UTC, afin de l'attacher aux données enregistrées.

js	function <b>set_timeUTC</b> ( <b>newval</b> )
nodejs	function <b>set_timeUTC</b> ( <b>newval</b> )
php	function <b>set_timeUTC</b> ( <b>\$newval</b> )
cpp	int <b>set_timeUTC</b> ( s64 <b>newval</b> )
m	-(int) setTimeUTC : (s64) <b>newval</b>
pas	function <b>set_timeUTC</b> ( <b>newval</b> : int64): integer
vb	function <b>set_timeUTC</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_timeUTC</b> ( long <b>newval</b> )
java	int <b>set_timeUTC</b> ( long <b>newval</b> )
py	def <b>set_timeUTC</b> ( <b>newval</b> )
cmd	YDataLogger <b>target set_timeUTC newval</b>

**Paramètres :**

**newval** un entier représentant la référence de temps UTC, afin de l'attacher aux données enregistrées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**datalogger**→**set\_userdata()****YDataLogger****datalogger**→**setUserData()**[**datalogger setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**datalogger**→**wait\_async()****YDataLogger**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.9. Séquence de données mise en forme

Un Run est un intervalle de temps pendant lequel un module est sous tension. Les objets YDataRun fournissent un accès facilité à toutes les mesures collectées durant un Run donné, y compris en permettant la lecture par mesure distantes d'un intervalle spécifié.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_datalogger.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YDataLogger = yoctolib.YDataLogger;
php	require_once('yocto_datalogger.php');
c++	#include "yocto_datalogger.h"
m	#import "yocto_datalogger.h"
pas	uses yocto_datalogger;
vb	yocto_datalogger.vb
cs	yocto_datalogger.cs
java	import com.yoctopuce.YoctoAPI.YDataLogger;
py	from yocto_datalogger import *

Méthodes des objets YDataRun
<b>datarun→get_averageValue(measureName, pos)</b> Retourne la valeur moyenne des mesures observées au moment choisi.
<b>datarun→get_duration()</b> Retourne la durée (en secondes) du Run.
<b>datarun→get_maxValue(measureName, pos)</b> Retourne la valeur maximale des mesures observées au moment choisi.
<b>datarun→get_measureNames()</b> Retourne les noms des valeurs mesurées par l'enregistreur de données.
<b>datarun→get_minValue(measureName, pos)</b> Retourne la valeur minimale des mesures observées au moment choisi.
<b>datarun→get_startTimeUTC()</b> Retourne l'heure absolue du début du Run, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).
<b>datarun→get_valueCount()</b> Retourne le nombre de valeurs accessibles dans ce Run, étant donné l'intervalle de temps choisi entre les valeurs.
<b>datarun→get_valueInterval()</b> Retourne l'intervalle de temps représenté par chaque valeur de ce run.
<b>datarun→set_valueInterval(valueInterval)</b> Change l'intervalle de temps représenté par chaque valeur de ce run.

datarun→get\_averageValue()

YDataRun

datarun→averageValue()

Retourne la valeur moyenne des mesures observées au moment choisi.

```
js function get_averageValue( measureName, pos)
nodejs function get_averageValue( measureName, pos)
php function get_averageValue( $measureName, $pos)
java double get_averageValue( String measureName, int pos)
py def get_averageValue( measureName, pos)
```

**Paramètres :**

**measureName** le nom de la mesure désirée (un des noms retournés par get\_measureNames)

**pos** l'index de la position désirée, entre 0 et la valeur de get\_valueCount

**Retourne :**

une nombre flottant (la valeur moyenne).

En cas d'erreur, déclenche une exception ou retourne Y\_AVERAGEVALUE\_INVALID.



**datarun→get\_duration()**  
**datarun→duration()****YDataRun**

Retourne la durée (en secondes) du Run.

js	function <b>get_duration</b> ( )
nodejs	function <b>get_duration</b> ( )
php	function <b>get_duration</b> ( )
java	long <b>get_duration</b> ( )
py	def <b>get_duration</b> ( )

Lorsque cette méthode est appelée dur le Run courant et que l'enregistreur de données est actif, l'appel à cette méthode force un rechargement de la dernière séquence du module pour s'assurer que la réponse prend en compte les dernières données enregistrées.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le début du Run (quand le module a été mis sous tension) et la dernière mesure enregistrée.

datarun→get\_maxValue()

YDataRun

datarun→maxValue()

Retourne la valeur maximale des mesures observées au moment choisi.

```
js function get_maxValue( measureName, pos)
nodejs function get_maxValue( measureName, pos)
php function get_maxValue( $measureName, $pos)
java double get_maxValue( String measureName, int pos)
py def get_maxValue( measureName, pos)
```

**Paramètres :**

**measureName** le nom de la mesure désirée (un des noms retournés par get\_measureNames)

**pos** l'index de la position désirée, entre 0 et la valeur de get\_valueCount

**Retourne :**

une nombre flottant (la valeur maximale).

En cas d'erreur, déclenche une exception ou retourne Y\_MAXVALUE\_INVALID.

**datarun→get\_measureNames()****YDataRun****datarun→measureNames()**

Retourne les noms des valeurs mesurées par l'enregistreur de données.

```
js function get_measureNames( )
nodejs function get_measureNames( )
php function get_measureNames( )
java ArrayList<String> get_measureNames( )
py def get_measureNames( )
```

Dans la plupart des cas, le nom des colonnes correspond à l'identifiant matériel du capteur qui a produit la mesure.

**Retourne :**

une liste de chaîne de caractères (les noms des mesures)

En cas d'erreur, déclenche une exception ou retourne une liste vide.

datarun→get\_minValue()

YDataRun

datarun→minValue()

Retourne la valeur minimale des mesures observées au moment choisi.

```
js function get_minValue( measureName, pos)
nodejs function get_minValue( measureName, pos)
php function get_minValue( $measureName, $pos)
java double get_minValue( String measureName, int pos)
py def get_minValue( measureName, pos)
```

**Paramètres :**

**measureName** le nom de la mesure désirée (un des noms retournés par get\_measureNames)

**pos** l'index de la position désirée, entre 0 et la valeur de get\_valueCount

**Retourne :**

une nombre flottant (la valeur minimale).

En cas d'erreur, déclenche une exception ou retourne Y\_MINVALUE\_INVALID.

---

**datarun→get\_startTimeUTC()****YDataRun****datarun→startTimeUTC()**

---

Retourne l'heure absolue du début du Run, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

Si l'heure UTC n'a jamais été configurée dans l'enregistreur de données durant le run, et si il ne s'agit pas du run courant, cette méthode retourne 0.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 et le début du Run.

**datarun**→**get\_valueCount()**

**YDataRun**

**datarun**→**valueCount()**

---

Retourne le nombre de valeurs accessibles dans ce Run, étant donné l'intervalle de temps choisi entre les valeurs.

```
js function get_valueCount( )
```

```
nodejs function get_valueCount( )
```

```
php function get_valueCount( )
```

```
java int get_valueCount( )
```

```
py def get_valueCount( )
```

Lorsque cette méthode est appelée dur le Run courant et que l'enregistreur de données est actif, l'appel à cette méthode force un rechargement de la dernière séquence du module pour s'assurer que la réponse prend en compte les dernières données enregistrées.

**Retourne :**

un entier positif correspondant à la durée du Run divisée par l'intervalle entre les valeurs.

**datarun→get\_valueInterval()****YDataRun****datarun→valueInterval()**

Retourne l'intervalle de temps représenté par chaque valeur de ce run.

js	function <b>get_valueInterval</b> ( )
nodejs	function <b>get_valueInterval</b> ( )
php	function <b>get_valueInterval</b> ( )
java	int <b>get_valueInterval</b> ( )
py	def <b>get_valueInterval</b> ( )

La valeur par défaut correspond à la plus grande granularité des mesures archivées dans la flash de l'enregistreur de données pour ce Run, mais l'intervalle à utiliser peut être configuré librement si désiré.

**Retourne :**

un entier positif correspondant au nombre de secondes couvertes par chaque valeur représentée dans le Run.

**datarun**→**set\_valueInterval()**

**YDataRun**

**datarun**→**setValueInterval()**

Change l'intervalle de temps représenté par chaque valeur de ce run.

```
js function set_valueInterval( valueInterval)
nodejs function set_valueInterval( valueInterval)
php function set_valueInterval( $valueInterval)
java void set_valueInterval( int valueInterval)
py def set_valueInterval( valueInterval)
```

La valeur par défaut correspond à la plus grande granularité des mesures archivées dans la flash de l'enregistreur de données pour ce Run, mais l'intervalle à utiliser peut être configuré librement si désiré.

#### Paramètres :

**valueInterval** un nombre entier de secondes.

#### Retourne :

nothing



## 3.10. Séquence de données enregistrées

Les objets `YDataSet` permettent de récupérer un ensemble de mesures enregistrées correspondant à un capteur donné, pour une période choisie. Ils permettent le chargement progressif des données. Lorsque l'objet `YDataSet` est instancié par la fonction `get_recordedData()`, aucune donnée n'est encore chargée du module. Ce sont les appels successifs à la méthode `loadMore()` qui procèdent au chargement effectif des données depuis l'enregistreur de données.

Un résumé des mesures disponibles est disponible via la fonction `get_preview()` dès le premier appel à `loadMore()`. Les mesures elles-même sont disponibles via la fonction `get_measures()` au fur et à mesure de leur chargement.

Cette classe ne fonctionne que si le module utilise un firmware récent, car les objets `YDataSet` ne sont pas supportés par les firmwares antérieurs à la révision 13000.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_api.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YAPI = yoctolib.YAPI;</code> <code>var YModule = yoctolib.YModule;</code>
php	<code>require_once('yocto_api.php');</code>
c++	<code>#include "yocto_api.h"</code>
m	<code>#import "yocto_api.h"</code>
pas	<code>uses yocto_api;</code>
vb	<code>yocto_api.vb</code>
cs	<code>yocto_api.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YModule;</code>
py	<code>from yocto_api import *</code>

### Méthodes des objets `YDataSet`

#### `dataset→get_endTimeUTC()`

Retourne l'heure absolue de la fin des mesures disponibles, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

#### `dataset→get_functionId()`

Retourne l'identifiant matériel de la fonction qui a effectué les mesures, sans référence au module.

#### `dataset→get_hardwareId()`

Retourne l'identifiant matériel unique de la fonction qui a effectué les mesures, au format `SERIAL.FUNCTIONID`.

#### `dataset→get_measures()`

Retourne toutes les mesures déjà disponibles pour le `DataSet`, sous forme d'une liste d'objets `YMeasure`.

#### `dataset→get_preview()`

Retourne une version résumée des mesures qui pourront être obtenues de ce `YDataSet`, sous forme d'une liste d'objets `YMeasure`.

#### `dataset→get_progress()`

Retourne l'état d'avancement du chargement des données, sur une échelle de 0 à 100.

#### `dataset→get_startTimeUTC()`

Retourne l'heure absolue du début des mesures disponibles, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

#### `dataset→get_summary()`

Retourne un objet `YMeasure` résumant tout le `YDataSet`.

#### `dataset→get_unit()`

### 3. Reference

---

Retourne l'unité dans laquelle la valeur mesurée est exprimée.

#### **dataset→loadMore()**

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, et met à jour l'indicateur d'avancement.

#### **dataset→loadMore\_async(callback, context)**

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

**dataset**→**get\_endTimeUTC()****YDataSet****dataset**→**endTimeUTC()[dataset endTimeUTC]**

Retourne l'heure absolue de la fin des mesures disponibles, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

js	function <b>get_endTimeUTC</b> ( )
nodejs	function <b>get_endTimeUTC</b> ( )
php	function <b>get_endTimeUTC</b> ( )
cpp	s64 <b>get_endTimeUTC</b> ( )
m	-(s64) endTimeUTC
pas	function <b>get_endTimeUTC</b> ( ): int64
vb	function <b>get_endTimeUTC</b> ( ) As Long
cs	long <b>get_endTimeUTC</b> ( )
java	long <b>get_endTimeUTC</b> ( )
py	def <b>get_endTimeUTC</b> ( )

Lorsque l'objet YDataSet est créé, l'heure de fin est celle qui a été passée en paramètre à la fonction `get_dataSet`. Dès le premier appel à la méthode `loadMore()`, l'heure de fin est mise à jour à la dernière mesure effectivement disponible dans l'enregistreur de données pour la plage spécifiée.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 et la dernière mesure.

**dataset**→**get\_functionId()****YDataSet****dataset**→**functionId()**[dataset functionId]

Retourne l'identifiant matériel de la fonction qui a effectué les mesures, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
pas	function <b>get_functionId</b> ( ): string
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple temperature1.

**Retourne :**

une chaîne de caractères identifiant la fonction (ex: temperature1)

**dataset**→**get\_hardwareId()****YDataSet****dataset**→**hardwareId()**[dataset hardwareId]

Retourne l'identifiant matériel unique de la fonction qui a effectué les mesures, au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
pas	function <b>get_hardwareId</b> ( ): string
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par exemple `THRMCPL1-123456.temperature1`).

**Retourne :**

une chaîne de caractères identifiant la fonction (ex: `THRMCPL1-123456.temperature1`)

En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**dataset**→**get\_measures()****YDataSet****dataset**→**measures()**[dataset measures]

Retourne toutes les mesures déjà disponibles pour le DataSet, sous forme d'une liste d'objets YMeasure.

js	function <b>get_measures</b> ( )
nodejs	function <b>get_measures</b> ( )
php	function <b>get_measures</b> ( )
cpp	vector<YMeasure> <b>get_measures</b> ( )
m	-(NSMutableArray*) measures
pas	function <b>get_measures</b> ( ): TYMeasureArray
vb	function <b>get_measures</b> ( ) As List
cs	List<YMeasure> <b>get_measures</b> ( )
java	ArrayList<YMeasure> <b>get_measures</b> ( )
py	def <b>get_measures</b> ( )

Chaque élément contient: - le moment où la mesure a débuté - le moment où la mesure s'est terminée - la valeur minimale observée dans l'intervalle de temps - la valeur moyenne observée dans l'intervalle de temps - la valeur maximale observée dans l'intervalle de temps

Avant d'appeler cette méthode, vous devez appeler `loadMore()` pour charger des données depuis l'enregistreur sur le module. L'appel doit être répété plusieurs fois pour charger toutes les données, mais vous pouvez commencer à utiliser les données disponibles avant qu'elles n'aient été toutes chargées

Les mesures les plus anciennes sont toujours chargées les premières, et les plus récentes en dernier. De ce fait, les timestamps dans la table des mesures sont normalement par ordre chronologique. La seule exception est dans le cas où il y a eu un ajustement de l'horloge UTC de l'enregistreur de données pendant l'enregistrement.

**Retourne :**

un tableau d'enregistrements, chaque enregistrement représentant une mesure effectuée à un moment précis.

En cas d'erreur, déclenche une exception ou retourne un tableau vide.

**dataset**→**get\_preview()****YDataSet****dataset**→**preview()**[dataset preview]

Retourne une version résumée des mesures qui pourront être obtenues de ce YDataSet, sous forme d'une liste d'objets YMeasure.

js	function <b>get_preview</b> ( )
nodejs	function <b>get_preview</b> ( )
php	function <b>get_preview</b> ( )
cpp	vector<YMeasure> <b>get_preview</b> ( )
m	-(NSMutableArray*) preview
pas	function <b>get_preview</b> ( ): TYMeasureArray
vb	function <b>get_preview</b> ( ) As List
cs	List<YMeasure> <b>get_preview</b> ( )
java	ArrayList<YMeasure> <b>get_preview</b> ( )
py	def <b>get_preview</b> ( )

Chaque élément contient: - le début d'un intervalle de temps - la fin d'un intervalle de temps - la valeur minimale observée dans l'intervalle de temps - la valeur moyenne observée dans l'intervalle de temps - la valeur maximale observée dans l'intervalle de temps

Le résumé des mesures est disponible dès que `loadMore()` a été appelé pour la première fois.

**Retourne :**

un tableau d'enregistrements, chaque enregistrement représentant les mesures observée durant un certain intervalle de temps.

En cas d'erreur, déclenche une exception ou retourne un tableau vide.

**dataset**→**get\_progress()****YDataSet****dataset**→**progress()**[dataset progress]

Retourne l'état d'avancement du chargement des données, sur une échelle de 0 à 100.

js	function <b>get_progress</b> ( )
nodejs	function <b>get_progress</b> ( )
php	function <b>get_progress</b> ( )
cpp	int <b>get_progress</b> ( )
m	-(int) progress
pas	function <b>get_progress</b> ( ): LongInt
vb	function <b>get_progress</b> ( ) As Integer
cs	int <b>get_progress</b> ( )
java	int <b>get_progress</b> ( )
py	def <b>get_progress</b> ( )

A l'instanciation de l'objet par la fonction `get_dataSet( )`, l'avancement est nul. Au fur et à mesure des appels à `loadMore( )`, l'avancement progresse pour atteindre la valeur 100 lorsque toutes les mesures ont été chargées.

**Retourne :**

un nombre entier entre 0 et 100 représentant l'avancement du chargement des données demandées.



**dataset**→**get\_startTimeUTC()****YDataSet****dataset**→**startTimeUTC()[dataset startTimeUTC]**

Retourne l'heure absolue du début des mesures disponibles, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

js	function <b>get_startTimeUTC</b> ( )
nodejs	function <b>get_startTimeUTC</b> ( )
php	function <b>get_startTimeUTC</b> ( )
cpp	s64 <b>get_startTimeUTC</b> ( )
m	-(s64) startTimeUTC
pas	function <b>get_startTimeUTC</b> ( ): int64
vb	function <b>get_startTimeUTC</b> ( ) As Long
cs	long <b>get_startTimeUTC</b> ( )
java	long <b>get_startTimeUTC</b> ( )
py	def <b>get_startTimeUTC</b> ( )

Lorsque l'objet YDataSet est créé, l'heure de départ est celle qui a été passée en paramètre à la fonction `get_dataSet`. Dès le premier appel à la méthode `loadMore()`, l'heure de départ est mise à jour à la première mesure effectivement disponible dans l'enregistreur de données pour la plage spécifiée.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 et la première mesure enregistrée.

**dataset**→**get\_summary()****YDataSet****dataset**→**summary()**[dataset summary]

Retourne un objet YMeasure résumant tout le YDataSet.

js	function <b>get_summary</b> ( )
nodejs	function <b>get_summary</b> ( )
php	function <b>get_summary</b> ( )
cpp	YMeasure <b>get_summary</b> ( )
m	-(YMeasure*) summary
pas	function <b>get_summary</b> ( ): TYMeasure
vb	function <b>get_summary</b> ( ) As YMeasure
cs	YMeasure <b>get_summary</b> ( )
java	YMeasure <b>get_summary</b> ( )
py	def <b>get_summary</b> ( )

Il inclut les information suivantes: - le moment de la première mesure - le moment de la dernière mesure - la valeur minimale observée dans l'intervalle de temps - la valeur moyenne observée dans l'intervalle de temps - la valeur maximale observée dans l'intervalle de temps

Ce résumé des mesures est disponible dès que `loadMore( )` a été appelé pour la première fois.

**Retourne :**

un objet YMeasure

**dataset**→**get\_unit()****YDataSet****dataset**→**unit()**[dataset unit]

Retourne l'unité dans laquelle la valeur mesurée est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )

**Retourne :**

une chaîne de caractères représentant une unité physique.

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**dataset→loadMore()[dataset loadMore]****YDataSet**

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, et met à jour l'indicateur d'avancement.

js	function <b>loadMore</b> ( )
nodejs	function <b>loadMore</b> ( )
php	function <b>loadMore</b> ( )
cpp	int <b>loadMore</b> ( )
m	-(int) <b>loadMore</b>
pas	function <b>loadMore</b> ( ): LongInt
vb	function <b>loadMore</b> ( ) As Integer
cs	int <b>loadMore</b> ( )
java	int <b>loadMore</b> ( )
py	def <b>loadMore</b> ( )

**Retourne :**

un nombre entier entre 0 et 100 représentant l'avancement du chargement des données demandées, ou un code d'erreur négatif en cas de problème.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**dataset→loadMore\_async()****YDataSet**

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

```
js function loadMore_async( callback, context)
```

```
nodejs function loadMore_async( callback, context)
```

**Paramètres :**

**callback** fonction fournie par l'utilisateur, qui sera appelée lorsque la suite du chargement aura été effectué. La fonction callback doit prendre trois arguments: - la variable de contexte à disposition de l'utilisateur - l'objet YDataSet dont la méthode loadMore\_async a été appelée - le résultat de l'appel: soit l'état d'avancement du chargement (0...100), ou un code d'erreur négatif en cas de problème.

**context** variable de contexte à disposition de l'utilisateur

**Retourne :**

rien.

## 3.11. Séquence de données enregistrées brute

Les objets YDataStream correspondent aux séquences de mesures enregistrées brutes, directement telles qu'obtenues par l'enregistreur de données présent dans les senseurs de Yoctopuce.

Dans la plupart des cas, il n'est pas nécessaire d'utiliser les objets DataStream, car les objets YDataSet (retournés par la méthode `get_recordedData()` des senseurs et la méthode `get_dataSets()` du DataLogger) fournissent une interface plus pratique.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_api.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YAPI = yoctolib.YAPI;</code> <code>var YModule = yoctolib.YModule;</code>
php	<code>require_once('yocto_api.php');</code>
c++	<code>#include "yocto_api.h"</code>
m	<code>#import "yocto_api.h"</code>
pas	<code>uses yocto_api;</code>
vb	<code>yocto_api.vb</code>
cs	<code>yocto_api.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YModule;</code>
py	<code>from yocto_api import *</code>

### Méthodes des objets YDataStream

#### **datastream→get\_averageValue()**

Retourne la moyenne des valeurs observées durant cette séquence.

#### **datastream→get\_columnCount()**

Retourne le nombre de colonnes de données contenus dans la séquence.

#### **datastream→get\_columnNames()**

Retourne le nom (la sémantique) des colonnes de données contenus dans la séquence.

#### **datastream→get\_data(row, col)**

Retourne une mesure unique de la séquence, spécifiée par l'index de l'enregistrement (ligne) et de la mesure (colonne).

#### **datastream→get\_dataRows()**

Retourne toutes les données mesurées contenues dans la séquence, sous forme d'une liste de vecteurs (table bidimensionnelle).

#### **datastream→get\_dataSamplesIntervalMs()**

Retourne le nombre de millisecondes entre chaque mesure de la séquence.

#### **datastream→get\_duration()**

Retourne la durée approximative de cette séquence, en secondes.

#### **datastream→get\_maxValue()**

Retourne la plus grande valeur observée durant cette séquence.

#### **datastream→get\_minValue()**

Retourne la plus petite valeur observée durant cette séquence.

#### **datastream→get\_rowCount()**

Retourne le nombre d'enregistrement contenus dans la séquence.

#### **datastream→get\_runIndex()**

Retourne le numéro de Run de la séquence de données.

#### **datastream→get\_startTime()**

Retourne le temps de départ relatif de la séquence (en secondes).

**`datastream→get_startTimeUTC()`**

Retourne l'heure absolue du début de la séquence de données, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

**datastream**→**get\_averageValue()****YDataStream****datastream**→**averageValue()**[**datastream**  
**averageValue**]

Retourne la moyenne des valeurs observées durant cette séquence.

js	function <b>get_averageValue</b> ( )
nodejs	function <b>get_averageValue</b> ( )
php	function <b>get_averageValue</b> ( )
cpp	double <b>get_averageValue</b> ( )
m	-(double) averageValue
pas	function <b>get_averageValue</b> ( ): double
vb	function <b>get_averageValue</b> ( ) As Double
cs	double <b>get_averageValue</b> ( )
java	double <b>get_averageValue</b> ( )
py	def <b>get_averageValue</b> ( )

Si le module utilise un firmware antérieur à la version 13000, cette méthode retournera toujours Y\_DATA\_INVALID.

**Retourne :**

un nombre décimal correspondant à la moyenne des valeurs, ou Y\_DATA\_INVALID si la séquence n'est pas encore terminée.

En cas d'erreur, déclenche une exception ou retourne Y\_DATA\_INVALID.



**datastream→get\_columnCount()**  
**datastream→columnCount()[datastream**  
**columnCount]**

**YDataStream**

Retourne le nombre de colonnes de données contenus dans la séquence.

js	function <b>get_columnCount</b> ( )
nodejs	function <b>get_columnCount</b> ( )
php	function <b>get_columnCount</b> ( )
cpp	int <b>get_columnCount</b> ( )
m	-(int) columnCount
pas	function <b>get_columnCount</b> ( ): LongInt
vb	function <b>get_columnCount</b> ( ) As Integer
cs	int <b>get_columnCount</b> ( )
java	int <b>get_columnCount</b> ( )
py	def <b>get_columnCount</b> ( )

La sémantique des données présentes dans chaque colonne peut être obtenue à l'aide de la méthode `get_columnNames()`.

Si le module utilise un firmware antérieur à la version 13000, cette méthode déclenche le chargement de toutes les données de la séquence si nécessaire, ce qui peut prendre un petit instant.

**Retourne :**

un entier positif correspondant au nombre de colonnes.

En cas d'erreur, déclenche une exception ou retourne zéro.

**datastream→get\_columnNames()**  
**datastream→columnNames()[datastream  
columnNames]**

**YDataStream**

Retourne le nom (la sémantique) des colonnes de données contenus dans la séquence.

js	function <b>get_columnNames</b> ( )
nodejs	function <b>get_columnNames</b> ( )
php	function <b>get_columnNames</b> ( )
cpp	vector<string> <b>get_columnNames</b> ( )
m	-(NSMutableArray*) columnNames
pas	function <b>get_columnNames</b> ( ): TStringArray
vb	function <b>get_columnNames</b> ( ) As List
cs	List<string> <b>get_columnNames</b> ( )
java	ArrayList<String> <b>get_columnNames</b> ( )
py	def <b>get_columnNames</b> ( )

Dans la plupart des cas, le nom des colonnes correspond à l'identifiant matériel du capteur qui a produit la mesure. Pour les séquences enregistrées à faible fréquence, l'enregistreur de donnée stocke la valeur min, moyenne et max observée durant chaque intervalle de temps dans des colonnes avec les suffixes `_min`, `_avg` et `_max` respectivement.

Si le module utilise un firmware antérieur à la version 13000, cette méthode déclenche le chargement de toutes les données de la séquence si nécessaire, ce qui peut prendre un petit instant.

**Retourne :**

une liste de chaîne de caractères.

En cas d'erreur, déclenche une exception ou retourne une liste vide.



**datastream**→**get\_dataRows()****YDataStream****datastream**→**dataRows()**[datastream dataRows]

Retourne toutes les données mesurées contenues dans la séquence, sous forme d'une liste de vecteurs (table bidimensionnelle).

js	function <b>get_dataRows</b> ( )
nodejs	function <b>get_dataRows</b> ( )
php	function <b>get_dataRows</b> ( )
cpp	vector< vector<double> > <b>get_dataRows</b> ( )
m	-(NSMutableArray*) dataRows
pas	function <b>get_dataRows</b> ( ): TDoubleArrayArray
vb	function <b>get_dataRows</b> ( ) As List
cs	List<List<double>> <b>get_dataRows</b> ( )
java	ArrayList<ArrayList<Double>> <b>get_dataRows</b> ( )
py	def <b>get_dataRows</b> ( )

La sémantique des données présentes dans chaque colonne peut être obtenue à l'aide de la méthode `get_columnNames()`.

Cette méthode déclenche le chargement de toutes les données de la séquence, si cela n'était pas encore fait.

**Retourne :**

une liste d'enregistrements, chaque enregistrement étant lui-même une liste de nombres décimaux.

En cas d'erreur, déclenche une exception ou retourne une liste vide.

**datastream**→**get\_dataSamplesIntervalMs()****YDataStream****datastream**→**dataSamplesIntervalMs()**[**datastream**  
**dataSamplesIntervalMs**]

Retourne le nombre de millisecondes entre chaque mesure de la séquence.

js	function <b>get_dataSamplesIntervalMs</b> ( )
nodejs	function <b>get_dataSamplesIntervalMs</b> ( )
php	function <b>get_dataSamplesIntervalMs</b> ( )
cpp	int <b>get_dataSamplesIntervalMs</b> ( )
m	-(int) dataSamplesIntervalMs
pas	function <b>get_dataSamplesIntervalMs</b> ( ): LongInt
vb	function <b>get_dataSamplesIntervalMs</b> ( ) As Integer
cs	int <b>get_dataSamplesIntervalMs</b> ( )
java	int <b>get_dataSamplesIntervalMs</b> ( )
py	def <b>get_dataSamplesIntervalMs</b> ( )

Par défaut, l'enregistreur mémorise une mesure par seconde, mais la fréquence d'enregistrement peut être changée pour chaque fonction.

**Retourne :**

un entier positif correspondant au nombre de millisecondes entre deux mesures consécutives.

**datastream**→**get\_duration()****YDataStream****datastream**→**duration()**[datastream duration]

Retourne la durée approximative de cette séquence, en secondes.

js	function <b>get_duration</b> ( )
nodejs	function <b>get_duration</b> ( )
php	function <b>get_duration</b> ( )
cpp	int <b>get_duration</b> ( )
m	-(int) duration
pas	function <b>get_duration</b> ( ): LongInt
vb	function <b>get_duration</b> ( ) As Integer
cs	int <b>get_duration</b> ( )
java	int <b>get_duration</b> ( )
py	def <b>get_duration</b> ( )

**Retourne :**

le nombre de secondes couvertes par cette séquence.

En cas d'erreur, déclenche une exception ou retourne Y\_DURATION\_INVALID.

**datastream**→**get\_maxValue()****YDataStream****datastream**→**maxValue()**[datastream maxValue]

Retourne la plus grande valeur observée durant cette séquence.

js	function <b>get_maxValue</b> ( )
nodejs	function <b>get_maxValue</b> ( )
php	function <b>get_maxValue</b> ( )
cpp	double <b>get_maxValue</b> ( )
m	-(double) maxValue
pas	function <b>get_maxValue</b> ( ): double
vb	function <b>get_maxValue</b> ( ) As Double
cs	double <b>get_maxValue</b> ( )
java	double <b>get_maxValue</b> ( )
py	def <b>get_maxValue</b> ( )

Si le module utilise un firmware antérieur à la version 13000, cette méthode retournera toujours Y\_DATA\_INVALID.

**Retourne :**

un nombre décimal correspondant à la plus grande valeur, ou Y\_DATA\_INVALID si la séquence n'est pas encore terminée.

En cas d'erreur, déclenche une exception ou retourne Y\_DATA\_INVALID.

**datastream**→**get\_minValue()****YDataStream****datastream**→**minValue()**[datastream minValue]

Retourne la plus petite valeur observée durant cette séquence.

js	function <b>get_minValue</b> ( )
nodejs	function <b>get_minValue</b> ( )
php	function <b>get_minValue</b> ( )
cpp	double <b>get_minValue</b> ( )
m	-(double) minValue
pas	function <b>get_minValue</b> ( ): double
vb	function <b>get_minValue</b> ( ) As Double
cs	double <b>get_minValue</b> ( )
java	double <b>get_minValue</b> ( )
py	def <b>get_minValue</b> ( )

Si le module utilise un firmware antérieur à la version 13000, cette méthode retournera toujours Y\_DATA\_INVALID.

**Retourne :**

un nombre décimal correspondant à la plus petite valeur, ou Y\_DATA\_INVALID si la séquence n'est pas encore terminée.

En cas d'erreur, déclenche une exception ou retourne Y\_DATA\_INVALID.



**datastream→get\_rowCount()****YDataStream****datastream→rowCount()[datastream rowCount]**

Retourne le nombre d'enregistrement contenus dans la séquence.

js	function <b>get_rowCount</b> ( )
nodejs	function <b>get_rowCount</b> ( )
php	function <b>get_rowCount</b> ( )
cpp	int <b>get_rowCount</b> ( )
m	-(int) rowCount
pas	function <b>get_rowCount</b> ( ): LongInt
vb	function <b>get_rowCount</b> ( ) As Integer
cs	int <b>get_rowCount</b> ( )
java	int <b>get_rowCount</b> ( )
py	def <b>get_rowCount</b> ( )

Si le module utilise un firmware antérieur à la version 13000, cette méthode déclenche le chargement de toutes les données de la séquence si nécessaire, ce qui peut prendre un petit instant.

**Retourne :**

un entier positif correspondant au nombre d'enregistrements.

En cas d'erreur, déclenche une exception ou retourne zéro.

**datastream**→**get\_runIndex()****YDataStream****datastream**→**runIndex()**[datastream runIndex]

Retourne le numéro de Run de la séquence de données.

js	function <b>get_runIndex</b> ( )
nodejs	function <b>get_runIndex</b> ( )
php	function <b>get_runIndex</b> ( )
cpp	int <b>get_runIndex</b> ( )
m	-(int) runIndex
pas	function <b>get_runIndex</b> ( ): LongInt
vb	function <b>get_runIndex</b> ( ) As Integer
cs	int <b>get_runIndex</b> ( )
java	int <b>get_runIndex</b> ( )
py	def <b>get_runIndex</b> ( )

Un Run peut être composé de plusieurs séquences, couvrant différents intervalles de temps.

**Retourne :**

un entier positif correspondant au numéro du Run

**datastream→get\_startTime()****YDataStream****datastream→startTime()[datastream startTime]**

Retourne le temps de départ relatif de la séquence (en secondes).

js	function <b>get_startTime</b> ( )
nodejs	function <b>get_startTime</b> ( )
php	function <b>get_startTime</b> ( )
cpp	int <b>get_startTime</b> ( )
m	-(int) startTime
pas	function <b>get_startTime</b> ( ): LongInt
vb	function <b>get_startTime</b> ( ) As Integer
cs	int <b>get_startTime</b> ( )
java	int <b>get_startTime</b> ( )
py	def <b>get_startTime</b> ( )

Pour les firmwares récents, la valeur est relative à l'heure courante (valeur négative). Pour les modules utilisant un firmware plus ancien que la version 13000, la valeur est le nombre de secondes depuis la mise sous tension du module (valeur positive). Si vous désirez obtenir l'heure absolue du début de la séquence, utilisez `get_startTimeUTC()`.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le début du Run et le début de la séquence enregistrée.

**datastream**→**get\_startTimeUTC()****YDataStream****datastream**→**startTimeUTC()**[**datastream**  
**startTimeUTC**]

Retourne l'heure absolue du début de la séquence de données, sous forme du nombre de secondes depuis le 1er janvier 1970 (date/heure au format Unix).

js	function <b>get_startTimeUTC</b> ( )
nodejs	function <b>get_startTimeUTC</b> ( )
php	function <b>get_startTimeUTC</b> ( )
cpp	s64 <b>get_startTimeUTC</b> ( )
m	-(s64) <b>startTimeUTC</b>
pas	function <b>get_startTimeUTC</b> ( ): int64
vb	function <b>get_startTimeUTC</b> ( ) As Long
cs	long <b>get_startTimeUTC</b> ( )
java	long <b>get_startTimeUTC</b> ( )
py	def <b>get_startTimeUTC</b> ( )

Si l'heure UTC n'était pas configurée dans l'enregistreur de données au début de la séquence, cette méthode retourne 0.

**Retourne :**

un entier positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 et le début de la séquence enregistrée.

## 3.12. Interface de la fonction DigitalIO

La librairie de programmation Yoctopuce permet simplement de changer l'état de chaque bit du port d'entrée sortie. Il est possible de changer tous les bits du port à la fois, ou de les changer indépendamment. La librairie permet aussi de créer des courtes impulsions de durée déterminée. Le comportement électrique de chaque entrée/sortie peut être modifié (open drain et polarité inverse).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_digitalio.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YDigitalIO = yoctolib.YDigitalIO;
php	require_once('yocto_digitalio.php');
c++	#include "yocto_digitalio.h"
m	#import "yocto_digitalio.h"
pas	uses yocto_digitalio;
vb	yocto_digitalio.vb
cs	yocto_digitalio.cs
java	import com.yoctopuce.YoctoAPI.YDigitalIO;
py	from yocto_digitalio import *

### Fonction globales

#### yFindDigitalIO(func)

Permet de retrouver un port d'E/S digital d'après un identifiant donné.

#### yFirstDigitalIO()

Commence l'énumération des ports d'E/S digitaux accessibles par la librairie.

### Méthodes des objets YDigitalIO

#### digitalio→delayedPulse(bitno, ms\_delay, ms\_duration)

Préprogramme une impulsion de durée spécifiée sur un bit choisi.

#### digitalio→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du port d'E/S digital au format TYPE ( NAME ) = SERIAL . FUNCTIONID.

#### digitalio→get\_advertisedValue()

Retourne la valeur courante du port d'E/S digital (pas plus de 6 caractères).

#### digitalio→get\_bitDirection(bitno)

Retourne la direction d'un seul bit du port d'E/S.

#### digitalio→get\_bitOpenDrain(bitno)

Retourne la direction d'un seul bit du port d'E/S.

#### digitalio→get\_bitPolarity(bitno)

Retourne la polarité d'un seul bit du port d'E/S.

#### digitalio→get\_bitState(bitno)

Retourne l'état d'un seul bit du port d'E/S.

#### digitalio→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port d'E/S digital.

#### digitalio→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port d'E/S digital.

#### digitalio→get\_friendlyName()

Retourne un identifiant global du port d'E/S digital au format NOM\_MODULE . NOM\_FONCTION.

#### digitalio→get\_functionDescriptor()

	Retourne un identifiant unique de type <code>YFUN_DESCR</code> correspondant à la fonction.
<b>digitalio→get_functionId()</b>	Retourne l'identifiant matériel du port d'E/S digital, sans référence au module.
<b>digitalio→get_hardwareId()</b>	Retourne l'identifiant matériel unique du port d'E/S digital au format <code>SERIAL . FUNCTIONID</code> .
<b>digitalio→get_logicalName()</b>	Retourne le nom logique du port d'E/S digital.
<b>digitalio→get_module()</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>digitalio→get_module_async(callback, context)</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>digitalio→get_outputVoltage()</b>	Retourne la source de tension utilisée pour piloter les bits en sortie.
<b>digitalio→get_portDirection()</b>	Retourne la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.
<b>digitalio→get_portOpenDrain()</b>	Retourne le type d'interface électrique de chaque bit du port (bitmap).
<b>digitalio→get_portPolarity()</b>	Retourne la polarité des bits du port (bitmap).
<b>digitalio→get_portSize()</b>	Retourne le nombre de bits implémentés dans le port d'E/S.
<b>digitalio→get_portState()</b>	Retourne l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.
<b>digitalio→get_userData()</b>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<b>digitalio→isOnline()</b>	Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.
<b>digitalio→isOnline_async(callback, context)</b>	Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.
<b>digitalio→load(msValidity)</b>	Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.
<b>digitalio→load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.
<b>digitalio→nextDigitalIO()</b>	Continue l'énumération des ports d'E/S digitaux commencée à l'aide de <code>yFirstDigitalIO()</code> .
<b>digitalio→pulse(bitno, ms_duration)</b>	Déclenche une impulsion de durée spécifiée sur un bit choisi.
<b>digitalio→registerValueCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b>digitalio→set_bitDirection(bitno, bitdirection)</b>	Change la direction d'un seul bit du port d'E/S.
<b>digitalio→set_bitOpenDrain(bitno, opendrain)</b>	Change le type d'interface électrique d'un seul bit du port d'E/S.
<b>digitalio→set_bitPolarity(bitno, bitpolarity)</b>	Change la polarité d'un seul bit du port d'E/S.

**digitalio→set\_bitState(bitno, bitstate)**

Change l'état d'un seul bit du port d'E/S.

**digitalio→set\_logicalName(newval)**

Modifie le nom logique du port d'E/S digital.

**digitalio→set\_outputVoltage(newval)**

Modifie la source de tension utilisée pour piloter les bits en sortie.

**digitalio→set\_portDirection(newval)**

Modifie la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.

**digitalio→set\_portOpenDrain(newval)**

Modifie le type d'interface électrique de chaque bit du port (bitmap).

**digitalio→set\_portPolarity(newval)**

Modifie la polarité des bits du port (bitmap): Pour chaque bit à 0 l'entrée sortie correspondante fonctionne manière normale, pour chaque bit à 1 elle fonctionne de manière inversée.

**digitalio→set\_portState(newval)**

Modifie l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.

**digitalio→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**digitalio→toggle\_bitState(bitno)**

Inverse l'état d'un seul bit du port d'E/S.

**digitalio→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YDigitalIO.FindDigitalIO() yFindDigitalIO()yFindDigitalIO()

## YDigitalIO

Permet de retrouver un port d'E/S digital d'après un identifiant donné.

js	function <b>yFindDigitalIO</b> ( <b>func</b> )
nodejs	function <b>FindDigitalIO</b> ( <b>func</b> )
php	function <b>yFindDigitalIO</b> ( <b>\$func</b> )
cpp	YDigitalIO* <b>yFindDigitalIO</b> ( const string& <b>func</b> )
m	YDigitalIO* <b>yFindDigitalIO</b> ( NSString* <b>func</b> )
pas	function <b>yFindDigitalIO</b> ( <b>func</b> : string): TYDigitalIO
vb	function <b>yFindDigitalIO</b> ( ByVal <b>func</b> As String) As YDigitalIO
cs	YDigitalIO <b>FindDigitalIO</b> ( string <b>func</b> )
java	YDigitalIO <b>FindDigitalIO</b> ( String <b>func</b> )
py	def <b>FindDigitalIO</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port d'E/S digital soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YDigitalIO.isOnline()` pour tester si le port d'E/S digital est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le port d'E/S digital sans ambiguïté

### Retourne :

un objet de classe `YDigitalIO` qui permet ensuite de contrôler le port d'E/S digital.



## YDigitalIO.FirstDigitalIO() yFirstDigitalIO()yFirstDigitalIO()

## YDigitalIO

Commence l'énumération des ports d'E/S digitaux accessibles par la librairie.

js	function <b>yFirstDigitalIO</b> ( )
nodejs	function <b>FirstDigitalIO</b> ( )
php	function <b>yFirstDigitalIO</b> ( )
cpp	YDigitalIO* <b>yFirstDigitalIO</b> ( )
m	YDigitalIO* <b>yFirstDigitalIO</b> ( )
pas	function <b>yFirstDigitalIO</b> ( ): TYDigitalIO
vb	function <b>yFirstDigitalIO</b> ( ) As YDigitalIO
cs	YDigitalIO <b>FirstDigitalIO</b> ( )
java	YDigitalIO <b>FirstDigitalIO</b> ( )
py	def <b>FirstDigitalIO</b> ( )

Utiliser la fonction `YDigitalIO.nextDigitalIO( )` pour itérer sur les autres ports d'E/S digitaux.

### Retourne :

un pointeur sur un objet `YDigitalIO`, correspondant à le premier port d'E/S digital accessible en ligne, ou `null` si il n'y a pas de ports d'E/S digitaux disponibles.

Préprogramme une impulsion de durée spécifiée sur un bit choisi.

js	function <b>delayedPulse</b> ( <b>bitno</b> , <b>ms_delay</b> , <b>ms_duration</b> )
nodejs	function <b>delayedPulse</b> ( <b>bitno</b> , <b>ms_delay</b> , <b>ms_duration</b> )
php	function <b>delayedPulse</b> ( <b>\$bitno</b> , <b>\$ms_delay</b> , <b>\$ms_duration</b> )
c++	int <b>delayedPulse</b> ( int <b>bitno</b> , int <b>ms_delay</b> , int <b>ms_duration</b> )
m	-(int) <b>delayedPulse</b> : (int) <b>bitno</b> : (int) <b>ms_delay</b> : (int) <b>ms_duration</b>
pas	function <b>delayedPulse</b> ( <b>bitno</b> : LongInt, <b>ms_delay</b> : LongInt, <b>ms_duration</b> : LongInt): LongInt
vb	function <b>delayedPulse</b> ( ) As Integer
cs	int <b>delayedPulse</b> ( int <b>bitno</b> , int <b>ms_delay</b> , int <b>ms_duration</b> )
java	int <b>delayedPulse</b> ( int <b>bitno</b> , int <b>ms_delay</b> , int <b>ms_duration</b> )
py	def <b>delayedPulse</b> ( <b>bitno</b> , <b>ms_delay</b> , <b>ms_duration</b> )
cmd	YDigitalIO <b>target</b> <b>delayedPulse</b> <b>bitno</b> <b>ms_delay</b> <b>ms_duration</b>

Le bit va passer à 1 puis automatiquement revenir à 0 après le temps donné.

#### Paramètres :

**bitno**            index du bit dans le port; le bit de poids faible est à l'index 0

**ms\_delay**        délai d'attente avant l'impulsion, en millisecondes

**ms\_duration**    durée de l'impulsion désirée, en millisecondes. Notez que la résolution temporelle du module n'est pas garantie à la milliseconde.

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio→describe()[digitalio describe]****YDigitalIO**

Retourne un court texte décrivant de manière non-ambigüe l'instance du port d'E/S digital au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès a la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomeName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant le port d'E/S digital (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## digitalio→get\_advertisedValue() digitalio→advertisedValue()[digitalio advertisedValue]

YDigitalIO

Retourne la valeur courante du port d'E/S digital (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du port d'E/S digital (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**digitalio**→**get\_bitDirection()****YDigitalIO****digitalio**→**bitDirection()**[**digitalio bitDirection:** ]

Retourne la direction d'un seul bit du port d'E/S.

js	function <b>get_bitDirection</b> ( <b>bitno</b> )
nodejs	function <b>get_bitDirection</b> ( <b>bitno</b> )
php	function <b>get_bitDirection</b> ( <b>\$bitno</b> )
cpp	int <b>get_bitDirection</b> ( int <b>bitno</b> )
m	-(int) bitDirection : (int) <b>bitno</b>
pas	function <b>get_bitDirection</b> ( <b>bitno</b> : LongInt): LongInt
vb	function <b>get_bitDirection</b> ( ) As Integer
cs	int <b>get_bitDirection</b> ( int <b>bitno</b> )
java	int <b>get_bitDirection</b> ( int <b>bitno</b> )
py	def <b>get_bitDirection</b> ( <b>bitno</b> )
cmd	<b>YDigitalIO target</b> <b>get_bitDirection bitno</b>

(0 signifie que le bit est une entrée, 1 une sortie)

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**get\_bitOpenDrain()****YDigitalIO****digitalio**→**bitOpenDrain()**[**digitalio bitOpenDrain:** ]

Retourne la direction d'un seul bit du port d'E/S.

js	function <b>get_bitOpenDrain</b> ( <b>bitno</b> )
nodejs	function <b>get_bitOpenDrain</b> ( <b>bitno</b> )
php	function <b>get_bitOpenDrain</b> ( <b>\$bitno</b> )
cpp	int <b>get_bitOpenDrain</b> ( int <b>bitno</b> )
m	-(int) bitOpenDrain : (int) <b>bitno</b>
pas	function <b>get_bitOpenDrain</b> ( <b>bitno</b> : LongInt): LongInt
vb	function <b>get_bitOpenDrain</b> ( ) As Integer
cs	int <b>get_bitOpenDrain</b> ( int <b>bitno</b> )
java	int <b>get_bitOpenDrain</b> ( int <b>bitno</b> )
py	def <b>get_bitOpenDrain</b> ( <b>bitno</b> )
cmd	YDigitalIO <b>target</b> <b>get_bitOpenDrain</b> <b>bitno</b>

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**Retourne :**

0 représente une entrée ou une sortie digitale standard, 1 représente une entrée ou sortie en mode collecteur ouvert (drain ouvert)..

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**get\_bitPolarity()****YDigitalIO****digitalio**→**bitPolarity()**[**digitalio bitPolarity:** ]

Retourne la polarité d'un seul bit du port d'E/S.

js	function <b>get_bitPolarity</b> ( <b>bitno</b> )
nodejs	function <b>get_bitPolarity</b> ( <b>bitno</b> )
php	function <b>get_bitPolarity</b> ( <b>\$bitno</b> )
cpp	int <b>get_bitPolarity</b> ( int <b>bitno</b> )
m	-(int) bitPolarity : (int) <b>bitno</b>
pas	function <b>get_bitPolarity</b> ( <b>bitno</b> : LongInt): LongInt
vb	function <b>get_bitPolarity</b> ( ) As Integer
cs	int <b>get_bitPolarity</b> ( int <b>bitno</b> )
java	int <b>get_bitPolarity</b> ( int <b>bitno</b> )
py	def <b>get_bitPolarity</b> ( <b>bitno</b> )
cmd	YDigitalIO <b>target</b> <b>get_bitPolarity</b> <b>bitno</b>

0 signifie que l'entrée sortie est en mode normal, 1 qu'elle est en mode inverse

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**get\_bitState()****digitalio**→**bitState()**[**digitalio bitState:** ]

Retourne l'état d'un seul bit du port d'E/S.

js	function <b>get_bitState</b> ( <b>bitno</b> )
nodejs	function <b>get_bitState</b> ( <b>bitno</b> )
php	function <b>get_bitState</b> ( <b>\$bitno</b> )
cpp	int <b>get_bitState</b> ( int <b>bitno</b> )
m	-(int) bitState : (int) <b>bitno</b>
pas	function <b>get_bitState</b> ( <b>bitno</b> : LongInt): LongInt
vb	function <b>get_bitState</b> ( ) As Integer
cs	int <b>get_bitState</b> ( int <b>bitno</b> )
java	int <b>get_bitState</b> ( int <b>bitno</b> )
py	def <b>get_bitState</b> ( <b>bitno</b> )
cmd	YDigitalIO <b>target</b> <b>get_bitState</b> <b>bitno</b>

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**Retourne :**

l'état du bit (0 ou 1).

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**digitalio**→**get\_errorMessage()****YDigitalIO****digitalio**→**errorMessage()**[digitalio errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port d'E/S digital.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du port d'E/S digital.

**digitalio**→**get\_errorType()****YDigitalIO****digitalio**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port d'E/S digital.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du port d'E/S digital.

**digitalio**→**get\_friendlyName()****YDigitalIO****digitalio**→**friendlyName()**[digitalio friendlyName]

Retourne un identifiant global du port d'E/S digital au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du port d'E/S digital si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du port d'E/S digital (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le port d'E/S digital en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **digitalio→get\_functionDescriptor()** **digitalio→functionDescriptor()[digitalio** **functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**digitalio**→**get\_functionId()****YDigitalIO****digitalio**→**functionId()**[digitalio functionId]

Retourne l'identifiant matériel du port d'E/S digital, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le port d'E/S digital (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**digitalio**→**get\_hardwareId()****YDigitalIO****digitalio**→**hardwareId()**[**digitalio hardwareId**]

Retourne l'identifiant matériel unique du port d'E/S digital au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port d'E/S digital (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le port d'E/S digital (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**digitalio**→**get\_logicalName()****YDigitalIO****digitalio**→**logicalName()**[digitalio logicalName]

Retourne le nom logique du port d'E/S digital.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du port d'E/S digital. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**digitalio**→**get\_module()****digitalio**→**module()**[digitalio module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule



**digitalio**→**get\_module\_async()****YDigitalIO****digitalio**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**digitalio→get\_outputVoltage()****YDigitalIO****digitalio→outputVoltage()[digitalio outputVoltage]**

Retourne la source de tension utilisée pour piloter les bits en sortie.

js	function <b>get_outputVoltage</b> ( )
nodejs	function <b>get_outputVoltage</b> ( )
php	function <b>get_outputVoltage</b> ( )
cpp	Y_OUTPUTVOLTAGE_enum <b>get_outputVoltage</b> ( )
m	-(Y_OUTPUTVOLTAGE_enum) outputVoltage
pas	function <b>get_outputVoltage</b> ( ): Integer
vb	function <b>get_outputVoltage</b> ( ) As Integer
cs	int <b>get_outputVoltage</b> ( )
java	int <b>get_outputVoltage</b> ( )
py	def <b>get_outputVoltage</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_outputVoltage</b>

**Retourne :**

une valeur parmi Y\_OUTPUTVOLTAGE\_USB\_5V, Y\_OUTPUTVOLTAGE\_USB\_3V et Y\_OUTPUTVOLTAGE\_EXT\_V représentant la source de tension utilisée pour piloter les bits en sortie

En cas d'erreur, déclenche une exception ou retourne Y\_OUTPUTVOLTAGE\_INVALID.

**digitalio**→**get\_portDirection()****YDigitalIO****digitalio**→**portDirection()[digitalio portDirection]**

Retourne la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.

js	function <b>get_portDirection</b> ( )
nodejs	function <b>get_portDirection</b> ( )
php	function <b>get_portDirection</b> ( )
cpp	int <b>get_portDirection</b> ( )
m	-(int) portDirection
pas	function <b>get_portDirection</b> ( ): LongInt
vb	function <b>get_portDirection</b> ( ) As Integer
cs	int <b>get_portDirection</b> ( )
java	int <b>get_portDirection</b> ( )
py	def <b>get_portDirection</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_portDirection</b>

**Retourne :**

un entier représentant la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie

En cas d'erreur, déclenche une exception ou retourne Y\_PORTDIRECTION\_INVALID.

**digitalio**→**get\_portOpenDrain()****YDigitalIO****digitalio**→**portOpenDrain()[digitalio portOpenDrain]**

Retourne le type d'interface électrique de chaque bit du port (bitmap).

js	function <b>get_portOpenDrain</b> ( )
nodejs	function <b>get_portOpenDrain</b> ( )
php	function <b>get_portOpenDrain</b> ( )
cpp	int <b>get_portOpenDrain</b> ( )
m	-(int) portOpenDrain
pas	function <b>get_portOpenDrain</b> ( ): LongInt
vb	function <b>get_portOpenDrain</b> ( ) As Integer
cs	int <b>get_portOpenDrain</b> ( )
java	int <b>get_portOpenDrain</b> ( )
py	def <b>get_portOpenDrain</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_portOpenDrain</b>

0 représente une entrée ou une sortie digitale standard, 1 représente une entrée ou sortie en mode collecteur ouvert (drain ouvert).

**Retourne :**

un entier représentant le type d'interface électrique de chaque bit du port (bitmap)

En cas d'erreur, déclenche une exception ou retourne Y\_PORTOPENDRAIN\_INVALID.

**digitalio**→**get\_portPolarity()****YDigitalIO****digitalio**→**portPolarity()**[**digitalio portPolarity**]

Retourne la polarité des bits du port (bitmap).

js	function <b>get_portPolarity</b> ( )
nodejs	function <b>get_portPolarity</b> ( )
php	function <b>get_portPolarity</b> ( )
cpp	int <b>get_portPolarity</b> ( )
m	-(int) portPolarity
pas	function <b>get_portPolarity</b> ( ): LongInt
vb	function <b>get_portPolarity</b> ( ) As Integer
cs	int <b>get_portPolarity</b> ( )
java	int <b>get_portPolarity</b> ( )
py	def <b>get_portPolarity</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_portPolarity</b>

Pour chaque bit à 0 l'entrée sortie correspondante fonctionne manière normale, pour chaque bit à 1 elle fonctionne ne manière inversée.

**Retourne :**

un entier représentant la polarité des bits du port (bitmap)

En cas d'erreur, déclenche une exception ou retourne Y\_PORTPOLARITY\_INVALID.

**digitalio**→**get\_portSize()****YDigitalIO****digitalio**→**portSize()**[**digitalio portSize**]

Retourne le nombre de bits implémentés dans le port d'E/S.

js	function <b>get_portSize</b> ( )
nodejs	function <b>get_portSize</b> ( )
php	function <b>get_portSize</b> ( )
cpp	int <b>get_portSize</b> ( )
m	-(int) portSize
pas	function <b>get_portSize</b> ( ): LongInt
vb	function <b>get_portSize</b> ( ) As Integer
cs	int <b>get_portSize</b> ( )
java	int <b>get_portSize</b> ( )
py	def <b>get_portSize</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_portSize</b>

**Retourne :**

un entier représentant le nombre de bits implémentés dans le port d'E/S

En cas d'erreur, déclenche une exception ou retourne Y\_PORTSIZE\_INVALID.

**digitalio**→**get\_portState()****YDigitalIO****digitalio**→**portState()[digitalio portState]**

Retourne l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.

js	function <b>get_portState</b> ( )
nodejs	function <b>get_portState</b> ( )
php	function <b>get_portState</b> ( )
cpp	int <b>get_portState</b> ( )
m	-(int) portState
pas	function <b>get_portState</b> ( ): LongInt
vb	function <b>get_portState</b> ( ) As Integer
cs	int <b>get_portState</b> ( )
java	int <b>get_portState</b> ( )
py	def <b>get_portState</b> ( )
cmd	YDigitalIO <b>target</b> <b>get_portState</b>

**Retourne :**

un entier représentant l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite

En cas d'erreur, déclenche une exception ou retourne Y\_PORTSTATE\_INVALID.

**digitalio**→**get\_userdata()****digitalio**→**userData()**[digitalio userData]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userdata`.

<code>js</code>	<code>function <b>get_userdata</b>( )</code>
<code>nodejs</code>	<code>function <b>get_userdata</b>( )</code>
<code>php</code>	<code>function <b>get_userdata</b>( )</code>
<code>cpp</code>	<code>void * <b>get_userdata</b>( )</code>
<code>m</code>	<code>-(void*) userData</code>
<code>pas</code>	<code>function <b>get_userdata</b>( ): Tobject</code>
<code>vb</code>	<code>function <b>get_userdata</b>( ) As Object</code>
<code>cs</code>	<code>object <b>get_userdata</b>( )</code>
<code>java</code>	<code>Object <b>get_userdata</b>( )</code>
<code>py</code>	<code>def <b>get_userdata</b>( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.



**digitalio→isOnline()[digitalio isOnline]****YDigitalIO**

Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du port d'E/S digital sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le port d'E/S digital est joignable, `false` sinon

Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du port d'E/S digital sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**digitalio→load()[digitalio load: ]****YDigitalIO**

Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**digitalio**→**nextDigitalIO()**[**digitalio** **nextDigitalIO**]**YDigitalIO**

Continue l'énumération des ports d'E/S digitaux commencée à l'aide de `yFirstDigitalIO()`.

js	function <b>nextDigitalIO</b> ( )
nodejs	function <b>nextDigitalIO</b> ( )
php	function <b>nextDigitalIO</b> ( )
cpp	YDigitalIO * <b>nextDigitalIO</b> ( )
m	-(YDigitalIO*) <b>nextDigitalIO</b>
pas	function <b>nextDigitalIO</b> ( ): TYDigitalIO
vb	function <b>nextDigitalIO</b> ( ) As YDigitalIO
cs	YDigitalIO <b>nextDigitalIO</b> ( )
java	YDigitalIO <b>nextDigitalIO</b> ( )
py	def <b>nextDigitalIO</b> ( )

**Retourne :**

un pointeur sur un objet `YDigitalIO` accessible en ligne, ou `null` lorsque l'énumération est terminée.

Déclenche une impulsion de durée spécifiée sur un bit choisi.

js	function pulse( bitno, ms_duration)
nodejs	function pulse( bitno, ms_duration)
php	function pulse( \$bitno, \$ms_duration)
cpp	int pulse( int bitno, int ms_duration)
m	-(int) pulse : (int) bitno : (int) ms_duration
pas	function pulse( bitno: LongInt, ms_duration: LongInt): LongInt
vb	function pulse( ) As Integer
cs	int pulse( int bitno, int ms_duration)
java	int pulse( int bitno, int ms_duration)
py	def pulse( bitno, ms_duration)
cmd	YDigitalIO target pulse bitno ms_duration

Le bit va passer à 1 puis automatiquement revenir à 0 après le temps donné.

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**ms\_duration** durée de l'impulsion désirée, en millisecondes. Notez que la résolution temporelle du module n'est pas garantie à la milliseconde.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## digitalio→registerValueCallback()[digitalio registerValueCallback: ]

YDigitalIO

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YDigitalIOValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YDigitalIOValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYDigitalIOValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**digitalio→set\_bitDirection()****digitalio→setBitDirection() [digitalio setBitDirection: ]**

Change la direction d'un seul bit du port d'E/S.

js	function <b>set_bitDirection</b> ( <b>bitno</b> , <b>bitdirection</b> )
nodejs	function <b>set_bitDirection</b> ( <b>bitno</b> , <b>bitdirection</b> )
php	function <b>set_bitDirection</b> ( <b>\$bitno</b> , <b>\$bitdirection</b> )
cpp	int <b>set_bitDirection</b> ( int <b>bitno</b> , int <b>bitdirection</b> )
m	-(int) setBitDirection : (int) <b>bitno</b> : (int) <b>bitdirection</b>
pas	function <b>set_bitDirection</b> ( <b>bitno</b> : LongInt, <b>bitdirection</b> : LongInt): LongInt
vb	function <b>set_bitDirection</b> ( ) As Integer
cs	int <b>set_bitDirection</b> ( int <b>bitno</b> , int <b>bitdirection</b> )
java	int <b>set_bitDirection</b> ( int <b>bitno</b> , int <b>bitdirection</b> )
py	def <b>set_bitDirection</b> ( <b>bitno</b> , <b>bitdirection</b> )
cmd	YDigitalIO <b>target set_bitDirection bitno bitdirection</b>

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**bitdirection** nouvelle valeur de la direction, 0=entrée, 1=sortie. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé après un redémarrage du module.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## digitalio→set\_bitOpenDrain() digitalio→setBitOpenDrain()[digitalio setBitOpenDrain: ]

YDigitalIO

Change le type d'interface électrique d'un seul bit du port d'E/S.

js	function <b>set_bitOpenDrain</b> ( <b>bitno</b> , <b>opendrain</b> )
nodejs	function <b>set_bitOpenDrain</b> ( <b>bitno</b> , <b>opendrain</b> )
php	function <b>set_bitOpenDrain</b> ( <b>\$bitno</b> , <b>\$opendrain</b> )
cpp	int <b>set_bitOpenDrain</b> ( int <b>bitno</b> , int <b>opendrain</b> )
m	-(int) setBitOpenDrain : (int) <b>bitno</b> : (int) <b>opendrain</b>
pas	function <b>set_bitOpenDrain</b> ( <b>bitno</b> : LongInt, <b>opendrain</b> : LongInt): LongInt
vb	function <b>set_bitOpenDrain</b> ( ) As Integer
cs	int <b>set_bitOpenDrain</b> ( int <b>bitno</b> , int <b>opendrain</b> )
java	int <b>set_bitOpenDrain</b> ( int <b>bitno</b> , int <b>opendrain</b> )
py	def <b>set_bitOpenDrain</b> ( <b>bitno</b> , <b>opendrain</b> )
cmd	YDigitalIO <b>target</b> <b>set_bitOpenDrain</b> <b>bitno</b> <b>opendrain</b>

### Paramètres :

- bitno** index du bit dans le port; le bit de poids faible est à l'index 0
- opendrain** 0 pour faire une entrée ou une sortie digitale standard, 1 pour une entrée ou sortie en mode collecteur ouvert (drain ouvert). N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé après un redémarrage du module.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**set\_bitPolarity()****YDigitalIO****digitalio**→**setBitPolarity()**[**digitalio setBitPolarity:** ]

Change la polarité d'un seul bit du port d'E/S.

js	function <b>set_bitPolarity</b> ( <b>bitno</b> , <b>bitpolarity</b> )
nodejs	function <b>set_bitPolarity</b> ( <b>bitno</b> , <b>bitpolarity</b> )
php	function <b>set_bitPolarity</b> ( <b>\$bitno</b> , <b>\$bitpolarity</b> )
cpp	int <b>set_bitPolarity</b> ( int <b>bitno</b> , int <b>bitpolarity</b> )
m	-(int) setBitPolarity : (int) <b>bitno</b> : (int) <b>bitpolarity</b>
pas	function <b>set_bitPolarity</b> ( <b>bitno</b> : LongInt, <b>bitpolarity</b> : LongInt): LongInt
vb	function <b>set_bitPolarity</b> ( ) As Integer
cs	int <b>set_bitPolarity</b> ( int <b>bitno</b> , int <b>bitpolarity</b> )
java	int <b>set_bitPolarity</b> ( int <b>bitno</b> , int <b>bitpolarity</b> )
py	def <b>set_bitPolarity</b> ( <b>bitno</b> , <b>bitpolarity</b> )
cmd	YDigitalIO <b>target set_bitPolarity bitno bitpolarity</b>

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**bitpolarity** nouvelle valeur de la polarité. 0=mode normal, 1=mode inverse. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé après un redémarrage du module.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**set\_bitState()****YDigitalIO****digitalio**→**setBitState()**[**digitalio setBitState:** ]

Change l'état d'un seul bit du port d'E/S.

js	function <b>set_bitState</b> ( <b>bitno</b> , <b>bitstate</b> )
nodejs	function <b>set_bitState</b> ( <b>bitno</b> , <b>bitstate</b> )
php	function <b>set_bitState</b> ( <b>\$bitno</b> , <b>\$bitstate</b> )
cpp	int <b>set_bitState</b> ( int <b>bitno</b> , int <b>bitstate</b> )
m	-(int) setBitState : (int) <b>bitno</b> : (int) <b>bitstate</b>
pas	function <b>set_bitState</b> ( <b>bitno</b> : LongInt, <b>bitstate</b> : LongInt): LongInt
vb	function <b>set_bitState</b> ( ) As Integer
cs	int <b>set_bitState</b> ( int <b>bitno</b> , int <b>bitstate</b> )
java	int <b>set_bitState</b> ( int <b>bitno</b> , int <b>bitstate</b> )
py	def <b>set_bitState</b> ( <b>bitno</b> , <b>bitstate</b> )
cmd	YDigitalIO <b>target</b> <b>set_bitState</b> <b>bitno</b> <b>bitstate</b>

**Paramètres :**

**bitno** index du bit dans le port; le bit de poids faible est à l'index 0

**bitstate** nouvel état du bit (1 ou 0)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**set\_logicalName()****YDigitalIO****digitalio**→**setLogicalName()**[**digitalio**  
**setLogicalName:** ]

Modifie le nom logique du port d'E/S digital.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du port d'E/S digital.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## digitalio→set\_outputVoltage() digitalio→setOutputVoltage()[digitalio setOutputVoltage: ]

YDigitalIO

Modifie la source de tension utilisée pour piloter les bits en sortie.

js	function <b>set_outputVoltage</b> ( <b>newval</b> )
nodejs	function <b>set_outputVoltage</b> ( <b>newval</b> )
php	function <b>set_outputVoltage</b> ( <b>\$newval</b> )
cpp	int <b>set_outputVoltage</b> ( Y_OUTPUTVOLTAGE_enum <b>newval</b> )
m	-(int) setOutputVoltage : (Y_OUTPUTVOLTAGE_enum) <b>newval</b>
pas	function <b>set_outputVoltage</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_outputVoltage</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_outputVoltage</b> ( int <b>newval</b> )
java	int <b>set_outputVoltage</b> ( int <b>newval</b> )
py	def <b>set_outputVoltage</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target</b> <b>set_outputVoltage</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé après un redémarrage du module.

### Paramètres :

**newval** une valeur parmi Y\_OUTPUTVOLTAGE\_USB\_5V, Y\_OUTPUTVOLTAGE\_USB\_3V et Y\_OUTPUTVOLTAGE\_EXT\_V représentant la source de tension utilisée pour piloter les bits en sortie

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## digitalio→set\_portDirection() digitalio→setPortDirection()[digitalio setPortDirection: ]

Modifie la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.

js	function <b>set_portDirection</b> ( <b>newval</b> )
nodejs	function <b>set_portDirection</b> ( <b>newval</b> )
php	function <b>set_portDirection</b> ( <b>\$newval</b> )
cpp	int <b>set_portDirection</b> ( int <b>newval</b> )
m	-(int) setPortDirection : (int) <b>newval</b>
pas	function <b>set_portDirection</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_portDirection</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_portDirection</b> ( int <b>newval</b> )
java	int <b>set_portDirection</b> ( int <b>newval</b> )
py	def <b>set_portDirection</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target</b> <b>set_portDirection</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** un entier représentant la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio→set\_portOpenDrain()****YDigitalIO****digitalio→setPortOpenDrain()[digitalio****setPortOpenDrain: ]**

Modifie le type d'interface électrique de chaque bit du port (bitmap).

js	function <b>set_portOpenDrain</b> ( <b>newval</b> )
nodejs	function <b>set_portOpenDrain</b> ( <b>newval</b> )
php	function <b>set_portOpenDrain</b> ( <b>\$newval</b> )
cpp	int <b>set_portOpenDrain</b> ( int <b>newval</b> )
m	-(int) setPortOpenDrain : (int) <b>newval</b>
pas	function <b>set_portOpenDrain</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_portOpenDrain</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_portOpenDrain</b> ( int <b>newval</b> )
java	int <b>set_portOpenDrain</b> ( int <b>newval</b> )
py	def <b>set_portOpenDrain</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target set_portOpenDrain newval</b>

0 représente une entrée ou une sortie digitale standard, 1 représente une entrée ou sortie en mode collecteur ouvert (drain ouvert). N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant le type d'interface électrique de chaque bit du port (bitmap)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**set\_portPolarity()****YDigitalIO****digitalio**→**setPortPolarity()**[**digitalio** **setPortPolarity:** ]

Modifie la polarité des bits du port (bitmap): Pour chaque bit à 0 l'entrée sortie correspondante fonctionne manière normale, pour chaque bit à 1 elle fonctionne ne manière inversée.

js	function <b>set_portPolarity</b> ( <b>newval</b> )
nodejs	function <b>set_portPolarity</b> ( <b>newval</b> )
php	function <b>set_portPolarity</b> ( <b>\$newval</b> )
cpp	int <b>set_portPolarity</b> ( int <b>newval</b> )
m	-(int) setPortPolarity : (int) <b>newval</b>
pas	function <b>set_portPolarity</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_portPolarity</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_portPolarity</b> ( int <b>newval</b> )
java	int <b>set_portPolarity</b> ( int <b>newval</b> )
py	def <b>set_portPolarity</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target</b> <b>set_portPolarity</b> <b>newval</b>

**Paramètres :**

**newval** un entier représentant la polarité des bits du port (bitmap): Pour chaque bit à 0 l'entrée sortie correspondante fonctionne manière normale, pour chaque bit à 1 elle fonctionne ne manière inversée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**digitalio→set\_portState()****YDigitalIO****digitalio→setPortState()[digitalio setPortState: ]**

Modifie l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.

js	function <b>set_portState</b> ( <b>newval</b> )
nodejs	function <b>set_portState</b> ( <b>newval</b> )
php	function <b>set_portState</b> ( <b>\$newval</b> )
cpp	int <b>set_portState</b> ( int <b>newval</b> )
m	-(int) setPortState : (int) <b>newval</b>
pas	function <b>set_portState</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_portState</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_portState</b> ( int <b>newval</b> )
java	int <b>set_portState</b> ( int <b>newval</b> )
py	def <b>set_portState</b> ( <b>newval</b> )
cmd	YDigitalIO <b>target</b> <b>set_portState</b> <b>newval</b>

Seuls les bits configurés en sortie dans `portDirection` sont affectés.

**Paramètres :**

**newval** un entier représentant l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**set\_userData()****digitalio**→**setUserData()**[**digitalio setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function <b>set_userData</b> ( <b>data</b> )
nodejs	function <b>set_userData</b> ( <b>data</b> )
php	function <b>set_userData</b> ( <b>\$data</b> )
cpp	void <b>set_userData</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userData</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userData</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userData</b> ( object <b>data</b> )
java	void <b>set_userData</b> ( Object <b>data</b> )
py	def <b>set_userData</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**digitalio→toggle\_bitState()[digitalio toggle\_bitState: ]****YDigitalIO**

Inverse l'état d'un seul bit du port d'E/S.

js	function <b>toggle_bitState</b> ( <b>bitno</b> )
nodejs	function <b>toggle_bitState</b> ( <b>bitno</b> )
php	function <b>toggle_bitState</b> ( <b>\$bitno</b> )
cpp	int <b>toggle_bitState</b> ( int <b>bitno</b> )
m	-(int) <b>toggle_bitState</b> : (int) <b>bitno</b>
pas	function <b>toggle_bitState</b> ( <b>bitno</b> : LongInt): LongInt
vb	function <b>toggle_bitState</b> ( ) As Integer
cs	int <b>toggle_bitState</b> ( int <b>bitno</b> )
java	int <b>toggle_bitState</b> ( int <b>bitno</b> )
py	def <b>toggle_bitState</b> ( <b>bitno</b> )
cmd	YDigitalIO <b>target toggle_bitState bitno</b>

**Paramètres :****bitno** index du bit dans le port; le bit de poids faible est à l'index 0**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**digitalio**→**wait\_async()****YDigitalIO**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

### 3.13. Interface de la fonction Display

L'interface de contrôle des écrans Yoctopuce est conçue pour afficher facilement des informations et des images. Le module est capable de gérer seul la superposition de plusieurs couches graphiques, qui peuvent être dessinées individuellement, sans affichage immédiat, puis librement positionnées sur l'écran. Il est aussi capable de rejouer des séquences de commandes pré-enregistrées (animations).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_display.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YDisplay = yoctolib.YDisplay;
php	require_once('yocto_display.php');
c++	#include "yocto_display.h"
m	#import "yocto_display.h"
pas	uses yocto_display;
vb	yocto_display.vb
cs	yocto_display.cs
java	import com.yoctopuce.YoctoAPI.YDisplay;
py	from yocto_display import *

#### Fonction globales

##### yFindDisplay(func)

Permet de retrouver un écran d'après un identifiant donné.

##### yFirstDisplay()

Commence l'énumération des écran accessibles par la librairie.

#### Méthodes des objets YDisplay

##### display→copyLayerContent(srcLayerId, dstLayerId)

Copie le contenu d'un couche d'affichage vers une autre couche.

##### display→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'écran au format TYPE (NAME) = SERIAL . FUNCTIONID.

##### display→fade(brightness, duration)

Change la luminosité de l'écran en douceur, pour produire un effet de fade-in ou fade-out.

##### display→get\_advertisedValue()

Retourne la valeur courante de l'écran (pas plus de 6 caractères).

##### display→get\_brightness()

Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).

##### display→get\_displayHeight()

Retourne la hauteur de l'écran, en pixels.

##### display→get\_displayLayer(layerId)

Retourne un objet YDisplayLayer utilisable pour dessiner sur la couche d'affichage correspondante.

##### display→get\_displayType()

Retourne le type de l'écran: monochrome, niveaux de gris ou couleur.

##### display→get\_displayWidth()

Retourne la largeur de l'écran, en pixels.

##### display→get\_enabled()

Retourne vrai si le l'écran est alimenté, faux sinon.

##### display→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'écran.

#### **display→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'écran.

#### **display→get\_friendlyName()**

Retourne un identifiant global de l'écran au format `NOM_MODULE . NOM_FONCTION`.

#### **display→get\_functionDescriptor()**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **display→get\_functionId()**

Retourne l'identifiant matériel de l'écran, sans référence au module.

#### **display→get\_hardwareId()**

Retourne l'identifiant matériel unique de l'écran au format `SERIAL . FUNCTIONID`.

#### **display→get\_layerCount()**

Retourne le nombre des couches affichables disponibles.

#### **display→get\_layerHeight()**

Retourne la hauteur des couches affichables, en pixels.

#### **display→get\_layerWidth()**

Retourne la largeur des couches affichables, en pixels.

#### **display→get\_logicalName()**

Retourne le nom logique de l'écran.

#### **display→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **display→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **display→get\_orientation()**

Retourne l'orientation sélectionnée pour l'écran.

#### **display→get\_startupSeq()**

Retourne le nom de la séquence à jouer à la mise sous tension de l'écran.

#### **display→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **display→isOnline()**

Vérifie si le module hébergeant l'écran est joignable, sans déclencher d'erreur.

#### **display→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'écran est joignable, sans déclencher d'erreur.

#### **display→load(msValidity)**

Met en cache les valeurs courantes de l'écran, avec une durée de validité spécifiée.

#### **display→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'écran, avec une durée de validité spécifiée.

#### **display→newSequence()**

Enclanche l'enregistrement de toutes les commandes d'affichage suivantes dans une séquence, qui pourra être rejouée ultérieurement.

#### **display→nextDisplay()**

Continue l'énumération des écran commencée à l'aide de `yFirstDisplay()`.

#### **display→pauseSequence(delay\_ms)**

Attend pour la durée spécifiée (en millisecondes) avant de jouer les commandes suivantes de la séquence active.

**display→playSequence(sequenceName)**

Joue une séquence d'affichage préalablement enregistrée à l'aide des méthodes `newSequence()` et `saveSequence()`.

**display→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**display→resetAll()**

Efface le contenu de l'écran et remet toutes les couches à leur état initial.

**display→saveSequence(sequenceName)**

Termine l'enregistrement d'une séquence et la sauvegarde sur la mémoire interne de l'écran, sous le nom choisi.

**display→set\_brightness(newval)**

Modifie la luminosité de l'écran.

**display→set\_enabled(newval)**

Modifie l'état d'activité de l'écran.

**display→set\_logicalName(newval)**

Modifie le nom logique de l'écran.

**display→set\_orientation(newval)**

Modifie l'orientation de l'écran.

**display→set\_startupSeq(newval)**

Modifie le nom de la séquence à jouer à la mise sous tension de l'écran.

**display→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**display→stopSequence(sequenceName)**

Arrête immédiatement la séquence d'affichage actuellement jouée sur l'écran.

**display→swapLayerContent(layerIdA, layerIdB)**

Permute le contenu de deux couches d'affichage.

**display→upload(pathname, content)**

Télécharge un contenu arbitraire (par exemple une image GIF) vers le système de fichier de l'écran, au chemin d'accès spécifié.

**display→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YDisplay.FindDisplay() yFindDisplay()yFindDisplay()

## YDisplay

Permet de retrouver un ecran d'après un identifiant donné.

js	function <b>yFindDisplay</b> ( <b>func</b> )
nodejs	function <b>FindDisplay</b> ( <b>func</b> )
php	function <b>yFindDisplay</b> ( <b>\$func</b> )
c++	YDisplay* <b>yFindDisplay</b> ( string <b>func</b> )
m	+(YDisplay*) <b>yFindDisplay</b> : (NSString*) <b>func</b>
pas	function <b>yFindDisplay</b> ( <b>func</b> : string): TYDisplay
vb	function <b>yFindDisplay</b> ( ByVal <b>func</b> As String) As YDisplay
cs	YDisplay <b>FindDisplay</b> ( string <b>func</b> )
java	YDisplay <b>FindDisplay</b> ( String <b>func</b> )
py	def <b>FindDisplay</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'ecran soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YDisplay.isOnline()` pour tester si l'ecran est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'ecran sans ambiguïté

### Retourne :

un objet de classe `YDisplay` qui permet ensuite de contrôler l'ecran.



## YDisplay.FirstDisplay() yFirstDisplay()yFirstDisplay()

## YDisplay

Commence l'énumération des écran accessibles par la librairie.

js	function <b>yFirstDisplay</b> ( )
nodejs	function <b>FirstDisplay</b> ( )
php	function <b>yFirstDisplay</b> ( )
cpp	YDisplay* <b>yFirstDisplay</b> ( )
m	YDisplay* <b>yFirstDisplay</b> ( )
pas	function <b>yFirstDisplay</b> ( ): TYDisplay
vb	function <b>yFirstDisplay</b> ( ) As YDisplay
cs	YDisplay <b>FirstDisplay</b> ( )
java	YDisplay <b>FirstDisplay</b> ( )
py	def <b>FirstDisplay</b> ( )

Utiliser la fonction `YDisplay.nextDisplay( )` pour itérer sur les autres écran.

### Retourne :

un pointeur sur un objet `YDisplay`, correspondant à le premier écran accessible en ligne, ou `null` si il n'y a pas de écran disponibles.

Copie le contenu d'un couche d'affichage vers une autre couche.

js	function <b>copyLayerContent</b> ( <b>srcLayerId</b> , <b>dstLayerId</b> )
nodejs	function <b>copyLayerContent</b> ( <b>srcLayerId</b> , <b>dstLayerId</b> )
php	function <b>copyLayerContent</b> ( <b>\$srcLayerId</b> , <b>\$dstLayerId</b> )
c++	int <b>copyLayerContent</b> ( int <b>srcLayerId</b> , int <b>dstLayerId</b> )
m	-(int) <b>copyLayerContent</b> : (int) <b>srcLayerId</b> : (int) <b>dstLayerId</b>
pas	function <b>copyLayerContent</b> ( <b>srcLayerId</b> : LongInt, <b>dstLayerId</b> : LongInt): LongInt
vb	function <b>copyLayerContent</b> ( ) As Integer
cs	int <b>copyLayerContent</b> ( int <b>srcLayerId</b> , int <b>dstLayerId</b> )
java	int <b>copyLayerContent</b> ( int <b>srcLayerId</b> , int <b>dstLayerId</b> )
py	def <b>copyLayerContent</b> ( <b>srcLayerId</b> , <b>dstLayerId</b> )
cmd	YDisplay <b>target copyLayerContent</b> <b>srcLayerId</b> <b>dstLayerId</b>

La couleur et la transparence de tous les pixels de la couche de destination sont changés pour correspondre à la couche source. Cette méthode modifie le contenu affiché, mais n'a aucun effet sur les propriétés de l'objet layer lui-même. Notez que la couche zéro n'a pas de transparence (elle est toujours opaque).

### Paramètres :

**srcLayerId** l'identifiant de la couche d'origine (un chiffre parmi 0..layerCount-1)

**dstLayerId** l'identifiant de la couche de destination (un chiffre parmi 0..layerCount-1)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→describe()[display describe]****YDisplay**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'ecran au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès a la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomeName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant l'ecran (ex: `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**display→fade()[display fade: ]****YDisplay**

Change la luminosité de l'écran en douceur, pour produire un effet de fade-in ou fade-out.

js	function <b>fade</b> ( <b>brightness</b> , <b>duration</b> )
nodejs	function <b>fade</b> ( <b>brightness</b> , <b>duration</b> )
php	function <b>fade</b> ( <b>\$brightness</b> , <b>\$duration</b> )
cpp	int <b>fade</b> ( int <b>brightness</b> , int <b>duration</b> )
m	-(int) <b>fade</b> : (int) <b>brightness</b> : (int) <b>duration</b>
pas	function <b>fade</b> ( <b>brightness</b> : LongInt, <b>duration</b> : LongInt): LongInt
vb	function <b>fade</b> ( ) As Integer
cs	int <b>fade</b> ( int <b>brightness</b> , int <b>duration</b> )
java	int <b>fade</b> ( int <b>brightness</b> , int <b>duration</b> )
py	def <b>fade</b> ( <b>brightness</b> , <b>duration</b> )
cmd	YDisplay <b>target fade brightness duration</b>

**Paramètres :**

**brightness** nouvelle valeur de luminosité de l'écran

**duration** durée en millisecondes de la transition.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→get\_advertisedValue()****YDisplay****display→advertisedValue()[display advertisedValue]**

Retourne la valeur courante de l'ecran (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YDisplay <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'ecran (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**display**→**get\_brightness()****YDisplay****display**→**brightness()**[display brightness]

Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).

js	function <b>get_brightness</b> ( )
nodejs	function <b>get_brightness</b> ( )
php	function <b>get_brightness</b> ( )
cpp	int <b>get_brightness</b> ( )
m	-(int) brightness
pas	function <b>get_brightness</b> ( ): LongInt
vb	function <b>get_brightness</b> ( ) As Integer
cs	int <b>get_brightness</b> ( )
java	int <b>get_brightness</b> ( )
py	def <b>get_brightness</b> ( )
cmd	YDisplay <b>target</b> <b>get_brightness</b>

**Retourne :**

un entier représentant la luminosité des leds informatives du module (valeur entre 0 et 100)

En cas d'erreur, déclenche une exception ou retourne Y\_BRIGHTNESS\_INVALID.

**display**→**get\_displayHeight()****YDisplay****display**→**displayHeight()**[**display displayHeight**]

Retourne la hauteur de l'écran, en pixels.

js	function <b>get_displayHeight</b> ( )
nodejs	function <b>get_displayHeight</b> ( )
php	function <b>get_displayHeight</b> ( )
cpp	int <b>get_displayHeight</b> ( )
m	-(int) displayHeight
pas	function <b>get_displayHeight</b> ( ): LongInt
vb	function <b>get_displayHeight</b> ( ) As Integer
cs	int <b>get_displayHeight</b> ( )
java	int <b>get_displayHeight</b> ( )
py	def <b>get_displayHeight</b> ( )
cmd	YDisplay <b>target</b> <b>get_displayHeight</b>

**Retourne :**

un entier représentant la hauteur de l'écran, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_DISPLAYHEIGHT\_INVALID.

**display**→**get\_displayLayer()****YDisplay****display**→**displayLayer()**[display displayLayer: ]

Retourne un objet YDisplayLayer utilisable pour dessiner sur la couche d'affichage correspondante.

js	function <b>get_displayLayer</b> ( <b>layerId</b> )
nodejs	function <b>get_displayLayer</b> ( <b>layerId</b> )
php	function <b>get_displayLayer</b> ( <b>\$layerId</b> )
cpp	YDisplayLayer* <b>get_displayLayer</b> ( unsigned <b>layerId</b> )
m	-(YDisplayLayer*) displayLayer : (unsigned) <b>layerId</b>
vb	function <b>get_displayLayer</b> ( ) As YDisplayLayer
cs	YDisplayLayer <b>get_displayLayer</b> ( int <b>layerId</b> )
java	synchronized YDisplayLayer <b>get_displayLayer</b> ( int <b>layerId</b> )
py	def <b>get_displayLayer</b> ( <b>layerId</b> )

Le contenu n'est visible sur l'écran que lorsque la couche est active sur l'écran (et non masquée par une couche supérieure).

**Paramètres :**

**layerId** l'identifiant de la couche d'affichage désirée (un chiffre parmi 0..layerCount-1)

**Retourne :**

un objet YDisplayLayer

En cas d'erreur, déclenche une exception ou retourne `null`.



**display**→**get\_displayType()****YDisplay****display**→**displayType()[display displayType]**

Retourne le type de l'écran: monochrome, niveaux de gris ou couleur.

js	function <b>get_displayType</b> ( )
nodejs	function <b>get_displayType</b> ( )
php	function <b>get_displayType</b> ( )
cpp	Y_DISPLAYTYPE_enum <b>get_displayType</b> ( )
m	-(Y_DISPLAYTYPE_enum) displayType
pas	function <b>get_displayType</b> ( ): Integer
vb	function <b>get_displayType</b> ( ) As Integer
cs	int <b>get_displayType</b> ( )
java	int <b>get_displayType</b> ( )
py	def <b>get_displayType</b> ( )
cmd	YDisplay <b>target</b> <b>get_displayType</b>

**Retourne :**

une valeur parmi Y\_DISPLAYTYPE\_MONO, Y\_DISPLAYTYPE\_GRAY et Y\_DISPLAYTYPE\_RGB représentant le type de l'écran: monochrome, niveaux de gris ou couleur

En cas d'erreur, déclenche une exception ou retourne Y\_DISPLAYTYPE\_INVALID.

**display**→**get\_displayWidth()****YDisplay****display**→**displayWidth()**[display displayWidth]

Retourne la largeur de l'écran, en pixels.

js	function <b>get_displayWidth</b> ( )
nodejs	function <b>get_displayWidth</b> ( )
php	function <b>get_displayWidth</b> ( )
cpp	int <b>get_displayWidth</b> ( )
m	-(int) displayWidth
pas	function <b>get_displayWidth</b> ( ): LongInt
vb	function <b>get_displayWidth</b> ( ) As Integer
cs	int <b>get_displayWidth</b> ( )
java	int <b>get_displayWidth</b> ( )
py	def <b>get_displayWidth</b> ( )
cmd	YDisplay <b>target</b> <b>get_displayWidth</b>

**Retourne :**

un entier représentant la largeur de l'écran, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_DISPLAYWIDTH\_INVALID.

## display→get\_enabled() display→enabled()[display enabled]

YDisplay

Retourne vrai si le l'ecran est alimenté, faux sinon.

js	function <b>get_enabled</b> ( )
nodejs	function <b>get_enabled</b> ( )
php	function <b>get_enabled</b> ( )
cpp	Y_ENABLED_enum <b>get_enabled</b> ( )
m	-(Y_ENABLED_enum) enabled
pas	function <b>get_enabled</b> ( ): Integer
vb	function <b>get_enabled</b> ( ) As Integer
cs	int <b>get_enabled</b> ( )
java	int <b>get_enabled</b> ( )
py	def <b>get_enabled</b> ( )
cmd	YDisplay <b>target</b> <b>get_enabled</b>

### Retourne :

soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon vrai si le l'ecran est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLED\_INVALID.

**display**→**get\_errorMessage()****YDisplay****display**→**errorMessage()**[display errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'écran.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'écran.

**display**→**get\_errorType()**  
**display**→**errorType()**

**YDisplay**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'écran.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'écran.

**display**→**get\_friendlyName()****YDisplay****display**→**friendlyName()**[display friendlyName]

Retourne un identifiant global de l'ecran au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'ecran si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'ecran (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'ecran en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**display**→**get\_functionDescriptor()**  
**display**→**functionDescriptor()[display**  
**functionDescriptor]**

YDisplay

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**display**→**get\_functionId()****YDisplay****display**→**functionId()**[**display functionId**]

Retourne l'identifiant matériel de l'ecran, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'ecran (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.



**display**→**get\_hardwareId()****YDisplay****display**→**hardwareId()**[**display hardwareId**]

Retourne l'identifiant matériel unique de l'écran au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'écran (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'écran (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**display→get\_layerCount()****YDisplay****display→layerCount()[display layerCount]**

Retourne le nombre des couches affichables disponibles.

js	function <b>get_layerCount</b> ( )
nodejs	function <b>get_layerCount</b> ( )
php	function <b>get_layerCount</b> ( )
cpp	int <b>get_layerCount</b> ( )
m	-(int) layerCount
pas	function <b>get_layerCount</b> ( ): LongInt
vb	function <b>get_layerCount</b> ( ) As Integer
cs	int <b>get_layerCount</b> ( )
java	int <b>get_layerCount</b> ( )
py	def <b>get_layerCount</b> ( )
cmd	YDisplay <b>target</b> <b>get_layerCount</b>

**Retourne :**

un entier représentant le nombre des couches affichables disponibles

En cas d'erreur, déclenche une exception ou retourne Y\_LAYERCOUNT\_INVALID.

**display→get\_layerHeight()****YDisplay****display→layerHeight()[display layerHeight]**

Retourne la hauteur des couches affichables, en pixels.

js	function <b>get_layerHeight</b> ( )
nodejs	function <b>get_layerHeight</b> ( )
php	function <b>get_layerHeight</b> ( )
cpp	int <b>get_layerHeight</b> ( )
m	-(int) layerHeight
pas	function <b>get_layerHeight</b> ( ): LongInt
vb	function <b>get_layerHeight</b> ( ) As Integer
cs	int <b>get_layerHeight</b> ( )
java	int <b>get_layerHeight</b> ( )
py	def <b>get_layerHeight</b> ( )
cmd	YDisplay <b>target</b> <b>get_layerHeight</b>

**Retourne :**

un entier représentant la hauteur des couches affichables, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_LAYERHEIGHT\_INVALID.

**display**→**get\_layerWidth()****YDisplay****display**→**layerWidth()**[display layerWidth]

Retourne la largeur des couches affichables, en pixels.

js	function <b>get_layerWidth</b> ( )
nodejs	function <b>get_layerWidth</b> ( )
php	function <b>get_layerWidth</b> ( )
cpp	int <b>get_layerWidth</b> ( )
m	-(int) layerWidth
pas	function <b>get_layerWidth</b> ( ): LongInt
vb	function <b>get_layerWidth</b> ( ) As Integer
cs	int <b>get_layerWidth</b> ( )
java	int <b>get_layerWidth</b> ( )
py	def <b>get_layerWidth</b> ( )
cmd	YDisplay <b>target</b> <b>get_layerWidth</b>

**Retourne :**

un entier représentant la largeur des couches affichables, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_LAYERWIDTH\_INVALID.

**display**→**get\_logicalName()****YDisplay****display**→**logicalName()**[**display logicalName**]

Retourne le nom logique de l'ecran.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YDisplay <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'ecran. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**display→get\_module()****YDisplay****display→module()[display module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**display→get\_module\_async()**  
**display→module\_async()**

**YDisplay**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**display**→**get\_orientation()****YDisplay****display**→**orientation()**[display orientation]

Retourne l'orientation sélectionnée pour l'écran.

js	function <b>get_orientation</b> ( )
nodejs	function <b>get_orientation</b> ( )
php	function <b>get_orientation</b> ( )
cpp	Y_ORIENTATION_enum <b>get_orientation</b> ( )
m	-(Y_ORIENTATION_enum) orientation
pas	function <b>get_orientation</b> ( ): Integer
vb	function <b>get_orientation</b> ( ) As Integer
cs	int <b>get_orientation</b> ( )
java	int <b>get_orientation</b> ( )
py	def <b>get_orientation</b> ( )
cmd	YDisplay <b>target</b> <b>get_orientation</b>

**Retourne :**

une valeur parmi Y\_ORIENTATION\_LEFT, Y\_ORIENTATION\_UP, Y\_ORIENTATION\_RIGHT et Y\_ORIENTATION\_DOWN représentant l'orientation sélectionnée pour l'écran

En cas d'erreur, déclenche une exception ou retourne Y\_ORIENTATION\_INVALID.



**display**→**get\_startupSeq()****YDisplay****display**→**startupSeq()[display startupSeq]**

Retourne le nom de la séquence à jouer à la mise sous tension de l'écran.

js	function <b>get_startupSeq</b> ( )
nodejs	function <b>get_startupSeq</b> ( )
php	function <b>get_startupSeq</b> ( )
cpp	string <b>get_startupSeq</b> ( )
m	-(NSString*) startupSeq
pas	function <b>get_startupSeq</b> ( ): string
vb	function <b>get_startupSeq</b> ( ) As String
cs	string <b>get_startupSeq</b> ( )
java	String <b>get_startupSeq</b> ( )
py	def <b>get_startupSeq</b> ( )
cmd	YDisplay <b>target</b> <b>get_startupSeq</b>

**Retourne :**

une chaîne de caractères représentant le nom de la séquence à jouer à la mise sous tension de l'écran

En cas d'erreur, déclenche une exception ou retourne Y\_STARTUPSEQ\_INVALID.

**display→get\_userdata()**

**YDisplay**

**display→userdata()[display userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

<code>js</code>	<code>function get_userdata( )</code>
<code>nodejs</code>	<code>function get_userdata( )</code>
<code>php</code>	<code>function get_userdata( )</code>
<code>cpp</code>	<code>void * get_userdata( )</code>
<code>m</code>	<code>-(void*) userData</code>
<code>pas</code>	<code>function get_userdata( ): Tobject</code>
<code>vb</code>	<code>function get_userdata( ) As Object</code>
<code>cs</code>	<code>object get_userdata( )</code>
<code>java</code>	<code>Object get_userdata( )</code>
<code>py</code>	<code>def get_userdata( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**display→isOnline()[display isOnline]****YDisplay**

Vérifie si le module hébergeant l'écran est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'écran sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'écran est joignable, `false` sinon

**display→isOnline\_async()****YDisplay**

Vérifie si le module hébergeant l'écran est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'écran sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**display→load()[display load: ]****YDisplay**

Met en cache les valeurs courantes de l'ecran, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→load\_async()****YDisplay**

Met en cache les valeurs courantes de l'écran, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**display→newSequence()[display newSequence]****YDisplay**

Enclanche l'enregistrement de toutes les commandes d'affichage suivantes dans une séquence, qui pourra être rejouée ultérieurement.

js	function <b>newSequence</b> ( )
nodejs	function <b>newSequence</b> ( )
php	function <b>newSequence</b> ( )
cpp	int <b>newSequence</b> ( )
m	-(int) <b>newSequence</b>
pas	function <b>newSequence</b> ( ): LongInt
vb	function <b>newSequence</b> ( ) As Integer
cs	int <b>newSequence</b> ( )
java	int <b>newSequence</b> ( )
py	def <b>newSequence</b> ( )
cmd	YDisplay <b>target newSequence</b>

Le nom de la séquence sera donné au moment de l'appel à `saveSequence( )`, une fois la séquence terminée.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→nextDisplay()[display nextDisplay]****YDisplay**

Continue l'énumération des écran commencée à l'aide de `yFirstDisplay()`.

js	function <b>nextDisplay</b> ( )
nodejs	function <b>nextDisplay</b> ( )
php	function <b>nextDisplay</b> ( )
cpp	YDisplay * <b>nextDisplay</b> ( )
m	-(YDisplay*) <b>nextDisplay</b>
pas	function <b>nextDisplay</b> ( ): TYDisplay
vb	function <b>nextDisplay</b> ( ) As YDisplay
cs	YDisplay <b>nextDisplay</b> ( )
java	YDisplay <b>nextDisplay</b> ( )
py	def <b>nextDisplay</b> ( )

**Retourne :**

un pointeur sur un objet `YDisplay` accessible en ligne, ou `null` lorsque l'énumération est terminée.



**display→pauseSequence()[display pauseSequence: ]****YDisplay**

Attend pour la durée spécifiée (en millisecondes) avant de jouer les commandes suivantes de la séquence active.

js	function <b>pauseSequence</b> ( <b>delay_ms</b> )
nodejs	function <b>pauseSequence</b> ( <b>delay_ms</b> )
php	function <b>pauseSequence</b> ( <b>\$delay_ms</b> )
cpp	int <b>pauseSequence</b> ( int <b>delay_ms</b> )
m	-(int) <b>pauseSequence</b> : (int) <b>delay_ms</b>
pas	function <b>pauseSequence</b> ( <b>delay_ms</b> : LongInt): LongInt
vb	function <b>pauseSequence</b> ( ) As Integer
cs	int <b>pauseSequence</b> ( int <b>delay_ms</b> )
java	int <b>pauseSequence</b> ( int <b>delay_ms</b> )
py	def <b>pauseSequence</b> ( <b>delay_ms</b> )
cmd	YDisplay <b>target</b> <b>pauseSequence</b> <b>delay_ms</b>

Cette méthode peut être utilisée lors de l'enregistrement d'une séquence d'affichage, pour insérer une attente mesurée lors de l'exécution (mais sans effet immédiat). Cette méthode peut aussi être appelée dynamiquement pendant l'exécution d'une séquence enregistrée, pour suspendre temporairement ou reprendre l'exécution. Pour annuler une attente, appelez simplement la méthode avec une attente de zéro.

**Paramètres :**

**delay\_ms** la durée de l'attente, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→playSequence()[display playSequence: ]****YDisplay**

Joue une séquence d'affichage préalablement enregistrée à l'aide des méthodes `newSequence ( )` et `saveSequence ( )`.

js	function <b>playSequence</b> ( <b>sequenceName</b> )
nodejs	function <b>playSequence</b> ( <b>sequenceName</b> )
php	function <b>playSequence</b> ( <b>\$sequenceName</b> )
cpp	int <b>playSequence</b> ( string <b>sequenceName</b> )
m	-(int) <b>playSequence</b> : (NSString*) <b>sequenceName</b>
pas	function <b>playSequence</b> ( <b>sequenceName</b> : string): LongInt
vb	function <b>playSequence</b> ( ) As Integer
cs	int <b>playSequence</b> ( string <b>sequenceName</b> )
java	int <b>playSequence</b> ( String <b>sequenceName</b> )
py	def <b>playSequence</b> ( <b>sequenceName</b> )
cmd	YDisplay <b>target playSequence</b> <b>sequenceName</b>

**Paramètres :**

**sequenceName** le nom de la nouvelle séquence créée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## display→registerValueCallback()[display registerValueCallback: ]

## YDisplay

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YDisplayValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YDisplayValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYDisplayValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**display→resetAll()[display resetAll]****YDisplay**

Efface le contenu de l'écran et remet toutes les couches à leur état initial.

js	function <b>resetAll</b> ( )
nodejs	function <b>resetAll</b> ( )
php	function <b>resetAll</b> ( )
cpp	int <b>resetAll</b> ( )
m	-(int) <b>resetAll</b>
pas	function <b>resetAll</b> ( ): LongInt
vb	function <b>resetAll</b> ( ) As Integer
cs	int <b>resetAll</b> ( )
java	int <b>resetAll</b> ( )
py	def <b>resetAll</b> ( )
cmd	YDisplay <b>target resetAll</b>

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→saveSequence()[display saveSequence: ]****YDisplay**

Termine l'enregistrement d'une séquence et la sauvegarde sur la mémoire interne de l'écran, sous le nom choisi.

js	function <b>saveSequence</b> ( <b>sequenceName</b> )
nodejs	function <b>saveSequence</b> ( <b>sequenceName</b> )
php	function <b>saveSequence</b> ( <b>\$sequenceName</b> )
cpp	int <b>saveSequence</b> ( string <b>sequenceName</b> )
m	-(int) <b>saveSequence</b> : (NSString*) <b>sequenceName</b>
pas	function <b>saveSequence</b> ( <b>sequenceName</b> : string): LongInt
vb	function <b>saveSequence</b> ( ) As Integer
cs	int <b>saveSequence</b> ( string <b>sequenceName</b> )
java	int <b>saveSequence</b> ( String <b>sequenceName</b> )
py	def <b>saveSequence</b> ( <b>sequenceName</b> )
cmd	YDisplay <b>target saveSequence</b> <b>sequenceName</b>

La séquence peut être rejouée ultérieurement à l'aide de la méthode `playSequence ( )`.

**Paramètres :**

**sequenceName** le nom de la nouvelle séquence créée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display**→**set\_brightness()****YDisplay****display**→**setBrightness()**[**display** **setBrightness:** ]

Modifie la luminosité de l'écran.

js	function <b>set_brightness</b> ( <b>newval</b> )
nodejs	function <b>set_brightness</b> ( <b>newval</b> )
php	function <b>set_brightness</b> ( <b>\$newval</b> )
cpp	int <b>set_brightness</b> ( int <b>newval</b> )
m	-(int) <b>setBrightness</b> : (int) <b>newval</b>
pas	function <b>set_brightness</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_brightness</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_brightness</b> ( int <b>newval</b> )
java	int <b>set_brightness</b> ( int <b>newval</b> )
py	def <b>set_brightness</b> ( <b>newval</b> )
cmd	YDisplay <b>target</b> <b>set_brightness</b> <b>newval</b>

Le paramètre est une valeur entre 0 et 100. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant la luminosité de l'écran

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display**→**set\_enabled()****YDisplay****display**→**setEnabled()**[**display** **setEnabled:** ]

Modifie l'état d'activité de l'écran.

js	function <b>set_enabled</b> ( <b>newval</b> )
nodejs	function <b>set_enabled</b> ( <b>newval</b> )
php	function <b>set_enabled</b> ( <b>\$newval</b> )
cpp	int <b>set_enabled</b> ( Y_ENABLED_enum <b>newval</b> )
m	-(int) <b>setEnabled</b> : (Y_ENABLED_enum) <b>newval</b>
pas	function <b>set_enabled</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabled</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabled</b> ( int <b>newval</b> )
java	int <b>set_enabled</b> ( int <b>newval</b> )
py	def <b>set_enabled</b> ( <b>newval</b> )
cmd	YDisplay <b>target</b> <b>set_enabled</b> <b>newval</b>

**Paramètres :**

**newval** soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon l'état d'activité de l'écran

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display**→**set\_logicalName()****YDisplay****display**→**setLogicalName()**[**display setLogicalName:**  
**]**

Modifie le nom logique de l'ecran.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YDisplay <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'ecran.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**display→set\_orientation()****YDisplay****display→setOrientation()[display setOrientation: ]**

Modifie l'orientation de l'écran.

js	function <b>set_orientation</b> ( <b>newval</b> )
nodejs	function <b>set_orientation</b> ( <b>newval</b> )
php	function <b>set_orientation</b> ( <b>\$newval</b> )
cpp	int <b>set_orientation</b> ( Y_ORIENTATION_enum <b>newval</b> )
m	-(int) setOrientation : (Y_ORIENTATION_enum) <b>newval</b>
pas	function <b>set_orientation</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_orientation</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_orientation</b> ( int <b>newval</b> )
java	int <b>set_orientation</b> ( int <b>newval</b> )
py	def <b>set_orientation</b> ( <b>newval</b> )
cmd	YDisplay <b>target set_orientation newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une valeur parmi Y\_ORIENTATION\_LEFT, Y\_ORIENTATION\_UP, Y\_ORIENTATION\_RIGHT et Y\_ORIENTATION\_DOWN représentant l'orientation de l'écran

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display**→**set\_startupSeq()****YDisplay****display**→**setStartupSeq()**[**display setStartupSeq:** ]

Modifie le nom de la séquence à jouer à la mise sous tension de l'écran.

js	function <b>set_startupSeq</b> ( <b>newval</b> )
nodejs	function <b>set_startupSeq</b> ( <b>newval</b> )
php	function <b>set_startupSeq</b> ( <b>\$newval</b> )
cpp	int <b>set_startupSeq</b> ( const string& <b>newval</b> )
m	-(int) setStartupSeq : (NSString*) <b>newval</b>
pas	function <b>set_startupSeq</b> ( <b>newval</b> : string): integer
vb	function <b>set_startupSeq</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_startupSeq</b> ( string <b>newval</b> )
java	int <b>set_startupSeq</b> ( String <b>newval</b> )
py	def <b>set_startupSeq</b> ( <b>newval</b> )
cmd	YDisplay <b>target set_startupSeq newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom de la séquence à jouer à la mise sous tension de l'écran

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display**→**set\_userdata()****YDisplay****display**→**setUserData()**[**display setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**display→stopSequence()[display stopSequence]****YDisplay**

Arrête immédiatement la séquence d'affichage actuellement jouée sur l'écran.

js	function <b>stopSequence</b> ( )
nodejs	function <b>stopSequence</b> ( )
php	function <b>stopSequence</b> ( )
cpp	int <b>stopSequence</b> ( )
m	-(int) <b>stopSequence</b>
pas	function <b>stopSequence</b> ( ): LongInt
vb	function <b>stopSequence</b> ( ) As Integer
cs	int <b>stopSequence</b> ( )
java	int <b>stopSequence</b> ( )
py	def <b>stopSequence</b> ( )
cmd	YDisplay <b>target</b> <b>stopSequence</b>

L'affichage est laissé tel quel.

**Paramètres :**

**sequenceName** le nom de la nouvelle séquence créée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**display→upload()[display upload: ]****YDisplay**

Télécharge un contenu arbitraire (par exemple une image GIF) vers le système de fichier de l'écran, au chemin d'accès spécifié.

js	function <b>upload</b> ( <b>pathname</b> , <b>content</b> )
nodejs	function <b>upload</b> ( <b>pathname</b> , <b>content</b> )
php	function <b>upload</b> ( <b>\$pathname</b> , <b>\$content</b> )
c++	int <b>upload</b> ( string <b>pathname</b> , string <b>content</b> )
m	-(int) <b>upload</b> : (NSString*) <b>pathname</b> : (NSData*) <b>content</b>
pas	function <b>upload</b> ( <b>pathname</b> : string, <b>content</b> : TByteArray): LongInt
vb	procedure <b>upload</b> ( )
cs	int <b>upload</b> ( string <b>pathname</b> )
java	int <b>upload</b> ( String <b>pathname</b> )
py	def <b>upload</b> ( <b>pathname</b> , <b>content</b> )
cmd	YDisplay <b>target upload</b> <b>pathname content</b>

Si un fichier existe déjà pour le même chemin d'accès, son contenu est remplacé.

**Paramètres :**

**pathname** nom complet du fichier, y compris le chemin d'accès.

**content** contenu du fichier à télécharger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**display→wait\_async()****YDisplay**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.14. Interface des objets DisplayLayer

Un DisplayLayer est une couche de contenu affichable (images, texte, etc.). Le contenu n'est visible sur l'écran que lorsque la couche est active sur l'écran (et non masquée par une couche supérieure).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_display.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YDisplay = yoctolib.YDisplay;
php	require_once('yocto_display.php');
c++	#include "yocto_display.h"
m	#import "yocto_display.h"
pas	uses yocto_display;
vb	yocto_display.vb
cs	yocto_display.cs
java	import com.yoctopuce.YoctoAPI.YDisplay;
py	from yocto_display import *

### Méthodes des objets YDisplayLayer

#### displaylayer→clear()

Efface tout le contenu de la couche de dessin, de sorte à ce qu'elle redevienne entièrement transparente.

#### displaylayer→clearConsole(text)

Efface le contenu de la zone de console, et repositionne le curseur de la console en haut à gauche de la zone.

#### displaylayer→consoleOut(text)

Affiche un message dans la zone de console, et déplace le curseur de la console à la fin du texte.

#### displaylayer→drawBar(x1, y1, x2, y2)

Dessine un rectangle plein à une position spécifiée.

#### displaylayer→drawBitmap(x, y, w, bitmap, bgcol)

Dessine un bitmap à la position spécifiée de la couche.

#### displaylayer→drawCircle(x, y, r)

Dessine un cercle vide à une position spécifiée.

#### displaylayer→drawDisc(x, y, r)

Dessine un disque plein à une position spécifiée.

#### displaylayer→drawImage(x, y, imagename)

Dessine une image GIF à la position spécifiée de la couche.

#### displaylayer→drawPixel(x, y)

Dessine un pixel unique à une position spécifiée.

#### displaylayer→drawRect(x1, y1, x2, y2)

Dessine un rectangle vide à une position spécifiée.

#### displaylayer→drawText(x, y, anchor, text)

Affiche un texte à la position spécifiée de la couche.

#### displaylayer→get\_display()

Retourne l'YDisplay parent.

#### displaylayer→get\_displayHeight()

Retourne la hauteur de l'écran, en pixels.

#### displaylayer→get\_displayWidth()

Retourne la largeur de l'écran, en pixels.



**displaylayer→get\_layerHeight()**

Retourne la hauteur des couches affichables, en pixels.

**displaylayer→get\_layerWidth()**

Retourne la largeur des couches affichables, en pixels.

**displaylayer→hide()**

Cache la couche de dessin.

**displaylayer→lineTo(x, y)**

Dessine une ligne depuis le point de dessin courant jusqu'à la position spécifiée.

**displaylayer→moveTo(x, y)**

Déplace le point de dessin courant de cette couche à la position spécifiée.

**displaylayer→reset()**

Remet la couche de dessin dans son état initial (entièrement transparente, réglages par défaut).

**displaylayer→selectColorPen(color)**

Choisit la couleur du crayon à utiliser pour tous les appels suivants aux fonctions de dessin.

**displaylayer→selectEraser()**

Choisit une gomme plutôt qu'un crayon pour tous les appels suivants aux fonctions de dessin, à l'exception de l'affichage de texte et de copie d'images bitmaps.

**displaylayer→selectFont(fontname)**

Sélectionne la police de caractères à utiliser pour les fonctions d'affichage de texte suivantes.

**displaylayer→selectGrayPen(graylevel)**

Choisit le niveau de gris à utiliser pour tous les appels suivants aux fonctions de dessin.

**displaylayer→setAntialiasingMode(mode)**

Active ou désactive l'anti-aliasing pour tracer les lignes et les cercles.

**displaylayer→setConsoleBackground(bgcol)**

Configure la couleur de fond utilisée par la fonction `clearConsole` et par le défilement automatique de la console.

**displaylayer→setConsoleMargins(x1, y1, x2, y2)**

Configure les marges d'affichage pour la fonction `consoleOut`.

**displaylayer→setConsoleWordWrap(wordwrap)**

Configure le mode de retour à la ligne utilisé par la fonction `consoleOut`.

**displaylayer→setLayerPosition(x, y, scrollTime)**

Déplace la position de la couche de dessin par rapport au coin supérieur gauche de l'écran.

**displaylayer→unhide()**

Affiche la couche.

**displaylayer→clear()[displaylayer clear]****YDisplayLayer**

Efface tout le contenu de la couche de dessin, de sorte à ce qu'elle redevienne entièrement transparente.

js	function <b>clear</b> ( )
nodejs	function <b>clear</b> ( )
php	function <b>clear</b> ( )
cpp	int <b>clear</b> ( )
m	-(int) <b>clear</b>
pas	function <b>clear</b> ( ): LongInt
vb	function <b>clear</b> ( ) As Integer
cs	int <b>clear</b> ( )
java	int <b>clear</b> ( )
py	def <b>clear</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>clear</b>

Cette méthode ne change pas les réglages de la couche. Si vous désirez remettre la couche dans son état initial, utilisez plutôt la méthode `reset( )`.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→clearConsole()[displaylayer clearConsole]

## YDisplayLayer

Efface le contenu de la zone de console, et repositionne le curseur de la console en haut à gauche de la zone.

js	function <b>clearConsole</b> ( )
nodejs	function <b>clearConsole</b> ( )
php	function <b>clearConsole</b> ( )
cpp	int <b>clearConsole</b> ( )
m	-(int) <b>clearConsole</b>
pas	function <b>clearConsole</b> ( ): LongInt
vb	function <b>clearConsole</b> ( ) As Integer
cs	int <b>clearConsole</b> ( )
java	int <b>clearConsole</b> ( )
py	def <b>clearConsole</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>clearConsole</b>

### Paramètres :

**text** le texte à afficher

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→consoleOut()[displaylayer consoleOut:  
]****YDisplayLayer**

Affiche un message dans la zone de console, et déplace le curseur de la console à la fin du texte.

js	function <b>consoleOut</b> ( <b>text</b> )
nodejs	function <b>consoleOut</b> ( <b>text</b> )
php	function <b>consoleOut</b> ( <b>\$text</b> )
c++	int <b>consoleOut</b> ( string <b>text</b> )
m	-(int) <b>consoleOut</b> : (NSString*) <b>text</b>
pas	function <b>consoleOut</b> ( <b>text</b> : string): LongInt
vb	function <b>consoleOut</b> ( ) As Integer
cs	int <b>consoleOut</b> ( string <b>text</b> )
java	int <b>consoleOut</b> ( String <b>text</b> )
py	def <b>consoleOut</b> ( <b>text</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>consoleOut</b> <b>text</b>

Le curseur revient automatiquement en début de ligne suivante lorsqu'un saut de ligne est rencontré, ou lorsque la marge droite est atteinte. Lorsque le texte à afficher s'apprête à dépasser la marge inférieure, le contenu de la zone de console est automatiquement décalé vers le haut afin de laisser la place à la nouvelle ligne de texte.

**Paramètres :**

**text** le message à afficher

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→drawBar()[displaylayer drawBar: ]****YDisplayLayer**

Dessine un rectangle plein à une position spécifiée.

<code>js</code>	<code>function drawBar( x1, y1, x2, y2)</code>
<code>nodejs</code>	<code>function drawBar( x1, y1, x2, y2)</code>
<code>php</code>	<code>function drawBar( \$x1, \$y1, \$x2, \$y2)</code>
<code>cpp</code>	<code>int drawBar( int x1, int y1, int x2, int y2)</code>
<code>m</code>	<code>-(int) drawBar : (int) x1                   : (int) y1                   : (int) x2                   : (int) y2</code>
<code>pas</code>	<code>function drawBar( x1: LongInt,                   y1: LongInt,                   x2: LongInt,                   y2: LongInt): LongInt</code>
<code>vb</code>	<code>function drawBar( ) As Integer</code>
<code>cs</code>	<code>int drawBar( int x1, int y1, int x2, int y2)</code>
<code>java</code>	<code>int drawBar( int x1, int y1, int x2, int y2)</code>
<code>py</code>	<code>def drawBar( x1, y1, x2, y2)</code>
<code>cmd</code>	<code>YDisplay target [-layer layerId] drawBar x1 y1 x2 y2</code>

**Paramètres :**

- x1** la distance en pixels depuis la gauche de la couche jusqu'au bord gauche du rectangle
- y1** la distance en pixels depuis le haut de la couche jusqu'au bord supérieur du rectangle
- x2** la distance en pixels depuis la gauche de la couche jusqu'au bord droit du rectangle
- y2** la distance en pixels depuis le haut de la couche jusqu'au bord inférieur du rectangle

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→drawBitmap()[displaylayer drawBitmap: ]

YDisplayLayer

Dessine un bitmap à la position spécifiée de la couche.

```

js function drawBitmap( x, y, w, bitmap, bgcol)
nodejs function drawBitmap( x, y, w, bitmap, bgcol)
php function drawBitmap( $x, $y, $w, $bitmap, $bgcol)
cpp int drawBitmap( int x, int y, int w, string bitmap, int bgcol)
m -(int) drawBitmap : (int) x
                        : (int) y
                        : (int) w
                        : (NSData*) bitmap
                        : (int) bgcol
pas function drawBitmap( x: LongInt,
                        y: LongInt,
                        w: LongInt,
                        bitmap: TByteArray,
                        bgcol: LongInt): LongInt
vb procedure drawBitmap( )
cs int drawBitmap( int x, int y, int w, int bgcol)
java int drawBitmap( int x, int y, int w, int bgcol)
py def drawBitmap( x, y, w, bitmap, bgcol)
cmd YDisplay target [-layer layerId] drawBitmap x y w bitmap bgcol

```

Le bitmap est passé sous forme d'un objet binaire, où chaque bit correspond à un pixel, de gauche à droite et de haut en bas. Le bit de poids fort de chaque octet correspond au pixel de gauche, et le bit de poids faible au pixel le plus à droite. Les bits à 1 sont dessinés avec la couleur active de la couche. Les bits à 0 avec la couleur de fond spécifiée, sauf si la valeur -1 a été choisie, auquel cas ils ne sont pas dessinés (ils sont considérés comme transparents). Chaque ligne commence sur un nouvel octet. La hauteur du bitmap est donnée implicitement par la taille de l'objet binaire.

### Paramètres :

- x** la distance en pixels depuis la gauche de la couche jusqu'au bord gauche du bitmap
- y** la distance en pixels depuis le haut de la couche jusqu'au bord supérieur du bitmap
- w** la largeur du bitmap, en pixels
- bitmap** l'objet binaire contenant le bitmap
- bgcol** le niveau de gris à utiliser pour les bits à zéro (0 = noir, 255 = blanc), ou -1 pour laisser les pixels inchangés

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Dessine un cercle vide à une position spécifiée.

js	function <b>drawCircle</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
node.js	function <b>drawCircle</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
php	function <b>drawCircle</b> ( <b>\$x</b> , <b>\$y</b> , <b>\$r</b> )
cpp	int <b>drawCircle</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
m	-(int) <b>drawCircle</b> : (int) <b>x</b> : (int) <b>y</b> : (int) <b>r</b>
pas	function <b>drawCircle</b> ( <b>x</b> : LongInt, <b>y</b> : LongInt, <b>r</b> : LongInt): LongInt
vb	function <b>drawCircle</b> ( ) As Integer
cs	int <b>drawCircle</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
java	int <b>drawCircle</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
py	def <b>drawCircle</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>drawCircle</b> <b>x y r</b>

### Paramètres :

- x** la distance en pixels depuis la gauche de la couche jusqu'au centre du cercle
- y** la distance en pixels depuis le haut de la couche jusqu'au centre du cercle
- r** le rayon du cercle, en pixels

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→drawDisc()[displaylayer drawDisc: ]**

## YDisplayLayer

Dessine un disque plein à une position spécifiée.

js	function <b>drawDisc</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
nodejs	function <b>drawDisc</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
php	function <b>drawDisc</b> ( <b>\$x</b> , <b>\$y</b> , <b>\$r</b> )
cpp	int <b>drawDisc</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
m	-(int) <b>drawDisc</b> : (int) <b>x</b> : (int) <b>y</b> : (int) <b>r</b>
pas	function <b>drawDisc</b> ( <b>x</b> : LongInt, <b>y</b> : LongInt, <b>r</b> : LongInt): LongInt
vb	function <b>drawDisc</b> ( ) As Integer
cs	int <b>drawDisc</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
java	int <b>drawDisc</b> ( int <b>x</b> , int <b>y</b> , int <b>r</b> )
py	def <b>drawDisc</b> ( <b>x</b> , <b>y</b> , <b>r</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>drawDisc</b> <b>x y r</b>

### Paramètres :

- x** la distance en pixels depuis la gauche de la couche jusqu'au centre du disque
- y** la distance en pixels depuis le haut de la couche jusqu'au centre du disque
- r** le rayon du disque, en pixels

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.





**displaylayer→drawPixel()[displaylayer drawPixel: ]****YDisplayLayer**

Dessine un pixel unique à une position spécifiée.

js	function <b>drawPixel</b> ( <b>x</b> , <b>y</b> )
nodejs	function <b>drawPixel</b> ( <b>x</b> , <b>y</b> )
php	function <b>drawPixel</b> ( <b>\$x</b> , <b>\$y</b> )
cpp	int <b>drawPixel</b> ( int <b>x</b> , int <b>y</b> )
m	-(int) <b>drawPixel</b> : (int) <b>x</b> : (int) <b>y</b>
pas	function <b>drawPixel</b> ( <b>x</b> : LongInt, <b>y</b> : LongInt): LongInt
vb	function <b>drawPixel</b> ( ) As Integer
cs	int <b>drawPixel</b> ( int <b>x</b> , int <b>y</b> )
java	int <b>drawPixel</b> ( int <b>x</b> , int <b>y</b> )
py	def <b>drawPixel</b> ( <b>x</b> , <b>y</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>drawPixel</b> <b>x y</b>

**Paramètres :**

- x** la distance en pixels depuis la gauche de la couche
- y** la distance en pixels depuis le haut de la couche

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→drawRect()[displaylayer drawRect: ]****YDisplayLayer**

Dessine un rectangle vide à une position spécifiée.

js	function <b>drawRect</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
nodejs	function <b>drawRect</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
php	function <b>drawRect</b> ( <b>\$x1</b> , <b>\$y1</b> , <b>\$x2</b> , <b>\$y2</b> )
cpp	int <b>drawRect</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
m	-(int) <b>drawRect</b> : (int) <b>x1</b> : (int) <b>y1</b> : (int) <b>x2</b> : (int) <b>y2</b>
pas	function <b>drawRect</b> ( <b>x1</b> : LongInt, <b>y1</b> : LongInt, <b>x2</b> : LongInt, <b>y2</b> : LongInt): LongInt
vb	function <b>drawRect</b> ( ) As Integer
cs	int <b>drawRect</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
java	int <b>drawRect</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
py	def <b>drawRect</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>drawRect</b> <b>x1</b> <b>y1</b> <b>x2</b> <b>y2</b>

**Paramètres :**

- x1** la distance en pixels depuis la gauche de la couche jusqu'au bord gauche du rectangle
- y1** la distance en pixels depuis le haut de la couche jusqu'au bord supérieur du rectangle
- x2** la distance en pixels depuis la gauche de la couche jusqu'au bord droit du rectangle
- y2** la distance en pixels depuis le haut de la couche jusqu'au bord inférieur du rectangle

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→drawText()[displaylayer drawText: ]****YDisplayLayer**

Affiche un texte à la position spécifiée de la couche.

js	function <b>drawText</b> ( <b>x</b> , <b>y</b> , <b>anchor</b> , <b>text</b> )
nodejs	function <b>drawText</b> ( <b>x</b> , <b>y</b> , <b>anchor</b> , <b>text</b> )
php	function <b>drawText</b> ( <b>\$x</b> , <b>\$y</b> , <b>\$anchor</b> , <b>\$text</b> )
cpp	int <b>drawText</b> ( int <b>x</b> , int <b>y</b> , Y_ALIGN <b>anchor</b> , string <b>text</b> )
m	-(int) <b>drawText</b> : (int) <b>x</b> : (int) <b>y</b> : (Y_ALIGN) <b>anchor</b> : (NSString*) <b>text</b>
pas	function <b>drawText</b> ( <b>x</b> : LongInt, <b>y</b> : LongInt, <b>anchor</b> : TYALIGN, <b>text</b> : string): LongInt
vb	function <b>drawText</b> ( ) As Integer
cs	int <b>drawText</b> ( int <b>x</b> , int <b>y</b> , ALIGN <b>anchor</b> , string <b>text</b> )
java	int <b>drawText</b> ( int <b>x</b> , int <b>y</b> , ALIGN <b>anchor</b> , String <b>text</b> )
py	def <b>drawText</b> ( <b>x</b> , <b>y</b> , <b>anchor</b> , <b>text</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>drawText</b> <b>x y anchor text</b>

Le point du texte qui sera aligné sur la position spécifiée est appelé point d'ancrage, et peut être choisi parmi plusieurs options.

**Paramètres :**

- x** la distance en pixels depuis la gauche de la couche jusqu'au point d'ancrage du texte
- y** la distance en pixels depuis le haut de la couche jusqu'au point d'ancrage du texte
- anchor** le point d'ancrage du texte, choisi parmi l'énumération Y\_ALIGN: Y\_ALIGN\_TOP\_LEFT, Y\_ALIGN\_CENTER\_LEFT, Y\_ALIGN\_BASELINE\_LEFT, Y\_ALIGN\_BOTTOM\_LEFT, Y\_ALIGN\_TOP\_CENTER, Y\_ALIGN\_CENTER, Y\_ALIGN\_BASELINE\_CENTER, Y\_ALIGN\_BOTTOM\_CENTER, Y\_ALIGN\_TOP\_DECIMAL, Y\_ALIGN\_CENTER\_DECIMAL, Y\_ALIGN\_BASELINE\_DECIMAL, Y\_ALIGN\_BOTTOM\_DECIMAL, Y\_ALIGN\_TOP\_RIGHT, Y\_ALIGN\_CENTER\_RIGHT, Y\_ALIGN\_BASELINE\_RIGHT, Y\_ALIGN\_BOTTOM\_RIGHT.
- text** le texte à afficher

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→get\_display()**  
**displaylayer→display()[displaylayer display]**

**YDisplayLayer**

Retourne l'YDisplay parent.

js	function <b>get_display</b> ( )
nodejs	function <b>get_display</b> ( )
php	function <b>get_display</b> ( )
cpp	YDisplay* <b>get_display</b> ( )
m	-(YDisplay*) display
pas	function <b>get_display</b> ( ): TYDisplay
vb	function <b>get_display</b> ( ) As YDisplay
cs	YDisplay <b>get_display</b> ( )
java	YDisplay <b>get_display</b> ( )
py	def <b>get_display</b> ( )

Retourne l'objet YDisplay parent du YDisplayLayer courant.

**Retourne :**

un objet YDisplay

## displaylayer→get\_displayHeight() displaylayer→displayHeight()[displaylayer displayHeight]

YDisplayLayer

Retourne la hauteur de l'écran, en pixels.

js	function <b>get_displayHeight</b> ( )
nodejs	function <b>get_displayHeight</b> ( )
php	function <b>get_displayHeight</b> ( )
cpp	int <b>get_displayHeight</b> ( )
m	-(int) displayHeight
pas	function <b>get_displayHeight</b> ( ): LongInt
vb	function <b>get_displayHeight</b> ( ) As Integer
cs	int <b>get_displayHeight</b> ( )
java	int <b>get_displayHeight</b> ( )
py	def <b>get_displayHeight</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>get_displayHeight</b>

### Retourne :

un entier représentant la hauteur de l'écran, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_DISPLAYHEIGHT\_INVALID.

**displaylayer**→**get\_displayWidth()**  
**displaylayer**→**displayWidth()**[**displaylayer**  
**displayWidth**]

**YDisplayLayer**

Retourne la largeur de l'écran, en pixels.

js	function <b>get_displayWidth</b> ( )
nodejs	function <b>get_displayWidth</b> ( )
php	function <b>get_displayWidth</b> ( )
cpp	int <b>get_displayWidth</b> ( )
m	-(int) displayWidth
pas	function <b>get_displayWidth</b> ( ): LongInt
vb	function <b>get_displayWidth</b> ( ) As Integer
cs	int <b>get_displayWidth</b> ( )
java	int <b>get_displayWidth</b> ( )
py	def <b>get_displayWidth</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>get_displayWidth</b>

**Retourne :**

un entier représentant la largeur de l'écran, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_DISPLAYWIDTH\_INVALID.

**displaylayer→get\_layerHeight()****YDisplayLayer****displaylayer→layerHeight()[displaylayer layerHeight]**

Retourne la hauteur des couches affichables, en pixels.

js	function <b>get_layerHeight</b> ( )
nodejs	function <b>get_layerHeight</b> ( )
php	function <b>get_layerHeight</b> ( )
cpp	int <b>get_layerHeight</b> ( )
m	-(int) layerHeight
pas	function <b>get_layerHeight</b> ( ): LongInt
vb	function <b>get_layerHeight</b> ( ) As Integer
cs	int <b>get_layerHeight</b> ( )
java	int <b>get_layerHeight</b> ( )
py	def <b>get_layerHeight</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>get_layerHeight</b>

**Retourne :**

un entier représentant la hauteur des couches affichables, en pixels. En cas d'erreur, déclenche une exception ou retourne Y\_LAYERHEIGHT\_INVALID.



**displaylayer→get\_layerWidth()****YDisplayLayer****displaylayer→layerWidth()[displaylayer layerWidth]**

Retourne la largeur des couches affichables, en pixels.

js	function <b>get_layerWidth</b> ( )
nodejs	function <b>get_layerWidth</b> ( )
php	function <b>get_layerWidth</b> ( )
cpp	int <b>get_layerWidth</b> ( )
m	-(int) layerWidth
pas	function <b>get_layerWidth</b> ( ): LongInt
vb	function <b>get_layerWidth</b> ( ) As Integer
cs	int <b>get_layerWidth</b> ( )
java	int <b>get_layerWidth</b> ( )
py	def <b>get_layerWidth</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>get_layerWidth</b>

**Retourne :**

un entier représentant la largeur des couches affichables, en pixels

En cas d'erreur, déclenche une exception ou retourne Y\_LAYERWIDTH\_INVALID.

**displaylayer→hide()[displaylayer hide]****YDisplayLayer**

Cache la couche de dessin.

js	function <b>hide</b> ( )
nodejs	function <b>hide</b> ( )
php	function <b>hide</b> ( )
cpp	int <b>hide</b> ( )
m	-(int) <b>hide</b>
pas	function <b>hide</b> ( ): LongInt
vb	function <b>hide</b> ( ) As Integer
cs	int <b>hide</b> ( )
java	int <b>hide</b> ( )
py	def <b>hide</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>hide</b>

L'état de la couche est préservé, mais la couche ne sera plus plus affichés à l'écran jusqu'au prochain appel à `unhide()`. Le fait de cacher la couche améliore les performances de toutes les primitives d'affichage, car il évite de consacrer inutilement des cycles de calcul à afficher les états intermédiaires (technique de double-buffering).

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**displaylayer→moveTo()[displaylayer moveTo: ]**

## YDisplayLayer

Déplace le point de dessin courant de cette couche à la position spécifiée.

js	function <b>moveTo</b> ( <b>x</b> , <b>y</b> )
nodejs	function <b>moveTo</b> ( <b>x</b> , <b>y</b> )
php	function <b>moveTo</b> ( <b>\$x</b> , <b>\$y</b> )
c++	int <b>moveTo</b> ( int <b>x</b> , int <b>y</b> )
m	-(int) <b>moveTo</b> : (int) <b>x</b> : (int) <b>y</b>
pas	function <b>moveTo</b> ( <b>x</b> : LongInt, <b>y</b> : LongInt): LongInt
vb	function <b>moveTo</b> ( ) As Integer
cs	int <b>moveTo</b> ( int <b>x</b> , int <b>y</b> )
java	int <b>moveTo</b> ( int <b>x</b> , int <b>y</b> )
py	def <b>moveTo</b> ( <b>x</b> , <b>y</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>moveTo</b> <b>x y</b>

### Paramètres :

- x la distance en pixels depuis la gauche de la couche de dessin

**y** la distance en pixels depuis le haut de la couche de dessin

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→reset()[displaylayer reset]****YDisplayLayer**

Remet la couche de dessin dans son état initial (entièrement transparente, réglages par défaut).

<code>js</code>	<code>function reset( )</code>
<code>nodejs</code>	<code>function reset( )</code>
<code>php</code>	<code>function reset( )</code>
<code>cpp</code>	<code>int reset( )</code>
<code>m</code>	<code>-(int) reset</code>
<code>pas</code>	<code>function reset( ): LongInt</code>
<code>vb</code>	<code>function reset( ) As Integer</code>
<code>cs</code>	<code>int reset( )</code>
<code>java</code>	<code>int reset( )</code>
<code>py</code>	<code>def reset( )</code>
<code>cmd</code>	<code>YDisplay <b>target</b> [-layer <b>layerId</b>] <b>reset</b></code>

Réinitialise la position du point de dessin courant au coin supérieur gauche, et la couleur de dessin à la valeur la plus lumineuse. Si vous désirez simplement effacer le contenu de la couche, utilisez plutôt la méthode `clear()`.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→selectColorPen()[displaylayer selectColorPen: ]

YDisplayLayer

Choisit la couleur du crayon à utiliser pour tous les appels suivants aux fonctions de dessin.

js	function <b>selectColorPen</b> ( <b>color</b> )
nodejs	function <b>selectColorPen</b> ( <b>color</b> )
php	function <b>selectColorPen</b> ( <b>\$color</b> )
cpp	int <b>selectColorPen</b> ( int <b>color</b> )
m	-(int) <b>selectColorPen</b> : (int) <b>color</b>
pas	function <b>selectColorPen</b> ( <b>color</b> : LongInt): LongInt
vb	function <b>selectColorPen</b> ( ) As Integer
cs	int <b>selectColorPen</b> ( int <b>color</b> )
java	int <b>selectColorPen</b> ( int <b>color</b> )
py	def <b>selectColorPen</b> ( <b>color</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>selectColorPen</b> <b>color</b>

La couleur est fournie sous forme de couleur RGB. Pour les écrans monochromes ou en niveaux de gris, la couleur est automatiquement ramenée dans les valeurs permises.

### Paramètres :

**color** la couleur RGB désirée (sous forme d'entier 24 bits)

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→selectEraser()[displaylayer selectEraser]

## YDisplayLayer

Choisit une gomme plutôt qu'un crayon pour tous les appels suivants aux fonctions de dessin, à l'exception de l'affichage de texte et de copie d'images bitmaps.

js	function <b>selectEraser</b> ( )
nodejs	function <b>selectEraser</b> ( )
php	function <b>selectEraser</b> ( )
cpp	int <b>selectEraser</b> ( )
m	-(int) <b>selectEraser</b>
pas	function <b>selectEraser</b> ( ): LongInt
vb	function <b>selectEraser</b> ( ) As Integer
cs	int <b>selectEraser</b> ( )
java	int <b>selectEraser</b> ( )
py	def <b>selectEraser</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>selectEraser</b>

Tous les points dessinés à la gomme redeviennent transparents (comme ils l'étaient lorsque la couche était vide), rendant ainsi visibles les couches inférieures.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**displaylayer→selectFont()[displaylayer selectFont: ]****YDisplayLayer**

Sélectionne la police de caractères à utiliser pour les fonctions d'affichage de texte suivantes.

js	function <b>selectFont</b> ( <b>fontname</b> )
nodejs	function <b>selectFont</b> ( <b>fontname</b> )
php	function <b>selectFont</b> ( <b>\$fontname</b> )
cpp	int <b>selectFont</b> ( string <b>fontname</b> )
m	-(int) <b>selectFont</b> : (NSString*) <b>fontname</b>
pas	function <b>selectFont</b> ( <b>fontname</b> : string): LongInt
vb	function <b>selectFont</b> ( ) As Integer
cs	int <b>selectFont</b> ( string <b>fontname</b> )
java	int <b>selectFont</b> ( String <b>fontname</b> )
py	def <b>selectFont</b> ( <b>fontname</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>selectFont</b> <b>fontname</b>

La police est spécifiée par le nom de son fichier. Vous pouvez utiliser l'une des polices prédéfinies dans le module, ou une autre police que vous avez préalablement préchargé dans la mémoire du module. Si vous rencontrez des problèmes à l'utilisation d'une police de caractères, consultez les logs du module pour voir si vous n'y trouvez pas un message à propos d'un fichier de police manquant ou d'un format de fichier invalide.

**Paramètres :**

**fontname** le nom du fichier définissant la police de caractères

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## displaylayer→selectGrayPen()[displaylayer selectGrayPen: ]

## YDisplayLayer

Choisit le niveau de gris à utiliser pour tous les appels suivants aux fonctions de dessin.

js	function <b>selectGrayPen</b> ( <b>graylevel</b> )
nodejs	function <b>selectGrayPen</b> ( <b>graylevel</b> )
php	function <b>selectGrayPen</b> ( <b>\$graylevel</b> )
cpp	int <b>selectGrayPen</b> ( int <b>graylevel</b> )
m	-(int) <b>selectGrayPen</b> : (int) <b>graylevel</b>
pas	function <b>selectGrayPen</b> ( <b>graylevel</b> : LongInt): LongInt
vb	function <b>selectGrayPen</b> ( ) As Integer
cs	int <b>selectGrayPen</b> ( int <b>graylevel</b> )
java	int <b>selectGrayPen</b> ( int <b>graylevel</b> )
py	def <b>selectGrayPen</b> ( <b>graylevel</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>selectGrayPen</b> <b>graylevel</b>

Le niveau de gris est fourni sous forme d'un chiffre allant de 0 (noir) à 255 (blanc, ou la couleur la plus claire de l'écran, quelle qu'elle soit). Pour les écrans monochromes (sans niveaux de gris), toute valeur inférieure à 128 conduit à un point noir, et toute valeur supérieure ou égale à 128 devient un point lumineux.

### Paramètres :

**graylevel** le niveau de gris désiré, de 0 à 255

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→setAntialiasingMode()[displaylayer setAntialiasingMode: ]

YDisplayLayer

Active ou désactive l'anti-aliasing pour tracer les lignes et les cercles.

js	function <b>setAntialiasingMode</b> ( <b>mode</b> )
nodejs	function <b>setAntialiasingMode</b> ( <b>mode</b> )
php	function <b>setAntialiasingMode</b> ( <b>\$mode</b> )
cpp	int <b>setAntialiasingMode</b> ( bool <b>mode</b> )
m	-(int) <b>setAntialiasingMode</b> : (bool) <b>mode</b>
pas	function <b>setAntialiasingMode</b> ( <b>mode</b> : boolean): LongInt
vb	function <b>setAntialiasingMode</b> ( ) As Integer
cs	int <b>setAntialiasingMode</b> ( bool <b>mode</b> )
java	int <b>setAntialiasingMode</b> ( boolean <b>mode</b> )
py	def <b>setAntialiasingMode</b> ( <b>mode</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>setAntialiasingMode</b> <b>mode</b>

L'anti-aliasing est atténue la pixelisation des images lorsqu'on regarde l'écran depuis une distance suffisante, mais peut aussi donner parfois une impression de flou lorsque l'écran est regardé de très près. Au final, c'est un choix esthétique qui vous revient. L'anti-aliasing est activé par défaut pour les écrans en niveaux de gris et les écrans couleurs, mais vous pouvez le désactiver si vous préférez. Ce réglage n'a pas d'effet sur les écrans monochromes.

### Paramètres :

**mode** true pour activer l'antialiasing, false pour le désactiver.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→setConsoleBackground()[displaylayer setConsoleBackground: ]

YDisplayLayer

Configure la couleur de fond utilisée par la fonction `clearConsole` et par le défilement automatique de la console.

js	function <b>setConsoleBackground</b> ( <b>bgcol</b> )
nodejs	function <b>setConsoleBackground</b> ( <b>bgcol</b> )
php	function <b>setConsoleBackground</b> ( <b>\$bgcol</b> )
c++	int <b>setConsoleBackground</b> ( int <b>bgcol</b> )
m	-(int) <b>setConsoleBackground</b> : (int) <b>bgcol</b>
pas	function <b>setConsoleBackground</b> ( <b>bgcol</b> : LongInt): LongInt
vb	function <b>setConsoleBackground</b> ( ) As Integer
cs	int <b>setConsoleBackground</b> ( int <b>bgcol</b> )
java	int <b>setConsoleBackground</b> ( int <b>bgcol</b> )
py	def <b>setConsoleBackground</b> ( <b>bgcol</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>setConsoleBackground</b> <b>bgcol</b>

### Paramètres :

**bgcol** le niveau de gris à utiliser pour le fond lors de défilement (0 = noir, 255 = blanc), ou -1 pour un fond transparent

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Configure les marges d'affichage pour la fonction `consoleOut`.

js	function <b>setConsoleMargins</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
nodejs	function <b>setConsoleMargins</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
php	function <b>setConsoleMargins</b> ( <b>\$x1</b> , <b>\$y1</b> , <b>\$x2</b> , <b>\$y2</b> )
cpp	int <b>setConsoleMargins</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
m	<b>-(int) setConsoleMargins</b> : (int) <b>x1</b> : (int) <b>y1</b> : (int) <b>x2</b> : (int) <b>y2</b>
pas	function <b>setConsoleMargins</b> ( <b>x1</b> : LongInt, <b>y1</b> : LongInt, <b>x2</b> : LongInt, <b>y2</b> : LongInt): LongInt
vb	function <b>setConsoleMargins</b> ( ) As Integer
cs	int <b>setConsoleMargins</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
java	int <b>setConsoleMargins</b> ( int <b>x1</b> , int <b>y1</b> , int <b>x2</b> , int <b>y2</b> )
py	def <b>setConsoleMargins</b> ( <b>x1</b> , <b>y1</b> , <b>x2</b> , <b>y2</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>setConsoleMargins</b> <b>x1 y1 x2 y2</b>

### Paramètres :

**x1** la distance en pixels depuis la gauche de la couche jusqu'à la marge gauche  
**y1** la distance en pixels depuis le haut de la couche jusqu'à la marge supérieure  
**x2** la distance en pixels depuis la gauche de la couche jusqu'à la marge droite  
**y2** la distance en pixels depuis le haut de la couche jusqu'à la marge inférieure

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## displaylayer→setConsoleWordWrap()[displaylayer setConsoleWordWrap: ]

## YDisplayLayer

Configure le mode de retour à la ligne utilisé par la fonction `consoleOut`.

js	function <b>setConsoleWordWrap</b> ( <b>wordwrap</b> )
nodejs	function <b>setConsoleWordWrap</b> ( <b>wordwrap</b> )
php	function <b>setConsoleWordWrap</b> ( <b>\$wordwrap</b> )
cpp	int <b>setConsoleWordWrap</b> ( bool <b>wordwrap</b> )
m	-(int) <b>setConsoleWordWrap</b> : (bool) <b>wordwrap</b>
pas	function <b>setConsoleWordWrap</b> ( <b>wordwrap</b> : boolean): LongInt
vb	function <b>setConsoleWordWrap</b> ( ) As Integer
cs	int <b>setConsoleWordWrap</b> ( bool <b>wordwrap</b> )
java	int <b>setConsoleWordWrap</b> ( boolean <b>wordwrap</b> )
py	def <b>setConsoleWordWrap</b> ( <b>wordwrap</b> )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>setConsoleWordWrap</b> <b>wordwrap</b>

### Paramètres :

**wordwrap** `true` pour retourner à la ligne entre les mots seulements, `false` pour retourner à l'extrême droite de chaque ligne.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**displaylayer→unhide())[displaylayer unhide]****YDisplayLayer**

Affiche la couche.

js	function <b>unhide</b> ( )
nodejs	function <b>unhide</b> ( )
php	function <b>unhide</b> ( )
cpp	int <b>unhide</b> ( )
m	-(int) <b>unhide</b>
pas	function <b>unhide</b> ( ): LongInt
vb	function <b>unhide</b> ( ) As Integer
cs	int <b>unhide</b> ( )
java	int <b>unhide</b> ( )
py	def <b>unhide</b> ( )
cmd	YDisplay <b>target</b> [-layer <b>layerId</b> ] <b>unhide</b>

Affiche a nouveau la couche après la command hide.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## 3.15. Interface de contrôle de l'alimentation

La librairie de programmation Yoctopuce permet de contrôler la source d'alimentation qui doit être utilisée pour les fonctions du module consommant beaucoup de courant. Le module est par ailleurs capable de couper automatiquement l'alimentation externe lorsqu'il détecte que la tension a trop chuté (batterie épuisée).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_dualpower.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib'); var YDualPower = yoctolib.YDualPower;</code>
php	<code>require_once('yocto_dualpower.php');</code>
c++	<code>#include "yocto_dualpower.h"</code>
m	<code>#import "yocto_dualpower.h"</code>
pas	<code>uses yocto_dualpower;</code>
vb	<code>yocto_dualpower.vb</code>
cs	<code>yocto_dualpower.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YDualPower;</code>
py	<code>from yocto_dualpower import *</code>

### Fonction globales

#### yFindDualPower(func)

Permet de retrouver un contrôle d'alimentation d'après un identifiant donné.

#### yFirstDualPower()

Commence l'énumération des contrôles d'alimentation accessibles par la librairie.

### Méthodes des objets YDualPower

#### dualpower→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du contrôle d'alimentation au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### dualpower→get\_advertisedValue()

Retourne la valeur courante du contrôle d'alimentation (pas plus de 6 caractères).

#### dualpower→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'alimentation.

#### dualpower→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'alimentation.

#### dualpower→get\_extVoltage()

Retourne la tension mesurée sur l'alimentation de puissance externe, en millivolts.

#### dualpower→get\_friendlyName()

Retourne un identifiant global du contrôle d'alimentation au format `NOM_MODULE . NOM_FONCTION`.

#### dualpower→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### dualpower→get\_functionId()

Retourne l'identifiant matériel du contrôle d'alimentation, sans référence au module.

#### dualpower→get\_hardwareId()

Retourne l'identifiant matériel unique du contrôle d'alimentation au format `SERIAL . FUNCTIONID`.

#### dualpower→get\_logicalName()



Retourne le nom logique du contrôle d'alimentation.

#### **dualpower→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **dualpower→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **dualpower→get\_powerControl()**

Retourne le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

#### **dualpower→get\_powerState()**

Retourne la source d'alimentation active pour les fonctions du module consommant beaucoup de courant.

#### **dualpower→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **dualpower→isOnline()**

Vérifie si le module hébergeant le contrôle d'alimentation est joignable, sans déclencher d'erreur.

#### **dualpower→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le contrôle d'alimentation est joignable, sans déclencher d'erreur.

#### **dualpower→load(msValidity)**

Met en cache les valeurs courantes du contrôle d'alimentation, avec une durée de validité spécifiée.

#### **dualpower→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du contrôle d'alimentation, avec une durée de validité spécifiée.

#### **dualpower→nextDualPower()**

Continue l'énumération des contrôles d'alimentation commencée à l'aide de `yFirstDualPower()`.

#### **dualpower→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **dualpower→set\_logicalName(newval)**

Modifie le nom logique du contrôle d'alimentation.

#### **dualpower→set\_powerControl(newval)**

Modifie le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

#### **dualpower→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

#### **dualpower→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YDualPower.FindDualPower() yFindDualPower()yFindDualPower()

YDualPower

Permet de retrouver un contrôle d'alimentation d'après un identifiant donné.

js	function <b>yFindDualPower</b> ( <b>func</b> )
nodejs	function <b>FindDualPower</b> ( <b>func</b> )
php	function <b>yFindDualPower</b> ( <b>\$func</b> )
cpp	YDualPower* <b>yFindDualPower</b> ( const string& <b>func</b> )
m	YDualPower* <b>yFindDualPower</b> ( NSString* <b>func</b> )
pas	function <b>yFindDualPower</b> ( <b>func</b> : string): TYDualPower
vb	function <b>yFindDualPower</b> ( ByVal <b>func</b> As String) As YDualPower
cs	YDualPower <b>FindDualPower</b> ( string <b>func</b> )
java	YDualPower <b>FindDualPower</b> ( String <b>func</b> )
py	def <b>FindDualPower</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le contrôle d'alimentation soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YDualPower.isOnline()` pour tester si le contrôle d'alimentation est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le contrôle d'alimentation sans ambiguïté

### Retourne :

un objet de classe `YDualPower` qui permet ensuite de contrôler le contrôle d'alimentation.

## YDualPower.FirstDualPower() yFirstDualPower()yFirstDualPower()

## YDualPower

Commence l'énumération des contrôles d'alimentation accessibles par la librairie.

js	function <b>yFirstDualPower</b> ( )
nodejs	function <b>FirstDualPower</b> ( )
php	function <b>yFirstDualPower</b> ( )
cpp	YDualPower* <b>yFirstDualPower</b> ( )
m	YDualPower* <b>yFirstDualPower</b> ( )
pas	function <b>yFirstDualPower</b> ( ): TYDualPower
vb	function <b>yFirstDualPower</b> ( ) As YDualPower
cs	YDualPower <b>FirstDualPower</b> ( )
java	YDualPower <b>FirstDualPower</b> ( )
py	def <b>FirstDualPower</b> ( )

Utiliser la fonction `YDualPower.nextDualPower()` pour itérer sur les autres contrôles d'alimentation.

### Retourne :

un pointeur sur un objet `YDualPower`, correspondant à le premier contrôle d'alimentation accessible en ligne, ou `null` si il n'y a pas de contrôles d'alimentation disponibles.

**dualpower→describe()[dualpower describe]****YDualPower**

Retourne un court texte décrivant de manière non-ambigüe l'instance du contrôle d'alimentation au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le contrôle d'alimentation (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**dualpower→get\_advertisedValue()**  
**dualpower→advertisedValue()[dualpower**  
**advertisedValue]**

**YDualPower**

Retourne la valeur courante du contrôle d'alimentation (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YDualPower <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du contrôle d'alimentation (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

## **dualpower→get\_errorMessage() dualpower→errorMessage()[dualpower errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'alimentation.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du contrôle d'alimentation.

**dualpower→get\_errorType()**  
**dualpower→errorType()****YDualPower**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'alimentation.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du contrôle d'alimentation.

**dualpower**→**get\_extVoltage()****YDualPower****dualpower**→**extVoltage()**[**dualpower extVoltage**]

Retourne la tension mesurée sur l'alimentation de puissance externe, en millivolts.

js	function <b>get_extVoltage</b> ( )
nodejs	function <b>get_extVoltage</b> ( )
php	function <b>get_extVoltage</b> ( )
cpp	int <b>get_extVoltage</b> ( )
m	-(int) extVoltage
pas	function <b>get_extVoltage</b> ( ): LongInt
vb	function <b>get_extVoltage</b> ( ) As Integer
cs	int <b>get_extVoltage</b> ( )
java	int <b>get_extVoltage</b> ( )
py	def <b>get_extVoltage</b> ( )
cmd	YDualPower <b>target</b> <b>get_extVoltage</b>

**Retourne :**

un entier représentant la tension mesurée sur l'alimentation de puissance externe, en millivolts

En cas d'erreur, déclenche une exception ou retourne Y\_EXTVOLTAGE\_INVALID.



**dualpower→get\_friendlyName()****YDualPower****dualpower→friendlyName()[dualpower friendlyName]**

Retourne un identifiant global du contrôle d'alimentation au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
c++	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du contrôle d'alimentation si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du contrôle d'alimentation (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le contrôle d'alimentation en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **dualpower→get\_functionDescriptor() dualpower→functionDescriptor()[dualpower functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**dualpower**→**get\_functionId()****YDualPower****dualpower**→**functionId()**[**dualpower functionId**]

Retourne l'identifiant matériel du contrôle d'alimentation, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le contrôle d'alimentation (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**dualpower**→**get\_hardwareId()****YDualPower****dualpower**→**hardwareId()**[**dualpower hardwareId**]

Retourne l'identifiant matériel unique du contrôle d'alimentation au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du contrôle d'alimentation (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le contrôle d'alimentation (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**dualpower**→**get\_logicalName()****YDualPower****dualpower**→**logicalName()**[**dualpower** **logicalName**]

Retourne le nom logique du contrôle d'alimentation.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YDualPower <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du contrôle d'alimentation. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**dualpower**→**get\_module()****dualpower**→**module()**[**dualpower module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

## **dualpower→get\_module\_async()** **dualpower→module\_async()**

**YDualPower**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

### **Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### **Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## **dualpower→get\_powerControl() dualpower→powerControl()[dualpower powerControl]**

YDualPower

Retourne le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

js	function <b>get_powerControl</b> ( )
nodejs	function <b>get_powerControl</b> ( )
php	function <b>get_powerControl</b> ( )
cpp	Y_POWERCONTROL_enum <b>get_powerControl</b> ( )
m	-(Y_POWERCONTROL_enum) powerControl
pas	function <b>get_powerControl</b> ( ): Integer
vb	function <b>get_powerControl</b> ( ) As Integer
cs	int <b>get_powerControl</b> ( )
java	int <b>get_powerControl</b> ( )
py	def <b>get_powerControl</b> ( )
cmd	YDualPower <b>target</b> <b>get_powerControl</b>

**Retourne :**

une valeur parmi Y\_POWERCONTROL\_AUTO, Y\_POWERCONTROL\_FROM\_USB, Y\_POWERCONTROL\_FROM\_EXT et Y\_POWERCONTROL\_OFF représentant le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant

En cas d'erreur, déclenche une exception ou retourne Y\_POWERCONTROL\_INVALID.



**dualpower→get\_powerState()****YDualPower****dualpower→powerState()[dualpower powerState]**

Retourne la source d'alimentation active pour les fonctions du module consommant beaucoup de courant.

js	function <b>get_powerState</b> ( )
nodejs	function <b>get_powerState</b> ( )
php	function <b>get_powerState</b> ( )
cpp	Y_POWERSTATE_enum <b>get_powerState</b> ( )
m	-(Y_POWERSTATE_enum) powerState
pas	function <b>get_powerState</b> ( ): Integer
vb	function <b>get_powerState</b> ( ) As Integer
cs	int <b>get_powerState</b> ( )
java	int <b>get_powerState</b> ( )
py	def <b>get_powerState</b> ( )
cmd	YDualPower <b>target</b> <b>get_powerState</b>

**Retourne :**

une valeur parmi Y\_POWERSTATE\_OFF, Y\_POWERSTATE\_FROM\_USB et Y\_POWERSTATE\_FROM\_EXT représentant la source d'alimentation active pour les fonctions du module consommant beaucoup de courant

En cas d'erreur, déclenche une exception ou retourne Y\_POWERSTATE\_INVALID.

**dualpower→get\_userdata()****YDualPower****dualpower→userdata()[dualpower userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**dualpower**→**isOnline()**[dualpower isOnline]**YDualPower**

Vérifie si le module hébergeant le contrôle d'alimentation est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du contrôle d'alimentation sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le contrôle d'alimentation est joignable, `false` sinon

Vérifie si le module hébergeant le contrôle d'alimentation est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du contrôle d'alimentation sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**dualpower→load()[dualpower load: ]****YDualPower**

Met en cache les valeurs courantes du contrôle d'alimentation, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes du contrôle d'alimentation, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## dualpower→nextDualPower()[dualpower nextDualPower]

## YDualPower

Continue l'énumération des contrôles d'alimentation commencée à l'aide de `yFirstDualPower()`.

js	function <b>nextDualPower</b> ( )
nodejs	function <b>nextDualPower</b> ( )
php	function <b>nextDualPower</b> ( )
cpp	YDualPower * <b>nextDualPower</b> ( )
m	-(YDualPower*) <b>nextDualPower</b>
pas	function <b>nextDualPower</b> ( ): TYDualPower
vb	function <b>nextDualPower</b> ( ) As YDualPower
cs	YDualPower <b>nextDualPower</b> ( )
java	YDualPower <b>nextDualPower</b> ( )
py	def <b>nextDualPower</b> ( )

### Retourne :

un pointeur sur un objet `YDualPower` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## dualpower→registerValueCallback()[dualpower registerValueCallback: ]

YDualPower

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
c++	int <b>registerValueCallback</b> ( YDualPowerValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YDualPowerValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYDualPowerValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.



**dualpower**→**set\_logicalName()****YDualPower****dualpower**→**setLogicalName()**[**dualpower**  
**setLogicalName:** ]

Modifie le nom logique du contrôle d'alimentation.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YDualPower <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du contrôle d'alimentation.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**dualpower→set\_powerControl()****YDualPower****dualpower→setPowerControl()[dualpower  
setPowerControl: ]**

Modifie le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

js	function <b>set_powerControl</b> ( <b>newval</b> )
nodejs	function <b>set_powerControl</b> ( <b>newval</b> )
php	function <b>set_powerControl</b> ( <b>\$newval</b> )
cpp	int <b>set_powerControl</b> ( Y_POWERCONTROL_enum <b>newval</b> )
m	-(int) setPowerControl : (Y_POWERCONTROL_enum) <b>newval</b>
pas	function <b>set_powerControl</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_powerControl</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_powerControl</b> ( int <b>newval</b> )
java	int <b>set_powerControl</b> ( int <b>newval</b> )
py	def <b>set_powerControl</b> ( <b>newval</b> )
cmd	YDualPower <b>target set_powerControl newval</b>

**Paramètres :**

**newval** une valeur parmi Y\_POWERCONTROL\_AUTO, Y\_POWERCONTROL\_FROM\_USB, Y\_POWERCONTROL\_FROM\_EXT et Y\_POWERCONTROL\_OFF représentant le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**dualpower**→**set\_userdata()****YDualPower****dualpower**→**setUserData()**[**dualpower setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**dualpower→wait\_async()****YDualPower**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.16. Interface de la fonction Files

L'interface de stockage de fichiers permet de stocker des fichiers sur certains modules, par exemple pour personnaliser un service web (dans le cas d'un module connecté au réseau) ou pour ajouter un police de caractères (dans le cas d'un module d'affichage).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_files.js'></script>
nodejs	var yocotolib = require('yocotolib'); var YFiles = yocotolib.YFiles;
php	require_once('yocto_files.php');
c++	#include "yocto_files.h"
m	#import "yocto_files.h"
pas	uses yocto_files;
vb	yocto_files.vb
cs	yocto_files.cs
java	import com.yoctopuce.YoctoAPI.YFiles;
py	from yocto_files import *

### Fonction globales

#### yFindFiles(func)

Permet de retrouver un système de fichier d'après un identifiant donné.

#### yFirstFiles()

Commence l'énumération des système de fichier accessibles par la librairie.

### Méthodes des objets YFiles

#### files→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### files→download(pathname)

Télécharge le fichier choisi du filesystem et retourne son contenu.

#### files→download\_async(pathname, callback, context)

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

#### files→format\_fs()

Rétabli le système de fichier dans on état original, défragmenté.

#### files→get\_advertisedValue()

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

#### files→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

#### files→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

#### files→get\_filesCount()

Retourne le nombre de fichiers présents dans le système de fichier.

#### files→get\_freeSpace()

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

#### files→get\_friendlyName()

Retourne un identifiant global du système de fichier au format `NOM_MODULE . NOM_FONCTION`.

#### files→get\_functionDescriptor()

### 3. Reference

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **files**→**get\_functionId()**

Retourne l'identifiant matériel du système de fichier, sans référence au module.

#### **files**→**get\_hardwareId()**

Retourne l'identifiant matériel unique du système de fichier au format `SERIAL . FUNCTIONID`.

#### **files**→**get\_list(pattern)**

Retourne une liste d'objets objet `YFileRecord` qui décrivent les fichiers présents dans le système de fichier.

#### **files**→**get\_logicalName()**

Retourne le nom logique du système de fichier.

#### **files**→**get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **files**→**get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **files**→**get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **files**→**isOnline()**

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

#### **files**→**isOnline\_async(callback, context)**

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

#### **files**→**load(msValidity)**

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

#### **files**→**load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

#### **files**→**nextFiles()**

Continue l'énumération des système de fichier commencée à l'aide de `yFirstFiles()`.

#### **files**→**registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **files**→**remove(pathname)**

Efface un fichier, spécifié par son path complet, du système de fichier.

#### **files**→**set\_logicalName(newval)**

Modifie le nom logique du système de fichier.

#### **files**→**set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

#### **files**→**upload(pathname, content)**

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

#### **files**→**wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YFiles.FindFiles() yFindFiles()yFindFiles()

## YFiles

Permet de retrouver un système de fichier d'après un identifiant donné.

js	function <b>yFindFiles</b> ( <b>func</b> )
nodejs	function <b>FindFiles</b> ( <b>func</b> )
php	function <b>yFindFiles</b> ( <b>\$func</b> )
cpp	YFiles* <b>yFindFiles</b> ( string <b>func</b> )
m	+(YFiles*) <b>yFindFiles</b> : (NSString*) <b>func</b>
pas	function <b>yFindFiles</b> ( <b>func</b> : string): TYFiles
vb	function <b>yFindFiles</b> ( ByVal <b>func</b> As String) As YFiles
cs	YFiles <b>FindFiles</b> ( string <b>func</b> )
java	YFiles <b>FindFiles</b> ( String <b>func</b> )
py	def <b>FindFiles</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le système de fichier soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YFiles.isOnline()` pour tester si le système de fichier est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le système de fichier sans ambiguïté

### Retourne :

un objet de classe `YFiles` qui permet ensuite de contrôler le système de fichier.

## YFiles.FirstFiles() yFirstFiles()yFirstFiles()

YFiles

Commence l'énumération des système de fichier accessibles par la librairie.

js	function <b>yFirstFiles</b> ( )
nodejs	function <b>FirstFiles</b> ( )
php	function <b>yFirstFiles</b> ( )
cpp	YFiles* <b>yFirstFiles</b> ( )
m	YFiles* <b>yFirstFiles</b> ( )
pas	function <b>yFirstFiles</b> ( ): TYFiles
vb	function <b>yFirstFiles</b> ( ) As YFiles
cs	YFiles <b>FirstFiles</b> ( )
java	YFiles <b>FirstFiles</b> ( )
py	def <b>FirstFiles</b> ( )

Utiliser la fonction `YFiles.nextFiles( )` pour itérer sur les autres système de fichier.

### Retourne :

un pointeur sur un objet `YFiles`, correspondant à le premier système de fichier accessible en ligne, ou `null` si il n'y a pas de système de fichier disponibles.



**files→describe()[files describe]****YFiles**

Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le système de fichier (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## files→download()[files download: ]

## YFiles

Télécharge le fichier choisi du filesystem et retourne son contenu.

js	function <b>download</b> ( <b>pathname</b> )
nodejs	function <b>download</b> ( <b>pathname</b> )
php	function <b>download</b> ( <b>\$pathname</b> )
cpp	string <b>download</b> ( string <b>pathname</b> )
m	-(NSData*) <b>download</b> : (NSString*) <b>pathname</b>
pas	function <b>download</b> ( <b>pathname</b> : string): TByteArray
vb	function <b>download</b> ( ) As Byte
py	def <b>download</b> ( <b>pathname</b> )
cmd	YFiles <b>target download</b> <b>pathname</b>

**Paramètres :**

**pathname** nom complet du fichier à charger, y compris le chemin d'accès.

**Retourne :**

le contenu du fichier chargé sous forme d'objet binaire

En cas d'erreur, déclenche une exception ou retourne un contenu vide.

**files→download\_async()****YFiles**

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

```
js function download_async( pathname, callback, context)
```

```
nodejs function download_async( pathname, callback, context)
```

**Paramètres :**

**pathname** nom complet du fichier à charger, y compris le chemin d'accès.

**callback** fonction fournie par l'utilisateur, qui sera appelée lorsque la suite du chargement aura été effectué. La fonction callback doit prendre trois arguments: - la variable de contexte à disposition de l'utilisateur - l'objet YFiles dont la méthode download\_async a été appelée - le contenu du fichier chargé sous forme d'objet binaire

**context** variable de contexte à disposition de l'utilisateur

**Retourne :**

rien.

**files→format\_fs()[files format\_fs]****YFiles**

Rétabli le système de fichier dans on état original, défragmenté.

js	function <b>format_fs</b> ( )
nodejs	function <b>format_fs</b> ( )
php	function <b>format_fs</b> ( )
cpp	int <b>format_fs</b> ( )
m	-(int) <b>format_fs</b>
pas	function <b>format_fs</b> ( ): LongInt
vb	function <b>format_fs</b> ( ) As Integer
cs	int <b>format_fs</b> ( )
java	int <b>format_fs</b> ( )
py	def <b>format_fs</b> ( )
cmd	YFiles <b>target</b> <b>format_fs</b>

entièrement vide. Tous les fichiers précédemment chargés sont irrémédiablement effacés.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**files**→**get\_advertisedValue()****YFiles****files**→**advertisedValue()**[files advertisedValue]

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YFiles <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du système de fichier (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**files→get\_errorMessage()****files→errorMessage()[files errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

**files→get\_errorType()****YFiles****files→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

**files**→**get\_filesCount()****files**→**filesCount()**[files filesCount]

Retourne le nombre de fichiers présents dans le système de fichier.

js	function <b>get_filesCount</b> ( )
nodejs	function <b>get_filesCount</b> ( )
php	function <b>get_filesCount</b> ( )
c++	int <b>get_filesCount</b> ( )
m	-(int) filesCount
pas	function <b>get_filesCount</b> ( ): LongInt
vb	function <b>get_filesCount</b> ( ) As Integer
cs	int <b>get_filesCount</b> ( )
java	int <b>get_filesCount</b> ( )
py	def <b>get_filesCount</b> ( )
cmd	YFiles <b>target</b> <b>get_filesCount</b>

**Retourne :**

un entier représentant le nombre de fichiers présents dans le système de fichier

En cas d'erreur, déclenche une exception ou retourne Y\_FILES\_COUNT\_INVALID.



**files**→**get\_freeSpace()****YFiles****files**→**freeSpace()**[files freeSpace]

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

js	function <b>get_freeSpace</b> ( )
nodejs	function <b>get_freeSpace</b> ( )
php	function <b>get_freeSpace</b> ( )
cpp	int <b>get_freeSpace</b> ( )
m	-(int) freeSpace
pas	function <b>get_freeSpace</b> ( ): LongInt
vb	function <b>get_freeSpace</b> ( ) As Integer
cs	int <b>get_freeSpace</b> ( )
java	int <b>get_freeSpace</b> ( )
py	def <b>get_freeSpace</b> ( )
cmd	YFiles <b>target</b> <b>get_freeSpace</b>

**Retourne :**

un entier représentant l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets

En cas d'erreur, déclenche une exception ou retourne Y\_FREESPACE\_INVALID.

**files**→**get\_friendlyName()****files**→**friendlyName()**[files friendlyName]

Retourne un identifiant global du système de fichier au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du système de fichier si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du système de fichier (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le système de fichier en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**files→get\_functionDescriptor()****YFiles****files→functionDescriptor()[files functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**files**→**get\_functionId()****files**→**functionId()**[files functionId]

Retourne l'identifiant matériel du système de fichier, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le système de fichier (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**files→get\_hardwareId()****YFiles****files→hardwareId()[files hardwareId]**

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du système de fichier (par exemple RELAYLO1-123456.relay1).

**Retourne :**

une chaîne de caractères identifiant le système de fichier (ex: RELAYLO1-123456.relay1) En cas d'erreur, déclenche une exception ou retourne Y\_HARDWAREID\_INVALID.

**files→get\_list()****files→list()[files list: ]**

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

js	function <b>get_list</b> ( <b>pattern</b> )
nodejs	function <b>get_list</b> ( <b>pattern</b> )
php	function <b>get_list</b> ( <b>\$pattern</b> )
cpp	vector<YFileRecord> <b>get_list</b> ( string <b>pattern</b> )
m	-(NSMutableArray*) list : (NSString*) <b>pattern</b>
pas	function <b>get_list</b> ( <b>pattern</b> : string): TYFileRecordArray
vb	function <b>get_list</b> ( ) As List
cs	List<YFileRecord> <b>get_list</b> ( string <b>pattern</b> )
java	ArrayList<YFileRecord> <b>get_list</b> ( String <b>pattern</b> )
py	def <b>get_list</b> ( <b>pattern</b> )
cmd	YFiles <b>target get_list pattern</b>

**Paramètres :**

**pattern** un filtre optionel sur les noms de fichiers retournés, pouvant contenir des astérisques et des points d'interrogations comme jokers. Si le pattern fourni est vide, tous les fichiers sont retournés.

**Retourne :**

une liste d'objets YFileRecord, contenant le nom complet (y compris le chemin d'accès), la taille en octets et le CRC 32-bit du contenu du fichier.

En cas d'erreur, déclenche une exception ou retourne une liste vide.

**files**→**get\_logicalName()****YFiles****files**→**logicalName()**[files logicalName]

Retourne le nom logique du système de fichier.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YFiles <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du système de fichier. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**files**→**get\_module()****files**→**module()**[files module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule



**files**→**get\_module\_async()****YFiles****files**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**files→get\_userdata()****files→userdata()[files userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**files→isOnline()[files isOnline]****YFiles**

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le système de fichier est joignable, `false` sinon

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**files→load()**[files load: ]**YFiles**

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**files→nextFiles()[files nextFiles]****YFiles**

Continue l'énumération des système de fichier commencée à l'aide de `yFirstFiles()`.

js	function <b>nextFiles</b> ( )
nodejs	function <b>nextFiles</b> ( )
php	function <b>nextFiles</b> ( )
cpp	YFiles * <b>nextFiles</b> ( )
m	-(YFiles*) <b>nextFiles</b>
pas	function <b>nextFiles</b> ( ): TYFiles
vb	function <b>nextFiles</b> ( ) As YFiles
cs	YFiles <b>nextFiles</b> ( )
java	YFiles <b>nextFiles</b> ( )
py	def <b>nextFiles</b> ( )

**Retourne :**

un pointeur sur un objet `YFiles` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## files→registerValueCallback()[files registerValueCallback: ]

YFiles

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YFilesValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YFilesValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYFilesValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.



**files→remove()[files remove: ]****YFiles**

Efface un fichier, spécifié par son path complet, du système de fichier.

js	function <b>remove</b> ( <b>pathname</b> )
nodejs	function <b>remove</b> ( <b>pathname</b> )
php	function <b>remove</b> ( <b>\$pathname</b> )
cpp	int <b>remove</b> ( string <b>pathname</b> )
m	-(int) <b>remove</b> : (NSString*) <b>pathname</b>
pas	function <b>remove</b> ( <b>pathname</b> : string): LongInt
vb	function <b>remove</b> ( ) As Integer
cs	int <b>remove</b> ( string <b>pathname</b> )
java	int <b>remove</b> ( String <b>pathname</b> )
py	def <b>remove</b> ( <b>pathname</b> )
cmd	YFiles <b>target remove pathname</b>

A cause de la fragmentation, l'effacement d'un fichier ne libère pas toujours la totalité de l'espace qu'il occupe. Par contre, la ré-écriture d'un fichier du même nom récupérera dans tout les cas l'espace qui n'aurait éventuellement pas été libéré. Pour s'assurer de libérer la totalité de l'espace du système de fichier, utilisez la fonction `format_fs`.

**Paramètres :**

**pathname** nom complet du fichier, y compris le chemin d'accès.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**files**→**set\_logicalName()****files**→**setLogicalName()**[files setLogicalName: ]

Modifie le nom logique du système de fichier.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YFiles <b>target set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du système de fichier.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**files→set\_userdata()****YFiles****files→setUserData()[files setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## files→upload()[files upload: ]

## YFiles

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

js	function <b>upload</b> ( <b>pathname</b> , <b>content</b> )
nodejs	function <b>upload</b> ( <b>pathname</b> , <b>content</b> )
php	function <b>upload</b> ( <b>\$pathname</b> , <b>\$content</b> )
c++	int <b>upload</b> ( string <b>pathname</b> , string <b>content</b> )
m	-(int) <b>upload</b> : (NSString*) <b>pathname</b> : (NSData*) <b>content</b>
pas	function <b>upload</b> ( <b>pathname</b> : string, <b>content</b> : TByteArray): LongInt
vb	procedure <b>upload</b> ( )
cs	int <b>upload</b> ( string <b>pathname</b> )
java	int <b>upload</b> ( String <b>pathname</b> )
py	def <b>upload</b> ( <b>pathname</b> , <b>content</b> )
cmd	YFiles <b>target upload</b> <b>pathname</b> <b>content</b>

Si un fichier existe déjà pour le même chemin d'accès, son contenu est remplacé.

**Paramètres :**

**pathname** nom complet du fichier, y compris le chemin d'accès.  
**content** contenu du fichier à télécharger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**files→wait\_async()****YFiles**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.17. Interface de la fonction GenericSensor

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_genericsensor.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YGenericSensor = yoctolib.YGenericSensor;
php	require_once('yocto_genericsensor.php');
c++	#include "yocto_genericsensor.h"
m	#import "yocto_genericsensor.h"
pas	uses yocto_genericsensor;
vb	yocto_genericsensor.vb
cs	yocto_genericsensor.cs
java	import com.yoctopuce.YoctoAPI.YGenericSensor;
py	from yocto_genericsensor import *

### Fonction globales

#### yFindGenericSensor(func)

Permet de retrouver un capteur générique d'après un identifiant donné.

#### yFirstGenericSensor()

Commence l'énumération des capteurs génériques accessibles par la librairie.

### Méthodes des objets YGenericSensor

#### genericsensor→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### genericsensor→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur générique au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### genericsensor→get\_advertisedValue()

Retourne la valeur courante du capteur générique (pas plus de 6 caractères).

#### genericsensor→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### genericsensor→get\_currentValue()

Retourne la valeur mesurée actuelle.

#### genericsensor→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur générique.

#### genericsensor→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur générique.

#### genericsensor→get\_friendlyName()

Retourne un identifiant global du capteur générique au format `NOM_MODULE . NOM_FONCTION`.

#### genericsensor→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### genericsensor→get\_functionId()

Retourne l'identifiant matériel du capteur générique, sans référence au module.

#### genericsensor→get\_hardwareId()

Retourne l'identifiant matériel unique du capteur générique au format `SERIAL.FUNCTIONID`.

**`genericsensor→get_highestValue()`**

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

**`genericsensor→get_logFrequency()`**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**`genericsensor→get_logicalName()`**

Retourne le nom logique du capteur générique.

**`genericsensor→get_lowestValue()`**

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

**`genericsensor→get_module()`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`genericsensor→get_module_async(callback, context)`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`genericsensor→get_recordedData(startTime, endTime)`**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**`genericsensor→get_reportFrequency()`**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**`genericsensor→get_resolution()`**

Retourne la résolution des valeurs mesurées.

**`genericsensor→get_signalRange()`**

Retourne la plage de signal électrique utilisée par le capteur.

**`genericsensor→get_signalUnit()`**

Retourne l'unité du signal électrique utilisée par le capteur.

**`genericsensor→get_signalValue()`**

Retourne la valeur mesurée du signal électrique utilisée par le capteur.

**`genericsensor→get_unit()`**

Retourne l'unité dans laquelle la mesure est exprimée.

**`genericsensor→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`genericsensor→get_valueRange()`**

Retourne la plage de valeurs physiques mesurés par le capteur.

**`genericsensor→isOnline()`**

Vérifie si le module hébergeant le capteur générique est joignable, sans déclencher d'erreur.

**`genericsensor→isOnline_async(callback, context)`**

Vérifie si le module hébergeant le capteur générique est joignable, sans déclencher d'erreur.

**`genericsensor→load(msValidity)`**

Met en cache les valeurs courantes du capteur générique, avec une durée de validité spécifiée.

**`genericsensor→loadCalibrationPoints(rawValues, refValues)`**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

**`genericsensor→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes du capteur générique, avec une durée de validité spécifiée.

**`genericsensor→nextGenericSensor()`**

### 3. Reference

Continue l'énumération des capteurs génériques commencée à l'aide de `yFirstGenericSensor()`.

**`genericsensor→registerTimedReportCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**`genericsensor→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`genericsensor→set_highestValue(newval)`**

Modifie la mémoire de valeur maximale observée.

**`genericsensor→set_logFrequency(newval)`**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**`genericsensor→set_logicalName(newval)`**

Modifie le nom logique du capteur générique.

**`genericsensor→set_lowestValue(newval)`**

Modifie la mémoire de valeur minimale observée.

**`genericsensor→set_reportFrequency(newval)`**

Modifie la fréquence de notification périodique des valeurs mesurées.

**`genericsensor→set_resolution(newval)`**

Modifie la résolution des valeurs physique mesurées.

**`genericsensor→set_signalRange(newval)`**

Modifie la plage de signal électrique utilisée par le capteur.

**`genericsensor→set_unit(newval)`**

Change l'unité dans laquelle la valeur mesurée est exprimée.

**`genericsensor→set_userData(data)`**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**`genericsensor→set_valueRange(newval)`**

Modifie la plage de valeurs physiques mesurés par le capteur.

**`genericsensor→wait_async(callback, context)`**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YGenericSensor.FindGenericSensor() yFindGenericSensor()yFindGenericSensor()

## YGenericSensor

Permet de retrouver un capteur générique d'après un identifiant donné.

js	function <b>yFindGenericSensor</b> ( <b>func</b> )
nodejs	function <b>FindGenericSensor</b> ( <b>func</b> )
php	function <b>yFindGenericSensor</b> ( <b>\$func</b> )
cpp	YGenericSensor* <b>yFindGenericSensor</b> ( const string& <b>func</b> )
m	YGenericSensor* <b>yFindGenericSensor</b> ( NSString* <b>func</b> )
pas	function <b>yFindGenericSensor</b> ( <b>func</b> : string): TYGenericSensor
vb	function <b>yFindGenericSensor</b> ( ByVal <b>func</b> As String) As YGenericSensor
cs	YGenericSensor <b>FindGenericSensor</b> ( string <b>func</b> )
java	YGenericSensor <b>FindGenericSensor</b> ( String <b>func</b> )
py	def <b>FindGenericSensor</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur générique soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YGenericSensor.isOnline()` pour tester si le capteur générique est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur générique sans ambiguïté

### Retourne :

un objet de classe `YGenericSensor` qui permet ensuite de contrôler le capteur générique.

## YGenericSensor.FirstGenericSensor() yFirstGenericSensor()yFirstGenericSensor()

## YGenericSensor

Commence l'énumération des capteurs génériques accessibles par la librairie.

js	function <b>yFirstGenericSensor</b> ( )
nodejs	function <b>FirstGenericSensor</b> ( )
php	function <b>yFirstGenericSensor</b> ( )
cpp	YGenericSensor* <b>yFirstGenericSensor</b> ( )
m	YGenericSensor* <b>yFirstGenericSensor</b> ( )
pas	function <b>yFirstGenericSensor</b> ( ): TYGenericSensor
vb	function <b>yFirstGenericSensor</b> ( ) As YGenericSensor
cs	YGenericSensor <b>FirstGenericSensor</b> ( )
java	YGenericSensor <b>FirstGenericSensor</b> ( )
py	def <b>FirstGenericSensor</b> ( )

Utiliser la fonction `YGenericSensor.nextGenericSensor( )` pour itérer sur les autres capteurs génériques.

### Retourne :

un pointeur sur un objet `YGenericSensor`, correspondant à le premier capteur générique accessible en ligne, ou `null` si il n'y a pas de capteurs génériques disponibles.

## genericsensor→calibrateFromPoints()[genericsensor calibrateFromPoints: ]

## YGenericSensor

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

js	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
nodejs	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
php	function <b>calibrateFromPoints</b> ( <b>\$rawValues</b> , <b>\$refValues</b> )
cpp	int <b>calibrateFromPoints</b> ( vector<double> <b>rawValues</b> , vector<double> <b>refValues</b> )
m	-(int) <b>calibrateFromPoints</b> : (NSMutableArray*) <b>rawValues</b> : (NSMutableArray*) <b>refValues</b>
pas	function <b>calibrateFromPoints</b> ( <b>rawValues</b> : TDoubleArray, <b>refValues</b> : TDoubleArray): LongInt
vb	procedure <b>calibrateFromPoints</b> ( )
cs	int <b>calibrateFromPoints</b> ( List<double> <b>rawValues</b> , List<double> <b>refValues</b> )
java	int <b>calibrateFromPoints</b> ( ArrayList<Double> <b>rawValues</b> , ArrayList<Double> <b>refValues</b> )
py	def <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
cmd	YGenericSensor <b>target</b> <b>calibrateFromPoints</b> <b>rawValues</b> <b>refValues</b>

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→describe()[genericsensor describe]****YGenericSensor**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur générique au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur générique (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**genericsensor→get\_advertisedValue()****YGenericSensor****genericsensor→advertisedValue()[genericsensor  
advertisedValue]**

Retourne la valeur courante du capteur générique (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur générique (pas plus de 6 caractères).  
En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**genericsensor→get\_currentRawValue()****YGenericSensor****genericsensor→currentRawValue()[genericsensor  
currentRawValue]**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**genericsensor**→**get\_currentValue()****YGenericSensor****genericsensor**→**currentValue()**[**genericsensor**  
**currentValue**]

Retourne la valeur mesurée actuelle.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur mesurée actuelle

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**genericsensor**→**get\_errorMessage()****YGenericSensor****genericsensor**→**errorMessage()**[**genericsensor**  
**errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur générique.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur générique.



**genericsensor→get\_errorType()**  
**genericsensor→errorType()****YGenericSensor**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur générique.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur générique.

**genericsensor**→**get\_friendlyName()****YGenericSensor****genericsensor**→**friendlyName()**[**genericsensor**  
**friendlyName**]

Retourne un identifiant global du capteur générique au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur générique si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur générique (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur générique en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**genericsensor→get\_functionDescriptor()****YGenericSensor****genericsensor→functionDescriptor()[genericsensor  
functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**genericsensor**→**get\_functionId()****YGenericSensor****genericsensor**→**functionId()**[**genericsensor**  
**functionId**]

Retourne l'identifiant matériel du capteur générique, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple relay1.

**Retourne :**

une chaîne de caractères identifiant le capteur générique (ex: relay1) En cas d'erreur, déclenche une exception ou retourne Y\_FUNCTIONID\_INVALID.

**genericsensor**→**get\_hardwareId()**  
**genericsensor**→**hardwareId()**[**genericsensor**  
**hardwareId**]

**YGenericSensor**

Retourne l'identifiant matériel unique du capteur générique au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur générique (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur générique (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**genericsensor**→**get\_highestValue()****YGenericSensor****genericsensor**→**highestValue()**[**genericsensor**  
**highestValue**]

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la mesure depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**genericsensor→get\_logFrequency()****YGenericSensor****genericsensor→logFrequency()[genericsensor  
logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
c++	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**genericsensor**→**get\_logicalName()****YGenericSensor****genericsensor**→**logicalName()**[**genericsensor**  
**logicalName**]

Retourne le nom logique du capteur générique.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur générique. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.



**genericsensor**→**get\_lowestValue()****YGenericSensor****genericsensor**→**lowestValue()**[**genericsensor**  
**lowestValue**]

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la mesure depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**genericsensor**→**get\_module()****YGenericSensor****genericsensor**→**module()**[**genericsensor module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**genericsensor→get\_module\_async()****YGenericSensor****genericsensor→module\_async()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**genericsensor**→**get\_recordedData()****YGenericSensor****genericsensor**→**recordedData()**[**genericsensor**  
**recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YGenericSensor <b>target get_recordedData startTime endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**genericsensor→get\_reportFrequency()****YGenericSensor****genericsensor→reportFrequency()[genericsensor  
reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**genericsensor**→**get\_resolution()**  
**genericsensor**→**resolution()**[**genericsensor**  
**resolution**]

**YGenericSensor**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**genericsensor→get\_signalRange()****YGenericSensor****genericsensor→signalRange()[genericsensor  
signalRange]**

Retourne la plage de signal électrique utilisée par le capteur.

js	function <b>get_signalRange</b> ( )
nodejs	function <b>get_signalRange</b> ( )
php	function <b>get_signalRange</b> ( )
cpp	string <b>get_signalRange</b> ( )
m	-(NSString*) signalRange
pas	function <b>get_signalRange</b> ( ): string
vb	function <b>get_signalRange</b> ( ) As String
cs	string <b>get_signalRange</b> ( )
java	String <b>get_signalRange</b> ( )
py	def <b>get_signalRange</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_signalRange</b>

**Retourne :**

une chaîne de caractères représentant la plage de signal électrique utilisée par le capteur

En cas d'erreur, déclenche une exception ou retourne Y\_SIGNALRANGE\_INVALID.

**genericsensor**→**get\_signalUnit()****YGenericSensor****genericsensor**→**signalUnit()**[**genericsensor**  
**signalUnit**]

Retourne l'unité du signal électrique utilisée par le capteur.

js	function <b>get_signalUnit</b> ( )
nodejs	function <b>get_signalUnit</b> ( )
php	function <b>get_signalUnit</b> ( )
cpp	string <b>get_signalUnit</b> ( )
m	-(NSString*) signalUnit
pas	function <b>get_signalUnit</b> ( ): string
vb	function <b>get_signalUnit</b> ( ) As String
cs	string <b>get_signalUnit</b> ( )
java	String <b>get_signalUnit</b> ( )
py	def <b>get_signalUnit</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_signalUnit</b>

**Retourne :**

une chaîne de caractères représentant l'unité du signal électrique utilisée par le capteur

En cas d'erreur, déclenche une exception ou retourne Y\_SIGNALUNIT\_INVALID.



**genericsensor**→**get\_signalValue()**  
**genericsensor**→**signalValue()**[**genericsensor**  
**signalValue**]

**YGenericSensor**

Retourne la valeur mesurée du signal électrique utilisée par le capteur.

js	function <b>get_signalValue</b> ( )
nodejs	function <b>get_signalValue</b> ( )
php	function <b>get_signalValue</b> ( )
cpp	double <b>get_signalValue</b> ( )
m	-(double) signalValue
pas	function <b>get_signalValue</b> ( ): double
vb	function <b>get_signalValue</b> ( ) As Double
cs	double <b>get_signalValue</b> ( )
java	double <b>get_signalValue</b> ( )
py	def <b>get_signalValue</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_signalValue</b>

**Retourne :**

une valeur numérique représentant la valeur mesurée du signal électrique utilisée par le capteur

En cas d'erreur, déclenche une exception ou retourne Y\_SIGNALVALUE\_INVALID.

**genericsensor**→**get\_unit()****YGenericSensor****genericsensor**→**unit()**[**genericsensor unit**]

Retourne l'unité dans laquelle la mesure est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la mesure est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**genericsensor→get\_userData()****YGenericSensor****genericsensor→userData()[genericsensor userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**genericsensor**→**get\_valueRange()****YGenericSensor****genericsensor**→**valueRange()**[**genericsensor**  
**valueRange**]

Retourne la plage de valeurs physiques mesurés par le capteur.

js	function <b>get_valueRange</b> ( )
nodejs	function <b>get_valueRange</b> ( )
php	function <b>get_valueRange</b> ( )
cpp	string <b>get_valueRange</b> ( )
m	-(NSString*) valueRange
pas	function <b>get_valueRange</b> ( ): string
vb	function <b>get_valueRange</b> ( ) As String
cs	string <b>get_valueRange</b> ( )
java	String <b>get_valueRange</b> ( )
py	def <b>get_valueRange</b> ( )
cmd	YGenericSensor <b>target</b> <b>get_valueRange</b>

**Retourne :**

une chaîne de caractères représentant la plage de valeurs physiques mesurés par le capteur

En cas d'erreur, déclenche une exception ou retourne Y\_VALUERANGE\_INVALID.

**genericsensor→isOnline()[genericsensor isOnline]****YGenericSensor**

Vérifie si le module hébergeant le capteur générique est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur générique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le capteur générique est joignable, `false` sinon

**genericsensor→isOnline\_async()****YGenericSensor**

Vérifie si le module hébergeant le capteur générique est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur générique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**genericsensor→load()[genericsensor load: ]****YGenericSensor**

Met en cache les valeurs courantes du capteur générique, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## genericsensor→loadCalibrationPoints() [genericsensor loadCalibrationPoints: ]

YGenericSensor

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)
java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)
py def loadCalibrationPoints( rawValues, refValues)
cmd YGenericSensor target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**genericsensor→load\_async()****YGenericSensor**

Met en cache les valeurs courantes du capteur générique, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**genericsensor→nextGenericSensor()[genericsensor  
nextGenericSensor]****YGenericSensor**

Continue l'énumération des capteurs génériques commencée à l'aide de `yFirstGenericSensor()`.

js	function <b>nextGenericSensor</b> ( )
nodejs	function <b>nextGenericSensor</b> ( )
php	function <b>nextGenericSensor</b> ( )
cpp	YGenericSensor * <b>nextGenericSensor</b> ( )
m	-(YGenericSensor*) <b>nextGenericSensor</b>
pas	function <b>nextGenericSensor</b> ( ): TYGenericSensor
vb	function <b>nextGenericSensor</b> ( ) As YGenericSensor
cs	YGenericSensor <b>nextGenericSensor</b> ( )
java	YGenericSensor <b>nextGenericSensor</b> ( )
py	def <b>nextGenericSensor</b> ( )

**Retourne :**

un pointeur sur un objet `YGenericSensor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## genericsensor→registerTimedReportCallback() [genericsensor registerTimedReportCallback: ]

## YGenericSensor

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YGenericSensorTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YGenericSensorTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYGenericSensorTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## genericsensor→registerValueCallback() [genericsensor registerValueCallback: ]

YGenericSensor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YGenericSensorValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YGenericSensorValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYGenericSensorValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**genericsensor→set\_highestValue()****YGenericSensor****genericsensor→setHighestValue()[genericsensor  
setHighestValue: ]**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target</b> <b>set_highestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→set\_logFrequency()****YGenericSensor****genericsensor→setLogFrequency()[genericsensor  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→set\_logicalName()****YGenericSensor****genericsensor→setLogicalName()[genericsensor  
setLogicalName: ]**

Modifie le nom logique du capteur générique.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur générique.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→set\_lowestValue()****YGenericSensor****genericsensor→setLowestValue()[genericsensor  
setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**genericsensor→set\_reportFrequency()**  
**genericsensor→setReportFrequency()**  
**[genericsensor setReportFrequency: ]**

**YGenericSensor**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→set\_resolution()****YGenericSensor****genericsensor→setResolution()[genericsensor  
setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→set\_signalRange()****YGenericSensor****genericsensor→setSignalRange()[genericsensor  
setSignalRange: ]**

Modifie la plage de signal électrique utilisée par le capteur.

js	function <b>set_signalRange</b> ( <b>newval</b> )
nodejs	function <b>set_signalRange</b> ( <b>newval</b> )
php	function <b>set_signalRange</b> ( <b>\$newval</b> )
cpp	int <b>set_signalRange</b> ( const string& <b>newval</b> )
m	-(int) setSignalRange : (NSString*) <b>newval</b>
pas	function <b>set_signalRange</b> ( <b>newval</b> : string): integer
vb	function <b>set_signalRange</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_signalRange</b> ( string <b>newval</b> )
java	int <b>set_signalRange</b> ( String <b>newval</b> )
py	def <b>set_signalRange</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target</b> <b>set_signalRange</b> <b>newval</b>

**Paramètres :**

**newval** une chaîne de caractères représentant la plage de signal électrique utilisée par le capteur

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor**→**set\_unit()****YGenericSensor****genericsensor**→**setUnit()**[**genericsensor setUnit:** ]

Change l'unité dans laquelle la valeur mesurée est exprimée.

js	function <b>set_unit</b> ( <b>newval</b> )
nodejs	function <b>set_unit</b> ( <b>newval</b> )
php	function <b>set_unit</b> ( <b>\$newval</b> )
cpp	int <b>set_unit</b> ( const string& <b>newval</b> )
m	-(int) setUnit : (NSString*) <b>newval</b>
pas	function <b>set_unit</b> ( <b>newval</b> : string): integer
vb	function <b>set_unit</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_unit</b> ( string <b>newval</b> )
java	int <b>set_unit</b> ( String <b>newval</b> )
py	def <b>set_unit</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target set_unit newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor**→**set\_userdata()****YGenericSensor****genericsensor**→**setUserData()**[**genericsensor**  
**setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**genericsensor→set\_valueRange()****YGenericSensor****genericsensor→setValueRange()[genericsensor  
setValueRange: ]**

Modifie la plage de valeurs physiques mesurés par le capteur.

js	function <b>set_valueRange</b> ( <b>newval</b> )
nodejs	function <b>set_valueRange</b> ( <b>newval</b> )
php	function <b>set_valueRange</b> ( <b>\$newval</b> )
cpp	int <b>set_valueRange</b> ( const string& <b>newval</b> )
m	-(int) setValueRange : (NSString*) <b>newval</b>
pas	function <b>set_valueRange</b> ( <b>newval</b> : string): integer
vb	function <b>set_valueRange</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_valueRange</b> ( string <b>newval</b> )
java	int <b>set_valueRange</b> ( String <b>newval</b> )
py	def <b>set_valueRange</b> ( <b>newval</b> )
cmd	YGenericSensor <b>target set_valueRange newval</b>

Le changement de plage peut avoir pour effet de bord un changement automatique de la résolution affichée.

**Paramètres :**

**newval** une chaîne de caractères représentant la plage de valeurs physiques mesurés par le capteur

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**genericsensor→wait\_async()****YGenericSensor**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.18. Interface de la fonction Gyro

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_gyro.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YGyro = yoctolib.YGyro;
php	require_once('yocto_gyro.php');
c++	#include "yocto_gyro.h"
m	#import "yocto_gyro.h"
pas	uses yocto_gyro;
vb	yocto_gyro.vb
cs	yocto_gyro.cs
java	import com.yoctopuce.YoctoAPI.YGyro;
py	from yocto_gyro import *

### Fonction globales

#### yFindGyro(func)

Permet de retrouver un gyroscope d'après un identifiant donné.

#### yFirstGyro()

Commence l'énumération des gyroscopes accessibles par la librairie.

### Méthodes des objets YGyro

#### gyro→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### gyro→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du gyroscope au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### gyro→get\_advertisedValue()

Retourne la valeur courante du gyroscope (pas plus de 6 caractères).

#### gyro→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### gyro→get\_currentValue()

Retourne la valeur actuelle de la vitesse angulaire.

#### gyro→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du gyroscope.

#### gyro→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du gyroscope.

#### gyro→get\_friendlyName()

Retourne un identifiant global du gyroscope au format `NOM_MODULE . NOM_FONCTION`.

#### gyro→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### gyro→get\_functionId()

Retourne l'identifiant matériel du gyroscope, sans référence au module.

#### gyro→get\_hardwareId()

Retourne l'identifiant matériel unique du gyroscope au format `SERIAL . FUNCTIONID`.



**gyro→get\_heading()**

Retourne une estimation du cap (angle de lacet), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_highestValue()**

Retourne la valeur maximale observée pour la vitesse angulaire depuis le démarrage du module.

**gyro→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**gyro→get\_logicalName()**

Retourne le nom logique du gyroscope.

**gyro→get\_lowestValue()**

Retourne la valeur minimale observée pour la vitesse angulaire depuis le démarrage du module.

**gyro→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**gyro→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**gyro→get\_pitch()**

Retourne une estimation de l'assiette (angle de tangage), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_quaternionW()**

Retourne la composante  $w$  (composante réelle) du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_quaternionX()**

Retourne la composante  $x$  du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_quaternionY()**

Retourne la composante  $y$  du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_quaternionZ()**

Retourne la composante  $z$  du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**gyro→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**gyro→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**gyro→get\_roll()**

Retourne une estimation de l'inclinaison (angle de roulis), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

**gyro→get\_unit()**

Retourne l'unité dans laquelle la vitesse angulaire est exprimée.

**gyro→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**gyro→get\_xValue()**

### 3. Reference

Retourne la vitesse angulaire autour de l'axe X du module, sous forme de nombre à virgule.

#### **gyro→get\_yValue()**

Retourne la vitesse angulaire autour de l'axe Y du module, sous forme de nombre à virgule.

#### **gyro→get\_zValue()**

Retourne la vitesse angulaire autour de l'axe Z du module, sous forme de nombre à virgule.

#### **gyro→isOnline()**

Vérifie si le module hébergeant le gyroscope est joignable, sans déclencher d'erreur.

#### **gyro→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le gyroscope est joignable, sans déclencher d'erreur.

#### **gyro→load(msValidity)**

Met en cache les valeurs courantes du gyroscope, avec une durée de validité spécifiée.

#### **gyro→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

#### **gyro→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du gyroscope, avec une durée de validité spécifiée.

#### **gyro→nextGyro()**

Continue l'énumération des gyroscopes commencée à l'aide de `yFirstGyro()`.

#### **gyro→registerAnglesCallback(callback)**

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

#### **gyro→registerQuaternionCallback(callback)**

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

#### **gyro→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

#### **gyro→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **gyro→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

#### **gyro→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

#### **gyro→set\_logicalName(newval)**

Modifie le nom logique du gyroscope.

#### **gyro→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

#### **gyro→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

#### **gyro→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

#### **gyro→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

#### **gyro→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YGyro.FindGyro() yFindGyro()yFindGyro()

## YGyro

Permet de retrouver un gyroscope d'après un identifiant donné.

js	function <b>yFindGyro</b> ( <b>func</b> )
nodejs	function <b>FindGyro</b> ( <b>func</b> )
php	function <b>yFindGyro</b> ( <b>\$func</b> )
cpp	YGyro* <b>yFindGyro</b> ( string <b>func</b> )
m	+(YGyro*) <b>yFindGyro</b> : (NSString*) <b>func</b>
pas	function <b>yFindGyro</b> ( <b>func</b> : string): TYGyro
vb	function <b>yFindGyro</b> ( ByVal <b>func</b> As String) As YGyro
cs	YGyro <b>FindGyro</b> ( string <b>func</b> )
java	YGyro <b>FindGyro</b> ( String <b>func</b> )
py	def <b>FindGyro</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le gyroscope soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YGyro.isOnline()` pour tester si le gyroscope est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le gyroscope sans ambiguïté

### Retourne :

un objet de classe `YGyro` qui permet ensuite de contrôler le gyroscope.

## YGyro.FirstGyro() yFirstGyro()yFirstGyro()

YGyro

Commence l'énumération des gyroscopes accessibles par la librairie.

js	function <b>yFirstGyro</b> ( )
nodejs	function <b>FirstGyro</b> ( )
php	function <b>yFirstGyro</b> ( )
cpp	YGyro* <b>yFirstGyro</b> ( )
m	YGyro* <b>yFirstGyro</b> ( )
pas	function <b>yFirstGyro</b> ( ): TYGyro
vb	function <b>yFirstGyro</b> ( ) As YGyro
cs	YGyro <b>FirstGyro</b> ( )
java	YGyro <b>FirstGyro</b> ( )
py	def <b>FirstGyro</b> ( )

Utiliser la fonction `YGyro.nextGyro( )` pour itérer sur les autres gyroscopes.

### Retourne :

un pointeur sur un objet `YGyro`, correspondant à le premier gyroscope accessible en ligne, ou `null` si il n'y a pas de gyroscopes disponibles.

## gyro→calibrateFromPoints()[gyro calibrateFromPoints: ]

YGyro

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YGyro target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**gyro→describe()[gyro describe]****YGyro**

Retourne un court texte décrivant de manière non-ambigüe l'instance du gyroscope au format  
 TYPE (NAME) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.relay1)=unresolved si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant le gyroscope (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

**gyro→get\_advertisedValue()****YGyro****gyro→advertisedValue()[gyro advertisedValue]**

Retourne la valeur courante du gyroscope (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YGyro <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du gyroscope (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**gyro**→**get\_currentRawValue()****YGyro****gyro**→**currentRawValue()**[**gyro currentRawValue**]

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YGyro <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.



**gyro→get\_currentValue()**  
**gyro→currentValue()[gyro currentValue]**

**YGyro**

Retourne la valeur actuelle de la vitesse angulaire.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YGyro <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de la vitesse angulaire

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**gyro→get\_errorMessage()****YGyro****gyro→errorMessage()[gyro errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du gyroscope.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du gyroscope.

**gyro→get\_errorType()****YGyro****gyro→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du gyroscope.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du gyroscope.

**gyro→get\_friendlyName()****YGyro****gyro→friendlyName()[gyro friendlyName]**

Retourne un identifiant global du gyroscope au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
c++	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du gyroscope si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du gyroscope (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le gyroscope en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## gyro→get\_functionDescriptor() gyro→functionDescriptor()[gyro functionDescriptor]

YGyro

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### Retourne :

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**gyro**→**get\_functionId()****YGyro****gyro**→**functionId()**[gyro functionId]

Retourne l'identifiant matériel du gyroscope, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le gyroscope (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**gyro→get\_hardwareId()**  
**gyro→hardwareId()[gyro hardwareId]**

**YGyro**

Retourne l'identifiant matériel unique du gyroscope au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du gyroscope (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le gyroscope (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**gyro**→**get\_heading()****YGyro****gyro**→**heading()**[gyro heading]

Retourne une estimation du cap (angle de lacet), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_heading</b> ( )
nodejs	function <b>get_heading</b> ( )
php	function <b>get_heading</b> ( )
cpp	double <b>get_heading</b> ( )
m	-(double) heading
pas	function <b>get_heading</b> ( ): double
vb	function <b>get_heading</b> ( ) As Double
cs	double <b>get_heading</b> ( )
java	double <b>get_heading</b> ( )
py	def <b>get_heading</b> ( )

L'axe de lacet peut être attribué à n'importe laquelle des direction physiques X, Y ou Z du module à l'aide des méthodes de la classe `YRefFrame`.

**Retourne :**

un nombre à virgule correspondant au cap, exprimé en degrés (entre 0 et 360).



**gyro→get\_highestValue()****YGyro****gyro→highestValue()[gyro highestValue]**

Retourne la valeur maximale observée pour la vitesse angulaire depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YGyro <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la vitesse angulaire depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**gyro→get\_logFrequency()****YGyro****gyro→logFrequency()[gyro logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YGyro <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**gyro→get\_logicalName()****YGyro****gyro→logicalName()[gyro logicalName]**

Retourne le nom logique du gyroscope.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YGyro <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du gyroscope. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**gyro→get\_lowestValue()****YGyro****gyro→lowestValue()[gyro lowestValue]**

Retourne la valeur minimale observée pour la vitesse angulaire depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YGyro <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la vitesse angulaire depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

## **gyro→get\_module()** **gyro→module()[gyro module]**

**YGyro**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**gyro**→**get\_module\_async()****YGyro****gyro**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**gyro→get\_pitch()****YGyro****gyro→pitch()[gyro pitch]**

Retourne une estimation de l'assiette (angle de tangage), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_pitch</b> ( )
nodejs	function <b>get_pitch</b> ( )
php	function <b>get_pitch</b> ( )
cpp	double <b>get_pitch</b> ( )
m	-(double) pitch
pas	function <b>get_pitch</b> ( ): double
vb	function <b>get_pitch</b> ( ) As Double
cs	double <b>get_pitch</b> ( )
java	double <b>get_pitch</b> ( )
py	def <b>get_pitch</b> ( )

L'axe de tangage peut être attribué à n'importe laquelle des direction physiques X, Y ou Z du module à l'aide des méthodes de la classe `YRefFrame`.

**Retourne :**

un nombre à virgule correspondant à l'assiette, exprimée en degrés (entre -90 et +90).

**gyro→get\_quaternionW()****YGyro****gyro→quaternionW()[gyro quaternionW]**

Retourne la composante  $w$  (composante réelle) du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_quaternionW</b> ( )
nodejs	function <b>get_quaternionW</b> ( )
php	function <b>get_quaternionW</b> ( )
cpp	double <b>get_quaternionW</b> ( )
m	-(double) quaternionW
pas	function <b>get_quaternionW</b> ( ): double
vb	function <b>get_quaternionW</b> ( ) As Double
cs	double <b>get_quaternionW</b> ( )
java	double <b>get_quaternionW</b> ( )
py	def <b>get_quaternionW</b> ( )

**Retourne :**

un nombre à virgule correspondant à la composante  $w$  du quaternion.



**gyro**→**get\_quaternionX()****YGyro****gyro**→**quaternionX()[gyro quaternionX]**

Retourne la composante  $x$  du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_quaternionX</b> ( )
nodejs	function <b>get_quaternionX</b> ( )
php	function <b>get_quaternionX</b> ( )
cpp	double <b>get_quaternionX</b> ( )
m	-(double) quaternionX
pas	function <b>get_quaternionX</b> ( ): double
vb	function <b>get_quaternionX</b> ( ) As Double
cs	double <b>get_quaternionX</b> ( )
java	double <b>get_quaternionX</b> ( )
py	def <b>get_quaternionX</b> ( )

La composante  $x$  est essentiellement corrélée aux rotations sur l'axe de roulis.

**Retourne :**

un nombre à virgule correspondant à la composante  $x$  du quaternion.

**gyro**→**get\_quaternionY()****YGyro****gyro**→**quaternionY()[gyro quaternionY]**

Retourne la composante  $y$  du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_quaternionY</b> ( )
nodejs	function <b>get_quaternionY</b> ( )
php	function <b>get_quaternionY</b> ( )
cpp	double <b>get_quaternionY</b> ( )
m	-(double) quaternionY
pas	function <b>get_quaternionY</b> ( ): double
vb	function <b>get_quaternionY</b> ( ) As Double
cs	double <b>get_quaternionY</b> ( )
java	double <b>get_quaternionY</b> ( )
py	def <b>get_quaternionY</b> ( )

La composante  $y$  est essentiellement corrélée aux rotations sur l'axe de tangage.

**Retourne :**

un nombre à virgule correspondant à la composante  $y$  du quaternion.

**gyro→get\_quaternionZ()****YGyro****gyro→quaternionZ()[gyro quaternionZ]**

Retourne la composante *z* du quaternion décrivant l'orientation estimée du module, basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_quaternionZ</b> ( )
nodejs	function <b>get_quaternionZ</b> ( )
php	function <b>get_quaternionZ</b> ( )
cpp	double <b>get_quaternionZ</b> ( )
m	-(double) quaternionZ
pas	function <b>get_quaternionZ</b> ( ): double
vb	function <b>get_quaternionZ</b> ( ) As Double
cs	double <b>get_quaternionZ</b> ( )
java	double <b>get_quaternionZ</b> ( )
py	def <b>get_quaternionZ</b> ( )

La composante *z* est essentiellement corrélée aux rotations sur l'axe de lacet.

**Retourne :**

un nombre à virgule correspondant à la composante *z* du quaternion.

**gyro→get\_recordedData()****YGyro****gyro→recordedData()[gyro recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
c++	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YGyro <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

- startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.
- endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**gyro→get\_reportFrequency()****YGyro****gyro→reportFrequency()[gyro reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YGyro <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

## **gyro→get\_resolution()** **gyro→resolution()[gyro resolution]**

YGyro

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YGyro <b>target get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**gyro→get\_roll()****YGyro****gyro→roll()[gyro roll]**

Retourne une estimation de l'inclinaison (angle de roulis), basée sur l'intégration de mesures gyroscopiques combinée à des mesures statiques d'accélération et de champ magnétique.

js	function <b>get_roll</b> ( )
nodejs	function <b>get_roll</b> ( )
php	function <b>get_roll</b> ( )
cpp	double <b>get_roll</b> ( )
m	-(double) roll
pas	function <b>get_roll</b> ( ): double
vb	function <b>get_roll</b> ( ) As Double
cs	double <b>get_roll</b> ( )
java	double <b>get_roll</b> ( )
py	def <b>get_roll</b> ( )

L'axe de roulis peut être attribué à n'importe laquelle des direction physiques X, Y ou Z du module à l'aide des méthodes de la classe `YRefFrame`.

**Retourne :**

un nombre à virgule correspondant à l'inclinaison, exprimée en degrés (entre -180 et +180).

**gyro**→**get\_unit()****YGyro****gyro**→**unit()**[gyro unit]

Retourne l'unité dans laquelle la vitesse angulaire est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YGyro <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la vitesse angulaire est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.



**gyro→get\_userdata()****YGyro****gyro→userdata()[gyro userdata]**

Retourne le contenu de l'attribut userdata, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userdata
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**gyro**→**get\_xValue()****YGyro****gyro**→**xValue()**[**gyro xValue**]

Retourne la vitesse angulaire autour de l'axe X du module, sous forme de nombre à virgule.

js	function <b>get_xValue</b> ( )
nodejs	function <b>get_xValue</b> ( )
php	function <b>get_xValue</b> ( )
cpp	double <b>get_xValue</b> ( )
m	-(double) xValue
pas	function <b>get_xValue</b> ( ): double
vb	function <b>get_xValue</b> ( ) As Double
cs	double <b>get_xValue</b> ( )
java	double <b>get_xValue</b> ( )
py	def <b>get_xValue</b> ( )

**Retourne :**

une valeur numérique représentant la vitesse angulaire autour de l'axe X du module, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_XVALUE\_INVALID.

**gyro→get\_yValue()**  
**gyro→yValue()[gyro yValue]**

**YGyro**

Retourne la vitesse angulaire autour de l'axe Y du module, sous forme de nombre à virgule.

js	function <b>get_yValue</b> ( )
nodejs	function <b>get_yValue</b> ( )
php	function <b>get_yValue</b> ( )
cpp	double <b>get_yValue</b> ( )
m	-(double) yValue
pas	function <b>get_yValue</b> ( ): double
vb	function <b>get_yValue</b> ( ) As Double
cs	double <b>get_yValue</b> ( )
java	double <b>get_yValue</b> ( )
py	def <b>get_yValue</b> ( )

**Retourne :**

une valeur numérique représentant la vitesse angulaire autour de l'axe Y du module, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_YVALUE\_INVALID.

**gyro**→**get\_zValue()****YGyro****gyro**→**zValue()**[gyro zValue]

Retourne la vitesse angulaire autour de l'axe Z du module, sous forme de nombre à virgule.

js	function <b>get_zValue</b> ( )
nodejs	function <b>get_zValue</b> ( )
php	function <b>get_zValue</b> ( )
cpp	double <b>get_zValue</b> ( )
m	-(double) zValue
pas	function <b>get_zValue</b> ( ): double
vb	function <b>get_zValue</b> ( ) As Double
cs	double <b>get_zValue</b> ( )
java	double <b>get_zValue</b> ( )
py	def <b>get_zValue</b> ( )

**Retourne :**

une valeur numérique représentant la vitesse angulaire autour de l'axe Z du module, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_ZVALUE\_INVALID.

**gyro→isOnline()[gyro isOnline]****YGyro**

Vérifie si le module hébergeant le gyroscope est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du gyroscope sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le gyroscope est joignable, `false` sinon

**gyro→isOnline\_async()****YGyro**

Vérifie si le module hébergeant le gyroscope est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du gyroscope sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**gyro→load()[gyro load: ]****YGyro**

Met en cache les valeurs courantes du gyroscope, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## gyro→loadCalibrationPoints()[gyro loadCalibrationPoints: ]

YGyro

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YGyro target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## gyro→load\_async()

## YGyro

Met en cache les valeurs courantes du gyroscope, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**gyro**→**nextGyro()**[**gyro nextGyro**]**YGyro**

Continue l'énumération des gyroscopes commencée à l'aide de `yFirstGyro()`.

js	function <b>nextGyro</b> ( )
nodejs	function <b>nextGyro</b> ( )
php	function <b>nextGyro</b> ( )
cpp	YGyro * <b>nextGyro</b> ( )
m	-(YGyro*) <b>nextGyro</b>
pas	function <b>nextGyro</b> ( ): TYGyro
vb	function <b>nextGyro</b> ( ) As YGyro
cs	YGyro <b>nextGyro</b> ( )
java	YGyro <b>nextGyro</b> ( )
py	def <b>nextGyro</b> ( )

**Retourne :**

un pointeur sur un objet `YGyro` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## gyro→registerAnglesCallback()[gyro registerAnglesCallback: ]

YGyro

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

js	function <b>registerAnglesCallback</b> ( <b>callback</b> )
nodejs	function <b>registerAnglesCallback</b> ( <b>callback</b> )
php	function <b>registerAnglesCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerAnglesCallback</b> ( YAnglesCallback <b>callback</b> )
m	-(int) <b>registerAnglesCallback</b> : (YAnglesCallback) <b>callback</b>
pas	function <b>registerAnglesCallback</b> ( <b>callback</b> : TYAnglesCallback): LongInt
vb	function <b>registerAnglesCallback</b> ( ) As Integer
cs	int <b>registerAnglesCallback</b> ( YAnglesCallback <b>callback</b> )
java	int <b>registerAnglesCallback</b> ( YAnglesCallback <b>callback</b> )
py	def <b>registerAnglesCallback</b> ( <b>callback</b> )

La fréquence d'appel est typiquement de 95Hz durant un mouvement. Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand le callback peut se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que le callback ne soit pas appelés trop tard. Pour désactiver le callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter quatre arguments: l'objet YGyro du module qui a tourné, et les valeurs des trois angles roll, pitch et heading en degrés (nombres à virgules).

## gyro→registerQuaternionCallback()[gyro registerQuaternionCallback: ]

YGyro

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

js	function <b>registerQuaternionCallback</b> ( <b>callback</b> )
nodejs	function <b>registerQuaternionCallback</b> ( <b>callback</b> )
php	function <b>registerQuaternionCallback</b> ( <b>\$callback</b> )
c++	int <b>registerQuaternionCallback</b> ( YQuatCallback <b>callback</b> )
m	-(int) <b>registerQuaternionCallback</b> : (YQuatCallback) <b>callback</b>
pas	function <b>registerQuaternionCallback</b> ( <b>callback</b> : TYQuatCallback): LongInt
vb	function <b>registerQuaternionCallback</b> ( ) As Integer
cs	int <b>registerQuaternionCallback</b> ( YQuatCallback <b>callback</b> )
java	int <b>registerQuaternionCallback</b> ( YQuatCallback <b>callback</b> )
py	def <b>registerQuaternionCallback</b> ( <b>callback</b> )

La fréquence d'appel est typiquement de 95Hz durant un mouvement. Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand le callback peut se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que le callback ne soit pas appelé trop tard. Pour désactiver le callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter cinq arguments: l'objet YGyro du module qui a tourné, et les valeurs des quatre composantes w, x, y et z du quaternion (nombres à virgules).

## gyro→registerTimedReportCallback()[gyro registerTimedReportCallback: ]

YGyro

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
c++	int <b>registerTimedReportCallback</b> ( YGyroTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YGyroTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYGyroTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## gyro→registerValueCallback()[gyro registerValueCallback: ]

YGyro

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YGyroValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YGyroValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYGyroValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**gyro→set\_highestValue()****YGyro****gyro→setHighestValue()[gyro setHighestValue: ]**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YGyro <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**gyro→set\_logFrequency()****gyro→setLogFrequency()[gyro setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YGyro <b>target set_logFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**gyro→set\_logicalName()****YGyro****gyro→setLogicalName()[gyro setLogicalName: ]**

Modifie le nom logique du gyroscope.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YGyro <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du gyroscope.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**gyro**→**set\_lowestValue()****YGyro****gyro**→**setLowestValue()**[**gyro setLowestValue:** ]

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YGyro <b>target set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## gyro→set\_reportFrequency() gyro→setReportFrequency()[gyro setReportFrequency: ]

YGyro

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YGyro <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**gyro→set\_resolution()****gyro→setResolution()[gyro setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YGyro <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**gyro→set\_userdata()****YGyro****gyro→setUserData()[gyro setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**gyro→wait\_async()****YGyro**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.19. Interface d'un port de Yocto-hub

Les objets YHubPort permettent de contrôler l'alimentation des ports d'un YoctoHub, ainsi que de détecter si un module y est raccordé et lequel. Un YHubPort reçoit toujours automatiquement comme nom logique le numéro de série unique du module Yoctopuce qui y est connecté.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_hubport.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YHubPort = yoctolib.YHubPort;
php	require_once('yocto_hubport.php');
c++	#include "yocto_hubport.h"
m	#import "yocto_hubport.h"
pas	uses yocto_hubport;
vb	yocto_hubport.vb
cs	yocto_hubport.cs
java	import com.yoctopuce.YoctoAPI.YHubPort;
py	from yocto_hubport import *

### Fonction globales

#### yFindHubPort(func)

Permet de retrouver un port de Yocto-hub d'après un identifiant donné.

#### yFirstHubPort()

Commence l'énumération des port de Yocto-hub accessibles par la librairie.

### Méthodes des objets YHubPort

#### hubport→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du port de Yocto-hub au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### hubport→get\_advertisedValue()

Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

#### hubport→get\_baudRate()

Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.

#### hubport→get\_enabled()

Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.

#### hubport→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

#### hubport→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

#### hubport→get\_friendlyName()

Retourne un identifiant global du port de Yocto-hub au format `NOM_MODULE . NOM_FONCTION`.

#### hubport→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### hubport→get\_functionId()

Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.

#### hubport→get\_hardwareId()

Retourne l'identifiant matériel unique du port de Yocto-hub au format `SERIAL . FUNCTIONID`.

#### hubport→get\_logicalName()

	Retourne le nom logique du port de Yocto-hub.
<b>hubport→get_module()</b>	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
<b>hubport→get_module_async(callback, context)</b>	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
<b>hubport→get_portState()</b>	Retourne l'état actuel du port de Yocto-hub.
<b>hubport→get_userData()</b>	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.
<b>hubport→isOnline()</b>	Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.
<b>hubport→isOnline_async(callback, context)</b>	Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.
<b>hubport→load(msValidity)</b>	Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.
<b>hubport→load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.
<b>hubport→nextHubPort()</b>	Continue l'énumération des port de Yocto-hub commencée à l'aide de yFirstHubPort( ).
<b>hubport→registerValueCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b>hubport→set_enabled(newval)</b>	Modifie le mode d'activation du port du Yocto-hub.
<b>hubport→set_logicalName(newval)</b>	Modifie le nom logique du port de Yocto-hub.
<b>hubport→set_userData(data)</b>	Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get_userData.
<b>hubport→wait_async(callback, context)</b>	Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YHubPort.FindHubPort() yFindHubPort()yFindHubPort()

## YHubPort

Permet de retrouver un port de Yocto-hub d'après un identifiant donné.

js	function <b>yFindHubPort</b> ( <b>func</b> )
nodejs	function <b>FindHubPort</b> ( <b>func</b> )
php	function <b>yFindHubPort</b> ( <b>\$func</b> )
cpp	YHubPort* <b>yFindHubPort</b> ( const string& <b>func</b> )
m	YHubPort* <b>yFindHubPort</b> ( NSString* <b>func</b> )
pas	function <b>yFindHubPort</b> ( <b>func</b> : string): TYHubPort
vb	function <b>yFindHubPort</b> ( ByVal <b>func</b> As String) As YHubPort
cs	YHubPort <b>FindHubPort</b> ( string <b>func</b> )
java	YHubPort <b>FindHubPort</b> ( String <b>func</b> )
py	def <b>FindHubPort</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port de Yocto-hub soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHubPort.isOnline()` pour tester si le port de Yocto-hub est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le port de Yocto-hub sans ambiguïté

### Retourne :

un objet de classe `YHubPort` qui permet ensuite de contrôler le port de Yocto-hub.

## **YHubPort.FirstHubPort()** **yFirstHubPort()****yFirstHubPort()**

**YHubPort**

Commence l'énumération des port de Yocto-hub accessibles par la librairie.

js	function <b>yFirstHubPort</b> ( )
nodejs	function <b>FirstHubPort</b> ( )
php	function <b>yFirstHubPort</b> ( )
cpp	YHubPort* <b>yFirstHubPort</b> ( )
m	YHubPort* <b>yFirstHubPort</b> ( )
pas	function <b>yFirstHubPort</b> ( ): TYHubPort
vb	function <b>yFirstHubPort</b> ( ) As YHubPort
cs	YHubPort <b>FirstHubPort</b> ( )
java	YHubPort <b>FirstHubPort</b> ( )
py	def <b>FirstHubPort</b> ( )

Utiliser la fonction `YHubPort.nextHubPort( )` pour itérer sur les autres port de Yocto-hub.

**Retourne :**

un pointeur sur un objet `YHubPort`, correspondant à le premier port de Yocto-hub accessible en ligne, ou `null` si il n'y a pas de port de Yocto-hub disponibles.

**hubport→describe()[hubport describe]****YHubPort**

Retourne un court texte décrivant de manière non-ambigüe l'instance du port de Yocto-hub au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le port de Yocto-hub (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## hubport→get\_advertisedValue() hubport→advertisedValue()[hubport advertisedValue]

YHubPort

Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YHubPort <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du port de Yocto-hub (pas plus de 6 caractères).  
En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**hubport**→**get\_baudRate()****YHubPort****hubport**→**baudRate()**[hubport baudRate]

Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.

js	function <b>get_baudRate</b> ( )
nodejs	function <b>get_baudRate</b> ( )
php	function <b>get_baudRate</b> ( )
cpp	int <b>get_baudRate</b> ( )
m	-(int) baudRate
pas	function <b>get_baudRate</b> ( ): LongInt
vb	function <b>get_baudRate</b> ( ) As Integer
cs	int <b>get_baudRate</b> ( )
java	int <b>get_baudRate</b> ( )
py	def <b>get_baudRate</b> ( )
cmd	YHubPort <b>target</b> <b>get_baudRate</b>

La valeur par défaut est 1000 kbps, une valeur inférieure révèle des problèmes de communication.

**Retourne :**

un entier représentant la vitesse de transfert utilisée par le port de Yocto-hub, en kbps

En cas d'erreur, déclenche une exception ou retourne Y\_BAUDRATE\_INVALID.

**hubport**→**get\_enabled()****YHubPort****hubport**→**enabled()**[hubport enabled]

Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.

js	function <b>get_enabled</b> ( )
nodejs	function <b>get_enabled</b> ( )
php	function <b>get_enabled</b> ( )
cpp	Y_ENABLED_enum <b>get_enabled</b> ( )
m	-(Y_ENABLED_enum) enabled
pas	function <b>get_enabled</b> ( ): Integer
vb	function <b>get_enabled</b> ( ) As Integer
cs	int <b>get_enabled</b> ( )
java	int <b>get_enabled</b> ( )
py	def <b>get_enabled</b> ( )
cmd	YHubPort <b>target</b> <b>get_enabled</b>

**Retourne :**

soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon vrai si le port du Yocto-hub est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLED\_INVALID.

**hubport**→**get\_errorMessage()****YHubPort****hubport**→**errorMessage()**[**hubport errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.

**hubport**→**get\_errorType()****YHubPort****hubport**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du port de Yocto-hub.



**hubport**→**get\_friendlyName()****YHubPort****hubport**→**friendlyName()**[hubport friendlyName]

Retourne un identifiant global du port de Yocto-hub au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du port de Yocto-hub si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du port de Yocto-hub (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**hubport**→**get\_functionDescriptor()**  
**hubport**→**functionDescriptor()[hubport**  
**functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**hubport**→**get\_functionId()****YHubPort****hubport**→**functionId()**[hubport functionId]

Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**hubport**→**get\_hardwareId()****YHubPort****hubport**→**hardwareId()**[**hubport hardwareId**]

Retourne l'identifiant matériel unique du port de Yocto-hub au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port de Yocto-hub (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le port de Yocto-hub (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

---

**hubport**→**get\_logicalName()****YHubPort****hubport**→**logicalName()**[hubport logicalName]

---

Retourne le nom logique du port de Yocto-hub.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YHubPort <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du port de Yocto-hub. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**hubport**→**get\_module()****YHubPort****hubport**→**module()**[**hubport module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

---

**hubport→get\_module\_async()****YHubPort****hubport→module\_async()**

---

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**hubport**→**get\_portState()****YHubPort****hubport**→**portState()**[hubport portState]

Retourne l'état actuel du port de Yocto-hub.

js	function <b>get_portState</b> ( )
nodejs	function <b>get_portState</b> ( )
php	function <b>get_portState</b> ( )
cpp	Y_PORTSTATE_enum <b>get_portState</b> ( )
m	-(Y_PORTSTATE_enum) portState
pas	function <b>get_portState</b> ( ): Integer
vb	function <b>get_portState</b> ( ) As Integer
cs	int <b>get_portState</b> ( )
java	int <b>get_portState</b> ( )
py	def <b>get_portState</b> ( )
cmd	YHubPort <b>target</b> <b>get_portState</b>

**Retourne :**

une valeur parmi Y\_PORTSTATE\_OFF, Y\_PORTSTATE\_OVRLD, Y\_PORTSTATE\_ON, Y\_PORTSTATE\_RUN et Y\_PORTSTATE\_PROG représentant l'état actuel du port de Yocto-hub

En cas d'erreur, déclenche une exception ou retourne Y\_PORTSTATE\_INVALID.



**hubport→get\_userdata()****YHubPort****hubport→userdata()[hubport userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**hubport→isOnline()[hubport isOnline]****YHubPort**

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le port de Yocto-hub est joignable, false sinon

**hubport→isOnline\_async()****YHubPort**

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du port de Yocto-hub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## hubport→load\_async()

## YHubPort

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**hubport**→**nextHubPort()**[hubport nextHubPort]**YHubPort**

Continue l'énumération des port de Yocto-hub commencée à l'aide de `yFirstHubPort()`.

js	function <b>nextHubPort</b> ( )
nodejs	function <b>nextHubPort</b> ( )
php	function <b>nextHubPort</b> ( )
cpp	YHubPort * <b>nextHubPort</b> ( )
m	-(YHubPort*) <b>nextHubPort</b>
pas	function <b>nextHubPort</b> ( ): TYHubPort
vb	function <b>nextHubPort</b> ( ) As YHubPort
cs	YHubPort <b>nextHubPort</b> ( )
java	YHubPort <b>nextHubPort</b> ( )
py	def <b>nextHubPort</b> ( )

**Retourne :**

un pointeur sur un objet `YHubPort` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## hubport→registerValueCallback()[hubport registerValueCallback: ]

YHubPort

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
c++	int <b>registerValueCallback</b> ( YHubPortValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YHubPortValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYHubPortValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**hubport**→**set\_enabled()****YHubPort****hubport**→**setEnabled()**[**hubport** **setEnabled:** ]

Modifie le mode d'activation du port du Yocto-hub.

js	function <b>set_enabled</b> ( <b>newval</b> )
nodejs	function <b>set_enabled</b> ( <b>newval</b> )
php	function <b>set_enabled</b> ( <b>\$newval</b> )
cpp	int <b>set_enabled</b> ( Y_ENABLED_enum <b>newval</b> )
m	-(int) <b>setEnabled</b> : (Y_ENABLED_enum) <b>newval</b>
pas	function <b>set_enabled</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabled</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabled</b> ( int <b>newval</b> )
java	int <b>set_enabled</b> ( int <b>newval</b> )
py	def <b>set_enabled</b> ( <b>newval</b> )
cmd	YHubPort <b>target</b> <b>set_enabled</b> <b>newval</b>

Si le port est actif, il sera alimenté. Sinon, l'alimentation du module est coupée.

**Paramètres :**

**newval** soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon le mode d'activation du port du Yocto-hub

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## hubport→set\_logicalName() hubport→setLogicalName()[hubport setLogicalName: ]

YHubPort

Modifie le nom logique du port de Yocto-hub.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YHubPort <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du port de Yocto-hub.

### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**hubport**→**set\_userData()****YHubPort****hubport**→**setUserData()**[**hubport setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function <b>set_userData</b> ( <b>data</b> )
nodejs	function <b>set_userData</b> ( <b>data</b> )
php	function <b>set_userData</b> ( <b>\$data</b> )
cpp	void <b>set_userData</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userData</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userData</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userData</b> ( object <b>data</b> )
java	void <b>set_userData</b> ( Object <b>data</b> )
py	def <b>set_userData</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## hubport→wait\_async()

YHubPort

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :

## 3.20. Interface de la fonction Humidity

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_humidity.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YHumidity = yoctolib.YHumidity;
php	require_once('yocto_humidity.php');
c++	#include "yocto_humidity.h"
m	#import "yocto_humidity.h"
pas	uses yocto_humidity;
vb	yocto_humidity.vb
cs	yocto_humidity.cs
java	import com.yoctopuce.YoctoAPI.YHumidity;
py	from yocto_humidity import *

### Fonction globales

#### yFindHumidity(func)

Permet de retrouver un capteur d'humidité d'après un identifiant donné.

#### yFirstHumidity()

Commence l'énumération des capteurs d'humidité accessibles par la librairie.

### Méthodes des objets YHumidity

#### humidity→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### humidity→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur d'humidité au format TYPE ( NAME ) = SERIAL . FUNCTIONID.

#### humidity→get\_advertisedValue()

Retourne la valeur courante du capteur d'humidité (pas plus de 6 caractères).

#### humidity→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### humidity→get\_currentValue()

Retourne la mesure actuelle de l'humidité.

#### humidity→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur d'humidité.

#### humidity→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur d'humidité.

#### humidity→get\_friendlyName()

Retourne un identifiant global du capteur d'humidité au format NOM\_MODULE . NOM\_FONCTION.

#### humidity→get\_functionDescriptor()

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

#### humidity→get\_functionId()

Retourne l'identifiant matériel du capteur d'humidité, sans référence au module.

#### humidity→get\_hardwareId()

Retourne l'identifiant matériel unique du capteur d'humidité au format `SERIAL.FUNCTIONID`.

#### **humidity→get\_highestValue()**

Retourne la valeur maximale observée pour l'humidité.

#### **humidity→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

#### **humidity→get\_logicalName()**

Retourne le nom logique du capteur d'humidité.

#### **humidity→get\_lowestValue()**

Retourne la valeur minimale observée pour l'humidité.

#### **humidity→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **humidity→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **humidity→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

#### **humidity→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

#### **humidity→get\_resolution()**

Retourne la résolution des valeurs mesurées.

#### **humidity→get\_unit()**

Retourne l'unité dans laquelle l'humidité est exprimée.

#### **humidity→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **humidity→isOnline()**

Vérifie si le module hébergeant le capteur d'humidité est joignable, sans déclencher d'erreur.

#### **humidity→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le capteur d'humidité est joignable, sans déclencher d'erreur.

#### **humidity→load(msValidity)**

Met en cache les valeurs courantes du capteur d'humidité, avec une durée de validité spécifiée.

#### **humidity→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

#### **humidity→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du capteur d'humidité, avec une durée de validité spécifiée.

#### **humidity→nextHumidity()**

Continue l'énumération des capteurs d'humidité commencée à l'aide de `yFirstHumidity()`.

#### **humidity→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

#### **humidity→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **humidity→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée pour l'humidité.

#### **humidity→set\_logFrequency(newval)**

### 3. Reference

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**humidity**→**set\_logicalName**(newval)

Modifie le nom logique du capteur d'humidité.

**humidity**→**set\_lowestValue**(newval)

Modifie la mémoire de valeur minimale observée pour l'humidité.

**humidity**→**set\_reportFrequency**(newval)

Modifie la fréquence de notification périodique des valeurs mesurées.

**humidity**→**set\_resolution**(newval)

Modifie la résolution des valeurs physique mesurées.

**humidity**→**set\_userData**(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**humidity**→**wait\_async**(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YHumidity.FindHumidity() yFindHumidity()yFindHumidity()

## YHumidity

Permet de retrouver un capteur d'humidité d'après un identifiant donné.

js	function <b>yFindHumidity</b> ( <b>func</b> )
nodejs	function <b>FindHumidity</b> ( <b>func</b> )
php	function <b>yFindHumidity</b> ( <b>\$func</b> )
cpp	YHumidity* <b>yFindHumidity</b> ( const string& <b>func</b> )
m	YHumidity* <b>yFindHumidity</b> ( NSString* <b>func</b> )
pas	function <b>yFindHumidity</b> ( <b>func</b> : string): TYHumidity
vb	function <b>yFindHumidity</b> ( ByVal <b>func</b> As String) As YHumidity
cs	YHumidity <b>FindHumidity</b> ( string <b>func</b> )
java	YHumidity <b>FindHumidity</b> ( String <b>func</b> )
py	def <b>FindHumidity</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur d'humidité soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHumidity.isOnline()` pour tester si le capteur d'humidité est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur d'humidité sans ambiguïté

### Retourne :

un objet de classe `YHumidity` qui permet ensuite de contrôler le capteur d'humidité.

## YHumidity.FirstHumidity() yFirstHumidity()yFirstHumidity()

YHumidity

Commence l'énumération des capteurs d'humidité accessibles par la librairie.

js	function <b>yFirstHumidity</b> ( )
nodejs	function <b>FirstHumidity</b> ( )
php	function <b>yFirstHumidity</b> ( )
cpp	YHumidity* <b>yFirstHumidity</b> ( )
m	YHumidity* <b>yFirstHumidity</b> ( )
pas	function <b>yFirstHumidity</b> ( ): TYHumidity
vb	function <b>yFirstHumidity</b> ( ) As YHumidity
cs	YHumidity <b>FirstHumidity</b> ( )
java	YHumidity <b>FirstHumidity</b> ( )
py	def <b>FirstHumidity</b> ( )

Utiliser la fonction `YHumidity.nextHumidity( )` pour itérer sur les autres capteurs d'humidité.

### Retourne :

un pointeur sur un objet `YHumidity`, correspondant à le premier capteur d'humidité accessible en ligne, ou `null` si il n'y a pas de capteurs d'humidité disponibles.



## humidity→calibrateFromPoints()[humidity calibrateFromPoints: ]

YHumidity

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                           vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                 refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                           List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                             ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YHumidity target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**humidity→describe()[humidity describe]****YHumidity**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur d'humidité au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur d'humidité (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## humidity→get\_advertisedValue() humidity→advertisedValue()[humidity advertisedValue]

YHumidity

Retourne la valeur courante du capteur d'humidité (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YHumidity <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du capteur d'humidité (pas plus de 6 caractères).  
En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**humidity→get\_currentRawValue()**  
**humidity→currentRawValue()[humidity**  
**currentRawValue]**

**YHumidity**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YHumidity <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**humidity→get\_currentValue()****YHumidity****humidity→currentValue()[humidity currentValue]**

Retourne la mesure actuelle de l'humidité.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YHumidity <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la mesure actuelle de l'humidité

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**humidity→get\_errorMessage()****YHumidity****humidity→errorMessage()[humidity errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur d'humidité.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur d'humidité.

## humidity→get\_errorType() humidity→errorType()

## YHumidity

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur d'humidité.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

### Retourne :

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur d'humidité.

**humidity**→**get\_friendlyName()****YHumidity****humidity**→**friendlyName()**[humidity friendlyName]

Retourne un identifiant global du capteur d'humidité au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur d'humidité si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur d'humidité (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur d'humidité en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.



## humidity→get\_functionDescriptor() humidity→functionDescriptor()[humidity functionDescriptor]

YHumidity

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### Retourne :

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**humidity**→**get\_functionId()****YHumidity****humidity**→**functionId()**[humidity functionId]

Retourne l'identifiant matériel du capteur d'humidité, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur d'humidité (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

humidity→get\_hardwareId()

YHumidity

humidity→hardwareId()[humidity hardwareId]

Retourne l'identifiant matériel unique du capteur d'humidité au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur d'humidité (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur d'humidité (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**humidity**→**get\_highestValue()****YHumidity****humidity**→**highestValue()**[humidity highestValue]

Retourne la valeur maximale observée pour l'humidité.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YHumidity <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour l'humidité

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**humidity→get\_logFrequency()****YHumidity****humidity→logFrequency()[humidity logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YHumidity <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**humidity→get\_logicalName()****YHumidity****humidity→logicalName()[humidity logicalName]**

Retourne le nom logique du capteur d'humidité.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YHumidity <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur d'humidité. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**humidity→get\_lowestValue()****YHumidity****humidity→lowestValue()[humidity lowestValue]**

Retourne la valeur minimale observée pour l'humidité.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YHumidity <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour l'humidité

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**humidity**→**get\_module()****YHumidity****humidity**→**module()**[humidity module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule



---

**humidity→get\_module\_async()****YHumidity****humidity→module\_async()**

---

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**humidity→get\_recordedData()****YHumidity****humidity→recordedData()[humidity recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YHumidity <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

## humidity→get\_reportFrequency() humidity→reportFrequency()[humidity reportFrequency]

YHumidity

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YHumidity <b>target</b> <b>get_reportFrequency</b>

### Retourne :

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**humidity**→**get\_resolution()****YHumidity****humidity**→**resolution()**[humidity resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YHumidity <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**humidity→get\_unit()****YHumidity****humidity→unit()[humidity unit]**

Retourne l'unité dans laquelle l'humidité est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YHumidity <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle l'humidité est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**humidity→get\_userdata()**

**YHumidity**

**humidity→userdata()[humidity userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

<code>js</code>	<code>function get_userdata( )</code>
<code>nodejs</code>	<code>function get_userdata( )</code>
<code>php</code>	<code>function get_userdata( )</code>
<code>cpp</code>	<code>void * get_userdata( )</code>
<code>m</code>	<code>-(void*) userData</code>
<code>pas</code>	<code>function get_userdata( ): Tobject</code>
<code>vb</code>	<code>function get_userdata( ) As Object</code>
<code>cs</code>	<code>object get_userdata( )</code>
<code>java</code>	<code>Object get_userdata( )</code>
<code>py</code>	<code>def get_userdata( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**humidity→isOnline()[humidity isOnline]****YHumidity**

Vérifie si le module hébergeant le capteur d'humidité est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur d'humidité sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le capteur d'humidité est joignable, `false` sinon

## humidity→isOnline\_async()

YHumidity

Vérifie si le module hébergeant le capteur d'humidité est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur d'humidité sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**humidity→load()[humidity load: ]****YHumidity**

Met en cache les valeurs courantes du capteur d'humidité, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## humidity→loadCalibrationPoints()[humidity loadCalibrationPoints: ]

YHumidity

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
node.js function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                    var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YHumidity target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## humidity→load\_async()

## YHumidity

Met en cache les valeurs courantes du capteur d'humidité, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

humidity→nextHumidity()[humidity nextHumidity]

YHumidity

Continue l'énumération des capteurs d'humidité commencée à l'aide de yFirstHumidity( ).

js	function nextHumidity( )
nodejs	function nextHumidity( )
php	function nextHumidity( )
cpp	YHumidity * nextHumidity( )
m	-(YHumidity*) nextHumidity
pas	function nextHumidity( ): TYHumidity
vb	function nextHumidity( ) As YHumidity
cs	YHumidity nextHumidity( )
java	YHumidity nextHumidity( )
py	def nextHumidity( )

**Retourne :**  
un pointeur sur un objet YHumidity accessible en ligne, ou null lorsque l'énumération est terminée.

## humidity→registerTimedReportCallback()[humidity registerTimedReportCallback: ]

YHumidity

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
c++	int <b>registerTimedReportCallback</b> ( YHumidityTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YHumidityTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYHumidityTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## humidity→registerValueCallback()[humidity registerValueCallback: ]

YHumidity

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YHumidityValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YHumidityValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYHumidityValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**humidity→set\_highestValue()**  
**humidity→setHighestValue()[humidity**  
**setHighestValue: ]**

**YHumidity**

Modifie la mémoire de valeur maximale observée pour l'humidité.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YHumidity <b>target</b> <b>set_highestValue</b> <b>newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour l'humidité

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## humidity→set\_logFrequency() humidity→setLogFrequency()[humidity setLogFrequency: ]

YHumidity

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YHumidity <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## humidity→set\_logicalName() humidity→setLogicalName()[humidity setLogicalName: ]

YHumidity

Modifie le nom logique du capteur d'humidité.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YHumidity <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du capteur d'humidité.

### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**humidity→set\_lowestValue()**  
**humidity→setLowestValue()[humidity**  
**setLowestValue: ]**

**YHumidity**

Modifie la mémoire de valeur minimale observée pour l'humidité.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YHumidity <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour l'humidité

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## humidity→set\_reportFrequency() humidity→setReportFrequency()[humidity setReportFrequency: ]

YHumidity

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YHumidity <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**humidity→set\_resolution()****YHumidity****humidity→setResolution() [humidity setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YHumidity <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**humidity→set\_userdata()****YHumidity****humidity→setUserData()[humidity setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**humidity→wait\_async()****YHumidity**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.21. Interface de la fonction Led

La librairie de programmation Yoctopuce permet non seulement d'allumer la led à une intensité donnée, mais aussi de la faire osciller à plusieurs fréquences.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_led.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib'); var YLed = yoctolib.YLed;</code>
php	<code>require_once('yocto_led.php');</code>
c++	<code>#include "yocto_led.h"</code>
m	<code>#import "yocto_led.h"</code>
pas	<code>uses yocto_led;</code>
vb	<code>yocto_led.vb</code>
cs	<code>yocto_led.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YLed;</code>
py	<code>from yocto_led import *</code>

### Fonction globales

#### yFindLed(func)

Permet de retrouver une led d'après un identifiant donné.

#### yFirstLed()

Commence l'énumération des leds accessibles par la librairie.

### Méthodes des objets YLed

#### led→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de la led au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

#### led→get\_advertisedValue()

Retourne la valeur courante de la led (pas plus de 6 caractères).

#### led→get\_blinking()

Retourne le mode de signalisation de la led.

#### led→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la led.

#### led→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la led.

#### led→get\_friendlyName()

Retourne un identifiant global de la led au format `NOM_MODULE . NOM_FONCTION`.

#### led→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### led→get\_functionId()

Retourne l'identifiant matériel de la led, sans référence au module.

#### led→get\_hardwareId()

Retourne l'identifiant matériel unique de la led au format `SERIAL . FUNCTIONID`.

#### led→get\_logicalName()

Retourne le nom logique de la led.

#### led→get\_luminosity()

Retourne l'intensité de la led en pour cent.

#### led→get\_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### `led→get_module_async(callback, context)`

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### `led→get_power()`

Retourne l'état courant de la led.

#### `led→get_userData()`

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### `led→isOnline()`

Vérifie si le module hébergeant la led est joignable, sans déclencher d'erreur.

#### `led→isOnline_async(callback, context)`

Vérifie si le module hébergeant la led est joignable, sans déclencher d'erreur.

#### `led→load(msValidity)`

Met en cache les valeurs courantes de la led, avec une durée de validité spécifiée.

#### `led→load_async(msValidity, callback, context)`

Met en cache les valeurs courantes de la led, avec une durée de validité spécifiée.

#### `led→nextLed()`

Continue l'énumération des leds commencée à l'aide de `yFirstLed()`.

#### `led→registerValueCallback(callback)`

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### `led→set_blinking(newval)`

Modifie le mode de signalisation de la led.

#### `led→set_logicalName(newval)`

Modifie le nom logique de la led.

#### `led→set_luminosity(newval)`

Modifie l'intensité lumineuse de la led (en pour cent).

#### `led→set_power(newval)`

Modifie l'état courant de la led.

#### `led→set_userData(data)`

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

#### `led→wait_async(callback, context)`

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YLed.FindLed() yFindLed()/yFindLed()

YLed

Permet de retrouver une led d'après un identifiant donné.

js	function <b>yFindLed</b> ( <b>func</b> )
nodejs	function <b>FindLed</b> ( <b>func</b> )
php	function <b>yFindLed</b> ( <b>\$func</b> )
cpp	YLed* <b>yFindLed</b> ( const string& <b>func</b> )
m	YLed* <b>yFindLed</b> ( NSString* <b>func</b> )
pas	function <b>yFindLed</b> ( <b>func</b> : string): TYLed
vb	function <b>yFindLed</b> ( ByVal <b>func</b> As String) As YLed
cs	YLed <b>FindLed</b> ( string <b>func</b> )
java	YLed <b>FindLed</b> ( String <b>func</b> )
py	def <b>FindLed</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que la led soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YLed.isOnline()` pour tester si la led est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence la led sans ambiguïté

### Retourne :

un objet de classe `YLed` qui permet ensuite de contrôler la led.

## YLed.FirstLed() yFirstLed()yFirstLed()

YLed

Commence l'énumération des leds accessibles par la librairie.

js	function <b>yFirstLed</b> ( )
nodejs	function <b>FirstLed</b> ( )
php	function <b>yFirstLed</b> ( )
cpp	YLed* <b>yFirstLed</b> ( )
m	YLed* <b>yFirstLed</b> ( )
pas	function <b>yFirstLed</b> ( ): TYLed
vb	function <b>yFirstLed</b> ( ) As YLed
cs	YLed <b>FirstLed</b> ( )
java	YLed <b>FirstLed</b> ( )
py	def <b>FirstLed</b> ( )

Utiliser la fonction `YLed.nextLed( )` pour itérer sur les autres leds.

### Retourne :

un pointeur sur un objet YLed, correspondant à la première led accessible en ligne, ou `null` si il n'y a pas de leds disponibles.

**led→describe()[led describe]****YLed**

Retourne un court texte décrivant de manière non-ambigüe l'instance de la led au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomeName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant la led (ex: `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**led**→**get\_advertisedValue()****YLed****led**→**advertisedValue()[led advertisedValue]**

Retourne la valeur courante de la led (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YLed <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de la led (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**led**→**get\_blinking()****YLed****led**→**blinking()**[**led blinking**]

Retourne le mode de signalisation de la led.

js	function <b>get_blinking</b> ( )
nodejs	function <b>get_blinking</b> ( )
php	function <b>get_blinking</b> ( )
cpp	Y_BLINKING_enum <b>get_blinking</b> ( )
m	-(Y_BLINKING_enum) blinking
pas	function <b>get_blinking</b> ( ): Integer
vb	function <b>get_blinking</b> ( ) As Integer
cs	int <b>get_blinking</b> ( )
java	int <b>get_blinking</b> ( )
py	def <b>get_blinking</b> ( )
cmd	YLed <b>target</b> <b>get_blinking</b>

**Retourne :**

une valeur parmi Y\_BLINKING\_STILL, Y\_BLINKING\_RELAX, Y\_BLINKING\_AWARE, Y\_BLINKING\_RUN, Y\_BLINKING\_CALL et Y\_BLINKING\_PANIC représentant le mode de signalisation de la led

En cas d'erreur, déclenche une exception ou retourne Y\_BLINKING\_INVALID.

**led**→**get\_errorMessage()****led**→**errorMessage()**[led errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la led.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de la led.

**led→get\_errorType()****YLed****led→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la led.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de la led.

**led**→**get\_friendlyName()****YLed****led**→**friendlyName()**[led friendlyName]

Retourne un identifiant global de la led au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de la led si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de la led (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant la led en utilisant les noms logiques (ex: `MyCustomName.relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.



**led**→**get\_functionDescriptor()****YLed****led**→**functionDescriptor()**[**led functionDescriptor**]

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**led**→**get\_functionId()****YLed****led**→**functionId()**[**led functionId**]

Retourne l'identifiant matériel de la led, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant la led (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**led**→**get\_hardwareId()****YLed****led**→**hardwareId()**[**led hardwareId**]

Retourne l'identifiant matériel unique de la led au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la led (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant la led (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**led**→**get\_logicalName()****YLed****led**→**logicalName()**[**led logicalName**]

Retourne le nom logique de la led.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YLed <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de la led. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**led**→**get\_luminosity()****YLed****led**→**luminosity()**[**led luminosity**]

Retourne l'intensité de la led en pour cent.

js	function <b>get_luminosity</b> ( )
nodejs	function <b>get_luminosity</b> ( )
php	function <b>get_luminosity</b> ( )
cpp	int <b>get_luminosity</b> ( )
m	-(int) luminosity
pas	function <b>get_luminosity</b> ( ): LongInt
vb	function <b>get_luminosity</b> ( ) As Integer
cs	int <b>get_luminosity</b> ( )
java	int <b>get_luminosity</b> ( )
py	def <b>get_luminosity</b> ( )
cmd	YLed <b>target</b> <b>get_luminosity</b>

**Retourne :**

un entier représentant l'intensité de la led en pour cent

En cas d'erreur, déclenche une exception ou retourne Y\_LUMINOSITY\_INVALID.

**led**→**get\_module()****YLed****led**→**module()**[led module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**led**→**get\_module\_async()****YLed****led**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**led**→**get\_power()****YLed****led**→**power()**[**led power**]

Retourne l'état courant de la led.

js	function <b>get_power</b> ( )
nodejs	function <b>get_power</b> ( )
php	function <b>get_power</b> ( )
cpp	Y_POWER_enum <b>get_power</b> ( )
m	-(Y_POWER_enum) power
pas	function <b>get_power</b> ( ): Integer
vb	function <b>get_power</b> ( ) As Integer
cs	int <b>get_power</b> ( )
java	int <b>get_power</b> ( )
py	def <b>get_power</b> ( )
cmd	YLed <b>target get_power</b>

**Retourne :**

soit Y\_POWER\_OFF, soit Y\_POWER\_ON, selon l'état courant de la led

En cas d'erreur, déclenche une exception ou retourne Y\_POWER\_INVALID.



**led**→**get\_userData()****YLed****led**→**userData()**[**led userData**]

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): TObject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**led→isOnline()[led isOnline]****YLed**

Vérifie si le module hébergeant la led est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de la led sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si la led est joignable, false sinon

## led→isOnline\_async()

YLed

Vérifie si le module hébergeant la led est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de la led sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**led→load()[led load: ]****YLed**

Met en cache les valeurs courantes de la led, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**led→load\_async()****YLed**

Met en cache les valeurs courantes de la led, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**led**→**nextLed()**[**led nextLed**]**YLed**

Continue l'énumération des leds commencée à l'aide de `yFirstLed()`.

js	function <b>nextLed</b> ( )
nodejs	function <b>nextLed</b> ( )
php	function <b>nextLed</b> ( )
cpp	YLed * <b>nextLed</b> ( )
m	-(YLed*) <b>nextLed</b>
pas	function <b>nextLed</b> ( ): TYLed
vb	function <b>nextLed</b> ( ) As YLed
cs	YLed <b>nextLed</b> ( )
java	YLed <b>nextLed</b> ( )
py	def <b>nextLed</b> ( )

**Retourne :**

un pointeur sur un objet YLed accessible en ligne, ou `null` lorsque l'énumération est terminée.

## led→registerValueCallback()[led registerValueCallback: ]

YLed

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YLedValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YLedValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYLedValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**led**→**set\_blinking()****YLed****led**→**setBlinking()**[**led setBlinking:** ]

Modifie le mode de signalisation de la led.

js	function <b>set_blinking</b> ( <b>newval</b> )
nodejs	function <b>set_blinking</b> ( <b>newval</b> )
php	function <b>set_blinking</b> ( <b>\$newval</b> )
cpp	int <b>set_blinking</b> ( Y_BLINKING_enum <b>newval</b> )
m	-(int) setBlinking : (Y_BLINKING_enum) <b>newval</b>
pas	function <b>set_blinking</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_blinking</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_blinking</b> ( int <b>newval</b> )
java	int <b>set_blinking</b> ( int <b>newval</b> )
py	def <b>set_blinking</b> ( <b>newval</b> )
cmd	YLed <b>target set_blinking newval</b>

**Paramètres :**

**newval** une valeur parmi Y\_BLINKING\_STILL, Y\_BLINKING\_RELAX, Y\_BLINKING\_AWARE, Y\_BLINKING\_RUN, Y\_BLINKING\_CALL et Y\_BLINKING\_PANIC représentant le mode de signalisation de la led

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**led**→**set\_logicalName()****YLed****led**→**setLogicalName()**[**led setLogicalName:** ]

Modifie le nom logique de la led.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YLed <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de la led.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**led**→**set\_luminosity()****YLed****led**→**setLuminosity()**[**led setLuminosity:** ]

Modifie l'intensité lumineuse de la led (en pour cent).

js	function <b>set_luminosity</b> ( <b>newval</b> )
nodejs	function <b>set_luminosity</b> ( <b>newval</b> )
php	function <b>set_luminosity</b> ( <b>\$newval</b> )
cpp	int <b>set_luminosity</b> ( int <b>newval</b> )
m	-(int) setLuminosity : (int) <b>newval</b>
pas	function <b>set_luminosity</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_luminosity</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_luminosity</b> ( int <b>newval</b> )
java	int <b>set_luminosity</b> ( int <b>newval</b> )
py	def <b>set_luminosity</b> ( <b>newval</b> )
cmd	YLed <b>target set_luminosity newval</b>

**Paramètres :**

**newval** un entier représentant l'intensité lumineuse de la led (en pour cent)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**led**→**set\_power()****YLed****led**→**setPower()**[led setPower: ]

Modifie l'état courant de la led.

js	function <b>set_power</b> ( <b>newval</b> )
nodejs	function <b>set_power</b> ( <b>newval</b> )
php	function <b>set_power</b> ( <b>\$newval</b> )
cpp	int <b>set_power</b> ( Y_POWER_enum <b>newval</b> )
m	-(int) setPower : (Y_POWER_enum) <b>newval</b>
pas	function <b>set_power</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_power</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_power</b> ( int <b>newval</b> )
java	int <b>set_power</b> ( int <b>newval</b> )
py	def <b>set_power</b> ( <b>newval</b> )
cmd	YLed <b>target set_power newval</b>

**Paramètres :**

**newval** soit Y\_POWER\_OFF, soit Y\_POWER\_ON, selon l'état courant de la led

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**led**→**set\_userdata()****YLed****led**→**setUserData()**[**led** **setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) <b>setUserData</b> : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**led→wait\_async()****YLed**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.22. Interface de la fonction LightSensor

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_lightsensor.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YLightSensor = yoctolib.YLightSensor;
php	require_once('yocto_lightsensor.php');
c++	#include "yocto_lightsensor.h"
m	#import "yocto_lightsensor.h"
pas	uses yocto_lightsensor;
vb	yocto_lightsensor.vb
cs	yocto_lightsensor.cs
java	import com.yoctopuce.YoctoAPI.YLightSensor;
py	from yocto_lightsensor import *

### Fonction globales

#### yFindLightSensor(func)

Permet de retrouver un capteur de lumière d'après un identifiant donné.

#### yFirstLightSensor()

Commence l'énumération des capteurs de lumière accessibles par la librairie.

### Méthodes des objets YLightSensor

#### lightsensor→calibrate(calibratedVal)

Modifie le paramètre de calibration spécifique du senseur de sorte à ce que la valeur actuelle corresponde à une consigne donnée (correction linéaire).

#### lightsensor→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### lightsensor→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de lumière au format TYPE ( NAME ) = SERIAL . FUNCTIONID.

#### lightsensor→get\_advertisedValue()

Retourne la valeur courante du capteur de lumière (pas plus de 6 caractères).

#### lightsensor→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### lightsensor→get\_currentValue()

Retourne la mesure actuelle de la lumière ambiante.

#### lightsensor→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de lumière.

#### lightsensor→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de lumière.

#### lightsensor→get\_friendlyName()

Retourne un identifiant global du capteur de lumière au format NOM\_MODULE . NOM\_FONCTION.

#### lightsensor→get\_functionDescriptor()

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

#### lightsensor→get\_functionId()

Retourne l'identifiant matériel du capteur de lumière, sans référence au module.

#### **lightsensor**→**get\_hardwareId()**

Retourne l'identifiant matériel unique du capteur de lumière au format `SERIAL.FUNCTIONID`.

#### **lightsensor**→**get\_highestValue()**

Retourne la valeur maximale observée pour la lumière ambiante.

#### **lightsensor**→**get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

#### **lightsensor**→**get\_logicalName()**

Retourne le nom logique du capteur de lumière.

#### **lightsensor**→**get\_lowestValue()**

Retourne la valeur minimale observée pour la lumière ambiante.

#### **lightsensor**→**get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **lightsensor**→**get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **lightsensor**→**get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

#### **lightsensor**→**get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

#### **lightsensor**→**get\_resolution()**

Retourne la résolution des valeurs mesurées.

#### **lightsensor**→**get\_unit()**

Retourne l'unité dans laquelle la lumière ambiante est exprimée.

#### **lightsensor**→**get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **lightsensor**→**isOnline()**

Vérifie si le module hébergeant le capteur de lumière est joignable, sans déclencher d'erreur.

#### **lightsensor**→**isOnline\_async(callback, context)**

Vérifie si le module hébergeant le capteur de lumière est joignable, sans déclencher d'erreur.

#### **lightsensor**→**load(msValidity)**

Met en cache les valeurs courantes du capteur de lumière, avec une durée de validité spécifiée.

#### **lightsensor**→**loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

#### **lightsensor**→**load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du capteur de lumière, avec une durée de validité spécifiée.

#### **lightsensor**→**nextLightSensor()**

Continue l'énumération des capteurs de lumière commencée à l'aide de `yFirstLightSensor()`.

#### **lightsensor**→**registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

#### **lightsensor**→**registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **lightsensor**→**set\_highestValue(newval)**

### 3. Reference

Modifie la mémoire de valeur maximale observée pour la lumière ambiante.

**lightsensor**→**set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**lightsensor**→**set\_logicalName(newval)**

Modifie le nom logique du capteur de lumière.

**lightsensor**→**set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée pour la lumière ambiante.

**lightsensor**→**set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**lightsensor**→**set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**lightsensor**→**set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userData.

**lightsensor**→**wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YLightSensor.FindLightSensor() yFindLightSensor()yFindLightSensor()

## YLightSensor

Permet de retrouver un capteur de lumière d'après un identifiant donné.

js	function <b>yFindLightSensor</b> ( <b>func</b> )
nodejs	function <b>FindLightSensor</b> ( <b>func</b> )
php	function <b>yFindLightSensor</b> ( <b>\$func</b> )
cpp	YLightSensor* <b>yFindLightSensor</b> ( const string& <b>func</b> )
m	YLightSensor* <b>yFindLightSensor</b> ( NSString* <b>func</b> )
pas	function <b>yFindLightSensor</b> ( <b>func</b> : string): TYLightSensor
vb	function <b>yFindLightSensor</b> ( ByVal <b>func</b> As String) As YLightSensor
cs	YLightSensor <b>FindLightSensor</b> ( string <b>func</b> )
java	YLightSensor <b>FindLightSensor</b> ( String <b>func</b> )
py	def <b>FindLightSensor</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de lumière soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YLightSensor.isOnline()` pour tester si le capteur de lumière est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de lumière sans ambiguïté

### Retourne :

un objet de classe `YLightSensor` qui permet ensuite de contrôler le capteur de lumière.

## YLightSensor.FirstLightSensor() yFirstLightSensor()yFirstLightSensor()

YLightSensor

Commence l'énumération des capteurs de lumière accessibles par la librairie.

js	function <b>yFirstLightSensor</b> ( )
nodejs	function <b>FirstLightSensor</b> ( )
php	function <b>yFirstLightSensor</b> ( )
cpp	YLightSensor* <b>yFirstLightSensor</b> ( )
m	YLightSensor* <b>yFirstLightSensor</b> ( )
pas	function <b>yFirstLightSensor</b> ( ): TYLightSensor
vb	function <b>yFirstLightSensor</b> ( ) As YLightSensor
cs	YLightSensor <b>FirstLightSensor</b> ( )
java	YLightSensor <b>FirstLightSensor</b> ( )
py	def <b>FirstLightSensor</b> ( )

Utiliser la fonction `YLightSensor.nextLightSensor( )` pour itérer sur les autres capteurs de lumière.

### Retourne :

un pointeur sur un objet `YLightSensor`, correspondant à le premier capteur de lumière accessible en ligne, ou `null` si il n'y a pas de capteurs de lumière disponibles.

**lightsensor→calibrate() [lightsensor calibrate: ]****YLightSensor**

Modifie le paramètre de calibration spécifique du senseur de sorte à ce que la valeur actuelle corresponde à une consigne donnée (correction linéaire).

<code>js</code>	<code>function calibrate( calibratedVal)</code>
<code>nodejs</code>	<code>function calibrate( calibratedVal)</code>
<code>php</code>	<code>function calibrate( \$calibratedVal)</code>
<code>cpp</code>	<code>int calibrate( double calibratedVal)</code>
<code>m</code>	<code>-(int) calibrate : (double) calibratedVal</code>
<code>pas</code>	<code>function calibrate( calibratedVal: double): integer</code>
<code>vb</code>	<code>function calibrate( ByVal calibratedVal As Double) As Integer</code>
<code>cs</code>	<code>int calibrate( double calibratedVal)</code>
<code>java</code>	<code>int calibrate( double calibratedVal)</code>
<code>py</code>	<code>def calibrate( calibratedVal)</code>
<code>cmd</code>	<code>YLightSensor target calibrate calibratedVal</code>

**Paramètres :**

**calibratedVal** la consigne de valeur désirée.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## lightsensor→calibrateFromPoints()[lightsensor calibrateFromPoints: ]

YLightSensor

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                   refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YLightSensor target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→describe() [lightsensor describe]****YLightSensor**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de lumière au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisée dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de lumière (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## lightsensor→get\_advertisedValue() lightsensor→advertisedValue()[lightsensor advertisedValue]

YLightSensor

Retourne la valeur courante du capteur de lumière (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YLightSensor <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du capteur de lumière (pas plus de 6 caractères).  
En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**lightsensor→get\_currentRawValue()**  
**lightsensor→currentRawValue()[lightsensor**  
**currentRawValue]**

**YLightSensor**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YLightSensor <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**lightsensor→get\_currentValue()****YLightSensor****lightsensor→currentValue()[lightsensor  
currentValue]**

Retourne la mesure actuelle de la lumière ambiante.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YLightSensor <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la mesure actuelle de la lumière ambiante

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.



**lightsensor→get\_errorMessage()****YLightSensor****lightsensor→errorMessage()[lightsensor  
errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de lumière.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de lumière.

**lightsensor→get\_errorType()****YLightSensor****lightsensor→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de lumière.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de lumière.

**lightsensor→get\_friendlyName()**  
**lightsensor→friendlyName()[lightsensor**  
**friendlyName]**

**YLightSensor**

Retourne un identifiant global du capteur de lumière au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de lumière si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de lumière (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de lumière en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## lightsensor→get\_functionDescriptor() lightsensor→functionDescriptor()[lightsensor functionDescriptor]

YLightSensor

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**lightsensor→get\_functionId()****YLightSensor****lightsensor→functionId()[lightsensor functionId]**

Retourne l'identifiant matériel du capteur de lumière, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de lumière (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**lightsensor→get\_hardwareId()****YLightSensor****lightsensor→hardwareId()[lightsensor hardwareId]**

Retourne l'identifiant matériel unique du capteur de lumière au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de lumière (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de lumière (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**lightsensor→get\_highestValue()**  
**lightsensor→highestValue()[lightsensor  
highestValue]**

**YLightSensor**

Retourne la valeur maximale observée pour la lumière ambiante.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YLightSensor <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la lumière ambiante

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

## lightsensor→get\_logFrequency() lightsensor→logFrequency()[lightsensor logFrequency]

YLightSensor

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YLightSensor <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.



**lightsensor→get\_logicalName()****YLightSensor****lightsensor→logicalName()[lightsensor logicalName]**

Retourne le nom logique du capteur de lumière.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YLightSensor <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de lumière. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**lightsensor→get\_lowestValue()****YLightSensor****lightsensor→lowestValue()[lightsensor lowestValue]**

Retourne la valeur minimale observée pour la lumière ambiante.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YLightSensor <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la lumière ambiante

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**lightsensor→get\_module()****YLightSensor****lightsensor→module()[lightsensor module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**lightsensor**→**get\_module\_async()****YLightSensor****lightsensor**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**lightsensor→get\_recordedData()****YLightSensor****lightsensor→recordedData()[lightsensor  
recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YLightSensor <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

## lightsensor→get\_reportFrequency() lightsensor→reportFrequency()[lightsensor reportFrequency]

YLightSensor

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YLightSensor <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**lightsensor**→**get\_resolution()****YLightSensor****lightsensor**→**resolution()**[lightsensor resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YLightSensor <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**lightsensor→get\_unit()****YLightSensor****lightsensor→unit()[lightsensor unit]**

Retourne l'unité dans laquelle la lumière ambiante est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YLightSensor <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la lumière ambiante est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.



**lightsensor→get\_userdata()****YLightSensor****lightsensor→userData()[lightsensor userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
c++	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**lightsensor→isOnline()[lightsensor isOnline]****YLightSensor**

Vérifie si le module hébergeant le capteur de lumière est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de lumière sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de lumière est joignable, false sinon

**lightsensor→isOnline\_async()****YLightSensor**

Vérifie si le module hébergeant le capteur de lumière est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de lumière sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**lightsensor→load()[lightsensor load: ]****YLightSensor**

Met en cache les valeurs courantes du capteur de lumière, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
c++	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## lightsensor→loadCalibrationPoints()[lightsensor loadCalibrationPoints: ]

## YLightSensor

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YLightSensor target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→load\_async()****YLightSensor**

Met en cache les valeurs courantes du capteur de lumière, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## lightsensor→nextLightSensor()[lightsensor nextLightSensor]

## YLightSensor

Continue l'énumération des capteurs de lumière commencée à l'aide de `yFirstLightSensor()`.

js	function nextLightSensor( )
nodejs	function nextLightSensor( )
php	function nextLightSensor( )
cpp	YLightSensor * nextLightSensor( )
m	-(YLightSensor*) nextLightSensor
pas	function nextLightSensor( ): TYLightSensor
vb	function nextLightSensor( ) As YLightSensor
cs	YLightSensor nextLightSensor( )
java	YLightSensor nextLightSensor( )
py	def nextLightSensor( )

### Retourne :

un pointeur sur un objet `YLightSensor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## lightsensor→registerTimedReportCallback() [lightsensor registerTimedReportCallback: ]

YLightSensor

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YLightSensorTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YLightSensorTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYLightSensorTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.



## lightsensor→registerValueCallback()[lightsensor registerValueCallback: ]

## YLightSensor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YLightSensorValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YLightSensorValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYLightSensorValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## lightsensor→set\_highestValue() lightsensor→setHighestValue()[lightsensor setHighestValue: ]

YLightSensor

Modifie la mémoire de valeur maximale observée pour la lumière ambiante.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_highestValue</b> <b>newval</b>

### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour la lumière ambiante

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_logFrequency()****YLightSensor****lightsensor→setLogFrequency() [lightsensor  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
c++	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_logicalName()****YLightSensor****lightsensor→setLogicalName()[lightsensor  
setLogicalName: ]**

Modifie le nom logique du capteur de lumière.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de lumière.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_lowestValue()**  
**lightsensor→setLowestValue()[lightsensor**  
**setLowestValue: ]**

**YLightSensor**

Modifie la mémoire de valeur minimale observée pour la lumière ambiante.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_lowestValue</b> <b>newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour la lumière ambiante

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_reportFrequency()****YLightSensor****lightsensor→setReportFrequency() [lightsensor  
setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_resolution()****YLightSensor****lightsensor→setResolution()[lightsensor  
setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YLightSensor <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**lightsensor→set\_userdata()****YLightSensor****lightsensor→setUserData() [lightsensor setUserData:  
]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser



**lightsensor→wait\_async()****YLightSensor**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.23. Interface de la fonction Magnetometer

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_magnetometer.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YMagnetometer = yoctolib.YMagnetometer;
php	require_once('yocto_magnetometer.php');
c++	#include "yocto_magnetometer.h"
m	#import "yocto_magnetometer.h"
pas	uses yocto_magnetometer;
vb	yocto_magnetometer.vb
cs	yocto_magnetometer.cs
java	import com.yoctopuce.YoctoAPI.YMagnetometer;
py	from yocto_magnetometer import *

### Fonction globales

#### yFindMagnetometer(func)

Permet de retrouver un magnétomètre d'après un identifiant donné.

#### yFirstMagnetometer()

Commence l'énumération des magnétomètres accessibles par la librairie.

### Méthodes des objets YMagnetometer

#### magnetometer→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### magnetometer→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du magnétomètre au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### magnetometer→get\_advertisedValue()

Retourne la valeur courante du magnétomètre (pas plus de 6 caractères).

#### magnetometer→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### magnetometer→get\_currentValue()

Retourne la valeur actuelle du champ magnétique.

#### magnetometer→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du magnétomètre.

#### magnetometer→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du magnétomètre.

#### magnetometer→get\_friendlyName()

Retourne un identifiant global du magnétomètre au format `NOM_MODULE . NOM_FONCTION`.

#### magnetometer→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### magnetometer→get\_functionId()

Retourne l'identifiant matériel du magnétomètre, sans référence au module.

#### magnetometer→get\_hardwareId()

Retourne l'identifiant matériel unique du magnétomètre au format `SERIAL . FUNCTIONID`.

**magnetometer→get\_highestValue()**

Retourne la valeur maximale observée pour le champ magnétique depuis le démarrage du module.

**magnetometer→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**magnetometer→get\_logicalName()**

Retourne le nom logique du magnétomètre.

**magnetometer→get\_lowestValue()**

Retourne la valeur minimale observée pour le champ magnétique depuis le démarrage du module.

**magnetometer→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**magnetometer→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**magnetometer→get\_recordedData(startTime, endTime)**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

**magnetometer→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**magnetometer→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**magnetometer→get\_unit()**

Retourne l'unité dans laquelle le champ magnétique est exprimée.

**magnetometer→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

**magnetometer→get\_xValue()**

Retourne la composante X du champ magnétique, sous forme de nombre à virgule.

**magnetometer→get\_yValue()**

Retourne la composante Y du champ magnétique, sous forme de nombre à virgule.

**magnetometer→get\_zValue()**

Retourne la composante Z du champ magnétique, sous forme de nombre à virgule.

**magnetometer→isOnline()**

Vérifie si le module hébergeant le magnétomètre est joignable, sans déclencher d'erreur.

**magnetometer→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le magnétomètre est joignable, sans déclencher d'erreur.

**magnetometer→load(msValidity)**

Met en cache les valeurs courantes du magnétomètre, avec une durée de validité spécifiée.

**magnetometer→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.

**magnetometer→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du magnétomètre, avec une durée de validité spécifiée.

**magnetometer→nextMagnetometer()**

Continue l'énumération des magnétomètres commencée à l'aide de yFirstMagnetometer().

**magnetometer→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

### 3. Reference

#### **magnetometer→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **magnetometer→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

#### **magnetometer→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

#### **magnetometer→set\_logicalName(newval)**

Modifie le nom logique du magnétomètre.

#### **magnetometer→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

#### **magnetometer→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

#### **magnetometer→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

#### **magnetometer→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

#### **magnetometer→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YMagnetometer.FindMagnetometer() yFindMagnetometer()yFindMagnetometer()

## YMagnetometer

Permet de retrouver un magnétomètre d'après un identifiant donné.

js	function <b>yFindMagnetometer</b> ( <b>func</b> )
nodejs	function <b>FindMagnetometer</b> ( <b>func</b> )
php	function <b>yFindMagnetometer</b> ( <b>\$func</b> )
cpp	YMagnetometer* <b>yFindMagnetometer</b> ( const string& <b>func</b> )
m	YMagnetometer* <b>yFindMagnetometer</b> ( NSString* <b>func</b> )
pas	function <b>yFindMagnetometer</b> ( <b>func</b> : string): TYMagnetometer
vb	function <b>yFindMagnetometer</b> ( ByVal <b>func</b> As String) As YMagnetometer
cs	YMagnetometer <b>FindMagnetometer</b> ( string <b>func</b> )
java	YMagnetometer <b>FindMagnetometer</b> ( String <b>func</b> )
py	def <b>FindMagnetometer</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le magnétomètre soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YMagnetometer.isOnline()` pour tester si le magnétomètre est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le magnétomètre sans ambiguïté

### Retourne :

un objet de classe `YMagnetometer` qui permet ensuite de contrôler le magnétomètre.

## YMagnetometer.FirstMagnetometer() yFirstMagnetometer()yFirstMagnetometer()

## YMagnetometer

Commence l'énumération des magnétomètres accessibles par la librairie.

js	function <b>yFirstMagnetometer</b> ( )
nodejs	function <b>FirstMagnetometer</b> ( )
php	function <b>yFirstMagnetometer</b> ( )
cpp	YMagnetometer* <b>yFirstMagnetometer</b> ( )
m	YMagnetometer* <b>yFirstMagnetometer</b> ( )
pas	function <b>yFirstMagnetometer</b> ( ): TYMagnetometer
vb	function <b>yFirstMagnetometer</b> ( ) As YMagnetometer
cs	YMagnetometer <b>FirstMagnetometer</b> ( )
java	YMagnetometer <b>FirstMagnetometer</b> ( )
py	def <b>FirstMagnetometer</b> ( )

Utiliser la fonction `YMagnetometer.nextMagnetometer()` pour itérer sur les autres magnétomètres.

### Retourne :

un pointeur sur un objet `YMagnetometer`, correspondant à le premier magnétomètre accessible en ligne, ou `null` si il n'y a pas de magnétomètres disponibles.

**magnetometer→calibrateFromPoints()****YMagnetometer****[magnetometer calibrateFromPoints: ]**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

js	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
nodejs	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
php	function <b>calibrateFromPoints</b> ( <b>\$rawValues</b> , <b>\$refValues</b> )
cpp	int <b>calibrateFromPoints</b> ( vector<double> <b>rawValues</b> , vector<double> <b>refValues</b> )
m	-(int) <b>calibrateFromPoints</b> : (NSMutableArray*) <b>rawValues</b> : (NSMutableArray*) <b>refValues</b>
pas	function <b>calibrateFromPoints</b> ( <b>rawValues</b> : TDoubleArray, <b>refValues</b> : TDoubleArray): LongInt
vb	procedure <b>calibrateFromPoints</b> ( )
cs	int <b>calibrateFromPoints</b> ( List<double> <b>rawValues</b> , List<double> <b>refValues</b> )
java	int <b>calibrateFromPoints</b> ( ArrayList<Double> <b>rawValues</b> , ArrayList<Double> <b>refValues</b> )
py	def <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
cmd	YMagnetometer <b>target</b> <b>calibrateFromPoints</b> <b>rawValues</b> <b>refValues</b>

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

**Paramètres :**

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer→describe()[magnetometer describe]****YMagnetometer**

Retourne un court texte décrivant de manière non-ambigüe l'instance du magnétomètre au format  
 TYPE ( NAME ) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.relay1)=unresolved si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le magnétomètre (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)



**magnetometer→get\_advertisedValue()**  
**magnetometer→advertisedValue()[magnetometer**  
**advertisedValue]**

**YMagnetometer**

Retourne la valeur courante du magnétomètre (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du magnétomètre (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**magnetometer**→**get\_currentRawValue()****YMagnetometer****magnetometer**→**currentRawValue()**[**magnetometer**  
**currentRawValue**]

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**magnetometer**→**get\_currentValue()**  
**magnetometer**→**currentValue()**[**magnetometer**  
**currentValue**]

**YMagnetometer**

Retourne la valeur actuelle du champ magnétique.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle du champ magnétique

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**magnetometer**→**get\_errorMessage()****YMagnetometer****magnetometer**→**errorMessage()**[**magnetometer**  
**errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du magnétomètre.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du magnétomètre.

## magnetometer→get\_errorType() magnetometer→errorType()

## YMagnetometer

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du magnétomètre.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

### Retourne :

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du magnétomètre.

**magnetometer**→**get\_friendlyName()****YMagnetometer****magnetometer**→**friendlyName()**[**magnetometer**  
**friendlyName**]

Retourne un identifiant global du magnétomètre au format `NOM_MODULE . NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du magnétomètre si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du magnétomètre (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le magnétomètre en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**magnetometer→get\_functionDescriptor()**  
**magnetometer→functionDescriptor()[magnetometer**  
**functionDescriptor]**

**YMagnetometer**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**magnetometer**→**get\_functionId()**

**YMagnetometer**

**magnetometer**→**functionId()**[**magnetometer**  
**functionId**]

Retourne l'identifiant matériel du magnétomètre, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple relay1.

**Retourne :**

une chaîne de caractères identifiant le magnétomètre (ex: relay1) En cas d'erreur, déclenche une exception ou retourne Y\_FUNCTIONID\_INVALID.



**magnetometer**→**get\_hardwareId()**  
**magnetometer**→**hardwareId()**[**magnetometer**  
**hardwareId**]

**YMagnetometer**

Retourne l'identifiant matériel unique du magnétomètre au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du magnétomètre (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le magnétomètre (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**magnetometer**→**get\_highestValue()****YMagnetometer****magnetometer**→**highestValue()**[**magnetometer**  
**highestValue**]

Retourne la valeur maximale observée pour le champ magnétique depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour le champ magnétique depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

magnetometer→get\_logFrequency()

YMagnetometer

magnetometer→logFrequency()[magnetometer  
logFrequency]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**magnetometer**→**get\_logicalName()****YMagnetometer****magnetometer**→**logicalName()**[**magnetometer**  
**logicalName**]

Retourne le nom logique du magnétomètre.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du magnétomètre. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**magnetometer**→**get\_lowestValue()****YMagnetometer****magnetometer**→**lowestValue()**[**magnetometer**  
**lowestValue**]

Retourne la valeur minimale observée pour le champ magnétique depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour le champ magnétique depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**magnetometer**→**get\_module()****YMagnetometer****magnetometer**→**module()**[magnetometer module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**magnetometer**→**get\_module\_async()****YMagnetometer****magnetometer**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**magnetometer**→**get\_recordedData()****YMagnetometer****magnetometer**→**recordedData()[magnetometer  
recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YMagnetometer <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.



**magnetometer→get\_reportFrequency()**  
**magnetometer→reportFrequency()[magnetometer**  
**reportFrequency]**

**YMagnetometer**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**magnetometer**→**get\_resolution()****YMagnetometer****magnetometer**→**resolution()**[**magnetometer**  
**resolution**]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**magnetometer**→**get\_unit()****YMagnetometer****magnetometer**→**unit()**[magnetometer unit]

Retourne l'unité dans laquelle le champ magnétique est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle le champ magnétique est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**magnetometer→get\_userdata()**

**YMagnetometer**

**magnetometer→userData()[magnetometer userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**magnetometer→get\_xValue()****YMagnetometer****magnetometer→xValue()[magnetometer xValue]**

Retourne la composante X du champ magnétique, sous forme de nombre à virgule.

js	function <b>get_xValue</b> ( )
nodejs	function <b>get_xValue</b> ( )
php	function <b>get_xValue</b> ( )
cpp	double <b>get_xValue</b> ( )
m	-(double) xValue
pas	function <b>get_xValue</b> ( ): double
vb	function <b>get_xValue</b> ( ) As Double
cs	double <b>get_xValue</b> ( )
java	double <b>get_xValue</b> ( )
py	def <b>get_xValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_xValue</b>

**Retourne :**

une valeur numérique représentant la composante X du champ magnétique, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_XVALUE\_INVALID.

magnetometer→get\_yValue()

YMagnetometer

magnetometer→yValue()[magnetometer yValue]

Retourne la composante Y du champ magnétique, sous forme de nombre à virgule.

js	function <b>get_yValue</b> ( )
nodejs	function <b>get_yValue</b> ( )
php	function <b>get_yValue</b> ( )
cpp	double <b>get_yValue</b> ( )
m	-(double) yValue
pas	function <b>get_yValue</b> ( ): double
vb	function <b>get_yValue</b> ( ) As Double
cs	double <b>get_yValue</b> ( )
java	double <b>get_yValue</b> ( )
py	def <b>get_yValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_yValue</b>

**Retourne :**

une valeur numérique représentant la composante Y du champ magnétique, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_YVALUE\_INVALID.

**magnetometer→get\_zValue()****YMagnetometer****magnetometer→zValue()[magnetometer zValue]**

Retourne la composante Z du champ magnétique, sous forme de nombre à virgule.

js	function <b>get_zValue</b> ( )
nodejs	function <b>get_zValue</b> ( )
php	function <b>get_zValue</b> ( )
cpp	double <b>get_zValue</b> ( )
m	-(double) zValue
pas	function <b>get_zValue</b> ( ): double
vb	function <b>get_zValue</b> ( ) As Double
cs	double <b>get_zValue</b> ( )
java	double <b>get_zValue</b> ( )
py	def <b>get_zValue</b> ( )
cmd	YMagnetometer <b>target</b> <b>get_zValue</b>

**Retourne :**

une valeur numérique représentant la composante Z du champ magnétique, sous forme de nombre à virgule

En cas d'erreur, déclenche une exception ou retourne Y\_ZVALUE\_INVALID.

**magnetometer→isOnline()[magnetometer isOnline]****YMagnetometer**

Vérifie si le module hébergeant le magnétomètre est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du magnétomètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le magnétomètre est joignable, false sinon



**magnetometer→isOnline\_async()****YMagnetometer**

Vérifie si le module hébergeant le magnétomètre est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du magnétomètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**magnetometer→load()[magnetometer load: ]****YMagnetometer**

Met en cache les valeurs courantes du magnétomètre, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
c++	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer→loadCalibrationPoints()****YMagnetometer****[magnetometer loadCalibrationPoints: ]**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

<code>js</code>	<code>function loadCalibrationPoints( rawValues, refValues)</code>
<code>nodejs</code>	<code>function loadCalibrationPoints( rawValues, refValues)</code>
<code>php</code>	<code>function loadCalibrationPoints( &amp;\$rawValues, &amp;\$refValues)</code>
<code>cpp</code>	<code>int loadCalibrationPoints( vector&lt;double&gt;&amp; rawValues, vector&lt;double&gt;&amp; refValues)</code>
<code>m</code>	<code>-(int) loadCalibrationPoints : (NSMutableArray*) rawValues : (NSMutableArray*) refValues</code>
<code>pas</code>	<code>function loadCalibrationPoints( var rawValues: TDoubleArray, var refValues: TDoubleArray): LongInt</code>
<code>vb</code>	<code>procedure loadCalibrationPoints( )</code>
<code>cs</code>	<code>int loadCalibrationPoints( List&lt;double&gt; rawValues, List&lt;double&gt; refValues)</code>
<code>java</code>	<code>int loadCalibrationPoints( ArrayList&lt;Double&gt; rawValues, ArrayList&lt;Double&gt; refValues)</code>
<code>py</code>	<code>def loadCalibrationPoints( rawValues, refValues)</code>
<code>cmd</code>	<code>YMagnetometer target loadCalibrationPoints rawValues refValues</code>

**Paramètres :**

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer→load\_async()****YMagnetometer**

Met en cache les valeurs courantes du magnétomètre, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**magnetometer**→**nextMagnetometer()**[**magnetometer**  
**nextMagnetometer**]

**YMagnetometer**

Continue l'énumération des magnétomètres commencée à l'aide de `yFirstMagnetometer()`.

js	function <b>nextMagnetometer</b> ( )
nodejs	function <b>nextMagnetometer</b> ( )
php	function <b>nextMagnetometer</b> ( )
cpp	YMagnetometer * <b>nextMagnetometer</b> ( )
m	-(YMagnetometer*) <b>nextMagnetometer</b>
pas	function <b>nextMagnetometer</b> ( ): TYMagnetometer
vb	function <b>nextMagnetometer</b> ( ) As YMagnetometer
cs	YMagnetometer <b>nextMagnetometer</b> ( )
java	YMagnetometer <b>nextMagnetometer</b> ( )
py	def <b>nextMagnetometer</b> ( )

**Retourne :**

un pointeur sur un objet `YMagnetometer` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## magnetometer→registerTimedReportCallback() [magnetometer registerTimedReportCallback: ]

YMagnetometer

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YMagnetometerTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YMagnetometerTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYMagnetometerTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

**magnetometer→registerValueCallback()****YMagnetometer****[magnetometer registerValueCallback: ]**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YMagnetometerValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YMagnetometerValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYMagnetometerValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

magnetometer→set\_highestValue()

YMagnetometer

magnetometer→setHighestValue()[magnetometer  
setHighestValue: ]

Modifie la mémoire de valeur maximale observée.

js	function set_highestValue( newval)
nodejs	function set_highestValue( newval)
php	function set_highestValue( \$newval)
cpp	int set_highestValue( double newval)
m	-(int) setHighestValue : (double) newval
pas	function set_highestValue( newval: double): integer
vb	function set_highestValue( ByVal newval As Double) As Integer
cs	int set_highestValue( double newval)
java	int set_highestValue( double newval)
py	def set_highestValue( newval)
cmd	YMagnetometer target set_highestValue newval

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**magnetometer→set\_logFrequency()****YMagnetometer****magnetometer→setLogFrequency()[magnetometer  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YMagnetometer <b>target set_logFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer**→**set\_logicalName()****YMagnetometer****magnetometer**→**setLogicalName()**[**magnetometer**  
**setLogicalName:** ]

Modifie le nom logique du magnétomètre.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YMagnetometer <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du magnétomètre.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer→set\_lowestValue()****YMagnetometer****magnetometer→setLowestValue()[magnetometer  
setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YMagnetometer <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer→set\_reportFrequency()****YMagnetometer****magnetometer→setReportFrequency()****[magnetometer setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YMagnetometer <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer**→**set\_resolution()****YMagnetometer****magnetometer**→**setResolution()**[**magnetometer**  
**setResolution:** ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YMagnetometer <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**magnetometer**→**set\_userdata()****YMagnetometer****magnetometer**→**setUserData()**[**magnetometer**  
**setUserData:** ]

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :****data** objet quelconque à mémoriser

**magnetometer→wait\_async()****YMagnetometer**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.24. Valeur mesurée

Les objets YMeasure sont utilisés dans l'interface de programmation Yoctopuce pour représenter une valeur observée un moment donnée. Ces objets sont utilisés en particulier en conjonction avec la classe YDataSet.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_api.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YAPI = yoctolib.YAPI; var YModule = yoctolib.YModule;
php	require_once('yocto_api.php');
cpp	#include "yocto_api.h"
m	#import "yocto_api.h"
pas	uses yocto_api;
vb	yocto_api.vb
cs	yocto_api.cs
java	import com.yoctopuce.YoctoAPI.YModule;
py	from yocto_api import *

### Méthodes des objets YMeasure

#### **measure→get\_averageValue()**

Retourne la valeur moyenne observée durant l'intervalle de temps couvert par la mesure.

#### **measure→get\_endTimeUTC()**

Retourne l'heure absolue de la fin de la mesure, sous forme du nombre de secondes depuis le 1er janvier 1970 UTC (date/heure au format Unix).

#### **measure→get\_maxValue()**

Retourne la plus grande valeur observée durant l'intervalle de temps couvert par la mesure.

#### **measure→get\_minValue()**

Retourne la plus petite valeur observée durant l'intervalle de temps couvert par la mesure.

#### **measure→get\_startTimeUTC()**

Retourne l'heure absolue du début de la mesure, sous forme du nombre de secondes depuis le 1er janvier 1970 UTC (date/heure au format Unix).



**measure**→**get\_averageValue()****YMeasure****measure**→**averageValue()[measure averageValue]**

Retourne la valeur moyenne observée durant l'intervalle de temps couvert par la mesure.

js	function <b>get_averageValue</b> ( )
nodejs	function <b>get_averageValue</b> ( )
php	function <b>get_averageValue</b> ( )
cpp	double <b>get_averageValue</b> ( )
m	-(double) averageValue
pas	function <b>get_averageValue</b> ( ): double
vb	function <b>get_averageValue</b> ( ) As Double
cs	double <b>get_averageValue</b> ( )
java	double <b>get_averageValue</b> ( )
py	def <b>get_averageValue</b> ( )

**Retourne :**

un nombre décimal correspondant à la valeur moyenne observée.

**measure**→**get\_endTimeUTC()**

**YMeasure**

**measure**→**endTimeUTC()**[**measure endTimeUTC**]

---

Retourne l'heure absolue de la fin de la mesure, sous forme du nombre de secondes depuis le 1er janvier 1970 UTC (date/heure au format Unix).

js	function <b>get_endTimeUTC</b> ( )
nodejs	function <b>get_endTimeUTC</b> ( )
php	function <b>get_endTimeUTC</b> ( )
cpp	double <b>get_endTimeUTC</b> ( )
m	-(double) endTimeUTC
pas	function <b>get_endTimeUTC</b> ( ): double
vb	function <b>get_endTimeUTC</b> ( ) As Double
cs	double <b>get_endTimeUTC</b> ( )
java	double <b>get_endTimeUTC</b> ( )
py	def <b>get_endTimeUTC</b> ( )

Lors que l'enregistrement de données se fait à une fréquence supérieure à une mesure par seconde, le timestamp peuvent inclure une fraction décimale.

**Retourne :**

un nombre réel positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 UTC et la fin de la mesure.

**measure**→**get\_maxValue()****YMeasure****measure**→**maxValue()**[**measure** **maxValue**]

Retourne la plus grande valeur observée durant l'intervalle de temps couvert par la mesure.

js	function <b>get_maxValue</b> ( )
nodejs	function <b>get_maxValue</b> ( )
php	function <b>get_maxValue</b> ( )
cpp	double <b>get_maxValue</b> ( )
m	-(double) <b>maxValue</b>
pas	function <b>get_maxValue</b> ( ): double
vb	function <b>get_maxValue</b> ( ) As Double
cs	double <b>get_maxValue</b> ( )
java	double <b>get_maxValue</b> ( )
py	def <b>get_maxValue</b> ( )

**Retourne :**

un nombre décimal correspondant à la plus grande valeur observée.

**measure**→**get\_minValue()**

**YMeasure**

**measure**→**minValue()**[**measure** **minValue**]

---

Retourne la plus petite valeur observée durant l'intervalle de temps couvert par la mesure.

js	function <b>get_minValue</b> ( )
nodejs	function <b>get_minValue</b> ( )
php	function <b>get_minValue</b> ( )
cpp	double <b>get_minValue</b> ( )
m	-(double) minValue
pas	function <b>get_minValue</b> ( ): double
vb	function <b>get_minValue</b> ( ) As Double
cs	double <b>get_minValue</b> ( )
java	double <b>get_minValue</b> ( )
py	def <b>get_minValue</b> ( )

**Retourne :**

un nombre décimal correspondant à la plus petite valeur observée.

**measure→get\_startTimeUTC()****YMeasure****measure→startTimeUTC()[measure startTimeUTC]**

Retourne l'heure absolue du début de la mesure, sous forme du nombre de secondes depuis le 1er janvier 1970 UTC (date/heure au format Unix).

js	function <b>get_startTimeUTC</b> ( )
nodejs	function <b>get_startTimeUTC</b> ( )
php	function <b>get_startTimeUTC</b> ( )
cpp	double <b>get_startTimeUTC</b> ( )
m	-(double) startTimeUTC
pas	function <b>get_startTimeUTC</b> ( ): double
vb	function <b>get_startTimeUTC</b> ( ) As Double
cs	double <b>get_startTimeUTC</b> ( )
java	double <b>get_startTimeUTC</b> ( )
py	def <b>get_startTimeUTC</b> ( )

Lors que l'enregistrement de données se fait à une fréquence supérieure à une mesure par seconde, le timestamp peuvent inclure une fraction décimale.

**Retourne :**

un nombre réel positif correspondant au nombre de secondes écoulées entre le 1er janvier 1970 UTC et le début de la mesure.

### 3.25. Interface de contrôle du module

Cette interface est la même pour tous les modules USB de Yoctopuce. Elle permet de contrôler les paramètres généraux du module, et d'énumérer les fonctions fournies par chaque module.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_api.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YAPI = yoctolib.YAPI; var YModule = yoctolib.YModule;
php	require_once('yocto_api.php');
cpp	#include "yocto_api.h"
m	#import "yocto_api.h"
pas	uses yocto_api;
vb	yocto_api.vb
cs	yocto_api.cs
java	import com.yoctopuce.YoctoAPI.YModule;
py	from yocto_api import *

Fonction globales
<b>yFindModule(func)</b> Permet de retrouver un module d'après son numéro de série ou son nom logique.
<b>yFirstModule()</b> Commence l'énumération des modules accessibles par la librairie.
Méthodes des objets YModule
<b>module→describe()</b> Retourne un court texte décrivant le module.
<b>module→download(pathname)</b> Télécharge le fichier choisi du module et retourne son contenu.
<b>module→functionCount()</b> Retourne le nombre de fonctions (sans compter l'interface "module") existant sur le module.
<b>module→functionId(functionIndex)</b> Retourne l'identifiant matériel de la <i>nième</i> fonction du module.
<b>module→functionName(functionIndex)</b> Retourne le nom logique de la <i>nième</i> fonction du module.
<b>module→functionValue(functionIndex)</b> Retourne la valeur publiée par la <i>nième</i> fonction du module.
<b>module→get_beacon()</b> Retourne l'état de la balise de localisation.
<b>module→get_errorMessage()</b> Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.
<b>module→get_errorType()</b> Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.
<b>module→get_firmwareRelease()</b> Retourne la version du logiciel embarqué du module.
<b>module→get_hardwareId()</b> Retourne l'identifiant unique du module.
<b>module→get_icon2d()</b>

	Retourne l'icône du module.
<b>module→get_lastLogs()</b>	Retourne une chaîne de caractère contenant les derniers logs du module.
<b>module→get_logicalName()</b>	Retourne le nom logique du module.
<b>module→get_luminosity()</b>	Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).
<b>module→get_persistentSettings()</b>	Retourne l'état courant des réglages persistents du module.
<b>module→get_productId()</b>	Retourne l'identifiant USB du module, préprogrammé en usine.
<b>module→get_productName()</b>	Retourne le nom commercial du module, préprogrammé en usine.
<b>module→get_productRelease()</b>	Retourne le numéro de version matériel du module, préprogrammé en usine.
<b>module→get_rebootCountdown()</b>	Retourne le nombre de secondes restantes avant un redémarrage du module, ou zéro si aucun redémarrage n'a été agendé.
<b>module→get_serialNumber()</b>	Retourne le numéro de série du module, préprogrammé en usine.
<b>module→get_upTime()</b>	Retourne le nombre de millisecondes écoulées depuis la mise sous tension du module
<b>module→get_usbBandwidth()</b>	Retourne le nombre d'interface USB utilisé par le module.
<b>module→get_usbCurrent()</b>	Retourne le courant consommé par le module sur le bus USB, en milliampères.
<b>module→get_userData()</b>	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userdata.
<b>module→isOnline()</b>	Vérifie si le module est joignable, sans déclencher d'erreur.
<b>module→isOnline_async(callback, context)</b>	Vérifie si le module est joignable, sans déclencher d'erreur.
<b>module→load(msValidity)</b>	Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.
<b>module→load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.
<b>module→nextModule()</b>	Continue l'énumération des modules commencée à l'aide de yFirstModule().
<b>module→reboot(secBeforeReboot)</b>	Agende un simple redémarrage du module dans un nombre donné de secondes.
<b>module→registerLogCallback(callback)</b>	todo
<b>module→revertFromFlash()</b>	Recharge les réglages stockés dans le mémoire non volatile du module, comme à la mise sous tension du module.
<b>module→saveToFlash()</b>	

### 3. Reference

Sauve les réglages courants dans la mémoire non volatile du module.

**module**→**set\_beacon**(**newval**)

Allume ou éteint la balise de localisation du module.

**module**→**set\_logicalName**(**newval**)

Change le nom logique du module.

**module**→**set\_luminosity**(**newval**)

Modifie la luminosité des leds informatives du module.

**module**→**set\_usbBandwidth**(**newval**)

Modifie le nombre d'interface USB utilisé par le module.

**module**→**set\_userData**(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**module**→**triggerFirmwareUpdate**(**secBeforeReboot**)

Agende un redémarrage du module en mode spécial de reprogrammation du logiciel embarqué.

**module**→**wait\_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YModule.FindModule() yFindModule()yFindModule()

## YModule

Permet de retrouver un module d'après son numéro de série ou son nom logique.

js	function <b>yFindModule</b> ( <b>func</b> )
nodejs	function <b>FindModule</b> ( <b>func</b> )
php	function <b>yFindModule</b> ( <b>\$func</b> )
cpp	YModule* <b>yFindModule</b> ( string <b>func</b> )
m	+(YModule*) <b>yFindModule</b> : (NSString*) <b>func</b>
pas	function <b>yFindModule</b> ( <b>func</b> : string): TYModule
vb	function <b>yFindModule</b> ( ByVal <b>func</b> As String) As YModule
cs	YModule <b>FindModule</b> ( string <b>func</b> )
java	YModule <b>FindModule</b> ( String <b>func</b> )
py	def <b>FindModule</b> ( <b>func</b> )

Cette fonction n'exige pas que le module soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YModule.isOnline()` pour tester si le module est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères contenant soit le numéro de série, soit le nom logique du module désiré

### Retourne :

un objet de classe `YModule` qui permet ensuite de contrôler le module ou d'obtenir de plus amples informations sur le module.

## YModule.FirstModule() yFirstModule()yFirstModule()

YModule

Commence l'énumération des modules accessibles par la librairie.

js	function <b>yFirstModule</b> ( )
nodejs	function <b>FirstModule</b> ( )
php	function <b>yFirstModule</b> ( )
cpp	YModule* <b>yFirstModule</b> ( )
m	YModule* <b>yFirstModule</b> ( )
pas	function <b>yFirstModule</b> ( ): TYModule
vb	function <b>yFirstModule</b> ( ) As YModule
cs	YModule <b>FirstModule</b> ( )
java	YModule <b>FirstModule</b> ( )
py	def <b>FirstModule</b> ( )

Utiliser la fonction `YModule.nextModule( )` pour itérer sur les autres modules.

### Retourne :

un pointeur sur un objet `YModule`, correspondant au premier module accessible en ligne, ou `null` si aucun module n'a été trouvé.

**module**→**describe()**[**module describe**]**YModule**

Retourne un court texte décrivant le module.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Ce texte peut contenir soit le nom logique du module, soit son numéro de série.

**Retourne :**

une chaîne de caractères décrivant le module

**module→download()[module download: ]****YModule**

Télécharge le fichier choisi du module et retourne son contenu.

js	function <b>download</b> ( <b>pathname</b> )
nodejs	function <b>download</b> ( <b>pathname</b> )
php	function <b>download</b> ( <b>\$pathname</b> )
cpp	string <b>download</b> ( string <b>pathname</b> )
m	-(NSData*) <b>download</b> : (NSString*) <b>pathname</b>
pas	function <b>download</b> ( <b>pathname</b> : string): TByteArray
vb	function <b>download</b> ( ) As Byte
py	def <b>download</b> ( <b>pathname</b> )
cmd	YModule <b>target download</b> <b>pathname</b>

**Paramètres :**

**pathname** nom complet du fichier

**Retourne :**

le contenu du fichier chargé

En cas d'erreur, déclenche une exception ou retourne un contenu vide.

---

**module**→**functionCount()**[**module functionCount**]**YModule**

---

Retourne le nombre de fonctions (sans compter l'interface "module") existant sur le module.

js	function <b>functionCount</b> ( )
nodejs	function <b>functionCount</b> ( )
php	function <b>functionCount</b> ( )
cpp	int <b>functionCount</b> ( )
m	-(int) <b>functionCount</b>
pas	function <b>functionCount</b> ( ): integer
vb	function <b>functionCount</b> ( ) As Integer
cs	int <b>functionCount</b> ( )
py	def <b>functionCount</b> ( )

**Retourne :**

le nombre de fonctions sur le module

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**functionId()**[**module functionId:** ]**YModule**

Retourne l'identifiant matériel de la *n*ième fonction du module.

js	function <b>functionId</b> ( <b>functionIndex</b> )
nodejs	function <b>functionId</b> ( <b>functionIndex</b> )
php	function <b>functionId</b> ( <b>\$functionIndex</b> )
cpp	string <b>functionId</b> ( int <b>functionIndex</b> )
m	-(NSString*) <b>functionId</b> : (int) <b>functionIndex</b>
pas	function <b>functionId</b> ( <b>functionIndex</b> : integer): string
vb	function <b>functionId</b> ( ByVal <b>functionIndex</b> As Integer) As String
cs	string <b>functionId</b> ( int <b>functionIndex</b> )
py	def <b>functionId</b> ( <b>functionIndex</b> )

**Paramètres :**

**functionIndex** l'index de la fonction pour laquelle l'information est désirée, en commençant à 0 pour la première fonction.

**Retourne :**

une chaîne de caractères correspondant à l'identifiant matériel unique de la fonction désirée

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

**module**→**functionName()**[**module functionName:** ]

**YModule**

Retourne le nom logique de la *nième* fonction du module.

js	function <b>functionName</b> ( <b>functionIndex</b> )
nodejs	function <b>functionName</b> ( <b>functionIndex</b> )
php	function <b>functionName</b> ( <b>\$functionIndex</b> )
cpp	string <b>functionName</b> ( int <b>functionIndex</b> )
m	-(NSString*) <b>functionName</b> : (int) <b>functionIndex</b>
pas	function <b>functionName</b> ( <b>functionIndex</b> : integer): string
vb	function <b>functionName</b> ( ByVal <b>functionIndex</b> As Integer) As String
cs	string <b>functionName</b> ( int <b>functionIndex</b> )
py	def <b>functionName</b> ( <b>functionIndex</b> )

#### Paramètres :

**functionIndex** l'index de la fonction pour laquelle l'information est désirée, en commençant à 0 pour la première fonction.

#### Retourne :

une chaîne de caractères correspondant au nom logique de la fonction désirée

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

**module**→**functionValue()**[**module functionValue:** ]**YModule**

Retourne la valeur publiée par la *nième* fonction du module.

js	function <b>functionValue</b> ( <b>functionIndex</b> )
nodejs	function <b>functionValue</b> ( <b>functionIndex</b> )
php	function <b>functionValue</b> ( <b>\$functionIndex</b> )
cpp	string <b>functionValue</b> ( int <b>functionIndex</b> )
m	-(NSString*) <b>functionValue</b> : (int) <b>functionIndex</b>
pas	function <b>functionValue</b> ( <b>functionIndex</b> : integer): string
vb	function <b>functionValue</b> ( ByVal <b>functionIndex</b> As Integer) As String
cs	string <b>functionValue</b> ( int <b>functionIndex</b> )
py	def <b>functionValue</b> ( <b>functionIndex</b> )

**Paramètres :**

**functionIndex** l'index de la fonction pour laquelle l'information est désirée, en commençant à 0 pour la première fonction.

**Retourne :**

une chaîne de caractères correspondant à la valeur publiée par la fonction désirée

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.



**module**→**get\_beacon()****YModule****module**→**beacon()**[**module beacon**]

Retourne l'état de la balise de localisation.

js	function <b>get_beacon</b> ( )
nodejs	function <b>get_beacon</b> ( )
php	function <b>get_beacon</b> ( )
cpp	Y_BEACON_enum <b>get_beacon</b> ( )
m	-(Y_BEACON_enum) beacon
pas	function <b>get_beacon</b> ( ): Integer
vb	function <b>get_beacon</b> ( ) As Integer
cs	int <b>get_beacon</b> ( )
java	int <b>get_beacon</b> ( )
py	def <b>get_beacon</b> ( )
cmd	YModule <b>target</b> <b>get_beacon</b>

**Retourne :**

soit Y\_BEACON\_OFF, soit Y\_BEACON\_ON, selon l'état de la balise de localisation

En cas d'erreur, déclenche une exception ou retourne Y\_BEACON\_INVALID.

**module**→**get\_errorMessage()****YModule****module**→**errorMessage()**[**module errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du module

**module**→**get\_errorType()**  
**module**→**errorType()**

**YModule**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du module

**module→get\_firmwareRelease()**  
**module→firmwareRelease()[module  
firmwareRelease]**

**YModule**

Retourne la version du logiciel embarqué du module.

js	function <b>get_firmwareRelease</b> ( )
nodejs	function <b>get_firmwareRelease</b> ( )
php	function <b>get_firmwareRelease</b> ( )
cpp	string <b>get_firmwareRelease</b> ( )
m	-(NSString*) firmwareRelease
pas	function <b>get_firmwareRelease</b> ( ): string
vb	function <b>get_firmwareRelease</b> ( ) As String
cs	string <b>get_firmwareRelease</b> ( )
java	String <b>get_firmwareRelease</b> ( )
py	def <b>get_firmwareRelease</b> ( )
cmd	YModule <b>target</b> <b>get_firmwareRelease</b>

**Retourne :**

une chaîne de caractères représentant la version du logiciel embarqué du module

En cas d'erreur, déclenche une exception ou retourne Y\_FIRMWARERELEASE\_INVALID.

---

**module**→**get\_hardwareId()****YModule****module**→**hardwareId()**[**module hardwareId**]

---

Retourne l'identifiant unique du module.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module suivi de la chaîne ".module".

**Retourne :**

une chaîne de caractères identifiant la fonction

**module**→**get\_icon2d()**

**YModule**

**module**→**icon2d()**[module icon2d]

Retourne l'icône du module.

js	function <b>get_icon2d</b> ( )
nodejs	function <b>get_icon2d</b> ( )
php	function <b>get_icon2d</b> ( )
cpp	string <b>get_icon2d</b> ( )
m	-(NSData*) icon2d
pas	function <b>get_icon2d</b> ( ): TByteArray
vb	function <b>get_icon2d</b> ( ) As Byte
py	def <b>get_icon2d</b> ( )
cmd	YModule <b>target</b> <b>get_icon2d</b>

L'icone est au format PNG et a une taille maximale de 1536 octets.

**Retourne :**

un buffer binaire contenant l'icone, au format png.

**module**→**get\_lastLogs()****YModule****module**→**lastLogs()**[**module lastLogs**]

Retourne une chaine de caractère contenant les derniers logs du module.

js	function <b>get_lastLogs</b> ( )
nodejs	function <b>get_lastLogs</b> ( )
php	function <b>get_lastLogs</b> ( )
cpp	string <b>get_lastLogs</b> ( )
m	-(NSString*) lastLogs
pas	function <b>get_lastLogs</b> ( ): string
vb	function <b>get_lastLogs</b> ( ) As String
cs	string <b>get_lastLogs</b> ( )
java	String <b>get_lastLogs</b> ( )
py	def <b>get_lastLogs</b> ( )
cmd	YModule <b>target</b> <b>get_lastLogs</b>

Cette methode retourne les derniers logs qui sont encore stocké dans le module.

**Retourne :**

une chaine de caractère contenant les derniers logs du module.

**module**→**get\_logicalName()****YModule****module**→**logicalName()**[**module** **logicalName**]

Retourne le nom logique du module.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YModule <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.



**module**→**get\_luminosity()****YModule****module**→**luminosity()**[**module luminosity**]

Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).

js	function <b>get_luminosity</b> ( )
nodejs	function <b>get_luminosity</b> ( )
php	function <b>get_luminosity</b> ( )
cpp	int <b>get_luminosity</b> ( )
m	-(int) luminosity
pas	function <b>get_luminosity</b> ( ): LongInt
vb	function <b>get_luminosity</b> ( ) As Integer
cs	int <b>get_luminosity</b> ( )
java	int <b>get_luminosity</b> ( )
py	def <b>get_luminosity</b> ( )
cmd	YModule <b>target</b> <b>get_luminosity</b>

**Retourne :**

un entier représentant la luminosité des leds informatives du module (valeur entre 0 et 100)

En cas d'erreur, déclenche une exception ou retourne Y\_LUMINOSITY\_INVALID.

**module**→**get\_persistentSettings()**  
**module**→**persistentSettings()[module**  
**persistentSettings]**

**YModule**

Retourne l'état courant des réglages persistents du module.

js	function <b>get_persistentSettings</b> ( )
nodejs	function <b>get_persistentSettings</b> ( )
php	function <b>get_persistentSettings</b> ( )
cpp	Y_PERSISTENTSETTINGS_enum <b>get_persistentSettings</b> ( )
m	-(Y_PERSISTENTSETTINGS_enum) persistentSettings
pas	function <b>get_persistentSettings</b> ( ): Integer
vb	function <b>get_persistentSettings</b> ( ) As Integer
cs	int <b>get_persistentSettings</b> ( )
java	int <b>get_persistentSettings</b> ( )
py	def <b>get_persistentSettings</b> ( )
cmd	YModule <b>target</b> <b>get_persistentSettings</b>

**Retourne :**

une valeur parmi Y\_PERSISTENTSETTINGS\_LOADED, Y\_PERSISTENTSETTINGS\_SAVED et Y\_PERSISTENTSETTINGS\_MODIFIED représentant l'état courant des réglages persistents du module

En cas d'erreur, déclenche une exception ou retourne Y\_PERSISTENTSETTINGS\_INVALID.

**module**→**get\_productId()****YModule****module**→**productId()**[**module productId**]

Retourne l'identifiant USB du module, préprogrammé en usine.

js	function <b>get_productId</b> ( )
nodejs	function <b>get_productId</b> ( )
php	function <b>get_productId</b> ( )
cpp	int <b>get_productId</b> ( )
m	-(int) productId
pas	function <b>get_productId</b> ( ): LongInt
vb	function <b>get_productId</b> ( ) As Integer
cs	int <b>get_productId</b> ( )
java	int <b>get_productId</b> ( )
py	def <b>get_productId</b> ( )
cmd	YModule <b>target</b> <b>get_productId</b>

**Retourne :**

un entier représentant l'identifiant USB du module, préprogrammé en usine

En cas d'erreur, déclenche une exception ou retourne Y\_PRODUCTID\_INVALID.

**module**→**get\_productName()****YModule****module**→**productName()**[**module productName**]

Retourne le nom commercial du module, préprogrammé en usine.

js	function <b>get_productName</b> ( )
nodejs	function <b>get_productName</b> ( )
php	function <b>get_productName</b> ( )
cpp	string <b>get_productName</b> ( )
m	-(NSString*) productName
pas	function <b>get_productName</b> ( ): string
vb	function <b>get_productName</b> ( ) As String
cs	string <b>get_productName</b> ( )
java	String <b>get_productName</b> ( )
py	def <b>get_productName</b> ( )
cmd	YModule <b>target</b> <b>get_productName</b>

**Retourne :**

une chaîne de caractères représentant le nom commercial du module, préprogrammé en usine

En cas d'erreur, déclenche une exception ou retourne Y\_PRODUCTNAME\_INVALID.

**module→get\_productRelease()****YModule****module→productRelease()[module productRelease]**

Retourne le numéro de version matériel du module, préprogrammé en usine.

js	function <b>get_productRelease</b> ( )
nodejs	function <b>get_productRelease</b> ( )
php	function <b>get_productRelease</b> ( )
cpp	int <b>get_productRelease</b> ( )
m	-(int) productRelease
pas	function <b>get_productRelease</b> ( ): LongInt
vb	function <b>get_productRelease</b> ( ) As Integer
cs	int <b>get_productRelease</b> ( )
java	int <b>get_productRelease</b> ( )
py	def <b>get_productRelease</b> ( )
cmd	YModule <b>target</b> <b>get_productRelease</b>

**Retourne :**

un entier représentant le numéro de version matériel du module, préprogrammé en usine

En cas d'erreur, déclenche une exception ou retourne Y\_PRODUCTRELEASE\_INVALID.

**module**→**get\_rebootCountdown()**  
**module**→**rebootCountdown()**[**module**  
**rebootCountdown**]

**YModule**

Retourne le nombre de secondes restantes avant un redémarrage du module, ou zéro si aucun redémarrage n'a été agendé.

js	function <b>get_rebootCountdown</b> ( )
nodejs	function <b>get_rebootCountdown</b> ( )
php	function <b>get_rebootCountdown</b> ( )
cpp	int <b>get_rebootCountdown</b> ( )
m	-(int) rebootCountdown
pas	function <b>get_rebootCountdown</b> ( ): LongInt
vb	function <b>get_rebootCountdown</b> ( ) As Integer
cs	int <b>get_rebootCountdown</b> ( )
java	int <b>get_rebootCountdown</b> ( )
py	def <b>get_rebootCountdown</b> ( )
cmd	YModule <b>target</b> <b>get_rebootCountdown</b>

**Retourne :**

un entier représentant le nombre de secondes restantes avant un redémarrage du module, ou zéro si aucun redémarrage n'a été agendé

En cas d'erreur, déclenche une exception ou retourne Y\_REBOOTCOUNTDOWN\_INVALID.

**module**→**get\_serialNumber()****YModule****module**→**serialNumber()**[**module serialNumber**]

Retourne le numéro de série du module, préprogrammé en usine.

js	function <b>get_serialNumber</b> ( )
nodejs	function <b>get_serialNumber</b> ( )
php	function <b>get_serialNumber</b> ( )
cpp	string <b>get_serialNumber</b> ( )
m	-(NSString*) serialNumber
pas	function <b>get_serialNumber</b> ( ): string
vb	function <b>get_serialNumber</b> ( ) As String
cs	string <b>get_serialNumber</b> ( )
java	String <b>get_serialNumber</b> ( )
py	def <b>get_serialNumber</b> ( )
cmd	YModule <b>target</b> <b>get_serialNumber</b>

**Retourne :**

une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine

En cas d'erreur, déclenche une exception ou retourne Y\_SERIALNUMBER\_INVALID.

**module**→**get\_upTime()****YModule****module**→**upTime()**[**module upTime**]

Retourne le nombre de millisecondes écoulées depuis la mise sous tension du module

js	function <b>get_upTime</b> ( )
nodejs	function <b>get_upTime</b> ( )
php	function <b>get_upTime</b> ( )
cpp	s64 <b>get_upTime</b> ( )
m	-(s64) upTime
pas	function <b>get_upTime</b> ( ): int64
vb	function <b>get_upTime</b> ( ) As Long
cs	long <b>get_upTime</b> ( )
java	long <b>get_upTime</b> ( )
py	def <b>get_upTime</b> ( )
cmd	YModule <b>target</b> <b>get_upTime</b>

**Retourne :**

un entier représentant le nombre de millisecondes écoulées depuis la mise sous tension du module

En cas d'erreur, déclenche une exception ou retourne Y\_UPTIME\_INVALID.



**module**→**get\_usbBandwidth()****YModule****module**→**usbBandwidth()**[**module usbBandwidth**]

Retourne le nombre d'interface USB utilisé par le module.

js	function <b>get_usbBandwidth</b> ( )
nodejs	function <b>get_usbBandwidth</b> ( )
php	function <b>get_usbBandwidth</b> ( )
cpp	Y_USBBANDWIDTH_enum <b>get_usbBandwidth</b> ( )
m	-(Y_USBBANDWIDTH_enum) usbBandwidth
pas	function <b>get_usbBandwidth</b> ( ): Integer
vb	function <b>get_usbBandwidth</b> ( ) As Integer
cs	int <b>get_usbBandwidth</b> ( )
java	int <b>get_usbBandwidth</b> ( )
py	def <b>get_usbBandwidth</b> ( )
cmd	YModule <b>target</b> <b>get_usbBandwidth</b>

**Retourne :**

soit Y\_USBBANDWIDTH\_SIMPLE, soit Y\_USBBANDWIDTH\_DOUBLE, selon le nombre d'interface USB utilisé par le module

En cas d'erreur, déclenche une exception ou retourne Y\_USBBANDWIDTH\_INVALID.

**module**→**get\_usbCurrent()****YModule****module**→**usbCurrent()**[**module usbCurrent**]

Retourne le courant consommé par le module sur le bus USB, en milliampères.

js	function <b>get_usbCurrent</b> ( )
nodejs	function <b>get_usbCurrent</b> ( )
php	function <b>get_usbCurrent</b> ( )
cpp	int <b>get_usbCurrent</b> ( )
m	-(int) usbCurrent
pas	function <b>get_usbCurrent</b> ( ): LongInt
vb	function <b>get_usbCurrent</b> ( ) As Integer
cs	int <b>get_usbCurrent</b> ( )
java	int <b>get_usbCurrent</b> ( )
py	def <b>get_usbCurrent</b> ( )
cmd	YModule <b>target</b> <b>get_usbCurrent</b>

**Retourne :**

un entier représentant le courant consommé par le module sur le bus USB, en milliampères

En cas d'erreur, déclenche une exception ou retourne Y\_USBCURRENT\_INVALID.

**module**→**get\_userData()****YModule****module**→**userData()**[**module userData**]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**module**→**isOnline()**[**module isOnline**]**YModule**

Vérifie si le module est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs du module en cache sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le module est joignable, false sinon

**module→isOnline\_async()****YModule**

Vérifie si le module est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs du module en cache sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet module concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**module→load()[module load: ]****YModule**

Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
c++	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module→load\_async()****YModule**

Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet module concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**module**→**nextModule()**[**module nextModule**]**YModule**

Continue l'énumération des modules commencée à l'aide de `yFirstModule()`.

js	function <b>nextModule</b> ( )
nodejs	function <b>nextModule</b> ( )
php	function <b>nextModule</b> ( )
cpp	YModule * <b>nextModule</b> ( )
m	-(YModule*) <b>nextModule</b>
pas	function <b>nextModule</b> ( ): TYModule
vb	function <b>nextModule</b> ( ) As YModule
cs	YModule <b>nextModule</b> ( )
java	YModule <b>nextModule</b> ( )
py	def <b>nextModule</b> ( )

**Retourne :**

un pointeur sur un objet `YModule` accessible en ligne, ou `null` lorsque l'énumération est terminée.



**module→reboot()[module reboot: ]****YModule**

Agende un simple redémarrage du module dans un nombre donné de secondes.

js	function <b>reboot</b> ( <b>secBeforeReboot</b> )
nodejs	function <b>reboot</b> ( <b>secBeforeReboot</b> )
php	function <b>reboot</b> ( <b>\$secBeforeReboot</b> )
cpp	int <b>reboot</b> ( int <b>secBeforeReboot</b> )
m	-(int) <b>reboot</b> : (int) <b>secBeforeReboot</b>
pas	function <b>reboot</b> ( <b>secBeforeReboot</b> : LongInt): LongInt
vb	function <b>reboot</b> ( ) As Integer
cs	int <b>reboot</b> ( int <b>secBeforeReboot</b> )
java	int <b>reboot</b> ( int <b>secBeforeReboot</b> )
py	def <b>reboot</b> ( <b>secBeforeReboot</b> )
cmd	YModule <b>target reboot secBeforeReboot</b>

**Paramètres :**

**secBeforeReboot** nombre de secondes avant de redémarrer

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

---

**module**→**registerLogCallback()**[**module**  
**registerLogCallback:** ]**YModule**

---

todo

cpp	void <b>registerLogCallback</b> ( YModuleLogCallback <b>callback</b> )
m	-(void) <b>registerLogCallback</b> : (YModuleLogCallback) <b>callback</b>
vb	function <b>registerLogCallback</b> ( ByVal <b>callback</b> As YModuleLogCallback) As Integer
cs	int <b>registerLogCallback</b> ( LogCallback <b>callback</b> )
py	def <b>registerLogCallback</b> ( <b>callback</b> )

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## module→revertFromFlash()[module revertFromFlash]

YModule

Recharge les réglages stockés dans le mémoire non volatile du module, comme à la mise sous tension du module.

js	function <b>revertFromFlash</b> ( )
nodejs	function <b>revertFromFlash</b> ( )
php	function <b>revertFromFlash</b> ( )
c++	int <b>revertFromFlash</b> ( )
m	-(int) <b>revertFromFlash</b>
pas	function <b>revertFromFlash</b> ( ): LongInt
vb	function <b>revertFromFlash</b> ( ) As Integer
cs	int <b>revertFromFlash</b> ( )
java	int <b>revertFromFlash</b> ( )
py	def <b>revertFromFlash</b> ( )
cmd	YModule <b>target</b> <b>revertFromFlash</b>

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module→saveToFlash()[module saveToFlash]****YModule**

Sauve les réglages courants dans la mémoire non volatile du module.

js	function <b>saveToFlash</b> ( )
nodejs	function <b>saveToFlash</b> ( )
php	function <b>saveToFlash</b> ( )
cpp	int <b>saveToFlash</b> ( )
m	-(int) <b>saveToFlash</b>
pas	function <b>saveToFlash</b> ( ): LongInt
vb	function <b>saveToFlash</b> ( ) As Integer
cs	int <b>saveToFlash</b> ( )
java	int <b>saveToFlash</b> ( )
py	def <b>saveToFlash</b> ( )
cmd	YModule <b>target saveToFlash</b>

Attention le nombre total de sauvegardes possibles durant la vie du module est limité (environ 100000 cycles). N'appellez pas cette fonction dans une boucle.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**set\_beacon()****YModule****module**→**setBeacon()**[**module setBeacon:** ]

Allume ou éteint la balise de localisation du module.

js	function <b>set_beacon</b> ( <b>newval</b> )
nodejs	function <b>set_beacon</b> ( <b>newval</b> )
php	function <b>set_beacon</b> ( <b>\$newval</b> )
cpp	int <b>set_beacon</b> ( Y_BEACON_enum <b>newval</b> )
m	-(int) setBeacon : (Y_BEACON_enum) <b>newval</b>
pas	function <b>set_beacon</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_beacon</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_beacon</b> ( int <b>newval</b> )
java	int <b>set_beacon</b> ( int <b>newval</b> )
py	def <b>set_beacon</b> ( <b>newval</b> )
cmd	YModule <b>target set_beacon newval</b>

**Paramètres :**

**newval** soit Y\_BEACON\_OFF, soit Y\_BEACON\_ON

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**set\_logicalName()****YModule****module**→**setLogicalName()**[**module setLogicalName:**  
**]**

Change le nom logique du module.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YModule <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**set\_luminosity()****YModule****module**→**setLuminosity()**[**module setLuminosity:** ]

Modifie la luminosité des leds informatives du module.

js	function <b>set_luminosity</b> ( <b>newval</b> )
nodejs	function <b>set_luminosity</b> ( <b>newval</b> )
php	function <b>set_luminosity</b> ( <b>\$newval</b> )
cpp	int <b>set_luminosity</b> ( int <b>newval</b> )
m	-(int) <b>setLuminosity</b> : (int) <b>newval</b>
pas	function <b>set_luminosity</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_luminosity</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_luminosity</b> ( int <b>newval</b> )
java	int <b>set_luminosity</b> ( int <b>newval</b> )
py	def <b>set_luminosity</b> ( <b>newval</b> )
cmd	YModule <b>target set_luminosity newval</b>

Le paramètre est une valeur entre 0 et 100. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** un entier représentant la luminosité des leds informatives du module

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**set\_usbBandwidth()**  
**module**→**setUsbBandwidth()**[**module**  
**setUsbBandwidth:** ]

YModule

Modifie le nombre d'interface USB utilisé par le module.

js	function <b>set_usbBandwidth</b> ( <b>newval</b> )
nodejs	function <b>set_usbBandwidth</b> ( <b>newval</b> )
php	function <b>set_usbBandwidth</b> ( <b>\$newval</b> )
cpp	int <b>set_usbBandwidth</b> ( Y_USBBANDWIDTH_enum <b>newval</b> )
m	-(int) setUsbBandwidth : (Y_USBBANDWIDTH_enum) <b>newval</b>
pas	function <b>set_usbBandwidth</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_usbBandwidth</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_usbBandwidth</b> ( int <b>newval</b> )
java	int <b>set_usbBandwidth</b> ( int <b>newval</b> )
py	def <b>set_usbBandwidth</b> ( <b>newval</b> )
cmd	YModule <b>target</b> <b>set_usbBandwidth</b> <b>newval</b>

Vous devez redémarrer le module après avoir changé ce réglage.

#### Paramètres :

**newval** soit Y\_USBBANDWIDTH\_SIMPLE, soit Y\_USBBANDWIDTH\_DOUBLE, selon le nombre d'interface USB utilisé par le module

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**module**→**set\_userdata()****YModule****module**→**setUserData()**[**module setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## module→triggerFirmwareUpdate()[module triggerFirmwareUpdate: ]

YModule

Agende un redémarrage du module en mode spécial de reprogrammation du logiciel embarqué.

js	function <b>triggerFirmwareUpdate</b> ( <b>secBeforeReboot</b> )
nodejs	function <b>triggerFirmwareUpdate</b> ( <b>secBeforeReboot</b> )
php	function <b>triggerFirmwareUpdate</b> ( <b>\$secBeforeReboot</b> )
cpp	int <b>triggerFirmwareUpdate</b> ( int <b>secBeforeReboot</b> )
m	-(int) <b>triggerFirmwareUpdate</b> : (int) <b>secBeforeReboot</b>
pas	function <b>triggerFirmwareUpdate</b> ( <b>secBeforeReboot</b> : LongInt): LongInt
vb	function <b>triggerFirmwareUpdate</b> ( ) As Integer
cs	int <b>triggerFirmwareUpdate</b> ( int <b>secBeforeReboot</b> )
java	int <b>triggerFirmwareUpdate</b> ( int <b>secBeforeReboot</b> )
py	def <b>triggerFirmwareUpdate</b> ( <b>secBeforeReboot</b> )
cmd	YModule <b>target triggerFirmwareUpdate secBeforeReboot</b>

### Paramètres :

**secBeforeReboot** nombre de secondes avant de redémarrer

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**module**→**wait\_async()****YModule**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.26. Interface de la fonction Network

Les objets YNetwork permettent de contrôler les paramètres TCP/IP des modules Yoctopuce dotés d'une interface réseau.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_network.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YNetwork = yoctolib.YNetwork;
php	require_once('yocto_network.php');
c++	#include "yocto_network.h"
m	#import "yocto_network.h"
pas	uses yocto_network;
vb	yocto_network.vb
cs	yocto_network.cs
java	import com.yoctopuce.YoctoAPI.YNetwork;
py	from yocto_network import *

### Fonction globales

#### yFindNetwork(func)

Permet de retrouver une interface réseau d'après un identifiant donné.

#### yFirstNetwork()

Commence l'énumération des interfaces réseau accessibles par la librairie.

### Méthodes des objets YNetwork

#### network→callbackLogin(username, password)

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

#### network→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format TYPE (NAME) = SERIAL . FUNCTIONID.

#### network→get\_adminPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

#### network→get\_advertisedValue()

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

#### network→get\_callbackCredentials()

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

#### network→get\_callbackEncoding()

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

#### network→get\_callbackMaxDelay()

Retourne l'attente maximale entre deux notifications par callback, en secondes.

#### network→get\_callbackMethod()

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

#### network→get\_callbackMinDelay()

Retourne l'attente minimale entre deux notifications par callback, en secondes.

#### network→get\_callbackUrl()

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

#### network→get\_discoverable()

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

#### **network→get\_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

#### **network→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

#### **network→get\_friendlyName()**

Retourne un identifiant global de l'interface réseau au format `NOM_MODULE . NOM_FONCTION`.

#### **network→get\_functionDescriptor()**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **network→get\_functionId()**

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

#### **network→get\_hardwareId()**

Retourne l'identifiant matériel unique de l'interface réseau au format `SERIAL . FUNCTIONID`.

#### **network→get\_ipAddress()**

Retourne l'adresse IP utilisée par le module Yoctopuce.

#### **network→get\_logicalName()**

Retourne le nom logique de l'interface réseau.

#### **network→get\_macAddress()**

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

#### **network→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **network→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **network→get\_poeCurrent()**

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

#### **network→get\_primaryDNS()**

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

#### **network→get\_readiness()**

Retourne l'état de fonctionnement atteint par l'interface réseau.

#### **network→get\_router()**

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

#### **network→get\_secondaryDNS()**

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

#### **network→get\_subnetMask()**

Retourne le masque de sous-réseau utilisé par le module.

#### **network→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **network→get\_userPassword()**

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

#### **network→get\_wwwWatchdogDelay()**

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

#### **network→isOnline()**

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

**network→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

**network→load(msValidity)**

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

**network→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

**network→nextNetwork()**

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

**network→ping(host)**

Ping `str_host` pour vérifier la connexion réseau.

**network→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**network→set\_adminPassword(newval)**

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

**network→set\_callbackCredentials(newval)**

Modifie le laisser-passer pour se connecter à l'adresse de callback.

**network→set\_callbackEncoding(newval)**

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

**network→set\_callbackMaxDelay(newval)**

Modifie l'attente maximale entre deux notifications par callback, en secondes.

**network→set\_callbackMethod(newval)**

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

**network→set\_callbackMinDelay(newval)**

Modifie l'attente minimale entre deux notifications par callback, en secondes.

**network→set\_callbackUrl(newval)**

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

**network→set\_discoverable(newval)**

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

**network→set\_logicalName(newval)**

Modifie le nom logique de l'interface réseau.

**network→set\_primaryDNS(newval)**

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

**network→set\_secondaryDNS(newval)**

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

**network→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**network→set\_userPassword(newval)**

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

**network→set\_wwwWatchdogDelay(newval)**

Modifie la durée de perte de connexion WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

**network→useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)**

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

**network→useStaticIP(ipAddress, subnetMaskLen, router)**

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

**network→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YNetwork.FindNetwork() yFindNetwork()yFindNetwork()

YNetwork

Permet de retrouver une interface réseau d'après un identifiant donné.

js	function <b>yFindNetwork</b> ( <b>func</b> )
nodejs	function <b>FindNetwork</b> ( <b>func</b> )
php	function <b>yFindNetwork</b> ( <b>\$func</b> )
cpp	YNetwork* <b>yFindNetwork</b> ( const string& <b>func</b> )
m	YNetwork* <b>yFindNetwork</b> ( NSString* <b>func</b> )
pas	function <b>yFindNetwork</b> ( <b>func</b> : string): TYNetwork
vb	function <b>yFindNetwork</b> ( ByVal <b>func</b> As String) As YNetwork
cs	YNetwork <b>FindNetwork</b> ( string <b>func</b> )
java	YNetwork <b>FindNetwork</b> ( String <b>func</b> )
py	def <b>FindNetwork</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YNetwork.isOnline()` pour tester si l'interface réseau est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'interface réseau sans ambiguïté

### Retourne :

un objet de classe `YNetwork` qui permet ensuite de contrôler l'interface réseau.



## YNetwork.FirstNetwork() yFirstNetwork()yFirstNetwork()

## YNetwork

Commence l'énumération des interfaces réseau accessibles par la librairie.

js	function <b>yFirstNetwork</b> ( )
nodejs	function <b>FirstNetwork</b> ( )
php	function <b>yFirstNetwork</b> ( )
cpp	YNetwork* <b>yFirstNetwork</b> ( )
m	YNetwork* <b>yFirstNetwork</b> ( )
pas	function <b>yFirstNetwork</b> ( ): TYNetwork
vb	function <b>yFirstNetwork</b> ( ) As YNetwork
cs	YNetwork <b>FirstNetwork</b> ( )
java	YNetwork <b>FirstNetwork</b> ( )
py	def <b>FirstNetwork</b> ( )

Utiliser la fonction `YNetwork.nextNetwork( )` pour itérer sur les autres interfaces réseau.

### Retourne :

un pointeur sur un objet `YNetwork`, correspondant à la première interface réseau accessible en ligne, ou `null` si il n'y a pas de interfaces réseau disponibles.

**network→callbackLogin()[network callbackLogin: ]****YNetwork**

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

js	function <b>callbackLogin</b> ( <b>username</b> , <b>password</b> )
nodejs	function <b>callbackLogin</b> ( <b>username</b> , <b>password</b> )
php	function <b>callbackLogin</b> ( <b>\$username</b> , <b>\$password</b> )
c++	int <b>callbackLogin</b> ( string <b>username</b> , string <b>password</b> )
m	-(int) <b>callbackLogin</b> : (NSString*) <b>username</b> : (NSString*) <b>password</b>
pas	function <b>callbackLogin</b> ( <b>username</b> : string, <b>password</b> : string): integer
vb	function <b>callbackLogin</b> ( ByVal <b>username</b> As String, ByVal <b>password</b> As String) As Integer
cs	int <b>callbackLogin</b> ( string <b>username</b> , string <b>password</b> )
java	int <b>callbackLogin</b> ( String <b>username</b> , String <b>password</b> )
py	def <b>callbackLogin</b> ( <b>username</b> , <b>password</b> )
cmd	YNetwork <b>target callbackLogin</b> <b>username password</b>

Le mot de passe ne sera pas stocké dans le module, mais seulement une version hashée non réversible. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**username** nom d'utilisateur pour s'identifier au callback  
**password** mot de passe pour s'identifier au callback

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→describe()[network describe]****YNetwork**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'interface réseau (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## network→get\_adminPassword() network→adminPassword()[network adminPassword]

YNetwork

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

js	function get_adminPassword( )
nodejs	function get_adminPassword( )
php	function get_adminPassword( )
cpp	string get_adminPassword( )
m	-(NSString*) adminPassword
pas	function get_adminPassword( ): string
vb	function get_adminPassword( ) As String
cs	string get_adminPassword( )
java	String get_adminPassword( )
py	def get_adminPassword( )
cmd	YNetwork target get_adminPassword

**Retourne :**

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne Y\_ADMINPASSWORD\_INVALID.

**network→get\_advertisedValue()**  
**network→advertisedValue()[network**  
**advertisedValue]**

**YNetwork**

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YNetwork <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'interface réseau (pas plus de 6 caractères).  
 En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

## **network→get\_callbackCredentials() network→callbackCredentials()[network callbackCredentials]**

YNetwork

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

js	function <b>get_callbackCredentials</b> ( )
nodejs	function <b>get_callbackCredentials</b> ( )
php	function <b>get_callbackCredentials</b> ( )
cpp	string <b>get_callbackCredentials</b> ( )
m	-(NSString*) callbackCredentials
pas	function <b>get_callbackCredentials</b> ( ): string
vb	function <b>get_callbackCredentials</b> ( ) As String
cs	string <b>get_callbackCredentials</b> ( )
java	String <b>get_callbackCredentials</b> ( )
py	def <b>get_callbackCredentials</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackCredentials</b>

**Retourne :**

une chaîne de caractères représentant une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKCREDENTIALS\_INVALID.

**network→get\_callbackEncoding()****YNetwork****network→callbackEncoding()[network  
callbackEncoding]**

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function <b>get_callbackEncoding</b> ( )
nodejs	function <b>get_callbackEncoding</b> ( )
php	function <b>get_callbackEncoding</b> ( )
cpp	Y_CALLBACKENCODING_enum <b>get_callbackEncoding</b> ( )
m	-(Y_CALLBACKENCODING_enum) callbackEncoding
pas	function <b>get_callbackEncoding</b> ( ): Integer
vb	function <b>get_callbackEncoding</b> ( ) As Integer
cs	int <b>get_callbackEncoding</b> ( )
java	int <b>get_callbackEncoding</b> ( )
py	def <b>get_callbackEncoding</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackEncoding</b>

**Retourne :**

une valeur parmi Y\_CALLBACKENCODING\_FORM, Y\_CALLBACKENCODING\_JSON, Y\_CALLBACKENCODING\_JSON\_ARRAY, Y\_CALLBACKENCODING\_CSV et Y\_CALLBACKENCODING\_YOCTO\_API représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKENCODING\_INVALID.

## network→get\_callbackMaxDelay() network→callbackMaxDelay()[network callbackMaxDelay]

Retourne l'attente maximale entre deux notifications par callback, en secondes.

js	function <b>get_callbackMaxDelay</b> ( )
nodejs	function <b>get_callbackMaxDelay</b> ( )
php	function <b>get_callbackMaxDelay</b> ( )
cpp	int <b>get_callbackMaxDelay</b> ( )
m	-(int) callbackMaxDelay
pas	function <b>get_callbackMaxDelay</b> ( ): LongInt
vb	function <b>get_callbackMaxDelay</b> ( ) As Integer
cs	int <b>get_callbackMaxDelay</b> ( )
java	int <b>get_callbackMaxDelay</b> ( )
py	def <b>get_callbackMaxDelay</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackMaxDelay</b>

### Retourne :

un entier représentant l'attente maximale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKMAXDELAY\_INVALID.



**network**→**get\_callbackMethod()****YNetwork****network**→**callbackMethod()**[**network callbackMethod**]

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

js	function <b>get_callbackMethod</b> ( )
nodejs	function <b>get_callbackMethod</b> ( )
php	function <b>get_callbackMethod</b> ( )
cpp	Y_CALLBACKMETHOD_enum <b>get_callbackMethod</b> ( )
m	-(Y_CALLBACKMETHOD_enum) callbackMethod
pas	function <b>get_callbackMethod</b> ( ): Integer
vb	function <b>get_callbackMethod</b> ( ) As Integer
cs	int <b>get_callbackMethod</b> ( )
java	int <b>get_callbackMethod</b> ( )
py	def <b>get_callbackMethod</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackMethod</b>

**Retourne :**

une valeur parmi Y\_CALLBACKMETHOD\_POST, Y\_CALLBACKMETHOD\_GET et Y\_CALLBACKMETHOD\_PUT représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKMETHOD\_INVALID.

## network→get\_callbackMinDelay() network→callbackMinDelay()[network callbackMinDelay]

Retourne l'attente minimale entre deux notifications par callback, en secondes.

js	function <b>get_callbackMinDelay</b> ( )
nodejs	function <b>get_callbackMinDelay</b> ( )
php	function <b>get_callbackMinDelay</b> ( )
cpp	int <b>get_callbackMinDelay</b> ( )
m	-(int) callbackMinDelay
pas	function <b>get_callbackMinDelay</b> ( ): LongInt
vb	function <b>get_callbackMinDelay</b> ( ) As Integer
cs	int <b>get_callbackMinDelay</b> ( )
java	int <b>get_callbackMinDelay</b> ( )
py	def <b>get_callbackMinDelay</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackMinDelay</b>

### Retourne :

un entier représentant l'attente minimale entre deux notifications par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKMINDELAY\_INVALID.

**network→get\_callbackUrl()****YNetwork****network→callbackUrl()[network callbackUrl]**

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

js	function <b>get_callbackUrl</b> ( )
nodejs	function <b>get_callbackUrl</b> ( )
php	function <b>get_callbackUrl</b> ( )
cpp	string <b>get_callbackUrl</b> ( )
m	-(NSString*) callbackUrl
pas	function <b>get_callbackUrl</b> ( ): string
vb	function <b>get_callbackUrl</b> ( ) As String
cs	string <b>get_callbackUrl</b> ( )
java	String <b>get_callbackUrl</b> ( )
py	def <b>get_callbackUrl</b> ( )
cmd	YNetwork <b>target</b> <b>get_callbackUrl</b>

**Retourne :**

une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

En cas d'erreur, déclenche une exception ou retourne Y\_CALLBACKURL\_INVALID.

**network**→**get\_discoverable()****network**→**discoverable()**[**network discoverable**]

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

js	function <b>get_discoverable</b> ( )
nodejs	function <b>get_discoverable</b> ( )
php	function <b>get_discoverable</b> ( )
cpp	Y_DISCOVERABLE_enum <b>get_discoverable</b> ( )
m	-(Y_DISCOVERABLE_enum) discoverable
pas	function <b>get_discoverable</b> ( ): Integer
vb	function <b>get_discoverable</b> ( ) As Integer
cs	int <b>get_discoverable</b> ( )
java	int <b>get_discoverable</b> ( )
py	def <b>get_discoverable</b> ( )
cmd	YNetwork <b>target</b> <b>get_discoverable</b>

**Retourne :**

soit Y\_DISCOVERABLE\_FALSE, soit Y\_DISCOVERABLE\_TRUE, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour)

En cas d'erreur, déclenche une exception ou retourne Y\_DISCOVERABLE\_INVALID.

**network→get\_errorMessage()****YNetwork****network→errorMessage()[network errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

**network**→**get\_errorType()****YNetwork****network**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

**network→get\_friendlyName()****YNetwork****network→friendlyName()[network friendlyName]**

Retourne un identifiant global de l'interface réseau au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'interface réseau (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'interface réseau en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **network→get\_functionDescriptor() network→functionDescriptor()[network functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID



**network**→**get\_functionId()****YNetwork****network**→**functionId()**[network functionId]

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'interface réseau (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**network**→**get\_hardwareId()****network**→**hardwareId()**[network hardwareId]

Retourne l'identifiant matériel unique de l'interface réseau au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'interface réseau (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**network→get\_ipAddress()****YNetwork****network→ipAddress()[network ipAddress]**

Retourne l'adresse IP utilisée par le module Yoctopuce.

js	function <b>get_ipAddress</b> ( )
nodejs	function <b>get_ipAddress</b> ( )
php	function <b>get_ipAddress</b> ( )
cpp	string <b>get_ipAddress</b> ( )
m	-(NSString*) ipAddress
pas	function <b>get_ipAddress</b> ( ): string
vb	function <b>get_ipAddress</b> ( ) As String
cs	string <b>get_ipAddress</b> ( )
java	String <b>get_ipAddress</b> ( )
py	def <b>get_ipAddress</b> ( )
cmd	YNetwork <b>target</b> <b>get_ipAddress</b>

Il peut s'agir d'une adresse configurée statiquement, ou d'une adresse reçue par un serveur DHCP.

**Retourne :**

une chaîne de caractères représentant l'adresse IP utilisée par le module Yoctopuce

En cas d'erreur, déclenche une exception ou retourne Y\_IPADDRESS\_INVALID.

**network**→**get\_logicalName()****YNetwork****network**→**logicalName()**[network logicalName]

Retourne le nom logique de l'interface réseau.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YNetwork <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'interface réseau. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**network→get\_macAddress()****YNetwork****network→macAddress()[network macAddress]**

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

js	function <b>get_macAddress</b> ( )
nodejs	function <b>get_macAddress</b> ( )
php	function <b>get_macAddress</b> ( )
cpp	string <b>get_macAddress</b> ( )
m	-(NSString*) <b>macAddress</b>
pas	function <b>get_macAddress</b> ( ): string
vb	function <b>get_macAddress</b> ( ) As String
cs	string <b>get_macAddress</b> ( )
java	String <b>get_macAddress</b> ( )
py	def <b>get_macAddress</b> ( )
cmd	YNetwork <b>target get_macAddress</b>

L'adresse MAC est aussi présente sur un autocollant sur le module, représentée en chiffres et en code-barres.

**Retourne :**

une chaîne de caractères représentant l'adresse MAC de l'interface réseau, unique pour chaque module

En cas d'erreur, déclenche une exception ou retourne Y\_MACADDRESS\_INVALID.

**network→get\_module()****network→module()[network module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

---

**network**→**get\_module\_async()****YNetwork****network**→**module\_async()**

---

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**network**→**get\_poeCurrent()****YNetwork****network**→**poeCurrent()**[**network poeCurrent**]

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

js	function <b>get_poeCurrent</b> ( )
nodejs	function <b>get_poeCurrent</b> ( )
php	function <b>get_poeCurrent</b> ( )
cpp	int <b>get_poeCurrent</b> ( )
m	-(int) poeCurrent
pas	function <b>get_poeCurrent</b> ( ): LongInt
vb	function <b>get_poeCurrent</b> ( ) As Integer
cs	int <b>get_poeCurrent</b> ( )
java	int <b>get_poeCurrent</b> ( )
py	def <b>get_poeCurrent</b> ( )
cmd	YNetwork <b>target</b> <b>get_poeCurrent</b>

La consommation est mesurée après conversion en 5 Volt, et ne doit jamais dépasser 1800 mA.

**Retourne :**

un entier représentant le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères

En cas d'erreur, déclenche une exception ou retourne Y\_POECURRENT\_INVALID.



**network→get\_primaryDNS()****YNetwork****network→primaryDNS()[network primaryDNS]**

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function <b>get_primaryDNS</b> ( )
nodejs	function <b>get_primaryDNS</b> ( )
php	function <b>get_primaryDNS</b> ( )
cpp	string <b>get_primaryDNS</b> ( )
m	-(NSString*) primaryDNS
pas	function <b>get_primaryDNS</b> ( ): string
vb	function <b>get_primaryDNS</b> ( ) As String
cs	string <b>get_primaryDNS</b> ( )
java	String <b>get_primaryDNS</b> ( )
py	def <b>get_primaryDNS</b> ( )
cmd	YNetwork <b>target</b> <b>get_primaryDNS</b>

**Retourne :**

une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne Y\_PRIMARYDNS\_INVALID.

**network**→**get\_readiness()****network**→**readiness()**[network readiness]

Retourne l'état de fonctionnement atteint par l'interface réseau.

js	function <b>get_readiness</b> ( )
nodejs	function <b>get_readiness</b> ( )
php	function <b>get_readiness</b> ( )
cpp	Y_READINESS_enum <b>get_readiness</b> ( )
m	-(Y_READINESS_enum) readiness
pas	function <b>get_readiness</b> ( ): Integer
vb	function <b>get_readiness</b> ( ) As Integer
cs	int <b>get_readiness</b> ( )
java	int <b>get_readiness</b> ( )
py	def <b>get_readiness</b> ( )
cmd	YNetwork <b>target</b> <b>get_readiness</b>

Le niveau zéro (DOWN\_0) signifie qu'aucun support réseau matériel n'a été détecté. Soit il n'y a pas de signal sur le câble réseau, soit le point d'accès sans fil choisi n'est pas détecté. Le niveau 1 (LIVE\_1) est atteint lorsque le réseau est détecté, mais n'est pas encore connecté. Pour un réseau sans fil, cela confirme l'existence du SSID configuré. Le niveau 2 (LINK\_2) est atteint lorsque le support matériel du réseau est fonctionnel. Pour une connection réseau filaire, le niveau 2 signifie que le câble est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurités configurés sont corrects. Pour une connection sans fil en mode ad-hoc, cela signifie qu'il y a au moins un partenaire sur le réseau ad-hoc. Le niveau 3 (DHCP\_3) est atteint lorsque qu'une adresse IP a été obtenue par DHCP. Le niveau 4 (DNS\_4) est atteint lorsqu'un serveur DNS est joignable par le réseau. Le niveau 5 (WWW\_5) est atteint lorsque la connectivité globale à internet est avérée par l'obtention de l'heure courante sur un serveur NTP.

**Retourne :**

une valeur parmi Y\_READINESS\_DOWN, Y\_READINESS\_EXISTS, Y\_READINESS\_LINKED, Y\_READINESS\_LAN\_OK et Y\_READINESS\_WWW\_OK représentant l'état de fonctionnement atteint par l'interface réseau

En cas d'erreur, déclenche une exception ou retourne Y\_READINESS\_INVALID.

**network→get\_router()****YNetwork****network→router()[network router]**

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

js	function <b>get_router</b> ( )
nodejs	function <b>get_router</b> ( )
php	function <b>get_router</b> ( )
cpp	string <b>get_router</b> ( )
m	-(NSString*) router
pas	function <b>get_router</b> ( ): string
vb	function <b>get_router</b> ( ) As String
cs	string <b>get_router</b> ( )
java	String <b>get_router</b> ( )
py	def <b>get_router</b> ( )
cmd	YNetwork <b>target get_router</b>

**Retourne :**

une chaîne de caractères représentant l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*)

En cas d'erreur, déclenche une exception ou retourne Y\_ROUTER\_INVALID.

**network**→**get\_secondaryDNS()****YNetwork****network**→**secondaryDNS()[network secondaryDNS]**

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

js	function <b>get_secondaryDNS</b> ( )
nodejs	function <b>get_secondaryDNS</b> ( )
php	function <b>get_secondaryDNS</b> ( )
cpp	string <b>get_secondaryDNS</b> ( )
m	-(NSString*) secondaryDNS
pas	function <b>get_secondaryDNS</b> ( ): string
vb	function <b>get_secondaryDNS</b> ( ) As String
cs	string <b>get_secondaryDNS</b> ( )
java	String <b>get_secondaryDNS</b> ( )
py	def <b>get_secondaryDNS</b> ( )
cmd	YNetwork <b>target</b> <b>get_secondaryDNS</b>

**Retourne :**

une chaîne de caractères représentant l'adresse IP du serveur de noms secondaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne Y\_SECONDARYDNS\_INVALID.

**network→get\_subnetMask()****YNetwork****network→subnetMask()[network subnetMask]**

Retourne le masque de sous-réseau utilisé par le module.

js	function <b>get_subnetMask</b> ( )
nodejs	function <b>get_subnetMask</b> ( )
php	function <b>get_subnetMask</b> ( )
cpp	string <b>get_subnetMask</b> ( )
m	-(NSString*) subnetMask
pas	function <b>get_subnetMask</b> ( ): string
vb	function <b>get_subnetMask</b> ( ) As String
cs	string <b>get_subnetMask</b> ( )
java	String <b>get_subnetMask</b> ( )
py	def <b>get_subnetMask</b> ( )
cmd	YNetwork <b>target</b> <b>get_subnetMask</b>

**Retourne :**

une chaîne de caractères représentant le masque de sous-réseau utilisé par le module

En cas d'erreur, déclenche une exception ou retourne Y\_SUBNETMASK\_INVALID.

**network**→**get\_userdata()****YNetwork****network**→**userData()**[network userData]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userdata`.

<code>js</code>	<code>function <b>get_userdata</b>( )</code>
<code>nodejs</code>	<code>function <b>get_userdata</b>( )</code>
<code>php</code>	<code>function <b>get_userdata</b>( )</code>
<code>cpp</code>	<code>void * <b>get_userdata</b>( )</code>
<code>m</code>	<code>-(void*) userData</code>
<code>pas</code>	<code>function <b>get_userdata</b>( ): Tobject</code>
<code>vb</code>	<code>function <b>get_userdata</b>( ) As Object</code>
<code>cs</code>	<code>object <b>get_userdata</b>( )</code>
<code>java</code>	<code>Object <b>get_userdata</b>( )</code>
<code>py</code>	<code>def <b>get_userdata</b>( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**network→get\_userPassword()****YNetwork****network→userPassword()[network userPassword]**

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

js	function <b>get_userPassword</b> ( )
nodejs	function <b>get_userPassword</b> ( )
php	function <b>get_userPassword</b> ( )
cpp	string <b>get_userPassword</b> ( )
m	-(NSString*) userPassword
pas	function <b>get_userPassword</b> ( ): string
vb	function <b>get_userPassword</b> ( ) As String
cs	string <b>get_userPassword</b> ( )
java	String <b>get_userPassword</b> ( )
py	def <b>get_userPassword</b> ( )
cmd	YNetwork <b>target</b> <b>get_userPassword</b>

**Retourne :**

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne Y\_USERPASSWORD\_INVALID.

**network→get\_wwwWatchdogDelay()**  
**network→wwwWatchdogDelay()[network**  
**wwwWatchdogDelay]**

**YNetwork**

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

js	function <b>get_wwwWatchdogDelay</b> ( )
nodejs	function <b>get_wwwWatchdogDelay</b> ( )
php	function <b>get_wwwWatchdogDelay</b> ( )
c++	int <b>get_wwwWatchdogDelay</b> ( )
m	-(int) wwwWatchdogDelay
pas	function <b>get_wwwWatchdogDelay</b> ( ): LongInt
vb	function <b>get_wwwWatchdogDelay</b> ( ) As Integer
cs	int <b>get_wwwWatchdogDelay</b> ( )
java	int <b>get_wwwWatchdogDelay</b> ( )
py	def <b>get_wwwWatchdogDelay</b> ( )
cmd	YNetwork <b>target</b> <b>get_wwwWatchdogDelay</b>

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW.

**Retourne :**

un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet

En cas d'erreur, déclenche une exception ou retourne Y\_WWWWATCHDOGDELAY\_INVALID.



**network→isOnline()[network isOnline]****YNetwork**

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'interface réseau est joignable, `false` sinon

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**network→load()[network load: ]****YNetwork**

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**network→nextNetwork()[network nextNetwork]****YNetwork**

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`.

js	function <b>nextNetwork</b> ( )
nodejs	function <b>nextNetwork</b> ( )
php	function <b>nextNetwork</b> ( )
cpp	YNetwork * <b>nextNetwork</b> ( )
m	-(YNetwork*) <b>nextNetwork</b>
pas	function <b>nextNetwork</b> ( ): TYNetwork
vb	function <b>nextNetwork</b> ( ) As YNetwork
cs	YNetwork <b>nextNetwork</b> ( )
java	YNetwork <b>nextNetwork</b> ( )
py	def <b>nextNetwork</b> ( )

**Retourne :**

un pointeur sur un objet `YNetwork` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**network→ping())[network ping: ]****YNetwork**

Ping str\_host pour vérifier la connexion réseau.

js	function <b>ping</b> ( <b>host</b> )
nodejs	function <b>ping</b> ( <b>host</b> )
php	function <b>ping</b> ( <b>\$host</b> )
cpp	string <b>ping</b> ( string <b>host</b> )
m	-(NSString*) <b>ping</b> : (NSString*) <b>host</b>
pas	function <b>ping</b> ( <b>host</b> : string): string
vb	function <b>ping</b> ( ) As String
cs	string <b>ping</b> ( string <b>host</b> )
java	String <b>ping</b> ( String <b>host</b> )
py	def <b>ping</b> ( <b>host</b> )
cmd	YNetwork <b>target ping host</b>

Envoie quatre requêtes ICMP ECHO\_RESPONDER à la cible str\_host depuis le module. Cette méthode retourne une chaîne de caractères avec le résultat des 4 requêtes ICMP ECHO\_RESPONSE.

**Paramètres :**

**host** le nom d'hôte ou l'adresse IP de la cible

**Retourne :**

une chaîne de caractères contenant le résultat du ping.

## network→registerValueCallback()[network registerValueCallback: ]

## YNetwork

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YNetworkValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YNetworkValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYNetworkValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**network→set\_adminPassword()****YNetwork****network→setAdminPassword()[network  
setAdminPassword: ]**

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

js	function <b>set_adminPassword</b> ( <b>newval</b> )
nodejs	function <b>set_adminPassword</b> ( <b>newval</b> )
php	function <b>set_adminPassword</b> ( <b>\$newval</b> )
cpp	int <b>set_adminPassword</b> ( const string& <b>newval</b> )
m	-(int) setAdminPassword : (NSString*) <b>newval</b>
pas	function <b>set_adminPassword</b> ( <b>newval</b> : string): integer
vb	function <b>set_adminPassword</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_adminPassword</b> ( string <b>newval</b> )
java	int <b>set_adminPassword</b> ( String <b>newval</b> )
py	def <b>set_adminPassword</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_adminPassword</b> <b>newval</b>

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## network→set\_callbackCredentials() network→setCallbackCredentials()[network setCallbackCredentials: ]

YNetwork

Modifie le laisser-passer pour se connecter à l'adresse de callback.

js	function <b>set_callbackCredentials</b> ( <b>newval</b> )
nodejs	function <b>set_callbackCredentials</b> ( <b>newval</b> )
php	function <b>set_callbackCredentials</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackCredentials</b> ( const string& <b>newval</b> )
m	-(int) setCallbackCredentials : (NSString*) <b>newval</b>
pas	function <b>set_callbackCredentials</b> ( <b>newval</b> : string): integer
vb	function <b>set_callbackCredentials</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_callbackCredentials</b> ( string <b>newval</b> )
java	int <b>set_callbackCredentials</b> ( String <b>newval</b> )
py	def <b>set_callbackCredentials</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_callbackCredentials</b> <b>newval</b>

Le laisser-passer doit être fourni tel que retourné par la fonction `get_callbackCredentials`, sous la forme `username:hash`. La valeur du hash dépend de la méthode d'autorisation implémentée par le callback. Pour une autorisation de type Basic, le hash est le MD5 de la chaîne `username:password`. Pour une autorisation de type Digest, le hash est le MD5 de la chaîne `username:realm:password`. Pour une utilisation simplifiée, utilisez la fonction `callbackLogin`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le laisser-passer pour se connecter à l'adresse de callback

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set\_callbackEncoding() network→setCallbackEncoding()[network setCallbackEncoding: ]

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function <b>set_callbackEncoding</b> ( <b>newval</b> )
nodejs	function <b>set_callbackEncoding</b> ( <b>newval</b> )
php	function <b>set_callbackEncoding</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackEncoding</b> ( Y_CALLBACKENCODING_enum <b>newval</b> )
m	-(int) setCallbackEncoding : (Y_CALLBACKENCODING_enum) <b>newval</b>
pas	function <b>set_callbackEncoding</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_callbackEncoding</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_callbackEncoding</b> ( int <b>newval</b> )
java	int <b>set_callbackEncoding</b> ( int <b>newval</b> )
py	def <b>set_callbackEncoding</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_callbackEncoding</b> <b>newval</b>

### Paramètres :

**newval** une valeur parmi Y\_CALLBACKENCODING\_FORM, Y\_CALLBACKENCODING\_JSON, Y\_CALLBACKENCODING\_JSON\_ARRAY, Y\_CALLBACKENCODING\_CSV et Y\_CALLBACKENCODING\_YOCTO\_API représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→set\_callbackMaxDelay()**  
**network→setCallbackMaxDelay()[network**  
**setCallbackMaxDelay: ]**

**YNetwork**

Modifie l'attente maximale entre deux notifications par callback, en secondes.

js	function <b>set_callbackMaxDelay</b> ( <b>newval</b> )
nodejs	function <b>set_callbackMaxDelay</b> ( <b>newval</b> )
php	function <b>set_callbackMaxDelay</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackMaxDelay</b> ( int <b>newval</b> )
m	-(int) setCallbackMaxDelay : (int) <b>newval</b>
pas	function <b>set_callbackMaxDelay</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_callbackMaxDelay</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_callbackMaxDelay</b> ( int <b>newval</b> )
java	int <b>set_callbackMaxDelay</b> ( int <b>newval</b> )
py	def <b>set_callbackMaxDelay</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_callbackMaxDelay</b> <b>newval</b>

#### Paramètres :

**newval** un entier représentant l'attente maximale entre deux notifications par callback, en secondes

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network**→**set\_callbackMethod()****YNetwork****network**→**setCallbackMethod()**[**network**  
**setCallbackMethod:** ]

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

js	function <b>set_callbackMethod</b> ( <b>newval</b> )
nodejs	function <b>set_callbackMethod</b> ( <b>newval</b> )
php	function <b>set_callbackMethod</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackMethod</b> ( Y_CALLBACKMETHOD_enum <b>newval</b> )
m	-(int) setCallbackMethod : (Y_CALLBACKMETHOD_enum) <b>newval</b>
pas	function <b>set_callbackMethod</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_callbackMethod</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_callbackMethod</b> ( int <b>newval</b> )
java	int <b>set_callbackMethod</b> ( int <b>newval</b> )
py	def <b>set_callbackMethod</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_callbackMethod</b> <b>newval</b>

**Paramètres :**

**newval** une valeur parmi Y\_CALLBACKMETHOD\_POST, Y\_CALLBACKMETHOD\_GET et Y\_CALLBACKMETHOD\_PUT représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set\_callbackMinDelay() network→setCallbackMinDelay()[network setCallbackMinDelay: ]

YNetwork

Modifie l'attente minimale entre deux notifications par callback, en secondes.

js	function <b>set_callbackMinDelay</b> ( <b>newval</b> )
nodejs	function <b>set_callbackMinDelay</b> ( <b>newval</b> )
php	function <b>set_callbackMinDelay</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackMinDelay</b> ( int <b>newval</b> )
m	-(int) setCallbackMinDelay : (int) <b>newval</b>
pas	function <b>set_callbackMinDelay</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_callbackMinDelay</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_callbackMinDelay</b> ( int <b>newval</b> )
java	int <b>set_callbackMinDelay</b> ( int <b>newval</b> )
py	def <b>set_callbackMinDelay</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_callbackMinDelay</b> <b>newval</b>

### Paramètres :

**newval** un entier représentant l'attente minimale entre deux notifications par callback, en secondes

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network**→**set\_callbackUrl()****YNetwork****network**→**setCallbackUrl()**[**network setCallbackUrl:** ]

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

js	function <b>set_callbackUrl</b> ( <b>newval</b> )
nodejs	function <b>set_callbackUrl</b> ( <b>newval</b> )
php	function <b>set_callbackUrl</b> ( <b>\$newval</b> )
cpp	int <b>set_callbackUrl</b> ( const string& <b>newval</b> )
m	-(int) setCallbackUrl : (NSString*) <b>newval</b>
pas	function <b>set_callbackUrl</b> ( <b>newval</b> : string): integer
vb	function <b>set_callbackUrl</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_callbackUrl</b> ( string <b>newval</b> )
java	int <b>set_callbackUrl</b> ( String <b>newval</b> )
py	def <b>set_callbackUrl</b> ( <b>newval</b> )
cmd	YNetwork <b>target set_callbackUrl newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→set\_discoverable()****YNetwork****network→setDiscoverable()[network  
setDiscoverable: ]**

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

js	function <b>set_discoverable</b> ( <b>newval</b> )
nodejs	function <b>set_discoverable</b> ( <b>newval</b> )
php	function <b>set_discoverable</b> ( <b>\$newval</b> )
c++	int <b>set_discoverable</b> ( Y_DISCOVERABLE_enum <b>newval</b> )
m	-(int) setDiscoverable : (Y_DISCOVERABLE_enum) <b>newval</b>
pas	function <b>set_discoverable</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_discoverable</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_discoverable</b> ( int <b>newval</b> )
java	int <b>set_discoverable</b> ( int <b>newval</b> )
py	def <b>set_discoverable</b> ( <b>newval</b> )
cmd	YNetwork <b>target set_discoverable newval</b>

**Paramètres :**

**newval** soit Y\_DISCOVERABLE\_FALSE, soit Y\_DISCOVERABLE\_TRUE, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## **network→set\_logicalName()** **network→setLogicalName()[network** **setLogicalName: ]**

Modifie le nom logique de l'interface réseau.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### **Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'interface réseau.

### **Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**network→set\_primaryDNS()****YNetwork****network→setPrimaryDNS()[network setPrimaryDNS:  
]**

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function <b>set_primaryDNS</b> ( <b>newval</b> )
nodejs	function <b>set_primaryDNS</b> ( <b>newval</b> )
php	function <b>set_primaryDNS</b> ( <b>\$newval</b> )
cpp	int <b>set_primaryDNS</b> ( const string& <b>newval</b> )
m	-(int) setPrimaryDNS : (NSString*) <b>newval</b>
pas	function <b>set_primaryDNS</b> ( <b>newval</b> : string): integer
vb	function <b>set_primaryDNS</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_primaryDNS</b> ( string <b>newval</b> )
java	int <b>set_primaryDNS</b> ( String <b>newval</b> )
py	def <b>set_primaryDNS</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_primaryDNS</b> <b>newval</b>

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**newval** une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## network→set\_secondaryDNS() network→setSecondaryDNS()[network setSecondaryDNS: ]

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

js	function <b>set_secondaryDNS</b> ( <b>newval</b> )
nodejs	function <b>set_secondaryDNS</b> ( <b>newval</b> )
php	function <b>set_secondaryDNS</b> ( <b>\$newval</b> )
c++	int <b>set_secondaryDNS</b> ( const string& <b>newval</b> )
m	-(int) setSecondaryDNS : (NSString*) <b>newval</b>
pas	function <b>set_secondaryDNS</b> ( <b>newval</b> : string): integer
vb	function <b>set_secondaryDNS</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_secondaryDNS</b> ( string <b>newval</b> )
java	int <b>set_secondaryDNS</b> ( String <b>newval</b> )
py	def <b>set_secondaryDNS</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_secondaryDNS</b> <b>newval</b>

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

### Paramètres :

**newval** une chaîne de caractères représentant l'adresse IP du serveur de nom secondaire que le module doit utiliser

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→set\_userdata()****YNetwork****network→setUserData()[network setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## network→set\_userPassword() network→setUserPassword()[network setUserPassword: ]

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

js	function <b>set_userPassword</b> ( <b>newval</b> )
nodejs	function <b>set_userPassword</b> ( <b>newval</b> )
php	function <b>set_userPassword</b> ( <b>\$newval</b> )
cpp	int <b>set_userPassword</b> ( const string& <b>newval</b> )
m	-(int) setUserPassword : (NSString*) <b>newval</b>
pas	function <b>set_userPassword</b> ( <b>newval</b> : string): integer
vb	function <b>set_userPassword</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_userPassword</b> ( string <b>newval</b> )
java	int <b>set_userPassword</b> ( String <b>newval</b> )
py	def <b>set_userPassword</b> ( <b>newval</b> )
cmd	YNetwork <b>target</b> <b>set_userPassword</b> <b>newval</b>

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network**→**set\_wwwWatchdogDelay()****YNetwork****network**→**setWwwWatchdogDelay()[network  
setWwwWatchdogDelay: ]**

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

js	function <b>set_wwwWatchdogDelay</b> ( <b>newval</b> )
nodejs	function <b>set_wwwWatchdogDelay</b> ( <b>newval</b> )
php	function <b>set_wwwWatchdogDelay</b> ( <b>\$newval</b> )
c++	int <b>set_wwwWatchdogDelay</b> ( int <b>newval</b> )
m	-(int) <b>setWwwWatchdogDelay</b> : (int) <b>newval</b>
pas	function <b>set_wwwWatchdogDelay</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_wwwWatchdogDelay</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_wwwWatchdogDelay</b> ( int <b>newval</b> )
java	int <b>set_wwwWatchdogDelay</b> ( int <b>newval</b> )
py	def <b>set_wwwWatchdogDelay</b> ( <b>newval</b> )
cmd	<b>YNetwork target set_wwwWatchdogDelay newval</b>

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW. La plus petite durée non-nulle utilisable est 90 secondes.

**Paramètres :**

**newval** un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→useDHCP()[network useDHCP: ]****YNetwork**

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

```

js function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
nodejs function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
php function useDHCP( $fallbackIpAddr, $fallbackSubnetMaskLen, $fallbackRouter)
cpp int useDHCP( string fallbackIpAddr,
                int fallbackSubnetMaskLen,
                string fallbackRouter)

m -(int) useDHCP : (NSString*) fallbackIpAddr
    : (int) fallbackSubnetMaskLen
    : (NSString*) fallbackRouter

pas function useDHCP( fallbackIpAddr: string,
                    fallbackSubnetMaskLen: LongInt,
                    fallbackRouter: string): integer

vb function useDHCP( ByVal fallbackIpAddr As String,
                    ByVal fallbackSubnetMaskLen As Integer,
                    ByVal fallbackRouter As String) As Integer

cs int useDHCP( string fallbackIpAddr,
                int fallbackSubnetMaskLen,
                string fallbackRouter)

java int useDHCP( String fallbackIpAddr,
                 int fallbackSubnetMaskLen,
                 String fallbackRouter)

py def useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
cmd YNetwork target useDHCP fallbackIpAddr fallbackSubnetMaskLen fallbackRouter

```

En attendant qu'une adresse soit reçue (et indéfiniment si aucun serveur DHCP ne répond), le module utilisera les paramètres IP spécifiés à cette fonction. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**fallbackIpAddr** adresse IP à utiliser si aucun serveur DHCP ne répond

**fallbackSubnetMaskLen** longueur du masque de sous-réseau à utiliser si aucun serveur DHCP ne répond. Par exemple, la valeur 24 représente 255.255.255.0.

**fallbackRouter** adresse de la passerelle à utiliser si aucun serveur DHCP ne répond

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→useStaticIP()[network useStaticIP: ]****YNetwork**

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

```

js function useStaticIP( ipAddress, subnetMaskLen, router)
nodejs function useStaticIP( ipAddress, subnetMaskLen, router)
php function useStaticIP( $ipAddress, $subnetMaskLen, $router)
cpp int useStaticIP( string ipAddress,
                    int subnetMaskLen,
                    string router)

m -(int) useStaticIP : (NSString*) ipAddress
    : (int) subnetMaskLen
    : (NSString*) router

pas function useStaticIP( ipAddress: string,
                        subnetMaskLen: LongInt,
                        router: string): integer

vb function useStaticIP( ByVal ipAddress As String,
                        ByVal subnetMaskLen As Integer,
                        ByVal router As String) As Integer

cs int useStaticIP( string ipAddress,
                    int subnetMaskLen,
                    string router)

java int useStaticIP( String ipAddress,
                     int subnetMaskLen,
                     String router)

py def useStaticIP( ipAddress, subnetMaskLen, router)
cmd YNetwork target useStaticIP ipAddress subnetMaskLen router

```

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**ipAddress** adresse IP à utiliser par le module

**subnetMaskLen** longueur du masque de sous-réseau à utiliser. Par exemple, la valeur 24 représente 255.255.255.0.

**router** adresse IP de la passerelle à utiliser ("default gateway")

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**network→wait\_async()****YNetwork**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :



## 3.27. contrôle d'OS

L'objet `OsControl` permet de contrôler le système d'exploitation sur lequel tourne un VirtualHub. `OsControl` n'est disponible que dans le VirtualHub software. Attention, cette fonctionnalité doit être explicitement activée au lancement du VirtualHub, avec l'option `-o`.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_oscontrol.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib'); var YOsControl = yoctolib.YOsControl;</code>
php	<code>require_once('yocto_oscontrol.php');</code>
c++	<code>#include "yocto_oscontrol.h"</code>
m	<code>#import "yocto_oscontrol.h"</code>
pas	<code>uses yocto_oscontrol;</code>
vb	<code>yocto_oscontrol.vb</code>
cs	<code>yocto_oscontrol.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YOsControl;</code>
py	<code>from yocto_oscontrol import *</code>

### Fonction globales

#### `yFindOsControl(func)`

Permet de retrouver un contrôle d'OS d'après un identifiant donné.

#### `yFirstOsControl()`

Commence l'énumération des contrôle d'OS accessibles par la librairie.

### Méthodes des objets `YOsControl`

#### `oscontrol→describe()`

Retourne un court texte décrivant de manière non-ambigüe l'instance du contrôle d'OS au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### `oscontrol→get_advertisedValue()`

Retourne la valeur courante du contrôle d'OS (pas plus de 6 caractères).

#### `oscontrol→get_errorMessage()`

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'OS.

#### `oscontrol→get_errorType()`

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'OS.

#### `oscontrol→get_friendlyName()`

Retourne un identifiant global du contrôle d'OS au format `NOM_MODULE . NOM_FONCTION`.

#### `oscontrol→get_functionDescriptor()`

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### `oscontrol→get_functionId()`

Retourne l'identifiant matériel du contrôle d'OS, sans référence au module.

#### `oscontrol→get_hardwareId()`

Retourne l'identifiant matériel unique du contrôle d'OS au format `SERIAL . FUNCTIONID`.

#### `oscontrol→get_logicalName()`

Retourne le nom logique du contrôle d'OS.

#### `oscontrol→get_module()`

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### `oscontrol→get_module_async(callback, context)`

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`oscontrol→get_shutdownCountdown()`**

Retourne le nombre de secondes restantes avant un arrêt de l'OS, ou zéro si aucun arrêt n'a été agendé.

**`oscontrol→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`oscontrol→isOnline()`**

Vérifie si le module hébergeant le contrôle d'OS est joignable, sans déclencher d'erreur.

**`oscontrol→isOnline_async(callback, context)`**

Vérifie si le module hébergeant le contrôle d'OS est joignable, sans déclencher d'erreur.

**`oscontrol→load(msValidity)`**

Met en cache les valeurs courantes du contrôle d'OS, avec une durée de validité spécifiée.

**`oscontrol→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes du contrôle d'OS, avec une durée de validité spécifiée.

**`oscontrol→nextOsControl()`**

Continue l'énumération des contrôle d'OS commencée à l'aide de `yFirstOsControl()`.

**`oscontrol→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`oscontrol→set_logicalName(newval)`**

Modifie le nom logique du contrôle d'OS.

**`oscontrol→set_userData(data)`**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**`oscontrol→shutdown(secBeforeShutDown)`**

Agende un arrêt de l'OS dans un nombre donné de secondes.

**`oscontrol→wait_async(callback, context)`**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YOsControl.FindOsControl() yFindOsControl()yFindOsControl()

## YOsControl

Permet de retrouver un contrôle d'OS d'après un identifiant donné.

js	function <b>yFindOsControl</b> ( <b>func</b> )
nodejs	function <b>FindOsControl</b> ( <b>func</b> )
php	function <b>yFindOsControl</b> ( <b>\$func</b> )
cpp	YOsControl* <b>yFindOsControl</b> ( const string& <b>func</b> )
m	YOsControl* <b>yFindOsControl</b> ( NSString* <b>func</b> )
pas	function <b>yFindOsControl</b> ( <b>func</b> : string): TYOsControl
vb	function <b>yFindOsControl</b> ( ByVal <b>func</b> As String) As YOsControl
cs	YOsControl <b>FindOsControl</b> ( string <b>func</b> )
java	YOsControl <b>FindOsControl</b> ( String <b>func</b> )
py	def <b>FindOsControl</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le contrôle d'OS soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YOsControl.isOnline()` pour tester si le contrôle d'OS est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le contrôle d'OS sans ambiguïté

### Retourne :

un objet de classe `YOsControl` qui permet ensuite de contrôler le contrôle d'OS.

## YOsControl.FirstOsControl() yFirstOsControl()yFirstOsControl()

YOsControl

Commence l'énumération des contrôle d'OS accessibles par la librairie.

js	function <b>yFirstOsControl</b> ( )
nodejs	function <b>FirstOsControl</b> ( )
php	function <b>yFirstOsControl</b> ( )
cpp	YOsControl* <b>yFirstOsControl</b> ( )
m	YOsControl* <b>yFirstOsControl</b> ( )
pas	function <b>yFirstOsControl</b> ( ): TYOsControl
vb	function <b>yFirstOsControl</b> ( ) As YOsControl
cs	YOsControl <b>FirstOsControl</b> ( )
java	YOsControl <b>FirstOsControl</b> ( )
py	def <b>FirstOsControl</b> ( )

Utiliser la fonction `YOsControl.nextOsControl( )` pour itérer sur les autres contrôle d'OS.

### Retourne :

un pointeur sur un objet `YOsControl`, correspondant à le premier contrôle d'OS accessible en ligne, ou `null` si il n'y a pas de contrôle d'OS disponibles.

**oscontrol→describe()[oscontrol describe]****YOsControl**

Retourne un court texte décrivant de manière non-ambigüe l'instance du contrôle d'OS au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le contrôle d'OS (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## oscontrol→get\_advertisedValue() oscontrol→advertisedValue()[oscontrol advertisedValue]

YOsControl

Retourne la valeur courante du contrôle d'OS (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YOsControl <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du contrôle d'OS (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**oscontrol→get\_errorMessage()****YOsControl****oscontrol→errorMessage()[oscontrol errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'OS.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du contrôle d'OS.

**oscontrol→get\_errorType()****YOsControl****oscontrol→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'OS.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du contrôle d'OS.



**oscontrol→get\_friendlyName()****YOsControl****oscontrol→friendlyName()[oscontrol friendlyName]**

Retourne un identifiant global du contrôle d'OS au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du contrôle d'OS si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du contrôle d'OS (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le contrôle d'OS en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **oscontrol→get\_functionDescriptor() oscontrol→functionDescriptor()[oscontrol functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**oscontrol**→**get\_functionId()****YOsControl****oscontrol**→**functionId()**[**oscontrol functionId**]

Retourne l'identifiant matériel du contrôle d'OS, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le contrôle d'OS (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**oscontrol→get\_hardwareId()****YOsControl****oscontrol→hardwareId()[oscontrol hardwareId]**

Retourne l'identifiant matériel unique du contrôle d'OS au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du contrôle d'OS (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le contrôle d'OS (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**oscontrol→get\_logicalName()****YOsControl****oscontrol→logicalName()[oscontrol logicalName]**

Retourne le nom logique du contrôle d'OS.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YOsControl <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du contrôle d'OS. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**oscontrol→get\_module()****YOsControl****oscontrol→module()[oscontrol module]**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

## oscontrol→get\_module\_async() oscontrol→module\_async()

YOsControl

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**oscontrol→get\_shutdownCountdown()****YOsControl****oscontrol→shutdownCountdown()[oscontrol  
shutdownCountdown]**

Retourne le nombre de secondes restantes avant un arrêt de l'OS, ou zéro si aucun arrêt n'a été agendé.

js	function <b>get_shutdownCountdown</b> ( )
nodejs	function <b>get_shutdownCountdown</b> ( )
php	function <b>get_shutdownCountdown</b> ( )
cpp	int <b>get_shutdownCountdown</b> ( )
m	-(int) shutdownCountdown
pas	function <b>get_shutdownCountdown</b> ( ): LongInt
vb	function <b>get_shutdownCountdown</b> ( ) As Integer
cs	int <b>get_shutdownCountdown</b> ( )
java	int <b>get_shutdownCountdown</b> ( )
py	def <b>get_shutdownCountdown</b> ( )
cmd	YOsControl <b>target</b> <b>get_shutdownCountdown</b>

**Retourne :**

un entier représentant le nombre de secondes restantes avant un arrêt de l'OS, ou zéro si aucun arrêt n'a été agendé

En cas d'erreur, déclenche une exception ou retourne Y\_SHUTDOWNCOUNTDOWN\_INVALID.



**oscontrol→get\_userdata()****YOsControl****oscontrol→userdata()[oscontrol userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

Vérifie si le module hébergeant le contrôle d'OS est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du contrôle d'OS sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le contrôle d'OS est joignable, false sinon

**oscontrol→isOnline\_async()****YOsControl**

Vérifie si le module hébergeant le contrôle d'OS est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du contrôle d'OS sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

Met en cache les valeurs courantes du contrôle d'OS, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

#### Paramètres :

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**oscontrol→load\_async()****YOsControl**

Met en cache les valeurs courantes du contrôle d'OS, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## oscontrol→nextOsControl()[oscontrol nextOsControl]

YOsControl

Continue l'énumération des contrôle d'OS commencée à l'aide de `yFirstOsControl()`.

js	function <b>nextOsControl</b> ( )
nodejs	function <b>nextOsControl</b> ( )
php	function <b>nextOsControl</b> ( )
c++	YOsControl * <b>nextOsControl</b> ( )
m	-(YOsControl*) <b>nextOsControl</b>
pas	function <b>nextOsControl</b> ( ): TYOsControl
vb	function <b>nextOsControl</b> ( ) As YOsControl
cs	YOsControl <b>nextOsControl</b> ( )
java	YOsControl <b>nextOsControl</b> ( )
py	def <b>nextOsControl</b> ( )

### Retourne :

un pointeur sur un objet `YOsControl` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## oscontrol→registerValueCallback()[oscontrol registerValueCallback: ]

## YOsControl

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
c++	int <b>registerValueCallback</b> ( YOsControlValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YOsControlValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYOsControlValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**oscontrol→set\_logicalName()****YOsControl****oscontrol→setLogicalName()[oscontrol  
setLogicalName: ]**

Modifie le nom logique du contrôle d'OS.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YOsControl <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du contrôle d'OS.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**oscontrol→set\_userdata()****YOsControl****oscontrol→setUserData()[oscontrol setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

Agende un arrêt de l'OS dans un nombre donné de secondes.

js	function <b>shutdown</b> ( <b>secBeforeShutDown</b> )
nodejs	function <b>shutdown</b> ( <b>secBeforeShutDown</b> )
php	function <b>shutdown</b> ( <b>\$secBeforeShutDown</b> )
c++	int <b>shutdown</b> ( int <b>secBeforeShutDown</b> )
m	-(int) <b>shutdown</b> : (int) <b>secBeforeShutDown</b>
pas	function <b>shutdown</b> ( <b>secBeforeShutDown</b> : LongInt): LongInt
vb	function <b>shutdown</b> ( ) As Integer
cs	int <b>shutdown</b> ( int <b>secBeforeShutDown</b> )
java	int <b>shutdown</b> ( int <b>secBeforeShutDown</b> )
py	def <b>shutdown</b> ( <b>secBeforeShutDown</b> )
cmd	YOsControl <b>target shutdown secBeforeShutDown</b>

**Paramètres :**

**secBeforeShutDown** nombre de secondes avant l'arrêt

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**oscontrol→wait\_async()****YOsControl**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.28. Interface de la fonction Power

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_power.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YPower = yoctolib.YPower;
php	require_once('yocto_power.php');
c++	#include "yocto_power.h"
m	#import "yocto_power.h"
pas	uses yocto_power;
vb	yocto_power.vb
cs	yocto_power.cs
java	import com.yoctopuce.YoctoAPI.YPower;
py	from yocto_power import *

### Fonction globales

#### yFindPower(func)

Permet de retrouver un capteur de puissance électrique d'après un identifiant donné.

#### yFirstPower()

Commence l'énumération des capteurs de puissance électrique accessibles par la librairie.

### Méthodes des objets YPower

#### power→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### power→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de puissance électrique au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### power→get\_advertisedValue()

Retourne la valeur courante du capteur de puissance électrique (pas plus de 6 caractères).

#### power→get\_cosPhi()

Retourne le facteur de puissance (rapport entre la puissance réelle consommée, en W, et la puissance apparente fournie, en VA).

#### power→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### power→get\_currentValue()

Retourne la valeur instantanée de la puissance électrique.

#### power→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de puissance électrique.

#### power→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de puissance électrique.

#### power→get\_friendlyName()

Retourne un identifiant global du capteur de puissance électrique au format `NOM_MODULE . NOM_FONCTION`.

#### power→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

**`power→get_functionId()`**

Retourne l'identifiant matériel du capteur de puissance électrique, sans référence au module.

**`power→get_hardwareId()`**

Retourne l'identifiant matériel unique du capteur de puissance électrique au format `SERIAL.FUNCTIONID`.

**`power→get_highestValue()`**

Retourne la valeur maximale observée pour la puissance électrique.

**`power→get_logFrequency()`**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**`power→get_logicalName()`**

Retourne le nom logique du capteur de puissance électrique.

**`power→get_lowestValue()`**

Retourne la valeur minimale observée pour la puissance électrique.

**`power→get_meter()`**

Retourne la valeur actuelle du compteur d'énergie, calculée par le wattmètre en intégrant la consommation instantanée.

**`power→get_meterTimer()`**

Retourne le temps écoulé depuis la dernière initialisation du compteur d'énergie, en secondes

**`power→get_module()`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`power→get_module_async(callback, context)`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`power→get_recordedData(startTime, endTime)`**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**`power→get_reportFrequency()`**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**`power→get_resolution()`**

Retourne la résolution des valeurs mesurées.

**`power→get_unit()`**

Retourne l'unité dans laquelle la puissance électrique est exprimée.

**`power→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`power→isOnline()`**

Vérifie si le module hébergeant le capteur de puissance électrique est joignable, sans déclencher d'erreur.

**`power→isOnline_async(callback, context)`**

Vérifie si le module hébergeant le capteur de puissance électrique est joignable, sans déclencher d'erreur.

**`power→load(msValidity)`**

Met en cache les valeurs courantes du capteur de puissance électrique, avec une durée de validité spécifiée.

**`power→loadCalibrationPoints(rawValues, refValues)`**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

**`power→load_async(msValidity, callback, context)`**

### 3. Reference

Met en cache les valeurs courantes du capteur de puissance électrique, avec une durée de validité spécifiée.

#### **power**→**nextPower()**

Continue l'énumération des capteurs de puissance électrique commencée à l'aide de `yFirstPower()`.

#### **power**→**registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

#### **power**→**registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **power**→**reset()**

Réinitialise le compteur d'énergie.

#### **power**→**set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée pour la puissance électrique.

#### **power**→**set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

#### **power**→**set\_logicalName(newval)**

Modifie le nom logique du capteur de puissance électrique.

#### **power**→**set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée pour la puissance électrique.

#### **power**→**set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

#### **power**→**set\_resolution(newval)**

Modifie la résolution des valeurs mesurées.

#### **power**→**set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

#### **power**→**wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YPower.FindPower() yFindPower()yFindPower()

## YPower

Permet de retrouver un capteur de puissance electrique d'après un identifiant donné.

js	function <b>yFindPower</b> ( <b>func</b> )
nodejs	function <b>FindPower</b> ( <b>func</b> )
php	function <b>yFindPower</b> ( <b>\$func</b> )
cpp	YPower* <b>yFindPower</b> ( const string& <b>func</b> )
m	YPower* <b>yFindPower</b> ( NSString* <b>func</b> )
pas	function <b>yFindPower</b> ( <b>func</b> : string): TYPower
vb	function <b>yFindPower</b> ( ByVal <b>func</b> As String) As YPower
cs	YPower <b>FindPower</b> ( string <b>func</b> )
java	YPower <b>FindPower</b> ( String <b>func</b> )
py	def <b>FindPower</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de puissance electrique soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YPower.isOnline()` pour tester si le capteur de puissance electrique est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de puissance electrique sans ambiguïté

### Retourne :

un objet de classe `YPower` qui permet ensuite de contrôler le capteur de puissance electrique.

## YPower.FirstPower() yFirstPower()yFirstPower()

YPower

Commence l'énumération des capteurs de puissance électrique accessibles par la librairie.

js	function <b>yFirstPower</b> ( )
nodejs	function <b>FirstPower</b> ( )
php	function <b>yFirstPower</b> ( )
cpp	YPower* <b>yFirstPower</b> ( )
m	YPower* <b>yFirstPower</b> ( )
pas	function <b>yFirstPower</b> ( ): TYPower
vb	function <b>yFirstPower</b> ( ) As YPower
cs	YPower <b>FirstPower</b> ( )
java	YPower <b>FirstPower</b> ( )
py	def <b>FirstPower</b> ( )

Utiliser la fonction `YPower.nextPower()` pour itérer sur les autres capteurs de puissance électrique.

### Retourne :

un pointeur sur un objet `YPower`, correspondant à le premier capteur de puissance électrique accessible en ligne, ou `null` si il n'y a pas de capteurs de puissance électrique disponibles.



## power→calibrateFromPoints()[power calibrateFromPoints: ]

YPower

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YPower target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power→describe()[power describe]****YPower**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de puissance électrique au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de puissance électrique (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**power**→**get\_advertisedValue()****YPower****power**→**advertisedValue()**[**power advertisedValue**]

Retourne la valeur courante du capteur de puissance electrique (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YPower <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de puissance electrique (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**power**→**get\_cosPhi()****YPower****power**→**cosPhi()**[**power cosPhi**]

Retourne le facteur de puissance (rapport entre la puissance réelle consommée, en W, et la puissance apparente fournie, en VA).

js	function <b>get_cosPhi</b> ( )
nodejs	function <b>get_cosPhi</b> ( )
php	function <b>get_cosPhi</b> ( )
cpp	double <b>get_cosPhi</b> ( )
m	-(double) cosPhi
pas	function <b>get_cosPhi</b> ( ): double
vb	function <b>get_cosPhi</b> ( ) As Double
cs	double <b>get_cosPhi</b> ( )
java	double <b>get_cosPhi</b> ( )
py	def <b>get_cosPhi</b> ( )
cmd	YPower <b>target</b> <b>get_cosPhi</b>

**Retourne :**

une valeur numérique représentant le facteur de puissance (rapport entre la puissance réelle consommée, en W, et la puissance apparente fournie, en VA)

En cas d'erreur, déclenche une exception ou retourne Y\_COSPHI\_INVALID.

**power**→**get\_currentRawValue()****YPower****power**→**currentRawValue()**[**power** **currentRawValue**]

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YPower <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**power**→**get\_currentValue()****YPower****power**→**currentValue()**[**power** **currentValue**]

Retourne la valeur instantanée de la puissance électrique.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YPower <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur instantanée de la puissance électrique

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**power→get\_errorMessage()****YPower****power→errorMessage()[power errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de puissance électrique.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de puissance électrique.

**power**→**get\_errorType()****YPower****power**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de puissance électrique.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de puissance électrique.



**power**→**get\_friendlyName()****YPower****power**→**friendlyName()**[**power friendlyName**]

Retourne un identifiant global du capteur de puissance electrique au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
c++	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de puissance electrique si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de puissance electrique (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de puissance electrique en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**power**→**get\_functionDescriptor()**  
**power**→**functionDescriptor()[power**  
**functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**power**→**get\_functionId()****YPower****power**→**functionId()**[**power functionId**]

Retourne l'identifiant matériel du capteur de puissance électrique, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de puissance électrique (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**power**→**get\_hardwareId()****YPower****power**→**hardwareId()**[**power hardwareId**]

Retourne l'identifiant matériel unique du capteur de puissance électrique au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de puissance électrique (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de puissance électrique (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**power**→**get\_highestValue()****YPower****power**→**highestValue()**[**power** **highestValue**]

Retourne la valeur maximale observée pour la puissance électrique.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YPower <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la puissance électrique

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**power**→**get\_logFrequency()****YPower****power**→**logFrequency()[power logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YPower <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**power**→**get\_logicalName()****YPower****power**→**logicalName()**[**power** **logicalName**]

Retourne le nom logique du capteur de puissance electrique.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YPower <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de puissance electrique. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**power**→**get\_lowestValue()****YPower****power**→**lowestValue()**[**power** **lowestValue**]

Retourne la valeur minimale observée pour la puissance électrique.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YPower <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la puissance électrique

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.



**power→get\_meter()****YPower****power→meter()[power meter]**

Retourne la valeur actuelle du compteur d'energie, calculée par le wattmètre en intégrant la consommation instantanée.

js	function <b>get_meter</b> ( )
nodejs	function <b>get_meter</b> ( )
php	function <b>get_meter</b> ( )
cpp	double <b>get_meter</b> ( )
m	-(double) meter
pas	function <b>get_meter</b> ( ): double
vb	function <b>get_meter</b> ( ) As Double
cs	double <b>get_meter</b> ( )
java	double <b>get_meter</b> ( )
py	def <b>get_meter</b> ( )
cmd	YPower <b>target</b> <b>get_meter</b>

Ce compteur est réinitialisé à chaque démarrage du module.

**Retourne :**

une valeur numérique représentant la valeur actuelle du compteur d'energie, calculée par le wattmètre en intégrant la consommation instantanée

En cas d'erreur, déclenche une exception ou retourne Y\_METER\_INVALID.

**power**→**get\_meterTimer()****YPower****power**→**meterTimer()**[**power meterTimer**]

Retourne le temps écoulé depuis la dernière initialisation du compteur d'énergie, en secondes

js	function <b>get_meterTimer</b> ( )
nodejs	function <b>get_meterTimer</b> ( )
php	function <b>get_meterTimer</b> ( )
cpp	int <b>get_meterTimer</b> ( )
m	-(int) meterTimer
pas	function <b>get_meterTimer</b> ( ): LongInt
vb	function <b>get_meterTimer</b> ( ) As Integer
cs	int <b>get_meterTimer</b> ( )
java	int <b>get_meterTimer</b> ( )
py	def <b>get_meterTimer</b> ( )
cmd	YPower <b>target</b> <b>get_meterTimer</b>

**Retourne :**

un entier représentant le temps écoulé depuis la dernière initialisation du compteur d'énergie, en secondes

En cas d'erreur, déclenche une exception ou retourne Y\_METERTIMER\_INVALID.

**power**→**get\_module()****YPower****power**→**module()**[**power module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**power→get\_module\_async()**  
**power→module\_async()**

YPower

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**power→get\_recordedData()****YPower****power→recordedData()[power recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YPower <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

- startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.
- endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**power**→**get\_reportFrequency()****YPower****power**→**reportFrequency()**[power reportFrequency]

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YPower <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**power**→**get\_resolution()****YPower****power**→**resolution()**[power resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YPower <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**power**→**get\_unit()****YPower****power**→**unit()**[power unit]

Retourne l'unité dans laquelle la puissance électrique est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YPower <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la puissance électrique est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.



**power→get\_userdata()****YPower****power→userdata()[power userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**power→isOnline()[power isOnline]****YPower**

Vérifie si le module hébergeant le capteur de puissance électrique est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de puissance électrique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de puissance électrique est joignable, false sinon

**power→isOnline\_async()****YPower**

Vérifie si le module hébergeant le capteur de puissance électrique est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de puissance électrique sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'il est à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**power→load()[power load: ]****YPower**

Met en cache les valeurs courantes du capteur de puissance électrique, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## power→loadCalibrationPoints()[power loadCalibrationPoints: ]

YPower

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
                             : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YPower target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power→load\_async()****YPower**

Met en cache les valeurs courantes du capteur de puissance électrique, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**power→nextPower()[power nextPower]****YPower**

Continue l'énumération des capteurs de puissance électrique commencée à l'aide de `yFirstPower()`.

<code>js</code>	<code>function nextPower( )</code>
<code>nodejs</code>	<code>function nextPower( )</code>
<code>php</code>	<code>function nextPower( )</code>
<code>cpp</code>	<code>YPower * nextPower( )</code>
<code>m</code>	<code>-(YPower*) nextPower</code>
<code>pas</code>	<code>function nextPower( ): TYPower</code>
<code>vb</code>	<code>function nextPower( ) As YPower</code>
<code>cs</code>	<code>YPower nextPower( )</code>
<code>java</code>	<code>YPower nextPower( )</code>
<code>py</code>	<code>def nextPower( )</code>

**Retourne :**

un pointeur sur un objet `YPower` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## power→registerTimedReportCallback()[power registerTimedReportCallback: ]

YPower

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YPowerTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YPowerTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYPowerTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.



## power→registerValueCallback()[power registerValueCallback: ]

YPower

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YPowerValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YPowerValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYPowerValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**power→reset()[power reset]****YPower**

Réinitialise le compteur d'énergie.

js	function <b>reset</b> ( )
nodejs	function <b>reset</b> ( )
php	function <b>reset</b> ( )
cpp	int <b>reset</b> ( )
m	-(int) <b>reset</b>
pas	function <b>reset</b> ( ): LongInt
vb	function <b>reset</b> ( ) As Integer
cs	int <b>reset</b> ( )
java	int <b>reset</b> ( )
py	def <b>reset</b> ( )
cmd	YPower <b>target reset</b>

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power**→**set\_highestValue()****YPower****power**→**setHighestValue()**[**power setHighestValue:** ]

Modifie la mémoire de valeur maximale observée pour la puissance électrique.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YPower <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour la puissance électrique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power→set\_logFrequency()****YPower****power→setLogFrequency()[power setLogFrequency:  
]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YPower <b>target set_logFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power→set\_logicalName()****YPower****power→setLogicalName()[power setLogicalName: ]**

Modifie le nom logique du capteur de puissance electrique.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YPower <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de puissance electrique.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power**→**set\_lowestValue()****YPower****power**→**setLowestValue()**[**power setLowestValue:** ]

Modifie la mémoire de valeur minimale observée pour la puissance électrique.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YPower <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour la puissance électrique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power→set\_reportFrequency()**  
**power→setReportFrequency()[power**  
**setReportFrequency: ]**

YPower

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YPower <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**power**→**set\_resolution()****power**→**setResolution()**[**power** **setResolution**: ]

Modifie la résolution des valeurs mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YPower <b>target set_resolution newval</b>

La résolution correspond à la précision de la représentation numérique des mesures. Changer la résolution ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**power→set\_userdata()****YPower****power→setUserData()[power setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**power→wait\_async()****YPower**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.29. Interface de la fonction Pressure

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_pressure.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YPressure = yoctolib.YPressure;</code>
php	<code>require_once('yocto_pressure.php');</code>
c++	<code>#include "yocto_pressure.h"</code>
m	<code>#import "yocto_pressure.h"</code>
pas	<code>uses yocto_pressure;</code>
vb	<code>yocto_pressure.vb</code>
cs	<code>yocto_pressure.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YPressure;</code>
py	<code>from yocto_pressure import *</code>

### Fonction globales

#### **yFindPressure(func)**

Permet de retrouver un capteur de pression d'après un identifiant donné.

#### **yFirstPressure()**

Commence l'énumération des capteurs de pression accessibles par la librairie.

### Méthodes des objets YPressure

#### **pressure→calibrateFromPoints(rawValues, refValues)**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### **pressure→describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de pression au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### **pressure→get\_advertisedValue()**

Retourne la valeur courante du capteur de pression (pas plus de 6 caractères).

#### **pressure→get\_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### **pressure→get\_currentValue()**

Retourne la mesure actuelle de la pression.

#### **pressure→get\_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de pression.

#### **pressure→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de pression.

#### **pressure→get\_friendlyName()**

Retourne un identifiant global du capteur de pression au format `NOM_MODULE . NOM_FONCTION`.

#### **pressure→get\_functionDescriptor()**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **pressure→get\_functionId()**

Retourne l'identifiant matériel du capteur de pression, sans référence au module.

#### **pressure→get\_hardwareId()**

<code>pressure→get_highestValue()</code>	Retourne l'identifiant matériel unique du capteur de pression au format <code>SERIAL.FUNCTIONID</code> .
<code>pressure→get_logFrequency()</code>	Retourne la valeur maximale observée pour la pression.
<code>pressure→get_logicalName()</code>	Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.
<code>pressure→get_lowestValue()</code>	Retourne le nom logique du capteur de pression.
<code>pressure→get_module()</code>	Retourne la valeur minimale observée pour la pression.
<code>pressure→get_module_async(callback, context)</code>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<code>pressure→get_recordedData(startTime, endTime)</code>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<code>pressure→get_reportFrequency()</code>	Retourne un objet <code>DataSet</code> représentant des mesures de ce capteur précédemment enregistrées à l'aide du <code>DataLogger</code> , pour l'intervalle de temps spécifié.
<code>pressure→get_resolution()</code>	Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.
<code>pressure→get_unit()</code>	Retourne la résolution des valeurs mesurées.
<code>pressure→get_userData()</code>	Retourne l'unité dans laquelle la pression est exprimée.
<code>pressure→isOnline()</code>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<code>pressure→isOnline_async(callback, context)</code>	Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.
<code>pressure→load(msValidity)</code>	Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.
<code>pressure→loadCalibrationPoints(rawValues, refValues)</code>	Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.
<code>pressure→load_async(msValidity, callback, context)</code>	Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode <code>calibrateFromPoints</code> .
<code>pressure→nextPressure()</code>	Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.
<code>pressure→registerTimedReportCallback(callback)</code>	Continue l'énumération des capteurs de pression commencée à l'aide de <code>yFirstPressure()</code> .
<code>pressure→registerValueCallback(callback)</code>	Enregistre la fonction de callback qui est appelée à chaque notification périodique.
<code>pressure→set_highestValue(newval)</code>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<code>pressure→set_logFrequency(newval)</code>	Modifie la mémoire de valeur maximale observée pour la pression.
<code>pressure→set_reportFrequency(newval)</code>	

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**pressure**→**set\_logicalName**(newval)

Modifie le nom logique du capteur de pression.

**pressure**→**set\_lowestValue**(newval)

Modifie la mémoire de valeur minimale observée pour la pression.

**pressure**→**set\_reportFrequency**(newval)

Modifie la fréquence de notification périodique des valeurs mesurées.

**pressure**→**set\_resolution**(newval)

Modifie la résolution des valeurs physique mesurées.

**pressure**→**set\_userData**(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**pressure**→**wait\_async**(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YPressure.FindPressure() yFindPressure()yFindPressure()

YPressure

Permet de retrouver un capteur de pression d'après un identifiant donné.

js	function <b>yFindPressure</b> ( <b>func</b> )
nodejs	function <b>FindPressure</b> ( <b>func</b> )
php	function <b>yFindPressure</b> ( <b>\$func</b> )
cpp	YPressure* <b>yFindPressure</b> ( const string& <b>func</b> )
m	YPressure* <b>yFindPressure</b> ( NSString* <b>func</b> )
pas	function <b>yFindPressure</b> ( <b>func</b> : string): TYPpressure
vb	function <b>yFindPressure</b> ( ByVal <b>func</b> As String) As YPressure
cs	YPressure <b>FindPressure</b> ( string <b>func</b> )
java	YPressure <b>FindPressure</b> ( String <b>func</b> )
py	def <b>FindPressure</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de pression soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YPressure.isOnline()` pour tester si le capteur de pression est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de pression sans ambiguïté

### Retourne :

un objet de classe `YPressure` qui permet ensuite de contrôler le capteur de pression.

## YPressure.FirstPressure() yFirstPressure()yFirstPressure()

## YPressure

Commence l'énumération des capteurs de pression accessibles par la librairie.

js	function <b>yFirstPressure</b> ( )
nodejs	function <b>FirstPressure</b> ( )
php	function <b>yFirstPressure</b> ( )
cpp	YPressure* <b>yFirstPressure</b> ( )
m	YPressure* <b>yFirstPressure</b> ( )
pas	function <b>yFirstPressure</b> ( ): TYPressure
vb	function <b>yFirstPressure</b> ( ) As YPressure
cs	YPressure <b>FirstPressure</b> ( )
java	YPressure <b>FirstPressure</b> ( )
py	def <b>FirstPressure</b> ( )

Utiliser la fonction `YPressure.nextPressure( )` pour itérer sur les autres capteurs de pression.

### Retourne :

un pointeur sur un objet `YPressure`, correspondant à le premier capteur de pression accessible en ligne, ou `null` si il n'y a pas de capteurs de pression disponibles.

## pressure→calibrateFromPoints()[pressure calibrateFromPoints: ]

YPressure

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

js	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
nodejs	function <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
php	function <b>calibrateFromPoints</b> ( <b>\$rawValues</b> , <b>\$refValues</b> )
cpp	int <b>calibrateFromPoints</b> ( vector<double> <b>rawValues</b> , vector<double> <b>refValues</b> )
m	-(int) <b>calibrateFromPoints</b> : (NSMutableArray*) <b>rawValues</b> : (NSMutableArray*) <b>refValues</b>
pas	function <b>calibrateFromPoints</b> ( <b>rawValues</b> : TDoubleArray, <b>refValues</b> : TDoubleArray): LongInt
vb	procedure <b>calibrateFromPoints</b> ( )
cs	int <b>calibrateFromPoints</b> ( List<double> <b>rawValues</b> , List<double> <b>refValues</b> )
java	int <b>calibrateFromPoints</b> ( ArrayList<Double> <b>rawValues</b> , ArrayList<Double> <b>refValues</b> )
py	def <b>calibrateFromPoints</b> ( <b>rawValues</b> , <b>refValues</b> )
cmd	YPressure <b>target</b> <b>calibrateFromPoints</b> <b>rawValues</b> <b>refValues</b>

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**pressure→describe()[pressure describe]****YPressure**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de pression au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de pression (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**pressure**→**get\_advertisedValue()**  
**pressure**→**advertisedValue()[pressure**  
**advertisedValue]**

**YPressure**

Retourne la valeur courante du capteur de pression (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YPressure <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de pression (pas plus de 6 caractères).  
En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**pressure**→**get\_currentRawValue()**  
**pressure**→**currentRawValue()[pressure**  
**currentRawValue]**

**YPressure**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YPressure <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**pressure**→**get\_currentValue()****YPressure****pressure**→**currentValue()**[**pressure** **currentValue**]

Retourne la mesure actuelle de la pression.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YPressure <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la mesure actuelle de la pression

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**pressure→get\_errorMessage()****YPressure****pressure→errorMessage()[pressure errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de pression.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de pression.

**pressure**→**get\_errorType()****YPressure****pressure**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de pression.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de pression.

**pressure→get\_friendlyName()****YPressure****pressure→friendlyName()[pressure friendlyName]**

Retourne un identifiant global du capteur de pression au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de pression si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de pression (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de pression en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**pressure**→**get\_functionDescriptor()**  
**pressure**→**functionDescriptor()[pressure**  
**functionDescriptor]**

YPressure

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID



**pressure**→**get\_functionId()****YPressure****pressure**→**functionId()**[**pressure functionId**]

Retourne l'identifiant matériel du capteur de pression, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de pression (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**pressure**→**get\_hardwareId()****YPressure****pressure**→**hardwareId()**[**pressure hardwareId**]

Retourne l'identifiant matériel unique du capteur de pression au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de pression (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de pression (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**pressure→get\_highestValue()****YPressure****pressure→highestValue()[pressure highestValue]**

Retourne la valeur maximale observée pour la pression.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YPressure <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la pression

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**pressure**→**get\_logFrequency()****YPressure****pressure**→**logFrequency()**[**pressure logFrequency**]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YPressure <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**pressure→get\_logicalName()****YPressure****pressure→logicalName()[pressure logicalName]**

Retourne le nom logique du capteur de pression.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YPressure <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de pression. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**pressure**→**get\_lowestValue()****YPressure****pressure**→**lowestValue()**[**pressure** **lowestValue**]

Retourne la valeur minimale observée pour la pression.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YPressure <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la pression

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**pressure→get\_module()****YPressure****pressure→module()[pressure module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**pressure**→**get\_module\_async()****YPressure****pressure**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**pressure→get\_recordedData()****YPressure****pressure→recordedData()[pressure recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YPressure <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**pressure→get\_reportFrequency()**  
**pressure→reportFrequency()[pressure**  
**reportFrequency]**

**YPressure**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YPressure <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**pressure→get\_resolution()****YPressure****pressure→resolution()[pressure resolution]**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YPressure <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**pressure**→**get\_unit()****YPressure****pressure**→**unit()**[pressure unit]

Retourne l'unité dans laquelle la pression est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YPressure <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la pression est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**pressure→get\_userData()****YPressure****pressure→userData()[pressure userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userData.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
c++	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**pressure**→**isOnline()**[**pressure isOnline**]**YPressure**

Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de pression sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de pression est joignable, false sinon

**pressure→isOnline\_async()****YPressure**

Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de pression sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**pressure→load()[pressure load: ]****YPressure**

Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



## pressure→loadCalibrationPoints()[pressure loadCalibrationPoints: ]

YPressure

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
                             : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YPressure target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure→load\_async()****YPressure**

Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**pressure→nextPressure()[pressure nextPressure]****YPressure**

Continue l'énumération des capteurs de pression commencée à l'aide de `yFirstPressure()`.

js	<code>function nextPressure( )</code>
nodejs	<code>function nextPressure( )</code>
php	<code>function nextPressure( )</code>
cpp	<code>YPressure * nextPressure( )</code>
m	<code>-(YPressure*) nextPressure</code>
pas	<code>function nextPressure( ): TYPressure</code>
vb	<code>function nextPressure( ) As YPressure</code>
cs	<code>YPressure nextPressure( )</code>
java	<code>YPressure nextPressure( )</code>
py	<code>def nextPressure( )</code>

**Retourne :**

un pointeur sur un objet `YPressure` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## pressure→registerTimedReportCallback()[pressure registerTimedReportCallback: ]

YPressure

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YPressureTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YPressureTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYPressureTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## pressure→registerValueCallback()[pressure registerValueCallback: ]

YPressure

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YPressureValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YPressureValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYPresureValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**pressure**→**set\_highestValue()****YPressure****pressure**→**setHighestValue()**[**pressure**  
**setHighestValue:** ]

Modifie la mémoire de valeur maximale observée pour la pression.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YPressure <b>target</b> <b>set_highestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour la pression

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure→set\_logFrequency()**  
**pressure→setLogFrequency()[pressure**  
**setLogFrequency: ]**

**YPressure**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
c++	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YPressure <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure→set\_logicalName()****YPressure****pressure→setLogicalName()[pressure  
setLogicalName: ]**

Modifie le nom logique du capteur de pression.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YPressure <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de pression.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**pressure→set\_lowestValue()**  
**pressure→setLowestValue()[pressure**  
**setLowestValue: ]**

**YPressure**

Modifie la mémoire de valeur minimale observée pour la pression.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YPressure <b>target set_lowestValue newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour la pression

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure→set\_reportFrequency()**  
**pressure→setReportFrequency()[pressure**  
**setReportFrequency: ]**

YPressure

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YPressure <b>target set_reportFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure**→**set\_resolution()****YPressure****pressure**→**setResolution()**[**pressure** **setResolution**: ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	<b>YPressure</b> <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pressure→set\_userdata()****YPressure****pressure→setUserData()[pressure setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

<code>js</code>	<code>function set_userdata( data )</code>
<code>nodejs</code>	<code>function set_userdata( data )</code>
<code>php</code>	<code>function set_userdata( \$data )</code>
<code>cpp</code>	<code>void set_userdata( void* data )</code>
<code>m</code>	<code>-(void) setUserData : (void*) data</code>
<code>pas</code>	<code>procedure set_userdata( data: Tobject )</code>
<code>vb</code>	<code>procedure set_userdata( ByVal data As Object )</code>
<code>cs</code>	<code>void set_userdata( object data )</code>
<code>java</code>	<code>void set_userdata( Object data )</code>
<code>py</code>	<code>def set_userdata( data )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**pressure→wait\_async()****YPressure**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.30. Interface de la fonction Pwm

La librairie de programmation Yoctopuce permet simplement de configurer, démarrer et arrêter le PWM.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_pwmoutput.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YPwmOutput = yoctolib.YPwmOutput;
php	require_once('yocto_pwmoutput.php');
c++	#include "yocto_pwmoutput.h"
m	#import "yocto_pwmoutput.h"
pas	uses yocto_pwmoutput;
vb	yocto_pwmoutput.vb
cs	yocto_pwmoutput.cs
java	import com.yoctopuce.YoctoAPI.YPwmOutput;
py	from yocto_pwmoutput import *

### Fonction globales

#### yFindPwmOutput(func)

Permet de retrouver un PWM d'après un identifiant donné.

#### yFirstPwmOutput()

Commence l'énumération des PWM accessibles par la librairie.

### Méthodes des objets YPwmOutput

#### pwmoutput→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du PWM au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

#### pwmoutput→dutyCycleMove(target, ms\_duration)

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

#### pwmoutput→get\_advertisedValue()

Retourne la valeur courante du PWM (pas plus de 6 caractères).

#### pwmoutput→get\_dutyCycle()

Retourne le duty cycle du \$FUNCTION\$ sous la forme d'un nombre à virgule entre 0 et 1.

#### pwmoutput→get\_dutyCycleAtPowerOn()

Retourne le duty cycle du \$FUNCTION\$ au démarrage du module, sous la forme d'un nombre à virgule entre 0.0% et 100.

#### pwmoutput→get\_enabled()

Retourne l'état de fonctionnement du \$FUNCTION\$.

#### pwmoutput→get\_enabledAtPowerOn()

Retourne l'état de fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

#### pwmoutput→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du PWM.

#### pwmoutput→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du PWM.

#### pwmoutput→get\_frequency()

Retourne la fréquence du \$FUNCTION\$ en Hz.

#### pwmoutput→get\_friendlyName()

Retourne un identifiant global du PWM au format `NOM_MODULE . NOM_FONCTION`.

**pwmoutput→get\_functionDescriptor()**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

**pwmoutput→get\_functionId()**

Retourne l'identifiant matériel du PWM, sans référence au module.

**pwmoutput→get\_hardwareId()**

Retourne l'identifiant matériel unique du PWM au format SERIAL.FUNCTIONID.

**pwmoutput→get\_logicalName()**

Retourne le nom logique du PWM.

**pwmoutput→get\_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**pwmoutput→get\_module\_async(callback, context)**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

**pwmoutput→get\_period()**

Retourne la période du \$FUNCTION\$ en nano secondes.

**pwmoutput→get\_pulseDuration()**

Retourne la longueur d'une impulsion du \$FUNCTION\$ en millisecondes.

**pwmoutput→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

**pwmoutput→isOnline()**

Vérifie si le module hébergeant le PWM est joignable, sans déclencher d'erreur.

**pwmoutput→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le PWM est joignable, sans déclencher d'erreur.

**pwmoutput→load(msValidity)**

Met en cache les valeurs courantes du PWM, avec une durée de validité spécifiée.

**pwmoutput→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du PWM, avec une durée de validité spécifiée.

**pwmoutput→nextPwmOutput()**

Continue l'énumération des PWM commencée à l'aide de yFirstPwmOutput().

**pwmoutput→pulseDurationMove(ms\_target, ms\_duration)**

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

**pwmoutput→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**pwmoutput→set\_dutyCycle(newval)**

Configure le duty cycle du \$FUNCTION\$.

**pwmoutput→set\_dutyCycleAtPowerOn(newval)**

Configure le duty cycle du \$FUNCTION\$ au démarrage du module.

**pwmoutput→set\_enabled(newval)**

Démarre ou arrête le \$FUNCTION\$.

**pwmoutput→set\_enabledAtPowerOn(newval)**

Configure l'état du fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

**pwmoutput→set\_frequency(newval)**

Configure la fréquence du \$FUNCTION\$.

**pwmoutput→set\_logicalName(newval)**

Modifie le nom logique du PWM.

**pwmoutput→set\_period(newval)**

### 3. Reference

---

Configure la période du \$FUNCTION\$.

**pwmoutput**→**set\_pulseDuration**(newval)

Configure la longueur d'une impulsion du \$FUNCTION\$.

**pwmoutput**→**set\_userdata**(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

**pwmoutput**→**wait\_async**(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.



## YPwmOutput.FindPwmOutput() yFindPwmOutput()/yFindPwmOutput()

## YPwmOutput

Permet de retrouver un PWM d'après un identifiant donné.

js	function <b>yFindPwmOutput</b> ( <b>func</b> )
nodejs	function <b>FindPwmOutput</b> ( <b>func</b> )
php	function <b>yFindPwmOutput</b> ( <b>\$func</b> )
cpp	YPwmOutput* <b>yFindPwmOutput</b> ( const string& <b>func</b> )
m	YPwmOutput* <b>yFindPwmOutput</b> ( NSString* <b>func</b> )
pas	function <b>yFindPwmOutput</b> ( <b>func</b> : string): TYPwmOutput
vb	function <b>yFindPwmOutput</b> ( ByVal <b>func</b> As String) As YPwmOutput
cs	YPwmOutput <b>FindPwmOutput</b> ( string <b>func</b> )
java	YPwmOutput <b>FindPwmOutput</b> ( String <b>func</b> )
py	def <b>FindPwmOutput</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le PWM soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YPwmOutput.isOnline()` pour tester si le PWM est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le PWM sans ambiguïté

### Retourne :

un objet de classe `YPwmOutput` qui permet ensuite de contrôler le PWM.

## YPwmOutput.FirstPwmOutput() yFirstPwmOutput()yFirstPwmOutput()

YPwmOutput

Commence l'énumération des PWM accessibles par la librairie.

js	function <b>yFirstPwmOutput</b> ( )
nodejs	function <b>FirstPwmOutput</b> ( )
php	function <b>yFirstPwmOutput</b> ( )
cpp	YPwmOutput* <b>yFirstPwmOutput</b> ( )
m	YPwmOutput* <b>yFirstPwmOutput</b> ( )
pas	function <b>yFirstPwmOutput</b> ( ): TYPwmOutput
vb	function <b>yFirstPwmOutput</b> ( ) As YPwmOutput
cs	YPwmOutput <b>FirstPwmOutput</b> ( )
java	YPwmOutput <b>FirstPwmOutput</b> ( )
py	def <b>FirstPwmOutput</b> ( )

Utiliser la fonction `YPwmOutput.nextPwmOutput( )` pour itérer sur les autres PWM.

### Retourne :

un pointeur sur un objet `YPwmOutput`, correspondant à le premier PWM accessible en ligne, ou `null` si il n'y a pas de PWM disponibles.

**pwmoutput→describe()[pwmoutput describe]****YPwmOutput**

Retourne un court texte décrivant de manière non-ambigüe l'instance du PWM au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le PWM (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## pwmoutput→dutyCycleMove()[pwmoutput dutyCycleMove: ]

YPwmOutput

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

js	function <b>dutyCycleMove</b> ( <b>target</b> , <b>ms_duration</b> )
nodejs	function <b>dutyCycleMove</b> ( <b>target</b> , <b>ms_duration</b> )
php	function <b>dutyCycleMove</b> ( <b>\$target</b> , <b>\$ms_duration</b> )
cpp	int <b>dutyCycleMove</b> ( double <b>target</b> , int <b>ms_duration</b> )
m	-(int) <b>dutyCycleMove</b> : (double) <b>target</b> : (int) <b>ms_duration</b>
pas	function <b>dutyCycleMove</b> ( <b>target</b> : double, <b>ms_duration</b> : LongInt): LongInt
vb	function <b>dutyCycleMove</b> ( ) As Integer
cs	int <b>dutyCycleMove</b> ( double <b>target</b> , int <b>ms_duration</b> )
java	int <b>dutyCycleMove</b> ( double <b>target</b> , int <b>ms_duration</b> )
py	def <b>dutyCycleMove</b> ( <b>target</b> , <b>ms_duration</b> )
cmd	YPwmOutput <b>target</b> <b>dutyCycleMove</b> <b>target</b> <b>ms_duration</b>

### Paramètres :

**target** nouveau duty cycle à la fin de la transition (nombre flottant, entre 0 et 1)  
**ms\_duration** durée totale de la transition, en millisecondes

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→get\_advertisedValue()****YPwmOutput****pwmoutput→advertisedValue()[pwmoutput  
advertisedValue]**

Retourne la valeur courante du PWM (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du PWM (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**pwmoutput→get\_dutyCycle()****YPwmOutput****pwmoutput→dutyCycle()[pwmoutput dutyCycle]**

Retourne le duty cycle du \$FUNCTION\$ sous la forme d'un nombre a virgule entre 0 et 1.

js	function <b>get_dutyCycle</b> ( )
nodejs	function <b>get_dutyCycle</b> ( )
php	function <b>get_dutyCycle</b> ( )
cpp	double <b>get_dutyCycle</b> ( )
m	-(double) dutyCycle
pas	function <b>get_dutyCycle</b> ( ): double
vb	function <b>get_dutyCycle</b> ( ) As Double
cs	double <b>get_dutyCycle</b> ( )
java	double <b>get_dutyCycle</b> ( )
py	def <b>get_dutyCycle</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_dutyCycle</b>

**Retourne :**

une valeur numérique représentant le duty cycle du \$FUNCTION\$ sous la forme d'un nombre a virgule entre 0 et 1

En cas d'erreur, déclenche une exception ou retourne Y\_DUTYCYCLE\_INVALID.

**pwmoutput→get\_dutyCycleAtPowerOn()**

**YPwmOutput**

**pwmoutput→dutyCycleAtPowerOn()[pwmoutput  
dutyCycleAtPowerOn]**

Retourne le duty cycle du \$FUNCTION\$ au démarrage du module, sous la forme d'un nombre à virgule entre 0.0% et 100.

js	function <b>get_dutyCycleAtPowerOn</b> ( )
nodejs	function <b>get_dutyCycleAtPowerOn</b> ( )
php	function <b>get_dutyCycleAtPowerOn</b> ( )
cpp	double <b>get_dutyCycleAtPowerOn</b> ( )
m	-(double) dutyCycleAtPowerOn
pas	function <b>get_dutyCycleAtPowerOn</b> ( ): double
vb	function <b>get_dutyCycleAtPowerOn</b> ( ) As Double
cs	double <b>get_dutyCycleAtPowerOn</b> ( )
java	double <b>get_dutyCycleAtPowerOn</b> ( )
py	def <b>get_dutyCycleAtPowerOn</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_dutyCycleAtPowerOn</b>

0%

#### Retourne :

une valeur numérique représentant le duty cycle du \$FUNCTION\$ au démarrage du module, sous la forme d'un nombre à virgule entre 0.0% et 100

En cas d'erreur, déclenche une exception ou retourne Y\_DUTYCYCLEATPOWERON\_INVALID.

**pwmoutput→get\_enabled()****YPwmOutput****pwmoutput→enabled()[pwmoutput enabled]**

Retourne l'état de fonctionnement du \$FUNCTION\$.

js	function <b>get_enabled</b> ( )
nodejs	function <b>get_enabled</b> ( )
php	function <b>get_enabled</b> ( )
cpp	Y_ENABLED_enum <b>get_enabled</b> ( )
m	-(Y_ENABLED_enum) enabled
pas	function <b>get_enabled</b> ( ): Integer
vb	function <b>get_enabled</b> ( ) As Integer
cs	int <b>get_enabled</b> ( )
java	int <b>get_enabled</b> ( )
py	def <b>get_enabled</b> ( )
cmd	YPwmOutput <b>target get_enabled</b>

**Retourne :**

soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon l'état de fonctionnement du \$FUNCTION\$

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLED\_INVALID.



**pwmoutput→get\_enabledAtPowerOn()**  
**pwmoutput→enabledAtPowerOn()[pwmoutput**  
**enabledAtPowerOn]**

**YPwmOutput**

Retourne l'état de fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

js	function <b>get_enabledAtPowerOn</b> ( )
nodejs	function <b>get_enabledAtPowerOn</b> ( )
php	function <b>get_enabledAtPowerOn</b> ( )
cpp	Y_ENABLEDATPOWERON_enum <b>get_enabledAtPowerOn</b> ( )
m	-(Y_ENABLEDATPOWERON_enum) enabledAtPowerOn
pas	function <b>get_enabledAtPowerOn</b> ( ): Integer
vb	function <b>get_enabledAtPowerOn</b> ( ) As Integer
cs	int <b>get_enabledAtPowerOn</b> ( )
java	int <b>get_enabledAtPowerOn</b> ( )
py	def <b>get_enabledAtPowerOn</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_enabledAtPowerOn</b>

#### Retourne :

soit Y\_ENABLEDATPOWERON\_FALSE, soit Y\_ENABLEDATPOWERON\_TRUE, selon l'état de fonctionnement du \$FUNCTION\$ à la mise sous tension du module

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLEDATPOWERON\_INVALID.

**pwmoutput→get\_errorMessage()**  
**pwmoutput→errorMessage()[pwmoutput**  
**errorMessage]**

**YPwmOutput**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du PWM.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du PWM.

**pwmoutput→get\_errorType()**  
**pwmoutput→errorType()**

**YPwmOutput**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du PWM.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du PWM.

**pwmoutput→get\_frequency()****YPwmOutput****pwmoutput→frequency()[pwmoutput frequency]**

Retourne la fréquence du \$FUNCTION\$ en Hz.

js	function <b>get_frequency</b> ( )
nodejs	function <b>get_frequency</b> ( )
php	function <b>get_frequency</b> ( )
cpp	int <b>get_frequency</b> ( )
m	-(int) frequency
pas	function <b>get_frequency</b> ( ): LongInt
vb	function <b>get_frequency</b> ( ) As Integer
cs	int <b>get_frequency</b> ( )
java	int <b>get_frequency</b> ( )
py	def <b>get_frequency</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_frequency</b>

**Retourne :**

un entier représentant la fréquence du \$FUNCTION\$ en Hz

En cas d'erreur, déclenche une exception ou retourne Y\_FREQUENCY\_INVALID.

**pwmoutput→get\_friendlyName()**  
**pwmoutput→friendlyName()[pwmoutput**  
**friendlyName]**

**YPwmOutput**

Retourne un identifiant global du PWM au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du PWM si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du PWM (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le PWM en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **pwmoutput→get\_functionDescriptor() pwmoutput→functionDescriptor()[pwmoutput functionDescriptor]**

**YPwmOutput**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

---

**pwmoutput→get\_functionId()****YPwmOutput****pwmoutput→functionId()[pwmoutput functionId]**

---

Retourne l'identifiant matériel du PWM, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le PWM (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**pwmoutput→get\_hardwareId()****YPwmOutput****pwmoutput→hardwareId()[pwmoutput hardwareId]**

Retourne l'identifiant matériel unique du PWM au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du PWM (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le PWM (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**pwmoutput→get\_logicalName()****YPwmOutput****pwmoutput→logicalName()[pwmoutput logicalName]**

Retourne le nom logique du PWM.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du PWM. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**pwmoutput→get\_module()****YPwmOutput****pwmoutput→module()[pwmoutput module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**pwmoutput→get\_module\_async()**  
**pwmoutput→module\_async()**

**YPwmOutput**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**pwmoutput→get\_period()****YPwmOutput****pwmoutput→period()[pwmoutput period]**

Retourne la période du \$FUNCTION\$ en nano secondes.

js	function <b>get_period</b> ( )
nodejs	function <b>get_period</b> ( )
php	function <b>get_period</b> ( )
cpp	double <b>get_period</b> ( )
m	-(double) period
pas	function <b>get_period</b> ( ): double
vb	function <b>get_period</b> ( ) As Double
cs	double <b>get_period</b> ( )
java	double <b>get_period</b> ( )
py	def <b>get_period</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_period</b>

**Retourne :**

une valeur numérique représentant la période du \$FUNCTION\$ en nano secondes

En cas d'erreur, déclenche une exception ou retourne Y\_PERIOD\_INVALID.

**pwmoutput→get\_pulseDuration()**  
**pwmoutput→pulseDuration()[pwmoutput**  
**pulseDuration]**

**YPwmOutput**

Retourne la longueur d'une impulsion du \$FUNCTION\$ en millisecondes.

js	function <b>get_pulseDuration</b> ( )
nodejs	function <b>get_pulseDuration</b> ( )
php	function <b>get_pulseDuration</b> ( )
cpp	double <b>get_pulseDuration</b> ( )
m	-(double) pulseDuration
pas	function <b>get_pulseDuration</b> ( ): double
vb	function <b>get_pulseDuration</b> ( ) As Double
cs	double <b>get_pulseDuration</b> ( )
java	double <b>get_pulseDuration</b> ( )
py	def <b>get_pulseDuration</b> ( )
cmd	YPwmOutput <b>target</b> <b>get_pulseDuration</b>

**Retourne :**

une valeur numérique représentant la longueur d'une impulsion du \$FUNCTION\$ en millisecondes

En cas d'erreur, déclenche une exception ou retourne Y\_PULSEDURATION\_INVALID.

**pwmoutput→get\_userdata()**

**YPwmOutput**

**pwmoutput→userdata()[pwmoutput userdata]**

Retourne le contenu de l'attribut `userdata`, précédemment stocké à l'aide de la méthode `set_userdata`.

<code>js</code>	<code>function <b>get_userdata</b>( )</code>
<code>nodejs</code>	<code>function <b>get_userdata</b>( )</code>
<code>php</code>	<code>function <b>get_userdata</b>( )</code>
<code>cpp</code>	<code>void * <b>get_userdata</b>( )</code>
<code>m</code>	<code>-(void*) <b>userdata</b></code>
<code>pas</code>	<code>function <b>get_userdata</b>( ): Tobject</code>
<code>vb</code>	<code>function <b>get_userdata</b>( ) As Object</code>
<code>cs</code>	<code>object <b>get_userdata</b>( )</code>
<code>java</code>	<code>Object <b>get_userdata</b>( )</code>
<code>py</code>	<code>def <b>get_userdata</b>( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**pwmoutput→isOnline()[pwmoutput isOnline]****YPwmOutput**

Vérifie si le module hébergeant le PWM est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du PWM sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le PWM est joignable, `false` sinon

**pwmoutput→isOnline\_async()****YPwmOutput**

Vérifie si le module hébergeant le PWM est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du PWM sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**pwmoutput→load()[pwmoutput load: ]****YPwmOutput**

Met en cache les valeurs courantes du PWM, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→load\_async()****YPwmOutput**

Met en cache les valeurs courantes du PWM, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## pwmoutput→nextPwmOutput()[pwmoutput nextPwmOutput]

## YPwmOutput

Continue l'énumération des PWM commencée à l'aide de `yFirstPwmOutput()`.

js	function <b>nextPwmOutput</b> ( )
nodejs	function <b>nextPwmOutput</b> ( )
php	function <b>nextPwmOutput</b> ( )
cpp	YPwmOutput * <b>nextPwmOutput</b> ( )
m	-(YPwmOutput*) <b>nextPwmOutput</b>
pas	function <b>nextPwmOutput</b> ( ): TYPwmOutput
vb	function <b>nextPwmOutput</b> ( ) As YPwmOutput
cs	YPwmOutput <b>nextPwmOutput</b> ( )
java	YPwmOutput <b>nextPwmOutput</b> ( )
py	def <b>nextPwmOutput</b> ( )

### Retourne :

un pointeur sur un objet `YPwmOutput` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## pwmoutput→pulseDurationMove()[pwmoutput pulseDurationMove: ]

YPwmOutput

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

js	function <b>pulseDurationMove</b> ( <b>ms_target</b> , <b>ms_duration</b> )
nodejs	function <b>pulseDurationMove</b> ( <b>ms_target</b> , <b>ms_duration</b> )
php	function <b>pulseDurationMove</b> ( <b>\$ms_target</b> , <b>\$ms_duration</b> )
c++	int <b>pulseDurationMove</b> ( double <b>ms_target</b> , int <b>ms_duration</b> )
m	-(int) <b>pulseDurationMove</b> : (double) <b>ms_target</b> : (int) <b>ms_duration</b>
pas	function <b>pulseDurationMove</b> ( <b>ms_target</b> : double, <b>ms_duration</b> : LongInt): LongInt
vb	function <b>pulseDurationMove</b> ( ) As Integer
cs	int <b>pulseDurationMove</b> ( double <b>ms_target</b> , int <b>ms_duration</b> )
java	int <b>pulseDurationMove</b> ( double <b>ms_target</b> , int <b>ms_duration</b> )
py	def <b>pulseDurationMove</b> ( <b>ms_target</b> , <b>ms_duration</b> )
cmd	YPwmOutput <b>target</b> <b>pulseDurationMove</b> <b>ms_target</b> <b>ms_duration</b>

### Paramètres :

- ms\_target** nouvelle longueur des impulsions à la fin de la transition (nombre flottant, représentant la longueur en millisecondes)
- ms\_duration** durée totale de la transition, en millisecondes

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## pwmoutput→registerValueCallback()[pwmoutput registerValueCallback: ]

## YPwmOutput

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YPwmOutputValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YPwmOutputValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYPwmOutputValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**pwmoutput→set\_dutyCycle()****YPwmOutput****pwmoutput→setDutyCycle()[pwmoutput  
setDutyCycle: ]**

Configure le duty cycle du \$FUNCTION\$.

js	function <b>set_dutyCycle</b> ( <b>newval</b> )
nodejs	function <b>set_dutyCycle</b> ( <b>newval</b> )
php	function <b>set_dutyCycle</b> ( <b>\$newval</b> )
cpp	int <b>set_dutyCycle</b> ( double <b>newval</b> )
m	-(int) setDutyCycle : (double) <b>newval</b>
pas	function <b>set_dutyCycle</b> ( <b>newval</b> : double): integer
vb	function <b>set_dutyCycle</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_dutyCycle</b> ( double <b>newval</b> )
java	int <b>set_dutyCycle</b> ( double <b>newval</b> )
py	def <b>set_dutyCycle</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target set_dutyCycle newval</b>

**Paramètres :**

**newval** une valeur numérique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_dutyCycleAtPowerOn()**

**YPwmOutput**

**pwmoutput→setDutyCycleAtPowerOn()[pwmoutput  
setDutyCycleAtPowerOn: ]**

Configure le duty cycle du \$FUNCTION\$ au démarrage du module.

js	function <b>set_dutyCycleAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_dutyCycleAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_dutyCycleAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_dutyCycleAtPowerOn</b> ( double <b>newval</b> )
m	-(int) <b>setDutyCycleAtPowerOn</b> : (double) <b>newval</b>
pas	function <b>set_dutyCycleAtPowerOn</b> ( <b>newval</b> : double): integer
vb	function <b>set_dutyCycleAtPowerOn</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_dutyCycleAtPowerOn</b> ( double <b>newval</b> )
java	int <b>set_dutyCycleAtPowerOn</b> ( double <b>newval</b> )
py	def <b>set_dutyCycleAtPowerOn</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target set_dutyCycleAtPowerOn newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module sinon la modification n'aura aucun effet.

#### Paramètres :

**newval** une valeur numérique

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_enabled()****YPwmOutput****pwmoutput→setEnabled()[pwmoutput setEnabled: ]**

Démarré ou arrêté le \$FUNCTION\$.

js	function <b>set_enabled</b> ( <b>newval</b> )
nodejs	function <b>set_enabled</b> ( <b>newval</b> )
php	function <b>set_enabled</b> ( <b>\$newval</b> )
cpp	int <b>set_enabled</b> ( Y_ENABLED_enum <b>newval</b> )
m	-(int) <b>setEnabled</b> : (Y_ENABLED_enum) <b>newval</b>
pas	function <b>set_enabled</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabled</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabled</b> ( int <b>newval</b> )
java	int <b>set_enabled</b> ( int <b>newval</b> )
py	def <b>set_enabled</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target set_enabled newval</b>

**Paramètres :****newval** soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**pwmoutput→set\_enabledAtPowerOn()**

**YPwmOutput**

**pwmoutput→setEnabledAtPowerOn()[pwmoutput  
setEnabledAtPowerOn: ]**

Configure l'état du fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

js	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_enabledAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_enabledAtPowerOn</b> ( Y_ENABLEDATPOWERON_enum <b>newval</b> )
m	-(int) <b>setEnabledAtPowerOn</b> : (Y_ENABLEDATPOWERON_enum) <b>newval</b>
pas	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabledAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabledAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_enabledAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target</b> <b>set_enabledAtPowerOn</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module sinon la modification n'aura aucun effet.

#### Paramètres :

**newval** soit Y\_ENABLEDATPOWERON\_FALSE, soit Y\_ENABLEDATPOWERON\_TRUE

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

pwmoutput→set\_frequency()

YPwmOutput

pwmoutput→setFrequency()[pwmoutput  
setFrequency: ]

Configure la fréquence du \$FUNCTION\$.

js	function <b>set_frequency</b> ( <b>newval</b> )
nodejs	function <b>set_frequency</b> ( <b>newval</b> )
php	function <b>set_frequency</b> ( <b>\$newval</b> )
cpp	int <b>set_frequency</b> ( int <b>newval</b> )
m	-(int) setFrequency : (int) <b>newval</b>
pas	function <b>set_frequency</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_frequency</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_frequency</b> ( int <b>newval</b> )
java	int <b>set_frequency</b> ( int <b>newval</b> )
py	def <b>set_frequency</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target set_frequency newval</b>

Le duty cycle est conservé grâce à un changement automatique de la longueur des impulsions.

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_logicalName()****YPwmOutput****pwmoutput→setLogicalName()[pwmoutput  
setLogicalName: ]**

Modifie le nom logique du PWM.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du PWM.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_period()****YPwmOutput****pwmoutput→setPeriod()[pwmoutput setPeriod: ]**

Configure la période du \$FUNCTION\$.

js	function <b>set_period</b> ( <b>newval</b> )
nodejs	function <b>set_period</b> ( <b>newval</b> )
php	function <b>set_period</b> ( <b>\$newval</b> )
cpp	int <b>set_period</b> ( double <b>newval</b> )
m	-(int) setPeriod : (double) <b>newval</b>
pas	function <b>set_period</b> ( <b>newval</b> : double): integer
vb	function <b>set_period</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_period</b> ( double <b>newval</b> )
java	int <b>set_period</b> ( double <b>newval</b> )
py	def <b>set_period</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target set_period newval</b>

**Paramètres :**

**newval** une valeur numérique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_pulseDuration()**  
**pwmoutput→setPulseDuration()[pwmoutput**  
**setPulseDuration: ]**

YPwmOutput

Configure la longueur d'une impulsion du \$FUNCTION\$.

js	function <b>set_pulseDuration</b> ( <b>newval</b> )
nodejs	function <b>set_pulseDuration</b> ( <b>newval</b> )
php	function <b>set_pulseDuration</b> ( <b>\$newval</b> )
cpp	int <b>set_pulseDuration</b> ( double <b>newval</b> )
m	-(int) setPulseDuration : (double) <b>newval</b>
pas	function <b>set_pulseDuration</b> ( <b>newval</b> : double): integer
vb	function <b>set_pulseDuration</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_pulseDuration</b> ( double <b>newval</b> )
java	int <b>set_pulseDuration</b> ( double <b>newval</b> )
py	def <b>set_pulseDuration</b> ( <b>newval</b> )
cmd	YPwmOutput <b>target</b> <b>set_pulseDuration</b> <b>newval</b>

Attention la longueur d'un impulsion ne peut pas être plus grande que la période, dans la cas contraire, la longueur sera automatiquement tronqué à la période.

**Paramètres :**

**newval** une valeur numérique

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmoutput→set\_userdata()****YPwmOutput****pwmoutput→setUserData()[pwmoutput setUserData:  
]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : TObject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**pwmoutput→wait\_async()****YPwmOutput**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.31. Interface de la fonction PwmPowerSource

La librairie de programmation Yoctopuce permet de configurer la source de tension utilisée par tous les PWM situés sur un même module.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_pwmpowersource.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YPwmPowerSource = yoctolib.YPwmPowerSource;
php	require_once('yocto_pwmpowersource.php');
c++	#include "yocto_pwmpowersource.h"
m	#import "yocto_pwmpowersource.h"
pas	uses yocto_pwmpowersource;
vb	yocto_pwmpowersource.vb
cs	yocto_pwmpowersource.cs
java	import com.yoctopuce.YoctoAPI.YPwmPowerSource;
py	from yocto_pwmpowersource import *

### Fonction globales

#### yFindPwmPowerSource(func)

Permet de retrouver une source de tension d'après un identifiant donné.

#### yFirstPwmPowerSource()

Commence l'énumération des Source de tension accessibles par la librairie.

### Méthodes des objets YPwmPowerSource

#### pwmpowersource→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de la source de tension au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### pwmpowersource→get\_advertisedValue()

Retourne la valeur courante de la source de tension (pas plus de 6 caractères).

#### pwmpowersource→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la source de tension.

#### pwmpowersource→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la source de tension.

#### pwmpowersource→get\_friendlyName()

Retourne un identifiant global de la source de tension au format `NOM_MODULE . NOM_FONCTION`.

#### pwmpowersource→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### pwmpowersource→get\_functionId()

Retourne l'identifiant matériel de la source de tension, sans référence au module.

#### pwmpowersource→get\_hardwareId()

Retourne l'identifiant matériel unique de la source de tension au format `SERIAL . FUNCTIONID`.

#### pwmpowersource→get\_logicalName()

Retourne le nom logique de la source de tension.

#### pwmpowersource→get\_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### pwmpowersource→get\_module\_async(callback, context)



Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`pwmpowersource→get_powerMode()`**

Retourne la source de tension utilisé par tous les PWM du même module.

**`pwmpowersource→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`pwmpowersource→isOnline()`**

Vérifie si le module hébergeant la source de tension est joignable, sans déclencher d'erreur.

**`pwmpowersource→isOnline_async(callback, context)`**

Vérifie si le module hébergeant la source de tension est joignable, sans déclencher d'erreur.

**`pwmpowersource→load(msValidity)`**

Met en cache les valeurs courantes de la source de tension, avec une durée de validité spécifiée.

**`pwmpowersource→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes de la source de tension, avec une durée de validité spécifiée.

**`pwmpowersource→nextPwmPowerSource()`**

Continue l'énumération des Source de tension commencée à l'aide de `yFirstPwmPowerSource()`.

**`pwmpowersource→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`pwmpowersource→set_logicalName(newval)`**

Modifie le nom logique de la source de tension.

**`pwmpowersource→set_powerMode(newval)`**

Modifie le mode fonctionnement des PWM qui peut sortir du 5 volts isolé issu de l'USB, du 3V isolé issu de l'USB, une tension arbitraire issue de l'alimentation externe.

**`pwmpowersource→set_userData(data)`**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**`pwmpowersource→wait_async(callback, context)`**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YPwmPowerSource.FindPwmPowerSource() yFindPwmPowerSource()yFindPwmPowerSource()

YPwmPowerSource

Permet de retrouver une source de tension d'après un identifiant donné.

js	function <b>yFindPwmPowerSource</b> ( <b>func</b> )
nodejs	function <b>FindPwmPowerSource</b> ( <b>func</b> )
php	function <b>yFindPwmPowerSource</b> ( <b>\$func</b> )
cpp	YPwmPowerSource* <b>yFindPwmPowerSource</b> ( const string& <b>func</b> )
m	YPwmPowerSource* <b>yFindPwmPowerSource</b> ( NSString* <b>func</b> )
pas	function <b>yFindPwmPowerSource</b> ( <b>func</b> : string): TYPwmPowerSource
vb	function <b>yFindPwmPowerSource</b> ( ByVal <b>func</b> As String) As YPwmPowerSource
cs	YPwmPowerSource <b>FindPwmPowerSource</b> ( string <b>func</b> )
java	YPwmPowerSource <b>FindPwmPowerSource</b> ( String <b>func</b> )
py	def <b>FindPwmPowerSource</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que la source de tension soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YPwmPowerSource.isOnline()` pour tester si la source de tension est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence la source de tension sans ambiguïté

### Retourne :

un objet de classe `YPwmPowerSource` qui permet ensuite de contrôler la source de tension.

## YPwmPowerSource.FirstPwmPowerSource() yFirstPwmPowerSource()yFirstPwmPowerSource()

## YPwmPowerSource

Commence l'énumération des Source de tension accessibles par la librairie.

js	function <b>yFirstPwmPowerSource</b> ( )
nodejs	function <b>FirstPwmPowerSource</b> ( )
php	function <b>yFirstPwmPowerSource</b> ( )
cpp	YPwmPowerSource* <b>yFirstPwmPowerSource</b> ( )
m	YPwmPowerSource* <b>yFirstPwmPowerSource</b> ( )
pas	function <b>yFirstPwmPowerSource</b> ( ): TYPwmPowerSource
vb	function <b>yFirstPwmPowerSource</b> ( ) As YPwmPowerSource
cs	YPwmPowerSource <b>FirstPwmPowerSource</b> ( )
java	YPwmPowerSource <b>FirstPwmPowerSource</b> ( )
py	def <b>FirstPwmPowerSource</b> ( )

Utiliser la fonction `YPwmPowerSource.nextPwmPowerSource( )` pour itérer sur les autres Source de tension.

### Retourne :

un pointeur sur un objet `YPwmPowerSource`, correspondant à la première source de tension accessible en ligne, ou `null` si il n'y a pas de Source de tension disponibles.

**pwmpowersource→describe()[pwmpowersource describe]**

**YPwmPowerSource**

Retourne un court texte décrivant de manière non-ambigüe l'instance de la source de tension au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant la source de tension (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**pwmpowersource→get\_advertisedValue()**

**YPwmPowerSource**

**pwmpowersource→advertisedValue()**

**[pwmpowersource advertisedValue]**

Retourne la valeur courante de la source de tension (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YPwmPowerSource <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de la source de tension (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**pwmpowersource→get\_errorMessage()**  
**pwmpowersource→errorMessage()**  
**[pwmpowersource errorMessage]**

**YPwmPowerSource**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la source de tension.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de la source de tension.

**pwmpowersource→get\_errorType()**  
**pwmpowersource→errorType()**

**YPwmPowerSource**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la source de tension.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de la source de tension.

**pwmpowersource**→**get\_friendlyName()****YPwmPowerSource****pwmpowersource**→**friendlyName()[pwmpowersource  
friendlyName]**

Retourne un identifiant global de la source de tension au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de la source de tension si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de la source de tension (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant la source de tension en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.



**pwmpowersource→get\_functionDescriptor()**  
**pwmpowersource→functionDescriptor()**  
**[pwmpowersource functionDescriptor]**

**YPwmPowerSource**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**pwmpowersource**→**get\_functionId()****YPwmPowerSource****pwmpowersource**→**functionId()**[**pwmpowersource**  
**functionId**]

Retourne l'identifiant matériel de la source de tension, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant la source de tension (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

---

**pwmpowersource→get\_hardwareId()****YPwmPowerSource****pwmpowersource→hardwareId()[pwmpowersource hardwareId]**

---

Retourne l'identifiant matériel unique de la source de tension au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la source de tension (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant la source de tension (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**pwmpowersource**→**get\_logicalName()****YPwmPowerSource****pwmpowersource**→**logicalName()**[**pwmpowersource**  
**logicalName**]

Retourne le nom logique de la source de tension.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YPwmPowerSource <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de la source de tension. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**pwmpowersource→get\_module()**  
**pwmpowersource→module()[pwmpowersource module]**

**YPwmPowerSource**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**pwmpowersource**→**get\_module\_async()**  
**pwmpowersource**→**module\_async()**

**YPwmPowerSource**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**pwmpowersource→get\_powerMode()****YPwmPowerSource****pwmpowersource→powerMode()[pwmpowersource  
powerMode]**

Retourne la source de tension utilisé par tous les PWM du même module.

js	function <b>get_powerMode</b> ( )
nodejs	function <b>get_powerMode</b> ( )
php	function <b>get_powerMode</b> ( )
cpp	Y_POWERMODE_enum <b>get_powerMode</b> ( )
m	-(Y_POWERMODE_enum) powerMode
pas	function <b>get_powerMode</b> ( ): Integer
vb	function <b>get_powerMode</b> ( ) As Integer
cs	int <b>get_powerMode</b> ( )
java	int <b>get_powerMode</b> ( )
py	def <b>get_powerMode</b> ( )

**Retourne :**

une valeur parmi Y\_POWERMODE\_USB\_5V, Y\_POWERMODE\_USB\_3V, Y\_POWERMODE\_EXT\_V et Y\_POWERMODE\_OPNDRN représentant la source de tension utilisé par tous les PWM du même module

En cas d'erreur, déclenche une exception ou retourne Y\_POWERMODE\_INVALID.

**pwmpowersource**→**get\_userData()****YPwmPowerSource****pwmpowersource**→**userData()**[**pwmpowersource**  
**userData**]

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.



## **pwmpowersource→isOnline()**[pwmpowersource **isOnline**]

## **YPwmPowerSource**

Vérifie si le module hébergeant la source de tension est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de la source de tension sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

### **Retourne :**

true si la source de tension est joignable, false sinon

**pwmpowersource**→**isOnline\_async()****YPwmPowerSource**

Vérifie si le module hébergeant la source de tension est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de la source de tension sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**pwmpowersource→load()[pwmpowersource load: ]****YPwmPowerSource**

Met en cache les valeurs courantes de la source de tension, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmpowersource**→**load\_async()****YPwmPowerSource**

Met en cache les valeurs courantes de la source de tension, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## **pwpmpowersource→nextPwmPowerSource()** **[pwpmpowersource nextPwmPowerSource]**

**YPwmPowerSource**

Continue l'énumération des Source de tension commencée à l'aide de `yFirstPwmPowerSource()`.

js	function <b>nextPwmPowerSource</b> ( )
nodejs	function <b>nextPwmPowerSource</b> ( )
php	function <b>nextPwmPowerSource</b> ( )
c++	YPwmPowerSource * <b>nextPwmPowerSource</b> ( )
m	-(YPwmPowerSource*) <b>nextPwmPowerSource</b>
pas	function <b>nextPwmPowerSource</b> ( ): TYPwmPowerSource
vb	function <b>nextPwmPowerSource</b> ( ) As YPwmPowerSource
cs	YPwmPowerSource <b>nextPwmPowerSource</b> ( )
java	YPwmPowerSource <b>nextPwmPowerSource</b> ( )
py	def <b>nextPwmPowerSource</b> ( )

### **Retourne :**

un pointeur sur un objet `YPwmPowerSource` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## pwmpowersource→registerValueCallback() [pwmpowersource registerValueCallback: ]

YPwmPowerSource

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YPwmPowerSourceValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YPwmPowerSourceValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYPwmPowerSourceValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**pwmpowersource→set\_logicalName()****YPwmPowerSource****pwmpowersource→setLogicalName()****[pwmpowersource setLogicalName: ]**

Modifie le nom logique de la source de tension.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YPwmPowerSource <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de la source de tension.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**pwmpowersource→set\_powerMode()****YPwmPowerSource****pwmpowersource→setPowerMode()****[pwmpowersource setPowerMode: ]**

Modifie le mode fonctionnement des PWM qui peut sortir du 5 volts isolé issu de l'USB, du 3V isolé issu de l'USB, une tension arbitraire issue de l'alimentation externe.

js	function <b>set_powerMode</b> ( <b>newval</b> )
nodejs	function <b>set_powerMode</b> ( <b>newval</b> )
php	function <b>set_powerMode</b> ( <b>\$newval</b> )
cpp	int <b>set_powerMode</b> ( Y_POWERMODE_enum <b>newval</b> )
m	-(int) <b>setPowerMode</b> : (Y_POWERMODE_enum) <b>newval</b>
pas	function <b>set_powerMode</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_powerMode</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_powerMode</b> ( int <b>newval</b> )
java	int <b>set_powerMode</b> ( int <b>newval</b> )
py	def <b>set_powerMode</b> ( <b>newval</b> )
cmd	YPwmPowerSource <b>target set_powerMode newval</b>

Le PWM peut aussi en mode open drain, dans ce code il tire activement la ligne à zéro volts. Attention ce paramètre est commun à tous les PWM du module, si vous changez le valeur de ce paramètre, tous les PWM situés sur le même module seront affectés. Si vous souhaitez que le changement de ce paramètre soit conservé après un redémarrage du module, n'oubliez pas d'appeler la méthode `saveToFlash()`.

**Paramètres :**

**newval** une valeur parmi Y\_POWERMODE\_USB\_5V, Y\_POWERMODE\_USB\_3V, Y\_POWERMODE\_EXT\_V et Y\_POWERMODE\_OPNDRN représentant le mode fonctionnement des PWM qui peut sortir du 5 volts isolé issu de l'USB, du 3V isolé issu de l'USB, une tension arbitraire issue de l'alimentation externe

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**pwmpowersource→set\_userdata()****YPwmPowerSource****pwmpowersource→setUserData()[pwmpowersource  
setUserData: ]**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## pwmpowersource→wait\_async()

## YPwmPowerSource

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :

## 3.32. Interface du quaternion

La class YQt de la librairie Yoctopuce permet d'accéder à l'estimation de l'orientation tridimensionnelle du Yocto-3D sous forme d'un quaternion. Il n'est en général pas nécessaire d'y accéder directement, la classe YGyro offrant une abstraction de plus haut niveau.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_gyro.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib'); var YGyro = yoctolib.YGyro;</code>
php	<code>require_once('yocto_gyro.php');</code>
c++	<code>#include "yocto_gyro.h"</code>
m	<code>#import "yocto_gyro.h"</code>
pas	<code>uses yocto_gyro;</code>
vb	<code>yocto_gyro.vb</code>
cs	<code>yocto_gyro.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YGyro;</code>
py	<code>from yocto_gyro import *</code>

### Fonction globales

#### yFindQt(func)

Permet de retrouver un élément de quaternion d'après un identifiant donné.

#### yFirstQt()

Commence l'énumération des éléments de quaternion accessibles par la librairie.

### Méthodes des objets YQt

#### qt→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### qt→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'élément de quaternion au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### qt→get\_advertisedValue()

Retourne la valeur courante de l'élément de quaternion (pas plus de 6 caractères).

#### qt→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### qt→get\_currentValue()

Retourne la valeur actuelle de la coordonnée.

#### qt→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'élément de quaternion.

#### qt→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'élément de quaternion.

#### qt→get\_friendlyName()

Retourne un identifiant global de l'élément de quaternion au format `NOM_MODULE . NOM_FONCTION`.

#### qt→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### qt→get\_functionId()

Retourne l'identifiant matériel de l'élément de quaternion, sans référence au module.

**qt→get\_hardwareId()**

Retourne l'identifiant matériel unique de l'élément de quaternion au format `SERIAL.FUNCTIONID`.

**qt→get\_highestValue()**

Retourne la valeur maximale observée pour la coordonnée depuis le démarrage du module.

**qt→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**qt→get\_logicalName()**

Retourne le nom logique de l'élément de quaternion.

**qt→get\_lowestValue()**

Retourne la valeur minimale observée pour la coordonnée depuis le démarrage du module.

**qt→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**qt→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**qt→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**qt→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**qt→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**qt→get\_unit()**

Retourne l'unité dans laquelle la coordonnée est exprimée.

**qt→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**qt→isOnline()**

Vérifie si le module hébergeant l'élément de quaternion est joignable, sans déclencher d'erreur.

**qt→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'élément de quaternion est joignable, sans déclencher d'erreur.

**qt→load(msValidity)**

Met en cache les valeurs courantes de l'élément de quaternion, avec une durée de validité spécifiée.

**qt→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

**qt→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'élément de quaternion, avec une durée de validité spécifiée.

**qt→nextQt()**

Continue l'énumération des éléments de quaternion commencée à l'aide de `yFirstQt()`.

**qt→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**qt→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**qt→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

**qt→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**qt→set\_logicalName(newval)**

Modifie le nom logique de l'élément de quaternion.

**qt→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

**qt→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**qt→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**qt→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userData.

**qt→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YQt.FindQt() yFindQt()yFindQt()

YQt

Permet de retrouver un élément de quaternion d'après un identifiant donné.

js	function <b>yFindQt</b> ( <b>func</b> )
nodejs	function <b>FindQt</b> ( <b>func</b> )
php	function <b>yFindQt</b> ( <b>\$func</b> )
cpp	YQt* <b>yFindQt</b> ( string <b>func</b> )
m	+(YQt*) <b>yFindQt</b> : (NSString*) <b>func</b>
pas	function <b>yFindQt</b> ( <b>func</b> : string): TYQt
vb	function <b>yFindQt</b> ( ByVal <b>func</b> As String) As YQt
cs	YQt <b>FindQt</b> ( string <b>func</b> )
java	YQt <b>FindQt</b> ( String <b>func</b> )
py	def <b>FindQt</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'élément de quaternion soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YQt.isOnline()` pour tester si l'élément de quaternion est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'élément de quaternion sans ambiguïté

### Retourne :

un objet de classe `YQt` qui permet ensuite de contrôler l'élément de quaternion.

## YQt.FirstQt() yFirstQt()yFirstQt()

YQt

Commence l'énumération des éléments de quaternion accessibles par la librairie.

js	function <b>yFirstQt</b> ( )
nodejs	function <b>FirstQt</b> ( )
php	function <b>yFirstQt</b> ( )
cpp	YQt* <b>yFirstQt</b> ( )
m	YQt* <b>yFirstQt</b> ( )
pas	function <b>yFirstQt</b> ( ): TYQt
vb	function <b>yFirstQt</b> ( ) As YQt
cs	YQt <b>FirstQt</b> ( )
java	YQt <b>FirstQt</b> ( )
py	def <b>FirstQt</b> ( )

Utiliser la fonction `YQt.nextQt()` pour itérer sur les autres éléments de quaternion.

### Retourne :

un pointeur sur un objet YQt, correspondant à le premier élément de quaternion accessible en ligne, ou `null` si il n'y a pas de éléments de quaternion disponibles.

**qt→calibrateFromPoints()[qt calibrateFromPoints: ]****YQt**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)
m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues
pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt
vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)
java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)
py def calibrateFromPoints( rawValues, refValues)
cmd YSensor target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

**Paramètres :**

**rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.

**refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**qt→describe()[qt describe]****YQt**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'élément de quaternion au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'élément de quaternion (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**qt→get\_advertisedValue()****YQt****qt→advertisedValue()[qt advertisedValue]**

Retourne la valeur courante de l'élément de quaternion (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YSensor <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'élément de quaternion (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**qt→get\_currentRawValue()****YQt****qt→currentRawValue()[qt currentRawValue]**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YSensor <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**qt→get\_currentValue()****YQt****qt→currentValue()[qt currentValue]**

Retourne la valeur actuelle de la coordonnée.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YSensor <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de la coordonnée

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**qt→get\_errorMessage()****YQt****qt→errorMessage()[qt errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'élément de quaternion.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'élément de quaternion.

**qt→get\_errorType()****YQt****qt→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'élément de quaternion.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'élément de quaternion.

**qt→get\_friendlyName()****YQt****qt→friendlyName()[qt friendlyName]**

Retourne un identifiant global de l'élément de quaternion au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'élément de quaternion si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'élément de quaternion (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'élément de quaternion en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**qt→get\_functionDescriptor()****YQt****qt→functionDescriptor()[qt functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID



**qt→get\_functionId()****YQt****qt→functionId()[qt functionId]**

Retourne l'identifiant matériel de l'élément de quaternion, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'élément de quaternion (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**qt→get\_hardwareId()****YQt****qt→hardwareId()[qt hardwareId]**

Retourne l'identifiant matériel unique de l'élément de quaternion au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'élément de quaternion (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'élément de quaternion (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**qt→get\_highestValue()****YQt****qt→highestValue()[qt highestValue]**

Retourne la valeur maximale observée pour la coordonnée depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YSensor <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la coordonnée depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**qt→get\_logFrequency()**

YQt

**qt→logFrequency()[qt logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YSensor <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**qt→get\_logicalName()****YQt****qt→logicalName()[qt logicalName]**

Retourne le nom logique de l'élément de quaternion.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YSensor <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'élément de quaternion. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**qt→get\_lowestValue()****qt→lowestValue()[qt lowestValue]**

Retourne la valeur minimale observée pour la coordonnée depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YSensor <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la coordonnée depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**qt→get\_module()****YQt****qt→module()[qt module]**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**qt→get\_module\_async()**  
**qt→module\_async()**

YQt

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**qt→get\_recordedData()****YQt****qt→recordedData()[qt recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YSensor <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

- startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.
- endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**qt→get\_reportFrequency()****YQt****qt→reportFrequency()[qt reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YSensor <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**qt→get\_resolution()****YQt****qt→resolution()[qt resolution]**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YSensor <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**qt→get\_unit()****qt→unit()[qt unit]**

Retourne l'unité dans laquelle la coordonnée est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YSensor <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la coordonnée est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**qt→get\_userdata()****YQt****qt→userdata()[qt userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): TObject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

Vérifie si le module hébergeant l'élément de quaternion est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'élément de quaternion sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si l'élément de quaternion est joignable, false sinon

**qt→isOnline\_async()****YQt**

Vérifie si le module hébergeant l'élément de quaternion est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'élément de quaternion sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

Met en cache les valeurs courantes de l'élément de quaternion, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

#### Paramètres :

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**qt→loadCalibrationPoints()**[qt loadCalibrationPoints:  
]

**YQt**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YSensor target loadCalibrationPoints rawValues refValues

```

#### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→load\_async()****YQt**

Met en cache les valeurs courantes de l'élément de quaternion, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**qt→nextQt())[qt nextQt]****YQt**

Continue l'énumération des éléments de quaternion commencée à l'aide de `yFirstQt()`.

<code>js</code>	<code>function nextQt( )</code>
<code>nodejs</code>	<code>function nextQt( )</code>
<code>php</code>	<code>function nextQt( )</code>
<code>cpp</code>	<code>YQt * nextQt( )</code>
<code>m</code>	<code>-(YQt*) nextQt</code>
<code>pas</code>	<code>function nextQt( ): TYQt</code>
<code>vb</code>	<code>function nextQt( ) As YQt</code>
<code>cs</code>	<code>YQt nextQt( )</code>
<code>java</code>	<code>YQt nextQt( )</code>
<code>py</code>	<code>def nextQt( )</code>

**Retourne :**

un pointeur sur un objet `YQt` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## qt→registerTimedReportCallback()[qt registerTimedReportCallback: ]

YQt

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YQtTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YQtTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYQtTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## qt→registerValueCallback()[qt registerValueCallback: ]

YQt

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YQtValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YQtValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYQtValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**qt→set\_highestValue()****qt→setHighestValue()[qt setHighestValue: ]**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YSensor <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→set\_logFrequency()****YQt****qt→setLogFrequency()[qt setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→set\_logicalName()****qt→setLogicalName()[qt setLogicalName: ]**

Modifie le nom logique de l'élément de quaternion.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'élément de quaternion.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**qt→set\_lowestValue()****qt→setLowestValue() [qt setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→set\_reportFrequency()****qt→setReportFrequency() [qt setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→set\_resolution()****YQt****qt→setResolution() [qt setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YSensor <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**qt→set\_userdata()****YQt****qt→setUserData()[qt setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**qt→wait\_async()****YQt**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

### 3.33. Interface de la fonction Horloge Temps Real

La fonction RealTimeClock fourni la date et l'heure courante de manière persistante, même en cas de coupure de courant de plusieurs jours. Elle est le fondement des fonctions de réveil automatique implémentées par le WakeUpScheduler. L'heure courante peut représenter aussi bien une heure locale qu'une heure UTC, mais aucune adaptation automatique n'est fait au changement d'heure été/hiver.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_realtimeclock.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YRealTimeClock = yoctolib.YRealTimeClock;
php	require_once('yocto_realtimeclock.php');
c++	#include "yocto_realtimeclock.h"
m	#import "yocto_realtimeclock.h"
pas	uses yocto_realtimeclock;
vb	yocto_realtimeclock.vb
cs	yocto_realtimeclock.cs
java	import com.yoctopuce.YoctoAPI.YRealTimeClock;
py	from yocto_realtimeclock import *

Fonction globales
<b>yFindRealTimeClock(func)</b> Permet de retrouver une horloge d'après un identifiant donné.
<b>yFirstRealTimeClock()</b> Commence l'énumération des horloge accessibles par la librairie.
Méthodes des objets YRealTimeClock
<b>realtimeclock→describe()</b> Retourne un court texte décrivant de manière non-ambigüe l'instance de l'horloge au format TYPE ( NAME ) = SERIAL . FUNCTIONID.
<b>realtimeclock→get_advertisedValue()</b> Retourne la valeur courante de l'horloge (pas plus de 6 caractères).
<b>realtimeclock→get_dateTime()</b> Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss"
<b>realtimeclock→get_errorMessage()</b> Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.
<b>realtimeclock→get_errorType()</b> Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.
<b>realtimeclock→get_friendlyName()</b> Retourne un identifiant global de l'horloge au format NOM_MODULE . NOM_FONCTION.
<b>realtimeclock→get_functionDescriptor()</b> Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
<b>realtimeclock→get_functionId()</b> Retourne l'identifiant matériel de l'horloge, sans référence au module.
<b>realtimeclock→get_hardwareId()</b> Retourne l'identifiant matériel unique de l'horloge au format SERIAL . FUNCTIONID.
<b>realtimeclock→get_logicalName()</b> Retourne le nom logique de l'horloge.
<b>realtimeclock→get_module()</b>

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`realtimeclock→get_module_async(callback, context)`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`realtimeclock→get_timeSet()`**

Retourne vrai si l'horloge à été mise à l'heure, sinon faux.

**`realtimeclock→get_unixTime()`**

Retourne l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

**`realtimeclock→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`realtimeclock→get_utcOffset()`**

Retourne le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

**`realtimeclock→isOnline()`**

Vérifie si le module hébergeant l'horloge est joignable, sans déclencher d'erreur.

**`realtimeclock→isOnline_async(callback, context)`**

Vérifie si le module hébergeant l'horloge est joignable, sans déclencher d'erreur.

**`realtimeclock→load(msValidity)`**

Met en cache les valeurs courantes de l'horloge, avec une durée de validité spécifiée.

**`realtimeclock→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes de l'horloge, avec une durée de validité spécifiée.

**`realtimeclock→nextRealTimeClock()`**

Continue l'énumération des horloge commencée à l'aide de `yFirstRealTimeClock()`.

**`realtimeclock→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`realtimeclock→set_logicalName(newval)`**

Modifie le nom logique de l'horloge.

**`realtimeclock→set_unixTime(newval)`**

Modifie l'heure courante.

**`realtimeclock→set_userData(data)`**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**`realtimeclock→set_utcOffset(newval)`**

Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

**`realtimeclock→wait_async(callback, context)`**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YRealTimeClock.FindRealTimeClock() yFindRealTimeClock()yFindRealTimeClock()

## YRealTimeClock

Permet de retrouver une horloge d'après un identifiant donné.

js	function <b>yFindRealTimeClock</b> ( <b>func</b> )
nodejs	function <b>FindRealTimeClock</b> ( <b>func</b> )
php	function <b>yFindRealTimeClock</b> ( <b>\$func</b> )
cpp	YRealTimeClock* <b>yFindRealTimeClock</b> ( const string& <b>func</b> )
m	YRealTimeClock* <b>yFindRealTimeClock</b> ( NSString* <b>func</b> )
pas	function <b>yFindRealTimeClock</b> ( <b>func</b> : string): TYRealTimeClock
vb	function <b>yFindRealTimeClock</b> ( ByVal <b>func</b> As String) As YRealTimeClock
cs	YRealTimeClock <b>FindRealTimeClock</b> ( string <b>func</b> )
java	YRealTimeClock <b>FindRealTimeClock</b> ( String <b>func</b> )
py	def <b>FindRealTimeClock</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'horloge soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YRealTimeClock.isOnline()` pour tester si l'horloge est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'horloge sans ambiguïté

### Retourne :

un objet de classe `YRealTimeClock` qui permet ensuite de contrôler l'horloge.



## YRealTimeClock.FirstRealTimeClock() yFirstRealTimeClock()yFirstRealTimeClock()

## YRealTimeClock

Commence l'énumération des horloge accessibles par la librairie.

js	function <b>yFirstRealTimeClock</b> ( )
nodejs	function <b>FirstRealTimeClock</b> ( )
php	function <b>yFirstRealTimeClock</b> ( )
cpp	YRealTimeClock* <b>yFirstRealTimeClock</b> ( )
m	YRealTimeClock* <b>yFirstRealTimeClock</b> ( )
pas	function <b>yFirstRealTimeClock</b> ( ): TYRealTimeClock
vb	function <b>yFirstRealTimeClock</b> ( ) As YRealTimeClock
cs	YRealTimeClock <b>FirstRealTimeClock</b> ( )
java	YRealTimeClock <b>FirstRealTimeClock</b> ( )
py	def <b>FirstRealTimeClock</b> ( )

Utiliser la fonction `YRealTimeClock.nextRealTimeClock( )` pour itérer sur les autres horloge.

### Retourne :

un pointeur sur un objet `YRealTimeClock`, correspondant à la première horloge accessible en ligne, ou `null` si il n'y a pas de horloge disponibles.

**realtimeclock→describe()[realtimeclock describe]****YRealTimeClock**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'horloge au format  
 TYPE (NAME) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.relay1)=unresolved si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant l'horloge (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

**realtimeclock→get\_advertisedValue()****YRealTimeClock****realtimeclock→advertisedValue()[realtimeclock  
advertisedValue]**

Retourne la valeur courante de l'horloge (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YRealTimeClock <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'horloge (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**realtimeclock→get\_dateTime()****YRealTimeClock****realtimeclock→dateTime()[realtimeclock dateTime]**

Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss"

js	function <b>get_dateTime</b> ( )
nodejs	function <b>get_dateTime</b> ( )
php	function <b>get_dateTime</b> ( )
cpp	string <b>get_dateTime</b> ( )
m	-(NSString*) <b>dateTime</b>
pas	function <b>get_dateTime</b> ( ): string
vb	function <b>get_dateTime</b> ( ) As String
cs	string <b>get_dateTime</b> ( )
java	String <b>get_dateTime</b> ( )
py	def <b>get_dateTime</b> ( )

**Retourne :**

une chaîne de caractères représentant l'heure courante au format "AAAA/MM/JJ hh:mm:ss"

En cas d'erreur, déclenche une exception ou retourne Y\_DATETIME\_INVALID.

**realtimeclock→get\_errorMessage()**  
**realtimeclock→errorMessage()[realtimeclock**  
**errorMessage]**

**YRealTimeClock**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'horloge.

**realtimeclock→get\_errorType()**  
**realtimeclock→errorType()****YRealTimeClock**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'horloge.

**realtimeclock→get\_friendlyName()****YRealTimeClock****realtimeclock→friendlyName()[realtimeclock  
friendlyName]**

Retourne un identifiant global de l'horloge au format `NOM_MODULE . NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'horloge si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'horloge (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'horloge en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**realtimeclock→get\_functionDescriptor()****YRealTimeClock****realtimeclock→functionDescriptor()[realtimeclock  
functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID



**realtimeclock→get\_functionId()****YRealTimeClock****realtimeclock→functionId()[realtimeclock functionId]**

Retourne l'identifiant matériel de l'horloge, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'horloge (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**realtimeclock→get\_hardwareId()**

**YRealTimeClock**

**realtimeclock→hardwareId()[realtimeclock  
hardwareId]**

Retourne l'identifiant matériel unique de l'horloge au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'horloge (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'horloge (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**realtimeclock→get\_logicalName()**  
**realtimeclock→logicalName()[realtimeclock**  
**logicalName]**

**YRealTimeClock**

Retourne le nom logique de l'horloge.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YRealTimeClock <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'horloge. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**realtimeclock→get\_module()****YRealTimeClock****realtimeclock→module())[realtimeclock module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**realtimeclock→get\_module\_async()**  
**realtimeclock→module\_async()**

**YRealTimeClock**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**realtimeclock→get\_timeSet()****YRealTimeClock****realtimeclock→timeSet()[realtimeclock timeSet]**

Retourne vrai si l'horloge à été mise à l'heure, sinon faux.

js	function <b>get_timeSet</b> ( )
nodejs	function <b>get_timeSet</b> ( )
php	function <b>get_timeSet</b> ( )
cpp	Y_TIMESET_enum <b>get_timeSet</b> ( )
m	-(Y_TIMESET_enum) timeSet
pas	function <b>get_timeSet</b> ( ): Integer
vb	function <b>get_timeSet</b> ( ) As Integer
cs	int <b>get_timeSet</b> ( )
java	int <b>get_timeSet</b> ( )
py	def <b>get_timeSet</b> ( )
cmd	YRealTimeClock <b>target</b> <b>get_timeSet</b>

**Retourne :**

soit Y\_TIMESET\_FALSE, soit Y\_TIMESET\_TRUE, selon vrai si l'horloge à été mise à l'heure, sinon faux

En cas d'erreur, déclenche une exception ou retourne Y\_TIMESET\_INVALID.

**realtimeclock→get\_unixTime()****YRealTimeClock****realtimeclock→unixTime()[realtimeclock unixTime]**

Retourne l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

js	function <b>get_unixTime</b> ( )
nodejs	function <b>get_unixTime</b> ( )
php	function <b>get_unixTime</b> ( )
cpp	s64 <b>get_unixTime</b> ( )
m	-(s64) unixTime
pas	function <b>get_unixTime</b> ( ): int64
vb	function <b>get_unixTime</b> ( ) As Long
cs	long <b>get_unixTime</b> ( )
java	long <b>get_unixTime</b> ( )
py	def <b>get_unixTime</b> ( )
cmd	YRealTimeClock <b>target</b> <b>get_unixTime</b>

**Retourne :**

un entier représentant l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970)

En cas d'erreur, déclenche une exception ou retourne Y\_UNIXTIME\_INVALID.

**realtimeclock→get\_userdata()****YRealTimeClock****realtimeclock→userdata()[realtimeclock userdata]**

Retourne le contenu de l'attribut `userdata`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) <b>userdata</b>
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.



**realtimeclock→get\_utcOffset()****YRealTimeClock****realtimeclock→utcOffset()[realtimeclock utcOffset]**

Retourne le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

js	function <b>get_utcOffset</b> ( )
nodejs	function <b>get_utcOffset</b> ( )
php	function <b>get_utcOffset</b> ( )
cpp	int <b>get_utcOffset</b> ( )
m	-(int) utcOffset
pas	function <b>get_utcOffset</b> ( ): LongInt
vb	function <b>get_utcOffset</b> ( ) As Integer
cs	int <b>get_utcOffset</b> ( )
java	int <b>get_utcOffset</b> ( )
py	def <b>get_utcOffset</b> ( )
cmd	YRealTimeClock <b>target</b> <b>get_utcOffset</b>

**Retourne :**

un entier représentant le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone)

En cas d'erreur, déclenche une exception ou retourne Y\_UTCOffset\_INVALID.

**realtimeclock→isOnline()[realtimeclock isOnline]****YRealTimeClock**

Vérifie si le module hébergeant l'horloge est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'horloge sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si l'horloge est joignable, false sinon

---

**realtimeclock→isOnline\_async()****YRealTimeClock**

---

Vérifie si le module hébergeant l'horloge est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'horloge sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**realtimeclock→load()[realtimeclock load: ]****YRealTimeClock**

Met en cache les valeurs courantes de l'horloge, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**realtimeclock→load\_async()****YRealTimeClock**

Met en cache les valeurs courantes de l'horloge, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**realtimeclock→nextRealTimeClock()[realtimeclock  
nextRealTimeClock]****YRealTimeClock**

Continue l'énumération des horloge commencée à l'aide de `yFirstRealTimeClock()`.

js	function <b>nextRealTimeClock</b> ( )
nodejs	function <b>nextRealTimeClock</b> ( )
php	function <b>nextRealTimeClock</b> ( )
cpp	YRealTimeClock * <b>nextRealTimeClock</b> ( )
m	-(YRealTimeClock*) <b>nextRealTimeClock</b>
pas	function <b>nextRealTimeClock</b> ( ): TYRealTimeClock
vb	function <b>nextRealTimeClock</b> ( ) As YRealTimeClock
cs	YRealTimeClock <b>nextRealTimeClock</b> ( )
java	YRealTimeClock <b>nextRealTimeClock</b> ( )
py	def <b>nextRealTimeClock</b> ( )

**Retourne :**

un pointeur sur un objet `YRealTimeClock` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## realtimeclock→registerValueCallback()[realtimeclock registerValueCallback: ]

## YRealTimeClock

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YRealTimeClockValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YRealTimeClockValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYRealTimeClockValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**realtimeclock→set\_logicalName()****YRealTimeClock****realtimeclock→setLogicalName()[realtimeclock  
setLogicalName: ]**

Modifie le nom logique de l'horloge.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YRealTimeClock <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'horloge.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**realtimeclock→set\_unixTime()**  
**realtimeclock→setUnixTime())[realtimeclock**  
**setUnixTime: ]**

**YRealTimeClock**

Modifie l'heure courante.

js	function <b>set_unixTime</b> ( <b>newval</b> )
nodejs	function <b>set_unixTime</b> ( <b>newval</b> )
php	function <b>set_unixTime</b> ( <b>\$newval</b> )
cpp	int <b>set_unixTime</b> ( s64 <b>newval</b> )
m	-(int) setUnixTime : (s64) <b>newval</b>
pas	function <b>set_unixTime</b> ( <b>newval</b> : int64): integer
vb	function <b>set_unixTime</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_unixTime</b> ( long <b>newval</b> )
java	int <b>set_unixTime</b> ( long <b>newval</b> )
py	def <b>set_unixTime</b> ( <b>newval</b> )
cmd	YRealTimeClock <b>target</b> <b>set_unixTime</b> <b>newval</b>

L'heure est passée au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970). Si l'heure UTC est connue, l'attribut utcOffset sera automatiquement ajusté en fonction de l'heure configurée.

#### Paramètres :

**newval** un entier représentant l'heure courante

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**realtimeclock→set\_userdata()****YRealTimeClock****realtimeclock→setUserData()[realtimeclock  
setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : TObject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**realtimeclock→set\_utcOffset()****YRealTimeClock****realtimeclock→setUtcOffset())[realtimeclock  
setUtcOffset: ]**

Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

js	function <b>set_utcOffset</b> ( <b>newval</b> )
nodejs	function <b>set_utcOffset</b> ( <b>newval</b> )
php	function <b>set_utcOffset</b> ( <b>\$newval</b> )
cpp	int <b>set_utcOffset</b> ( int <b>newval</b> )
m	-(int) setUtcOffset : (int) <b>newval</b>
pas	function <b>set_utcOffset</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_utcOffset</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_utcOffset</b> ( int <b>newval</b> )
java	int <b>set_utcOffset</b> ( int <b>newval</b> )
py	def <b>set_utcOffset</b> ( <b>newval</b> )
cmd	YRealTimeClock <b>target</b> <b>set_utcOffset</b> <b>newval</b>

Le décalage est automatiquement arrondi au quart d'heure le plus proche. Si l'heure UTC est connue, l'heure courante sera automatiquement adaptée en fonction du décalage choisi.

**Paramètres :**

**newval** un entier représentant le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## realtimeclock→wait\_async()

## YRealTimeClock

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :

## 3.34. Configuration du référentiel

Cette classe permet de configurer l'orientation dans laquelle le Yocto-3D est utilisé, afin que les fonctions d'orientation relatives au plan de la surface terrestre utilisent le référentiel approprié. La classe offre aussi un processus de recalibration tridimensionnel des capteurs, permettant de compenser les variations locales de l'accélération terrestre et d'améliorer la précision des capteurs d'inclinaisons.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_refframe.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib'); var YRefFrame = yoctolib.YRefFrame;</code>
php	<code>require_once('yocto_refframe.php');</code>
c++	<code>#include "yocto_refframe.h"</code>
m	<code>#import "yocto_refframe.h"</code>
pas	<code>uses yocto_refframe;</code>
vb	<code>yocto_refframe.vb</code>
cs	<code>yocto_refframe.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YRefFrame;</code>
py	<code>from yocto_refframe import *</code>

### Fonction globales

#### **yFindRefFrame(func)**

Permet de retrouver un référentiel d'après un identifiant donné.

#### **yFirstRefFrame()**

Commence l'énumération des référentiels accessibles par la librairie.

### Méthodes des objets YRefFrame

#### **refframe→cancel3DCalibration()**

Annule la calibration tridimensionnelle en cours, et rétabli les réglages normaux.

#### **refframe→describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance du référentiel au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

#### **refframe→get\_3DCalibrationHint()**

Retourne les instructions à suivre pour procéder à la calibration tridimensionnelle initiée avec la méthode `start3DCalibration`.

#### **refframe→get\_3DCalibrationLogMsg()**

Retourne le dernier message de log produit par le processus de calibration.

#### **refframe→get\_3DCalibrationProgress()**

Retourne l'avancement global du processus de calibration tridimensionnelle initié avec la méthode `start3DCalibration`.

#### **refframe→get\_3DCalibrationStage()**

Retourne l'index de l'étape courante de la calibration initiée avec la méthode `start3DCalibration`.

#### **refframe→get\_3DCalibrationStageProgress()**

Retourne l'avancement de l'étape courante de la calibration initiée avec la méthode `start3DCalibration`.

#### **refframe→get\_advertisedValue()**

Retourne la valeur courante du référentiel (pas plus de 6 caractères).

#### **refframe→get\_bearing()**

	Retourne le cap de référence utilisé par le compas.
<b>refframe</b> → <b>get_errorMessage()</b>	Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.
<b>refframe</b> → <b>get_errorType()</b>	Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.
<b>refframe</b> → <b>get_friendlyName()</b>	Retourne un identifiant global du référentiel au format <code>NOM_MODULE . NOM_FONCTION</code> .
<b>refframe</b> → <b>get_functionDescriptor()</b>	Retourne un identifiant unique de type <code>YFUN_DESCR</code> correspondant à la fonction.
<b>refframe</b> → <b>get_functionId()</b>	Retourne l'identifiant matériel du référentiel, sans référence au module.
<b>refframe</b> → <b>get_hardwareId()</b>	Retourne l'identifiant matériel unique du référentiel au format <code>SERIAL . FUNCTIONID</code> .
<b>refframe</b> → <b>get_logicalName()</b>	Retourne le nom logique du référentiel.
<b>refframe</b> → <b>get_module()</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>refframe</b> → <b>get_module_async(callback, context)</b>	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b>refframe</b> → <b>get_mountOrientation()</b>	Retourne l'orientation à l'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.
<b>refframe</b> → <b>get_mountPosition()</b>	Retourne la position d'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.
<b>refframe</b> → <b>get_userData()</b>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<b>refframe</b> → <b>isOnline()</b>	Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.
<b>refframe</b> → <b>isOnline_async(callback, context)</b>	Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.
<b>refframe</b> → <b>load(msValidity)</b>	Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.
<b>refframe</b> → <b>load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.
<b>refframe</b> → <b>more3DCalibration()</b>	Continue le processus de calibration tridimensionnelle des capteurs initié avec la méthode <code>start3DCalibration</code> .
<b>refframe</b> → <b>nextRefFrame()</b>	Continue l'énumération des référentiels commencée à l'aide de <code>yFirstRefFrame()</code> .
<b>refframe</b> → <b>registerValueCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b>refframe</b> → <b>save3DCalibration()</b>	Applique les paramètres de calibration tridimensionnelle précédemment calculés.
<b>refframe</b> → <b>set_bearing(newval)</b>	

Modifie le cap de référence utilisé par le compas.

**refframe**→**set\_logicalName**(**newval**)

Modifie le nom logique du référentiel.

**refframe**→**set\_mountPosition**(**position**, **orientation**)

Modifie le référentiel de la boussole et des inclinomètres.

**refframe**→**set\_userData**(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**refframe**→**start3DCalibration**()

Initie le processus de calibration tridimensionnelle des capteurs.

**refframe**→**wait\_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YRefFrame.FindRefFrame() yFindRefFrame()yFindRefFrame()

## YRefFrame

Permet de retrouver un référentiel d'après un identifiant donné.

js	function <b>yFindRefFrame</b> ( <b>func</b> )
nodejs	function <b>FindRefFrame</b> ( <b>func</b> )
php	function <b>yFindRefFrame</b> ( <b>\$func</b> )
cpp	YRefFrame* <b>yFindRefFrame</b> ( const string& <b>func</b> )
m	YRefFrame* <b>yFindRefFrame</b> ( NSString* <b>func</b> )
pas	function <b>yFindRefFrame</b> ( <b>func</b> : string): TYRefFrame
vb	function <b>yFindRefFrame</b> ( ByVal <b>func</b> As String) As YRefFrame
cs	YRefFrame <b>FindRefFrame</b> ( string <b>func</b> )
java	YRefFrame <b>FindRefFrame</b> ( String <b>func</b> )
py	def <b>FindRefFrame</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le référentiel soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YRefFrame.isOnline()` pour tester si le référentiel est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le référentiel sans ambiguïté

### Retourne :

un objet de classe `YRefFrame` qui permet ensuite de contrôler le référentiel.



## **YRefFrame.FirstRefFrame() yFirstRefFrame()yFirstRefFrame()**

## **YRefFrame**

Commence l'énumération des référentiels accessibles par la librairie.

js	function <b>yFirstRefFrame</b> ( )
nodejs	function <b>FirstRefFrame</b> ( )
php	function <b>yFirstRefFrame</b> ( )
cpp	YRefFrame* <b>yFirstRefFrame</b> ( )
m	YRefFrame* <b>yFirstRefFrame</b> ( )
pas	function <b>yFirstRefFrame</b> ( ): TYRefFrame
vb	function <b>yFirstRefFrame</b> ( ) As YRefFrame
cs	YRefFrame <b>FirstRefFrame</b> ( )
java	YRefFrame <b>FirstRefFrame</b> ( )
py	def <b>FirstRefFrame</b> ( )

Utiliser la fonction `YRefFrame.nextRefFrame( )` pour itérer sur les autres référentiels.

### **Retourne :**

un pointeur sur un objet `YRefFrame`, correspondant à le premier référentiel accessible en ligne, ou `null` si il n'y a pas de référentiels disponibles.

## refframe→cancel3DCalibration()[refframe cancel3DCalibration]

YRefFrame

Annule la calibration tridimensionnelle en cours, et rétabli les réglages normaux.

js	function <b>cancel3DCalibration</b> ( )
nodejs	function <b>cancel3DCalibration</b> ( )
php	function <b>cancel3DCalibration</b> ( )
cpp	int <b>cancel3DCalibration</b> ( )
m	-(int) <b>cancel3DCalibration</b>
pas	function <b>cancel3DCalibration</b> ( ): LongInt
vb	function <b>cancel3DCalibration</b> ( ) As Integer
cs	int <b>cancel3DCalibration</b> ( )
java	int <b>cancel3DCalibration</b> ( )
py	def <b>cancel3DCalibration</b> ( )
cmd	YRefFrame <b>target</b> <b>cancel3DCalibration</b>

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→describe()[refframe describe]****YRefFrame**

Retourne un court texte décrivant de manière non-ambigüe l'instance du référentiel au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le référentiel (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## refframe→get\_3DCalibrationHint() refframe→3DCalibrationHint()[refframe 3DCalibrationHint]

YRefFrame

Retourne les instructions à suivre pour procéder à la calibration tridimensionnelle initiée avec la méthode `start3DCalibration`.

js	function <b>get_3DCalibrationHint</b> ( )
nodejs	function <b>get_3DCalibrationHint</b> ( )
php	function <b>get_3DCalibrationHint</b> ( )
cpp	string <b>get_3DCalibrationHint</b> ( )
m	-(NSString*) 3DCalibrationHint
pas	function <b>get_3DCalibrationHint</b> ( ): string
vb	function <b>get_3DCalibrationHint</b> ( ) As String
cs	string <b>get_3DCalibrationHint</b> ( )
java	String <b>get_3DCalibrationHint</b> ( )
py	def <b>get_3DCalibrationHint</b> ( )
cmd	YRefFrame <b>target</b> <b>get_3DCalibrationHint</b>

**Retourne :**

une chaîne de caractères.

**refframe→get\_3DCalibrationLogMsg()**  
**refframe→3DCalibrationLogMsg()[refframe**  
**3DCalibrationLogMsg]**

**YRefFrame**

Retourne le dernier message de log produit par le processus de calibration.

js	function <b>get_3DCalibrationLogMsg</b> ( )
nodejs	function <b>get_3DCalibrationLogMsg</b> ( )
php	function <b>get_3DCalibrationLogMsg</b> ( )
cpp	string <b>get_3DCalibrationLogMsg</b> ( )
m	-(NSString*) 3DCalibrationLogMsg
pas	function <b>get_3DCalibrationLogMsg</b> ( ): string
vb	function <b>get_3DCalibrationLogMsg</b> ( ) As String
cs	string <b>get_3DCalibrationLogMsg</b> ( )
java	String <b>get_3DCalibrationLogMsg</b> ( )
py	def <b>get_3DCalibrationLogMsg</b> ( )
cmd	YRefFrame <b>target</b> <b>get_3DCalibrationLogMsg</b>

Si aucun nouveau message n'est disponible, retourne une chaîne vide.

**Retourne :**

une chaîne de caractères.

**refframe→get\_3DCalibrationProgress()**  
**refframe→3DCalibrationProgress()[refframe**  
**3DCalibrationProgress]**

**YRefFrame**

Retourne l'avancement global du processus de calibration tridimensionnelle initié avec la méthode `start3DCalibration`.

js	function <b>get_3DCalibrationProgress</b> ( )
nodejs	function <b>get_3DCalibrationProgress</b> ( )
php	function <b>get_3DCalibrationProgress</b> ( )
cpp	int <b>get_3DCalibrationProgress</b> ( )
m	-(int) 3DCalibrationProgress
pas	function <b>get_3DCalibrationProgress</b> ( ): LongInt
vb	function <b>get_3DCalibrationProgress</b> ( ) As Integer
cs	int <b>get_3DCalibrationProgress</b> ( )
java	int <b>get_3DCalibrationProgress</b> ( )
py	def <b>get_3DCalibrationProgress</b> ( )
cmd	YRefFrame <b>target</b> <b>get_3DCalibrationProgress</b>

**Retourne :**

une nombre entier entre 0 (pas commencé) et 100 (terminé).

**refframe→get\_3DCalibrationStage()**  
**refframe→3DCalibrationStage()[refframe**  
**3DCalibrationStage]**

**YRefFrame**

Retourne l'index de l'étape courante de la calibration initiée avec la méthode start3DCalibration.

js	function <b>get_3DCalibrationStage</b> ( )
nodejs	function <b>get_3DCalibrationStage</b> ( )
php	function <b>get_3DCalibrationStage</b> ( )
cpp	int <b>get_3DCalibrationStage</b> ( )
m	-(int) 3DCalibrationStage
pas	function <b>get_3DCalibrationStage</b> ( ): LongInt
vb	function <b>get_3DCalibrationStage</b> ( ) As Integer
cs	int <b>get_3DCalibrationStage</b> ( )
java	int <b>get_3DCalibrationStage</b> ( )
py	def <b>get_3DCalibrationStage</b> ( )
cmd	YRefFrame <b>target</b> <b>get_3DCalibrationStage</b>

**Retourne :**

une nombre entier, croissant au fur et à mesure de la complétion des étapes.

**refframe→get\_3DCalibrationStageProgress()**  
**refframe→3DCalibrationStageProgress()[refframe**  
**3DCalibrationStageProgress]**

**YRefFrame**

Retourne l'avancement de l'étape courante de la calibration initiée avec la méthode `start3DCalibration`.

js	function <b>get_3DCalibrationStageProgress</b> ( )
nodejs	function <b>get_3DCalibrationStageProgress</b> ( )
php	function <b>get_3DCalibrationStageProgress</b> ( )
cpp	int <b>get_3DCalibrationStageProgress</b> ( )
m	-(int) 3DCalibrationStageProgress
pas	function <b>get_3DCalibrationStageProgress</b> ( ): LongInt
vb	function <b>get_3DCalibrationStageProgress</b> ( ) As Integer
cs	int <b>get_3DCalibrationStageProgress</b> ( )
java	int <b>get_3DCalibrationStageProgress</b> ( )
py	def <b>get_3DCalibrationStageProgress</b> ( )
cmd	YRefFrame <b>target</b> <b>get_3DCalibrationStageProgress</b>

**Retourne :**

une nombre entier entre 0 (pas commencé) et 100 (terminé).



**refframe→get\_advertisedValue()**  
**refframe→advertisedValue()[refframe**  
**advertisedValue]**

**YRefFrame**

Retourne la valeur courante du référentiel (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YRefFrame <b>target</b> <b>get_advertisedValue</b>

#### Retourne :

une chaîne de caractères représentant la valeur courante du référentiel (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

refframe→**get\_bearing()**

YRefFrame

refframe→**bearing()**[refframe bearing]

Retourne le cap de référence utilisé par le compas.

js	function <b>get_bearing</b> ( )
nodejs	function <b>get_bearing</b> ( )
php	function <b>get_bearing</b> ( )
cpp	double <b>get_bearing</b> ( )
m	-(double) bearing
pas	function <b>get_bearing</b> ( ): double
vb	function <b>get_bearing</b> ( ) As Double
cs	double <b>get_bearing</b> ( )
java	double <b>get_bearing</b> ( )
py	def <b>get_bearing</b> ( )
cmd	YRefFrame <b>target</b> <b>get_bearing</b>

Le cap relatif indiqué par le compas est la différence entre le Nord magnétique mesuré et le cap de référence spécifié ici.

**Retourne :**

une valeur numérique représentant le cap de référence utilisé par le compas

En cas d'erreur, déclenche une exception ou retourne Y\_BEARING\_INVALID.

**refframe→get\_errorMessage()****YRefFrame****refframe→errorMessage()[refframe errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du référentiel.

refframe→**get\_errorType()**

YRefFrame

refframe→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du référentiel.

**refframe→get\_friendlyName()****YRefFrame****refframe→friendlyName()[refframe friendlyName]**

Retourne un identifiant global du référentiel au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du référentiel si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du référentiel (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le référentiel en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**refframe→get\_functionDescriptor()**  
**refframe→functionDescriptor()[refframe**  
**functionDescriptor]**

YRefFrame

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**refframe→get\_functionId()****YRefFrame****refframe→functionId()[refframe functionId]**

Retourne l'identifiant matériel du référentiel, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le référentiel (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**refframe→get\_hardwareId()****YRefFrame****refframe→hardwareId()[refframe hardwareId]**

Retourne l'identifiant matériel unique du référentiel au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du référentiel (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le référentiel (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**refframe→get\_logicalName()****YRefFrame****refframe→logicalName()[refframe logicalName]**

Retourne le nom logique du référentiel.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YRefFrame <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du référentiel. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**refframe→get\_module()****YRefFrame****refframe→module()[refframe module]**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

---

**refframe→get\_module\_async()****YRefFrame****refframe→module\_async()**

---

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**refframe→get\_mountOrientation()**  
**refframe→mountOrientation()[refframe**  
**mountOrientation]**

YRefFrame

Retourne l'orientation à l'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.

js	function <b>get_mountOrientation</b> ( )
nodejs	function <b>get_mountOrientation</b> ( )
php	function <b>get_mountOrientation</b> ( )
cpp	Y_MOUNTORIENTATION <b>get_mountOrientation</b> ( )
m	-(Y_MOUNTORIENTATION) mountOrientation
pas	function <b>get_mountOrientation</b> ( ): TY_MOUNTORIENTATION
vb	function <b>get_mountOrientation</b> ( ) As Y_MOUNTORIENTATION
cs	MOUNTORIENTATION <b>get_mountOrientation</b> ( )
java	MOUNTORIENTATION <b>get_mountOrientation</b> ( )
py	def <b>get_mountOrientation</b> ( )
cmd	YRefFrame <b>target</b> <b>get_mountOrientation</b>

#### Retourne :

une valeur parmi l'énumération Y\_MOUNTORIENTATION (Y\_MOUNTORIENTATION\_TWELVE, Y\_MOUNTORIENTATION\_THREE, Y\_MOUNTORIENTATION\_SIX, Y\_MOUNTORIENTATION\_NINE) correspondant à la l'orientation de la flèche "X" sur le module par rapport à un cadran d'horloge vu par un observateur au centre de la boîte. Sur la face BOTTOM le 12h pointe vers l'avant, tandis que sur la face TOP le 12h pointe vers l'arrière.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→get\_mountPosition()****YRefFrame****refframe→mountPosition()[refframe mountPosition]**

Retourne la position d'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.

js	function <b>get_mountPosition</b> ( )
nodejs	function <b>get_mountPosition</b> ( )
php	function <b>get_mountPosition</b> ( )
cpp	Y_MOUNTPOSITION <b>get_mountPosition</b> ( )
m	-(Y_MOUNTPOSITION) mountPosition
pas	function <b>get_mountPosition</b> ( ): TY_MOUNTPOSITION
vb	function <b>get_mountPosition</b> ( ) As Y_MOUNTPOSITION
cs	MOUNTPOSITION <b>get_mountPosition</b> ( )
java	MOUNTPOSITION <b>get_mountPosition</b> ( )
py	def <b>get_mountPosition</b> ( )
cmd	YRefFrame <b>target</b> <b>get_mountPosition</b>

**Retourne :**

une valeur parmi l'énumération Y\_MOUNTPOSITION (Y\_MOUNTPOSITION\_BOTTOM, Y\_MOUNTPOSITION\_TOP, Y\_MOUNTPOSITION\_FRONT, Y\_MOUNTPOSITION\_RIGHT, Y\_MOUNTPOSITION\_REAR, Y\_MOUNTPOSITION\_LEFT), correspondant à l'installation dans une boîte, sur l'une des six faces

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→get\_userdata()**

**YRefFrame**

**refframe→userdata()[refframe userData]**

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**refframe→isOnline()[refframe isOnline]****YRefFrame**

Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du référentiel sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le référentiel est joignable, `false` sinon

Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du référentiel sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**refframe→load()[refframe load: ]****YRefFrame**

Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## refframe→more3DCalibration()[refframe more3DCalibration]

YRefFrame

Continue le processus de calibration tridimensionnelle des capteurs initié avec la méthode `start3DCalibration`.

js	function <b>more3DCalibration</b> ( )
nodejs	function <b>more3DCalibration</b> ( )
php	function <b>more3DCalibration</b> ( )
cpp	int <b>more3DCalibration</b> ( )
m	-(int) <b>more3DCalibration</b>
pas	function <b>more3DCalibration</b> ( ): LongInt
vb	function <b>more3DCalibration</b> ( ) As Integer
cs	int <b>more3DCalibration</b> ( )
java	int <b>more3DCalibration</b> ( )
py	def <b>more3DCalibration</b> ( )
cmd	YRefFrame <b>target</b> <b>more3DCalibration</b>

Cette méthode doit être appelée environ 5 fois par secondes après avoir positionné le module selon les instructions fournies par la méthode `get_3DCalibrationHint` (les instructions changent pendant la procédure de calibration). En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→nextRefFrame()[refframe nextRefFrame]****YRefFrame**

Continue l'énumération des référentiels commencée à l'aide de `yFirstRefFrame()`.

js	function <b>nextRefFrame</b> ( )
nodejs	function <b>nextRefFrame</b> ( )
php	function <b>nextRefFrame</b> ( )
cpp	YRefFrame * <b>nextRefFrame</b> ( )
m	-(YRefFrame*) <b>nextRefFrame</b>
pas	function <b>nextRefFrame</b> ( ): TYRefFrame
vb	function <b>nextRefFrame</b> ( ) As YRefFrame
cs	YRefFrame <b>nextRefFrame</b> ( )
java	YRefFrame <b>nextRefFrame</b> ( )
py	def <b>nextRefFrame</b> ( )

**Retourne :**

un pointeur sur un objet `YRefFrame` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## refframe→registerValueCallback()[refframe registerValueCallback: ]

## YRefFrame

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YRefFrameValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YRefFrameValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYRefFrameValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**refframe→save3DCalibration()[refframe  
save3DCalibration]****YRefFrame**

Applique les paramètres de calibration tridimensionnelle précédemment calculés.

js	function <b>save3DCalibration</b> ( )
nodejs	function <b>save3DCalibration</b> ( )
php	function <b>save3DCalibration</b> ( )
cpp	int <b>save3DCalibration</b> ( )
m	-(int) <b>save3DCalibration</b>
pas	function <b>save3DCalibration</b> ( ): LongInt
vb	function <b>save3DCalibration</b> ( ) As Integer
cs	int <b>save3DCalibration</b> ( )
java	int <b>save3DCalibration</b> ( )
py	def <b>save3DCalibration</b> ( )
cmd	YRefFrame <b>target save3DCalibration</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé après le redémarrage du module. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→set\_bearing()****YRefFrame****refframe→setBearing()[refframe setBearing: ]**

Modifie le cap de référence utilisé par le compas.

js	function <b>set_bearing</b> ( <b>newval</b> )
nodejs	function <b>set_bearing</b> ( <b>newval</b> )
php	function <b>set_bearing</b> ( <b>\$newval</b> )
cpp	int <b>set_bearing</b> ( double <b>newval</b> )
m	-(int) setBearing : (double) <b>newval</b>
pas	function <b>set_bearing</b> ( <b>newval</b> : double): integer
vb	function <b>set_bearing</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_bearing</b> ( double <b>newval</b> )
java	int <b>set_bearing</b> ( double <b>newval</b> )
py	def <b>set_bearing</b> ( <b>newval</b> )
cmd	YRefFrame <b>target set_bearing newval</b>

Le cap relatif indiqué par le compas est la différence entre le Nord magnétique mesuré et le cap de référence spécifié ici. Par exemple, si vous indiquez comme cap de référence la valeur de la déclinaison magnétique terrestre, le compas donnera l'orientation par rapport au Nord géographique. De même, si le capteur n'est pas positionné dans une des directions standard à cause d'un angle de lacet supplémentaire, vous pouvez le configurer comme cap de référence afin que le compas donne la direction naturelle attendue.

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une valeur numérique représentant le cap de référence utilisé par le compas

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

refframe→set\_logicalName()

YRefFrame

refframe→setLogicalName()[refframe

setLogicalName: ]

Modifie le nom logique du référentiel.

js	function set_logicalName( newval)
nodejs	function set_logicalName( newval)
php	function set_logicalName( \$newval)
cpp	int set_logicalName( const string& newval)
m	-(int) setLogicalName : (NSString*) newval
pas	function set_logicalName( newval: string): integer
vb	function set_logicalName( ByVal newval As String) As Integer
cs	int set_logicalName( string newval)
java	int set_logicalName( String newval)
py	def set_logicalName( newval)
cmd	YRefFrame target set_logicalName newval

Vous pouvez utiliser yCheckLogicalName( ) pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode saveToFlash( ) du module si le réglage doit être préservé.

#### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du référentiel.

#### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**refframe→set\_mountPosition()****YRefFrame****refframe→setMountPosition()[refframe****setMountPosition: ]**

Modifie le référentiel de la boussole et des inclinomètres.

js	function <b>set_mountPosition</b> ( <b>position</b> , <b>orientation</b> )
nodejs	function <b>set_mountPosition</b> ( <b>position</b> , <b>orientation</b> )
php	function <b>set_mountPosition</b> ( <b>\$position</b> , <b>\$orientation</b> )
cpp	int <b>set_mountPosition</b> ( Y_MOUNTPOSITION <b>position</b> , Y_MOUNTORIENTATION <b>orientation</b> )
m	-(int) setMountPosition : (Y_MOUNTPOSITION) <b>position</b> : (Y_MOUNTORIENTATION) <b>orientation</b>
pas	function <b>set_mountPosition</b> ( <b>position</b> : TYMOUNTPOSITION, <b>orientation</b> : TYMOUNTORIENTATION): LongInt
vb	function <b>set_mountPosition</b> ( ) As Integer
cs	int <b>set_mountPosition</b> ( MOUNTPOSITION <b>position</b> , MOUNTORIENTATION <b>orientation</b> )
java	int <b>set_mountPosition</b> ( MOUNTPOSITION <b>position</b> , MOUNTORIENTATION <b>orientation</b> )
py	def <b>set_mountPosition</b> ( <b>position</b> , <b>orientation</b> )
cmd	YRefFrame <b>target set_mountPosition position orientation</b>

La boussole magnétique et les inclinomètres gravitationnels fonctionnent par rapport au plan parallèle à la surface terrestre. Dans les cas où le module n'est pas utilisé horizontalement et à l'endroit, il faut indiquer son orientation de référence (parallèle à la surface terrestre) afin que les mesures soient faites relativement à cette position.

**Paramètres :**

- position** une valeur parmi l'énumération Y\_MOUNTPOSITION (Y\_MOUNTPOSITION\_BOTTOM, Y\_MOUNTPOSITION\_TOP, Y\_MOUNTPOSITION\_FRONT, Y\_MOUNTPOSITION\_RIGHT, Y\_MOUNTPOSITION\_REAR, Y\_MOUNTPOSITION\_LEFT), correspondant à l'installation dans une boîte, sur l'une des six faces.
- orientation** une valeur parmi l'énumération Y\_MOUNTORIENTATION (Y\_MOUNTORIENTATION\_TWELVE, Y\_MOUNTORIENTATION\_THREE, Y\_MOUNTORIENTATION\_SIX, Y\_MOUNTORIENTATION\_NINE) correspondant à la l'orientation de la flèche "X" sur le module par rapport à un cadran d'horloge vu par un observateur au centre de la boîte. Sur la face BOTTOM le 12h pointe vers l'avant, tandis que sur la face TOP le 12h pointe vers l'arrière. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**refframe→set\_userdata()****YRefFrame****refframe→setUserData()[refframe setUserData: ]**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

## refframe→start3DCalibration()[refframe start3DCalibration]

YRefFrame

Initie le processus de calibration tridimensionnelle des capteurs.

js	function <b>start3DCalibration</b> ( )
nodejs	function <b>start3DCalibration</b> ( )
php	function <b>start3DCalibration</b> ( )
cpp	int <b>start3DCalibration</b> ( )
m	-(int) <b>start3DCalibration</b>
pas	function <b>start3DCalibration</b> ( ): LongInt
vb	function <b>start3DCalibration</b> ( ) As Integer
cs	int <b>start3DCalibration</b> ( )
java	int <b>start3DCalibration</b> ( )
py	def <b>start3DCalibration</b> ( )
cmd	YRefFrame <b>target</b> <b>start3DCalibration</b>

Cette calibration est utilisée à bas niveau pour l'estimation innertielle de position et pour améliorer la précision des mesures d'inclinaison. Après avoir appelé cette méthode, il faut positionner le module selon les instructions fournies par la méthode `get_3DCalibrationHint` et appeler `more3DCalibration` environ 5 fois par secondes. La procédure de calibration est terminée lorsque la méthode `get_3DCalibrationProgress` retourne 100. Il est alors possible d'appliquer les paramètres calculés, à l'aide de la méthode `save3DCalibration`. A tout moment, la calibration peut être abandonnée à l'aide de `cancel3DCalibration`. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**refframe→wait\_async()****YRefFrame**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.35. Interface de la fonction Relay

La librairie de programmation Yoctopuce permet simplement de changer l'état du relais. Le changement d'état n'est pas persistant: le relais retournera spontanément à sa position de repos dès que le module est mis hors tension ou redémarré. La librairie permet aussi de créer des courtes impulsions de durée déterminée. Pour les modules dotés de deux sorties par relais (relai inverseur), les deux sorties sont appelées A et B, la sortie A correspondant à la position de repos (hors tension) et la sortie B correspondant à l'état actif. Si vous préféreriez l'état par défaut opposé, vous pouvez simplement changer vos fils sur le bornier.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_relay.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YRelay = yoctolib.YRelay;</code>
php	<code>require_once('yocto_relay.php');</code>
c++	<code>#include "yocto_relay.h"</code>
m	<code>#import "yocto_relay.h"</code>
pas	<code>uses yocto_relay;</code>
vb	<code>yocto_relay.vb</code>
cs	<code>yocto_relay.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YRelay;</code>
py	<code>from yocto_relay import *</code>

### Fonction globales

#### yFindRelay(func)

Permet de retrouver un relais d'après un identifiant donné.

#### yFirstRelay()

Commence l'énumération des relais accessibles par la librairie.

### Méthodes des objets YRelay

#### relay→delayedPulse(ms\_delay, ms\_duration)

Pré-programme une impulsion

#### relay→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du relais au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### relay→get\_advertisedValue()

Retourne la valeur courante du relais (pas plus de 6 caractères).

#### relay→get\_countdown()

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`.

#### relay→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du relais.

#### relay→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du relais.

#### relay→get\_friendlyName()

Retourne un identifiant global du relais au format `NOM_MODULE . NOM_FONCTION`.

#### relay→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### relay→get\_functionId()

Retourne l'identifiant matériel du relais, sans référence au module.

**relay→get\_hardwareId()**

Retourne l'identifiant matériel unique du relais au format `SERIAL.FUNCTIONID`.

**relay→get\_logicalName()**

Retourne le nom logique du relais.

**relay→get\_maxTimeOnStateA()**

Retourne le temps maximal (en ms) pendant lequel le relais peut rester dans l'état A avant de basculer automatiquement dans l'état B.

**relay→get\_maxTimeOnStateB()**

Retourne le temps maximal (en ms) pendant lequel le relais peut rester dans l'état B avant de basculer automatiquement dans l'état A.

**relay→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**relay→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**relay→get\_output()**

Retourne l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

**relay→get\_pulseTimer()**

Retourne le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée.

**relay→get\_state()**

Retourne l'état du relais (A pour la position de repos, B pour l'état actif).

**relay→get\_stateAtPowerOn()**

Retourne l'état du relais au démarrage du module (A pour la position de repos, B pour l'état actif, `UNCHANGED` pour aucun changement).

**relay→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**relay→isOnline()**

Vérifie si le module hébergeant le relais est joignable, sans déclencher d'erreur.

**relay→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le relais est joignable, sans déclencher d'erreur.

**relay→load(msValidity)**

Met en cache les valeurs courantes du relais, avec une durée de validité spécifiée.

**relay→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du relais, avec une durée de validité spécifiée.

**relay→nextRelay()**

Continue l'énumération des relais commencée à l'aide de `yFirstRelay()`.

**relay→pulse(ms\_duration)**

Commute le relais à l'état B (actif) pour une durée spécifiée, puis revient ensuite spontanément vers l'état A (état de repos).

**relay→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**relay→set\_logicalName(newval)**

Modifie le nom logique du relais.

**relay→set\_maxTimeOnStateA(newval)**

Règle le temps maximal (en ms) pendant lequel le relais peut rester dans l'état A avant de basculer automatiquement dans l'état B.

**relay→set\_maxTimeOnStateB(newval)**

Règle le temps maximal (en ms) pendant lequel le relais peut rester dans l'état B avant de basculer automatiquement dans l'état A.

**relay→set\_output(newval)**

Modifie l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

**relay→set\_state(newval)**

Modifie l'état du relais (A pour la position de repos, B pour l'état actif).

**relay→set\_stateAtPowerOn(newval)**

Pré-programme l'état du relais au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

**relay→set\_userdata(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

**relay→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YRelay.FindRelay() yFindRelay()yFindRelay()

YRelay

Permet de retrouver un relais d'après un identifiant donné.

js	function <b>yFindRelay</b> ( <b>func</b> )
nodejs	function <b>FindRelay</b> ( <b>func</b> )
php	function <b>yFindRelay</b> ( <b>\$func</b> )
cpp	YRelay* <b>yFindRelay</b> ( const string& <b>func</b> )
m	YRelay* <b>yFindRelay</b> ( NSString* <b>func</b> )
pas	function <b>yFindRelay</b> ( <b>func</b> : string): TYRelay
vb	function <b>yFindRelay</b> ( ByVal <b>func</b> As String) As YRelay
cs	YRelay <b>FindRelay</b> ( string <b>func</b> )
java	YRelay <b>FindRelay</b> ( String <b>func</b> )
py	def <b>FindRelay</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le relais soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YRelay.isOnline()` pour tester si le relais est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le relais sans ambiguïté

### Retourne :

un objet de classe `YRelay` qui permet ensuite de contrôler le relais.



## YRelay.FirstRelay() yFirstRelay()yFirstRelay()

## YRelay

Commence l'énumération des relais accessibles par la librairie.

js	function <b>yFirstRelay</b> ( )
nodejs	function <b>FirstRelay</b> ( )
php	function <b>yFirstRelay</b> ( )
cpp	YRelay* <b>yFirstRelay</b> ( )
m	YRelay* <b>yFirstRelay</b> ( )
pas	function <b>yFirstRelay</b> ( ): TYRelay
vb	function <b>yFirstRelay</b> ( ) As YRelay
cs	YRelay <b>FirstRelay</b> ( )
java	YRelay <b>FirstRelay</b> ( )
py	def <b>FirstRelay</b> ( )

Utiliser la fonction `YRelay.nextRelay( )` pour itérer sur les autres relais.

### Retourne :

un pointeur sur un objet `YRelay`, correspondant à le premier relais accessible en ligne, ou `null` si il n'y a pas de relais disponibles.

**relay→delayedPulse()[relay delayedPulse: ]****YRelay**

Pré-programme une impulsion

js	function <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
nodejs	function <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
php	function <b>delayedPulse</b> ( <b>\$ms_delay</b> , <b>\$ms_duration</b> )
c++	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
m	-(int) <b>delayedPulse</b> : (int) <b>ms_delay</b> : (int) <b>ms_duration</b>
pas	function <b>delayedPulse</b> ( <b>ms_delay</b> : LongInt, <b>ms_duration</b> : LongInt): integer
vb	function <b>delayedPulse</b> ( ByVal <b>ms_delay</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
java	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
py	def <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
cmd	YRelay <b>target delayedPulse ms_delay ms_duration</b>

**Paramètres :****ms\_delay**    delai d'attente avant l'impulsion, en millisecondes**ms\_duration**    durée de l'impulsion, en millisecondes**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay→describe()[relay describe]****YRelay**

Retourne un court texte décrivant de manière non-ambigüe l'instance du relais au format `TYPE (NAME) = SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le relais (ex: `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**relay**→**get\_advertisedValue()****YRelay****relay**→**advertisedValue()**[relay advertisedValue]

Retourne la valeur courante du relais (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YRelay <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du relais (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**relay→get\_countdown()****YRelay****relay→countdown()[relay countdown]**

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à delayedPulse().

js	function <b>get_countdown</b> ( )
nodejs	function <b>get_countdown</b> ( )
php	function <b>get_countdown</b> ( )
c++	s64 <b>get_countdown</b> ( )
m	-(s64) countdown
pas	function <b>get_countdown</b> ( ): int64
vb	function <b>get_countdown</b> ( ) As Long
cs	long <b>get_countdown</b> ( )
java	long <b>get_countdown</b> ( )
py	def <b>get_countdown</b> ( )
cmd	YRelay <b>target</b> <b>get_countdown</b>

Si aucune impulsion n'est programmée, retourne zéro.

**Retourne :**

un entier représentant le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à delayedPulse()

En cas d'erreur, déclenche une exception ou retourne Y\_COUNTDOWN\_INVALID.

**relay**→**get\_errorMessage()****YRelay****relay**→**errorMessage()**[**relay errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du relais.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du relais.

**relay→get\_errorType()****YRelay****relay→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du relais.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du relais.

**relay**→**get\_friendlyName()****YRelay****relay**→**friendlyName()**[relay friendlyName]

Retourne un identifiant global du relais au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du relais si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du relais (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le relais en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.



**relay**→**get\_functionDescriptor()****YRelay****relay**→**functionDescriptor()**[**relay functionDescriptor**]

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**relay**→**get\_functionId()****YRelay****relay**→**functionId()**[relay functionId]

Retourne l'identifiant matériel du relais, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le relais (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**relay**→**get\_hardwareId()**  
**relay**→**hardwareId()[relay hardwareId]**

**YRelay**

Retourne l'identifiant matériel unique du relais au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du relais (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le relais (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**relay**→**get\_logicalName()****YRelay****relay**→**logicalName()**[relay logicalName]

Retourne le nom logique du relais.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YRelay <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du relais. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**relay→get\_maxTimeOnStateA()****YRelay****relay→maxTimeOnStateA()[relay maxTimeOnStateA]**

Retourne le temps maximal (en ms) pendant lequel le relais peut rester dans l'état A avant de basculer automatiquement dans l'état B.

js	function <b>get_maxTimeOnStateA</b> ( )
nodejs	function <b>get_maxTimeOnStateA</b> ( )
php	function <b>get_maxTimeOnStateA</b> ( )
cpp	s64 <b>get_maxTimeOnStateA</b> ( )
m	-(s64) maxTimeOnStateA
pas	function <b>get_maxTimeOnStateA</b> ( ): int64
vb	function <b>get_maxTimeOnStateA</b> ( ) As Long
cs	long <b>get_maxTimeOnStateA</b> ( )
java	long <b>get_maxTimeOnStateA</b> ( )
py	def <b>get_maxTimeOnStateA</b> ( )
cmd	YRelay <b>target</b> <b>get_maxTimeOnStateA</b>

Zéro signifie qu'il n'y a pas de limitation

**Retourne :**

un entier représentant le temps maximal (en ms) pendant lequel le relais peut rester dans l'état A avant de basculer automatiquement dans l'état B

En cas d'erreur, déclenche une exception ou retourne Y\_MAXTIMEONSTATEA\_INVALID.

**relay**→**get\_maxTimeOnStateB()****YRelay****relay**→**maxTimeOnStateB()**[**relay maxTimeOnStateB**]

Retourne le temps maximal (en ms) pendant lequel le relais peut rester dans l'état B avant de basculer automatiquement dans l'état A.

js	function <b>get_maxTimeOnStateB</b> ( )
nodejs	function <b>get_maxTimeOnStateB</b> ( )
php	function <b>get_maxTimeOnStateB</b> ( )
cpp	s64 <b>get_maxTimeOnStateB</b> ( )
m	-(s64) maxTimeOnStateB
pas	function <b>get_maxTimeOnStateB</b> ( ): int64
vb	function <b>get_maxTimeOnStateB</b> ( ) As Long
cs	long <b>get_maxTimeOnStateB</b> ( )
java	long <b>get_maxTimeOnStateB</b> ( )
py	def <b>get_maxTimeOnStateB</b> ( )
cmd	YRelay <b>target</b> <b>get_maxTimeOnStateB</b>

Zéro signifie qu'il n'y a pas de limitation

**Retourne :**

un entier représentant le temps maximal (en ms) pendant lequel le relais peut rester dans l'état B avant de basculer automatiquement dans l'état A

En cas d'erreur, déclenche une exception ou retourne Y\_MAXTIMEONSTATEB\_INVALID.

**relay**→**get\_module()**  
**relay**→**module()**[relay module]

**YRelay**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	-( <code>YModule*</code> ) module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**relay**→**get\_module\_async()****YRelay****relay**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**relay**→**get\_output()****YRelay****relay**→**output()**[relay output]

Retourne l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

js	function <b>get_output</b> ( )
nodejs	function <b>get_output</b> ( )
php	function <b>get_output</b> ( )
cpp	Y_OUTPUT_enum <b>get_output</b> ( )
m	-(Y_OUTPUT_enum) output
pas	function <b>get_output</b> ( ): Integer
vb	function <b>get_output</b> ( ) As Integer
cs	int <b>get_output</b> ( )
java	int <b>get_output</b> ( )
py	def <b>get_output</b> ( )
cmd	YRelay <b>target</b> <b>get_output</b>

**Retourne :**

soit Y\_OUTPUT\_OFF, soit Y\_OUTPUT\_ON, selon l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur

En cas d'erreur, déclenche une exception ou retourne Y\_OUTPUT\_INVALID.

**relay→get\_pulseTimer()****YRelay****relay→pulseTimer()[relay pulseTimer]**

Retourne le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée.

js	function <b>get_pulseTimer</b> ( )
nodejs	function <b>get_pulseTimer</b> ( )
php	function <b>get_pulseTimer</b> ( )
cpp	s64 <b>get_pulseTimer</b> ( )
m	-(s64) pulseTimer
pas	function <b>get_pulseTimer</b> ( ): int64
vb	function <b>get_pulseTimer</b> ( ) As Long
cs	long <b>get_pulseTimer</b> ( )
java	long <b>get_pulseTimer</b> ( )
py	def <b>get_pulseTimer</b> ( )
cmd	YRelay <b>target</b> <b>get_pulseTimer</b>

Si aucune impulsion n'est en cours, retourne zéro.

**Retourne :**

un entier représentant le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée

En cas d'erreur, déclenche une exception ou retourne Y\_PULSETIMER\_INVALID.

**relay→get\_state()****YRelay****relay→state()[relay state]**

Retourne l'état du relais (A pour la position de repos, B pour l'état actif).

js	function <b>get_state</b> ( )
nodejs	function <b>get_state</b> ( )
php	function <b>get_state</b> ( )
cpp	Y_STATE_enum <b>get_state</b> ( )
m	-(Y_STATE_enum) state
pas	function <b>get_state</b> ( ): Integer
vb	function <b>get_state</b> ( ) As Integer
cs	int <b>get_state</b> ( )
java	int <b>get_state</b> ( )
py	def <b>get_state</b> ( )
cmd	YRelay <b>target</b> <b>get_state</b>

**Retourne :**

soit Y\_STATE\_A, soit Y\_STATE\_B, selon l'état du relais (A pour la position de repos, B pour l'état actif)

En cas d'erreur, déclenche une exception ou retourne Y\_STATE\_INVALID.

**relay→get\_stateAtPowerOn()****YRelay****relay→stateAtPowerOn()[relay stateAtPowerOn]**

Retourne l'état du relais au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

js	function <b>get_stateAtPowerOn</b> ( )
nodejs	function <b>get_stateAtPowerOn</b> ( )
php	function <b>get_stateAtPowerOn</b> ( )
cpp	Y_STATEATPOWERON_enum <b>get_stateAtPowerOn</b> ( )
m	-(Y_STATEATPOWERON_enum) stateAtPowerOn
pas	function <b>get_stateAtPowerOn</b> ( ): Integer
vb	function <b>get_stateAtPowerOn</b> ( ) As Integer
cs	int <b>get_stateAtPowerOn</b> ( )
java	int <b>get_stateAtPowerOn</b> ( )
py	def <b>get_stateAtPowerOn</b> ( )
cmd	YRelay <b>target</b> <b>get_stateAtPowerOn</b>

**Retourne :**

une valeur parmi Y\_STATEATPOWERON\_UNCHANGED, Y\_STATEATPOWERON\_A et Y\_STATEATPOWERON\_B représentant l'état du relais au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement)

En cas d'erreur, déclenche une exception ou retourne Y\_STATEATPOWERON\_INVALID.

**relay**→**get\_userData()****YRelay****relay**→**userData()**[**relay userData**]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): TObject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**relay**→**isOnline()**[relay isOnline]**YRelay**

Vérifie si le module hébergeant le relais est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du relais sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le relais est joignable, false sinon

---

**relay→isOnline\_async()**

---

**YRelay**

Vérifie si le module hébergeant le relais est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du relais sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**relay→load()[relay load: ]****YRelay**

Met en cache les valeurs courantes du relais, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**relay→load\_async()****YRelay**

Met en cache les valeurs courantes du relais, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**relay**→**nextRelay()**[**relay** **nextRelay**]**YRelay**

Continue l'énumération des relais commencée à l'aide de `yFirstRelay()`.

js	function <b>nextRelay</b> ( )
nodejs	function <b>nextRelay</b> ( )
php	function <b>nextRelay</b> ( )
cpp	YRelay * <b>nextRelay</b> ( )
m	-(YRelay*) <b>nextRelay</b>
pas	function <b>nextRelay</b> ( ): TYRelay
vb	function <b>nextRelay</b> ( ) As YRelay
cs	YRelay <b>nextRelay</b> ( )
java	YRelay <b>nextRelay</b> ( )
py	def <b>nextRelay</b> ( )

**Retourne :**

un pointeur sur un objet `YRelay` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**relay→pulse()[relay pulse: ]****YRelay**

Commute le relais à l'état B (actif) pour un durée spécifiée, puis revient ensuite spontanément vers l'état A (état de repos).

js	function pulse( ms_duration)
nodejs	function pulse( ms_duration)
php	function pulse( \$ms_duration)
cpp	int pulse( int ms_duration)
m	-(int) pulse : (int) ms_duration
pas	function pulse( ms_duration: LongInt): integer
vb	function pulse( ByVal ms_duration As Integer) As Integer
cs	int pulse( int ms_duration)
java	int pulse( int ms_duration)
py	def pulse( ms_duration)
cmd	YRelay target pulse ms_duration

**Paramètres :**

**ms\_duration** durée de l'impulsion, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## relay→registerValueCallback()[relay registerValueCallback: ]

YRelay

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function registerValueCallback( callback)
nodejs	function registerValueCallback( callback)
php	function registerValueCallback( \$callback)
cpp	int registerValueCallback( YRelayValueCallback callback)
m	-(int) registerValueCallback : (YRelayValueCallback) callback
pas	function registerValueCallback( callback: TYRelayValueCallback): LongInt
vb	function registerValueCallback( ) As Integer
cs	int registerValueCallback( ValueCallback callback)
java	int registerValueCallback( UpdateCallback callback)
py	def registerValueCallback( callback)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**relay→set\_logicalName()****YRelay****relay→setLogicalName()[relay setLogicalName: ]**

Modifie le nom logique du relais.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YRelay <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du relais.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay**→**set\_maxTimeOnStateA()****YRelay****relay**→**setMaxTimeOnStateA()**[**relay****setMaxTimeOnStateA: ]**

Règle le temps maximal (en ms) pendant lequel le relais peut rester dans l'état A avant de basculer automatiquement dans l'état B.

js	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
nodejs	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
php	function <b>set_maxTimeOnStateA</b> ( <b>\$newval</b> )
cpp	int <b>set_maxTimeOnStateA</b> ( s64 <b>newval</b> )
m	-(int) setMaxTimeOnStateA : (s64) <b>newval</b>
pas	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> : int64): integer
vb	function <b>set_maxTimeOnStateA</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_maxTimeOnStateA</b> ( long <b>newval</b> )
java	int <b>set_maxTimeOnStateA</b> ( long <b>newval</b> )
py	def <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
cmd	YRelay <b>target set_maxTimeOnStateA newval</b>

Zéro signifie qu'il n'y a pas de limitation

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay→set\_maxTimeOnStateB()**  
**relay→setMaxTimeOnStateB()[relay**  
**setMaxTimeOnStateB: ]**

YRelay

Règle le temps maximal (en ms) pendant lequel le relais peut rester dans l'état B avant de basculer automatiquement dans l'état A.

js	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
nodejs	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
php	function <b>set_maxTimeOnStateB</b> ( <b>\$newval</b> )
cpp	int <b>set_maxTimeOnStateB</b> ( s64 <b>newval</b> )
m	-(int) setMaxTimeOnStateB : (s64) <b>newval</b>
pas	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> : int64): integer
vb	function <b>set_maxTimeOnStateB</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_maxTimeOnStateB</b> ( long <b>newval</b> )
java	int <b>set_maxTimeOnStateB</b> ( long <b>newval</b> )
py	def <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
cmd	YRelay <b>target set_maxTimeOnStateB newval</b>

Zéro signifie qu'il n'y a pas de limitation

#### Paramètres :

**newval** un entier

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay**→**set\_output()****YRelay****relay**→**setOutput()**[**relay setOutput:** ]

Modifie l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

js	function <b>set_output</b> ( <b>newval</b> )
nodejs	function <b>set_output</b> ( <b>newval</b> )
php	function <b>set_output</b> ( <b>\$newval</b> )
cpp	int <b>set_output</b> ( Y_OUTPUT_enum <b>newval</b> )
m	-(int) setOutput : (Y_OUTPUT_enum) <b>newval</b>
pas	function <b>set_output</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_output</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_output</b> ( int <b>newval</b> )
java	int <b>set_output</b> ( int <b>newval</b> )
py	def <b>set_output</b> ( <b>newval</b> )
cmd	YRelay <b>target set_output newval</b>

**Paramètres :**

**newval** soit Y\_OUTPUT\_OFF, soit Y\_OUTPUT\_ON, selon l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**relay→set\_state()****YRelay****relay→setState()[relay setState: ]**

Modifie l'état du relais (A pour la position de repos, B pour l'état actif).

js	function <b>set_state</b> ( <b>newval</b> )
nodejs	function <b>set_state</b> ( <b>newval</b> )
php	function <b>set_state</b> ( <b>\$newval</b> )
cpp	int <b>set_state</b> ( Y_STATE_enum <b>newval</b> )
m	-(int) <b>setState</b> : (Y_STATE_enum) <b>newval</b>
pas	function <b>set_state</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_state</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_state</b> ( int <b>newval</b> )
java	int <b>set_state</b> ( int <b>newval</b> )
py	def <b>set_state</b> ( <b>newval</b> )
cmd	YRelay <b>target</b> <b>set_state</b> <b>newval</b>

**Paramètres :**

**newval** soit Y\_STATE\_A, soit Y\_STATE\_B, selon l'état du relais (A pour la position de repos, B pour l'état actif)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay→set\_stateAtPowerOn()****YRelay****relay→setStateAtPowerOn()[relay****setStateAtPowerOn: ]**

Pré-programme l'état du relais au démarrage du module(A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

js	function <b>set_stateAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_stateAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_stateAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_stateAtPowerOn</b> ( Y_STATEATPOWERON_enum <b>newval</b> )
m	-(int) setStateAtPowerOn : (Y_STATEATPOWERON_enum) <b>newval</b>
pas	function <b>set_stateAtPowerOn</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_stateAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_stateAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_stateAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_stateAtPowerOn</b> ( <b>newval</b> )
cmd	YRelay <b>target set_stateAtPowerOn newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module sinon la modification n'aura aucun effet.

**Paramètres :**

**newval** une valeur parmi Y\_STATEATPOWERON\_UNCHANGED, Y\_STATEATPOWERON\_A et Y\_STATEATPOWERON\_B

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**relay**→**set\_userData()****YRelay****relay**→**setUserData()**[**relay setUserData:** ]

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function <b>set_userData</b> ( <b>data</b> )
nodejs	function <b>set_userData</b> ( <b>data</b> )
php	function <b>set_userData</b> ( <b>\$data</b> )
cpp	void <b>set_userData</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userData</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userData</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userData</b> ( object <b>data</b> )
java	void <b>set_userData</b> ( Object <b>data</b> )
py	def <b>set_userData</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**relay** → **wait\_async()****YRelay**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.36. Interface des fonctions de type senseur

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_api.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YAPI = yoctolib.YAPI; var YModule = yoctolib.YModule;
php	require_once('yocto_api.php');
cpp	#include "yocto_api.h"
m	#import "yocto_api.h"
pas	uses yocto_api;
vb	yocto_api.vb
cs	yocto_api.cs
java	import com.yoctopuce.YoctoAPI.YModule;
py	from yocto_api import *

### Fonction globales

#### yFindSensor(func)

Permet de retrouver un senseur d'après un identifiant donné.

#### yFirstSensor()

Commence l'énumération des senseurs accessibles par la librairie.

### Méthodes des objets YSensor

#### sensor→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### sensor→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du senseur au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### sensor→get\_advertisedValue()

Retourne la valeur courante du senseur (pas plus de 6 caractères).

#### sensor→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### sensor→get\_currentValue()

Retourne la valeur actuelle de la mesure.

#### sensor→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du senseur.

#### sensor→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du senseur.

#### sensor→get\_friendlyName()

Retourne un identifiant global du senseur au format `NOM_MODULE . NOM_FONCTION`.

#### sensor→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### sensor→get\_functionId()

Retourne l'identifiant matériel du senseur, sans référence au module.

#### sensor→get\_hardwareId()

Retourne l'identifiant matériel unique du capteur au format `SERIAL.FUNCTIONID`.

#### **sensor→get\_highestValue()**

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

#### **sensor→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

#### **sensor→get\_logicalName()**

Retourne le nom logique du capteur.

#### **sensor→get\_lowestValue()**

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

#### **sensor→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **sensor→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **sensor→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

#### **sensor→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

#### **sensor→get\_resolution()**

Retourne la résolution des valeurs mesurées.

#### **sensor→get\_unit()**

Retourne l'unité dans laquelle la mesure est exprimée.

#### **sensor→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **sensor→isOnline()**

Vérifie si le module hébergeant le capteur est joignable, sans déclencher d'erreur.

#### **sensor→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le capteur est joignable, sans déclencher d'erreur.

#### **sensor→load(msValidity)**

Met en cache les valeurs courantes du capteur, avec une durée de validité spécifiée.

#### **sensor→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

#### **sensor→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du capteur, avec une durée de validité spécifiée.

#### **sensor→nextSensor()**

Continue l'énumération des capteurs commencée à l'aide de `yFirstSensor()`.

#### **sensor→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

#### **sensor→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **sensor→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

#### **sensor→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**sensor**→**set\_logicalName**(newval)

Modifie le nom logique du senseur.

**sensor**→**set\_lowestValue**(newval)

Modifie la mémoire de valeur minimale observée.

**sensor**→**set\_reportFrequency**(newval)

Modifie la fréquence de notification périodique des valeurs mesurées.

**sensor**→**set\_resolution**(newval)

Modifie la résolution des valeurs physique mesurées.

**sensor**→**set\_userData**(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**sensor**→**wait\_async**(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YSensor.FindSensor() yFindSensor()yFindSensor()

YSensor

Permet de retrouver un senseur d'après un identifiant donné.

js	function <b>yFindSensor</b> ( <b>func</b> )
nodejs	function <b>FindSensor</b> ( <b>func</b> )
php	function <b>yFindSensor</b> ( <b>\$func</b> )
cpp	YSensor* <b>yFindSensor</b> ( string <b>func</b> )
m	+(YSensor*) <b>yFindSensor</b> : (NSString*) <b>func</b>
pas	function <b>yFindSensor</b> ( <b>func</b> : string): TYSensor
vb	function <b>yFindSensor</b> ( ByVal <b>func</b> As String) As YSensor
cs	YSensor <b>FindSensor</b> ( string <b>func</b> )
java	YSensor <b>FindSensor</b> ( String <b>func</b> )
py	def <b>FindSensor</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le senseur soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YSensor.isOnline()` pour tester si le senseur est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le senseur sans ambiguïté

### Retourne :

un objet de classe `YSensor` qui permet ensuite de contrôler le senseur.



## YSensor.FirstSensor() yFirstSensor()yFirstSensor()

## YSensor

Commence l'énumération des senseurs accessibles par la librairie.

js	function <b>yFirstSensor</b> ( )
nodejs	function <b>FirstSensor</b> ( )
php	function <b>yFirstSensor</b> ( )
cpp	YSensor* <b>yFirstSensor</b> ( )
m	YSensor* <b>yFirstSensor</b> ( )
pas	function <b>yFirstSensor</b> ( ): TYSensor
vb	function <b>yFirstSensor</b> ( ) As YSensor
cs	YSensor <b>FirstSensor</b> ( )
java	YSensor <b>FirstSensor</b> ( )
py	def <b>FirstSensor</b> ( )

Utiliser la fonction `YSensor.nextSensor( )` pour itérer sur les autres senseurs.

### Retourne :

un pointeur sur un objet `YSensor`, correspondant à le premier senseur accessible en ligne, ou `null` si il n'y a pas de senseurs disponibles.

## sensor→calibrateFromPoints()[sensor calibrateFromPoints: ]

YSensor

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YSensor target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor→describe()[sensor describe]****YSensor**

Retourne un court texte décrivant de manière non-ambigüe l'instance du senseur au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le senseur (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**sensor**→**get\_advertisedValue()****YSensor****sensor**→**advertisedValue()**[**sensor advertisedValue**]

Retourne la valeur courante du senseur (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YSensor <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du senseur (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**sensor**→**get\_currentRawValue()**  
**sensor**→**currentRawValue()**[**sensor**  
**currentRawValue**]

**YSensor**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YSensor <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**sensor**→**get\_currentValue()****YSensor****sensor**→**currentValue()**[**sensor currentValue**]

Retourne la valeur actuelle de la mesure.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YSensor <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de la mesure

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**sensor**→**get\_errorMessage()****YSensor****sensor**→**errorMessage()**[**sensor errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la bibliothèque Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur.

**sensor**→**get\_errorType()****YSensor****sensor**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du senseur.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du senseur.



**sensor**→**get\_friendlyName()****YSensor****sensor**→**friendlyName()**[sensor friendlyName]

Retourne un identifiant global du senseur au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du senseur si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du senseur (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le senseur en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**sensor**→**get\_functionDescriptor()**  
**sensor**→**functionDescriptor()**[**sensor**  
**functionDescriptor**]

**YSensor**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**sensor**→**get\_functionId()****YSensor****sensor**→**functionId()**[**sensor functionId**]

Retourne l'identifiant matériel du senseur, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le senseur (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**sensor**→**get\_hardwareId()****YSensor****sensor**→**hardwareId()**[**sensor hardwareId**]

Retourne l'identifiant matériel unique du senseur au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du senseur (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le senseur (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**sensor**→**get\_highestValue()****YSensor****sensor**→**highestValue()**[sensor highestValue]

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YSensor <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la mesure depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**sensor**→**get\_logFrequency()****YSensor****sensor**→**logFrequency()**[**sensor logFrequency**]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YSensor <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**sensor**→**get\_logicalName()**  
**sensor**→**logicalName()**[sensor logicalName]

**YSensor**

Retourne le nom logique du senseur.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YSensor <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du senseur. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**sensor**→**get\_lowestValue()****YSensor****sensor**→**lowestValue()**[**sensor lowestValue**]

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YSensor <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la mesure depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.



**sensor**→**get\_module()****YSensor****sensor**→**module()**[sensor module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**sensor**→**get\_module\_async()****YSensor****sensor**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**sensor**→**get\_recordedData()****YSensor****sensor**→**recordedData()**[**sensor recordedData:** ]

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YSensor <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**sensor**→**get\_reportFrequency()****YSensor****sensor**→**reportFrequency()**[sensor reportFrequency]

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YSensor <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**sensor**→**get\_resolution()****YSensor****sensor**→**resolution()**[sensor resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YSensor <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**sensor**→**get\_unit()****sensor**→**unit()**[sensor unit]

Retourne l'unité dans laquelle la mesure est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YSensor <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la mesure est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**sensor**→**get\_userData()****YSensor****sensor**→**userData()**[sensor userData]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**sensor**→**isOnline()**[**sensor isOnline**]**YSensor**

Vérifie si le module hébergeant le senseur est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du senseur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le senseur est joignable, false sinon



**sensor**→**isOnline\_async()****YSensor**

Vérifie si le module hébergeant le senseur est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du senseur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**sensor→load()[sensor load: ]****YSensor**

Met en cache les valeurs courantes du senseur, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
c++	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## sensor→loadCalibrationPoints()[sensor loadCalibrationPoints: ]

YSensor

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
                             : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YSensor target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**load\_async()****YSensor**

Met en cache les valeurs courantes du senseur, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**sensor**→**nextSensor()**[**sensor** **nextSensor**]**YSensor**

Continue l'énumération des senseurs commencée à l'aide de `yFirstSensor()`.

js	function <b>nextSensor</b> ( )
nodejs	function <b>nextSensor</b> ( )
php	function <b>nextSensor</b> ( )
cpp	YSensor * <b>nextSensor</b> ( )
m	-(YSensor*) <b>nextSensor</b>
pas	function <b>nextSensor</b> ( ): TYSensor
vb	function <b>nextSensor</b> ( ) As YSensor
cs	YSensor <b>nextSensor</b> ( )
java	YSensor <b>nextSensor</b> ( )
py	def <b>nextSensor</b> ( )

**Retourne :**

un pointeur sur un objet `YSensor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## sensor→registerTimedReportCallback()[sensor registerTimedReportCallback: ]

YSensor

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YSensorTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YSensorTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYSensorTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## sensor→registerValueCallback()[sensor registerValueCallback: ]

YSensor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YSensorValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YSensorValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYSensorValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**sensor**→**set\_highestValue()****YSensor****sensor**→**setHighestValue()**[**sensor setHighestValue:** ]

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YSensor <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**sensor**→**set\_logFrequency()**  
**sensor**→**setLogFrequency()**[**sensor**  
**setLogFrequency:** ]

**YSensor**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
c++	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**set\_logicalName()****YSensor****sensor**→**setLogicalName()**[**sensor setLogicalName:** ]

Modifie le nom logique du senseur.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YSensor <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du senseur.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**set\_lowestValue()****YSensor****sensor**→**setLowestValue()**[**sensor setLowestValue:** ]

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**set\_reportFrequency()**  
**sensor**→**setReportFrequency()**[**sensor**  
**setReportFrequency:** ]

YSensor

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YSensor <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**set\_resolution()****YSensor****sensor**→**setResolution()**[**sensor setResolution:** ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YSensor <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**sensor**→**set\_userData()****YSensor****sensor**→**setUserData()**[**sensor setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function <b>set_userData</b> ( <b>data</b> )
nodejs	function <b>set_userData</b> ( <b>data</b> )
php	function <b>set_userData</b> ( <b>\$data</b> )
cpp	void <b>set_userData</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userData</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userData</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userData</b> ( object <b>data</b> )
java	void <b>set_userData</b> ( Object <b>data</b> )
py	def <b>set_userData</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**sensor→wait\_async()****YSensor**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.37. Interface de la fonction Servo

La librairie de programmation Yoctopuce permet non seulement de déplacer le servo vers une position donnée, mais aussi de spécifier l'intervalle de temps dans lequel le mouvement doit être fait, de sorte à pouvoir synchroniser un mouvement sur plusieurs servos.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_servo.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YServo = yoctolib.YServo;</code>
php	<code>require_once('yocto_servo.php');</code>
c++	<code>#include "yocto_servo.h"</code>
m	<code>#import "yocto_servo.h"</code>
pas	<code>uses yocto_servo;</code>
vb	<code>yocto_servo.vb</code>
cs	<code>yocto_servo.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YServo;</code>
py	<code>from yocto_servo import *</code>

### Fonction globales

#### yFindServo(func)

Permet de retrouver un servo d'après un identifiant donné.

#### yFirstServo()

Commence l'énumération des servo accessibles par la librairie.

### Méthodes des objets YServo

#### servo→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du servo au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### servo→get\_advertisedValue()

Retourne la valeur courante du servo (pas plus de 6 caractères).

#### servo→get\_enabled()

Retourne l'état de fonctionnement du \$FUNCTION\$.

#### servo→get\_enabledAtPowerOn()

Retourne l'état du générateur de signal de commande du servo au démarrage du module.

#### servo→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du servo.

#### servo→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du servo.

#### servo→get\_friendlyName()

Retourne un identifiant global du servo au format `NOM_MODULE . NOM_FONCTION`.

#### servo→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### servo→get\_functionId()

Retourne l'identifiant matériel du servo, sans référence au module.

#### servo→get\_hardwareId()

Retourne l'identifiant matériel unique du servo au format `SERIAL . FUNCTIONID`.

#### servo→get\_logicalName()

Retourne le nom logique du servo.



**servo→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**servo→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**servo→get\_neutral()**

Retourne la durée en microsecondes de l'impulsion correspondant au neutre du servo.

**servo→get\_position()**

Retourne la position courante du servo.

**servo→get\_positionAtPowerOn()**

Retourne la position du servo au démarrage du module.

**servo→get\_range()**

Retourne la plage d'utilisation du servo.

**servo→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**servo→isOnline()**

Vérifie si le module hébergeant le servo est joignable, sans déclencher d'erreur.

**servo→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le servo est joignable, sans déclencher d'erreur.

**servo→load(msValidity)**

Met en cache les valeurs courantes du servo, avec une durée de validité spécifiée.

**servo→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du servo, avec une durée de validité spécifiée.

**servo→move(target, ms\_duration)**

Déclenche un mouvement à vitesse constante vers une position donnée.

**servo→nextServo()**

Continue l'énumération des servo commencée à l'aide de `yFirstServo()`.

**servo→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**servo→set\_enabled(newval)**

Démarre ou arrête le \$FUNCTION\$.

**servo→set\_enabledAtPowerOn(newval)**

Configure l'état du générateur de signal de commande du servo au démarrage du module.

**servo→set\_logicalName(newval)**

Modifie le nom logique du servo.

**servo→set\_neutral(newval)**

Modifie la durée de l'impulsion correspondant à la position neutre du servo.

**servo→set\_position(newval)**

Modifie immédiatement la consigne de position du servo.

**servo→set\_positionAtPowerOn(newval)**

Configure la position du servo au démarrage du module.

**servo→set\_range(newval)**

Modifie la plage d'utilisation du servo, en pourcents.

**servo→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**servo→wait\_async(callback, context)**

### 3. Reference

---

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YServo.FindServo() yFindServo()yFindServo()

## YServo

Permet de retrouver un servo d'après un identifiant donné.

js	function <b>yFindServo</b> ( <b>func</b> )
nodejs	function <b>FindServo</b> ( <b>func</b> )
php	function <b>yFindServo</b> ( <b>\$func</b> )
cpp	YServo* <b>yFindServo</b> ( const string& <b>func</b> )
m	YServo* <b>yFindServo</b> ( NSString* <b>func</b> )
pas	function <b>yFindServo</b> ( <b>func</b> : string): TYServo
vb	function <b>yFindServo</b> ( ByVal <b>func</b> As String) As YServo
cs	YServo <b>FindServo</b> ( string <b>func</b> )
java	YServo <b>FindServo</b> ( String <b>func</b> )
py	def <b>FindServo</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le servo soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YServo.isOnline()` pour tester si le servo est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le servo sans ambiguïté

### Retourne :

un objet de classe `YServo` qui permet ensuite de contrôler le servo.

## YServo.FirstServo() yFirstServo()yFirstServo()

YServo

Commence l'énumération des servo accessibles par la librairie.

js	function <b>yFirstServo</b> ( )
nodejs	function <b>FirstServo</b> ( )
php	function <b>yFirstServo</b> ( )
cpp	YServo* <b>yFirstServo</b> ( )
m	YServo* <b>yFirstServo</b> ( )
pas	function <b>yFirstServo</b> ( ): TYServo
vb	function <b>yFirstServo</b> ( ) As YServo
cs	YServo <b>FirstServo</b> ( )
java	YServo <b>FirstServo</b> ( )
py	def <b>FirstServo</b> ( )

Utiliser la fonction `YServo.nextServo( )` pour itérer sur les autres servo.

### Retourne :

un pointeur sur un objet `YServo`, correspondant à le premier servo accessible en ligne, ou `null` si il n'y a pas de servo disponibles.

**servo→describe()[servo describe]****YServo**

Retourne un court texte décrivant de manière non-ambigüe l'instance du servo au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le servo (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**servo**→**get\_advertisedValue()****YServo****servo**→**advertisedValue()**[**servo advertisedValue**]

Retourne la valeur courante du servo (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YServo <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du servo (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**servo→get\_enabled()****YServo****servo→enabled()[servo enabled]**

Retourne l'état de fonctionnement du \$FUNCTION\$.

js	function <b>get_enabled</b> ( )
nodejs	function <b>get_enabled</b> ( )
php	function <b>get_enabled</b> ( )
cpp	Y_ENABLED_enum <b>get_enabled</b> ( )
m	-(Y_ENABLED_enum) enabled
pas	function <b>get_enabled</b> ( ): Integer
vb	function <b>get_enabled</b> ( ) As Integer
cs	int <b>get_enabled</b> ( )
java	int <b>get_enabled</b> ( )
py	def <b>get_enabled</b> ( )
cmd	YServo <b>target</b> <b>get_enabled</b>

**Retourne :**

soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE, selon l'état de fonctionnement du \$FUNCTION\$

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLED\_INVALID.

**servo→get\_enabledAtPowerOn()**  
**servo→enabledAtPowerOn()[servo**  
**enabledAtPowerOn]**

Retourne l'état du générateur de signal de commande du servo au démarrage du module.

js	function <b>get_enabledAtPowerOn</b> ( )
nodejs	function <b>get_enabledAtPowerOn</b> ( )
php	function <b>get_enabledAtPowerOn</b> ( )
cpp	Y_ENABLEDATPOWERON_enum <b>get_enabledAtPowerOn</b> ( )
m	-(Y_ENABLEDATPOWERON_enum) enabledAtPowerOn
pas	function <b>get_enabledAtPowerOn</b> ( ): Integer
vb	function <b>get_enabledAtPowerOn</b> ( ) As Integer
cs	int <b>get_enabledAtPowerOn</b> ( )
java	int <b>get_enabledAtPowerOn</b> ( )
py	def <b>get_enabledAtPowerOn</b> ( )
cmd	YServo <b>target</b> <b>get_enabledAtPowerOn</b>

**Retourne :**

soit Y\_ENABLEDATPOWERON\_FALSE, soit Y\_ENABLEDATPOWERON\_TRUE, selon l'état du générateur de signal de commande du servo au démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_ENABLEDATPOWERON\_INVALID.



**servo→get\_errorMessage()****YServo****servo→errorMessage()[servo errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du servo.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du servo.

**servo**→**get\_errorType()****YServo****servo**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du servo.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du servo.

**servo→get\_friendlyName()****YServo****servo→friendlyName()[servo friendlyName]**

Retourne un identifiant global du servo au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du servo si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du servo (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le servo en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**servo**→**get\_functionDescriptor()**  
**servo**→**functionDescriptor()**[**servo**  
**functionDescriptor**]

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**servo**→**get\_functionId()**  
**servo**→**functionId()**[servo functionId]

**YServo**

Retourne l'identifiant matériel du servo, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le servo (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**servo**→**get\_hardwareId()****servo**→**hardwareId()**[servo hardwareId]

Retourne l'identifiant matériel unique du servo au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du servo (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le servo (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**servo→get\_logicalName()****YServo****servo→logicalName()[servo logicalName]**

Retourne le nom logique du servo.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YServo <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du servo. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**servo**→**get\_module()****servo**→**module()**[servo module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule



**servo→get\_module\_async()**  
**servo→module\_async()**

**YServo**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**servo**→**get\_neutral()****servo**→**neutral()**[servo neutral]

Retourne la durée en microsecondes de l'impulsion correspondant au neutre du servo.

js	function <b>get_neutral</b> ( )
nodejs	function <b>get_neutral</b> ( )
php	function <b>get_neutral</b> ( )
cpp	int <b>get_neutral</b> ( )
m	-(int) neutral
pas	function <b>get_neutral</b> ( ): LongInt
vb	function <b>get_neutral</b> ( ) As Integer
cs	int <b>get_neutral</b> ( )
java	int <b>get_neutral</b> ( )
py	def <b>get_neutral</b> ( )
cmd	YServo <b>target</b> <b>get_neutral</b>

**Retourne :**

un entier représentant la durée en microsecondes de l'impulsion correspondant au neutre du servo

En cas d'erreur, déclenche une exception ou retourne Y\_NEUTRAL\_INVALID.

**servo→get\_position()****YServo****servo→position()[servo position]**

Retourne la position courante du servo.

js	function <b>get_position</b> ( )
nodejs	function <b>get_position</b> ( )
php	function <b>get_position</b> ( )
cpp	int <b>get_position</b> ( )
m	-(int) position
pas	function <b>get_position</b> ( ): LongInt
vb	function <b>get_position</b> ( ) As Integer
cs	int <b>get_position</b> ( )
java	int <b>get_position</b> ( )
py	def <b>get_position</b> ( )
cmd	YServo <b>target</b> <b>get_position</b>

**Retourne :**

un entier représentant la position courante du servo

En cas d'erreur, déclenche une exception ou retourne Y\_POSITION\_INVALID.

**servo**→**get\_positionAtPowerOn()**  
**servo**→**positionAtPowerOn()**[**servo**  
**positionAtPowerOn**]

Retourne la position du servo au démarrage du module.

js	function <b>get_positionAtPowerOn</b> ( )
nodejs	function <b>get_positionAtPowerOn</b> ( )
php	function <b>get_positionAtPowerOn</b> ( )
cpp	int <b>get_positionAtPowerOn</b> ( )
m	-(int) positionAtPowerOn
pas	function <b>get_positionAtPowerOn</b> ( ): LongInt
vb	function <b>get_positionAtPowerOn</b> ( ) As Integer
cs	int <b>get_positionAtPowerOn</b> ( )
java	int <b>get_positionAtPowerOn</b> ( )
py	def <b>get_positionAtPowerOn</b> ( )
cmd	YServo <b>target</b> <b>get_positionAtPowerOn</b>

**Retourne :**

un entier représentant la position du servo au démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_POSITIONATPOWERON\_INVALID.

**servo→get\_range()**  
**servo→range()[servo range]**

**YServo**

Retourne la plage d'utilisation du servo.

js	function <b>get_range</b> ( )
nodejs	function <b>get_range</b> ( )
php	function <b>get_range</b> ( )
cpp	int <b>get_range</b> ( )
m	-(int) range
pas	function <b>get_range</b> ( ): LongInt
vb	function <b>get_range</b> ( ) As Integer
cs	int <b>get_range</b> ( )
java	int <b>get_range</b> ( )
py	def <b>get_range</b> ( )
cmd	YServo <b>target</b> <b>get_range</b>

**Retourne :**

un entier représentant la plage d'utilisation du servo

En cas d'erreur, déclenche une exception ou retourne Y\_RANGE\_INVALID.

**servo**→**get\_userData()**

**YServo**

**servo**→**userData()**[servo userData]

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
nodejs	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**servo→isOnline()[servo isOnline]****YServo**

Vérifie si le module hébergeant le servo est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du servo sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le servo est joignable, `false` sinon

Vérifie si le module hébergeant le servo est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du servo sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**servo→load()[servo load: ]****YServo**

Met en cache les valeurs courantes du servo, avec une durée de validité spécifiée.

js	function load( <b>msValidity</b> )
node.js	function load( <b>msValidity</b> )
php	function load( <b>\$msValidity</b> )
cpp	YRETCODE load( int <b>msValidity</b> )
m	-(YRETCODE) load : (int) <b>msValidity</b>
pas	function load( <b>msValidity</b> : integer): YRETCODE
vb	function load( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE load( int <b>msValidity</b> )
java	int load( long <b>msValidity</b> )
py	def load( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo→load\_async()****YServo**

Met en cache les valeurs courantes du servo, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**servo→move()[servo move: ]****YServo**

Déclenche un mouvement à vitesse constante vers une position donnée.

js	function <b>move</b> ( <b>target</b> , <b>ms_duration</b> )
nodejs	function <b>move</b> ( <b>target</b> , <b>ms_duration</b> )
php	function <b>move</b> ( <b>\$target</b> , <b>\$ms_duration</b> )
cpp	int <b>move</b> ( int <b>target</b> , int <b>ms_duration</b> )
m	-(int) <b>move</b> : (int) <b>target</b> : (int) <b>ms_duration</b>
pas	function <b>move</b> ( <b>target</b> : LongInt, <b>ms_duration</b> : LongInt): integer
vb	function <b>move</b> ( ByVal <b>target</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>move</b> ( int <b>target</b> , int <b>ms_duration</b> )
java	int <b>move</b> ( int <b>target</b> , int <b>ms_duration</b> )
py	def <b>move</b> ( <b>target</b> , <b>ms_duration</b> )
cmd	YServo <b>target move target ms_duration</b>

**Paramètres :**

**target** nouvelle position à la fin du mouvement  
**ms\_duration** durée totale du mouvement, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo**→**nextServo()**[**servo nextServo**]**YServo**

Continue l'énumération des servo commencée à l'aide de `yFirstServo()`.

js	function <b>nextServo</b> ( )
nodejs	function <b>nextServo</b> ( )
php	function <b>nextServo</b> ( )
cpp	YServo * <b>nextServo</b> ( )
m	-(YServo*) <b>nextServo</b>
pas	function <b>nextServo</b> ( ): TYServo
vb	function <b>nextServo</b> ( ) As YServo
cs	YServo <b>nextServo</b> ( )
java	YServo <b>nextServo</b> ( )
py	def <b>nextServo</b> ( )

**Retourne :**

un pointeur sur un objet `YServo` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## servo→registerValueCallback()[servo registerValueCallback: ]

YServo

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YServoValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YServoValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYServoValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**servo**→**set\_enabled()****servo**→**setEnabled()**[**servo** **setEnabled:** ]

Démarré ou arrête le \$FUNCTION\$.

js	function <b>set_enabled</b> ( <b>newval</b> )
nodejs	function <b>set_enabled</b> ( <b>newval</b> )
php	function <b>set_enabled</b> ( <b>\$newval</b> )
cpp	int <b>set_enabled</b> ( Y_ENABLED_enum <b>newval</b> )
m	-(int) <b>setEnabled</b> : (Y_ENABLED_enum) <b>newval</b>
pas	function <b>set_enabled</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabled</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabled</b> ( int <b>newval</b> )
java	int <b>set_enabled</b> ( int <b>newval</b> )
py	def <b>set_enabled</b> ( <b>newval</b> )
cmd	YServo <b>target set_enabled</b> <b>newval</b>

**Paramètres :****newval** soit Y\_ENABLED\_FALSE, soit Y\_ENABLED\_TRUE**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo→set\_enabledAtPowerOn()**  
**servo→setEnabledAtPowerOn()[servo**  
**setEnabledAtPowerOn: ]**

**YServo**

Configure l'état du générateur de signal de commande du servo au démarrage du module.

js	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_enabledAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_enabledAtPowerOn</b> ( Y_ENABLEDATPOWERON_enum <b>newval</b> )
m	-(int) <b>setEnabledAtPowerOn</b> : (Y_ENABLEDATPOWERON_enum) <b>newval</b>
pas	function <b>set_enabledAtPowerOn</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_enabledAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_enabledAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_enabledAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_enabledAtPowerOn</b> ( <b>newval</b> )
cmd	YServo <b>target</b> <b>set_enabledAtPowerOn</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash( )` du module sinon la modification n'aura aucun effet.

**Paramètres :**

**newval** soit Y\_ENABLEDATPOWERON\_FALSE, soit Y\_ENABLEDATPOWERON\_TRUE

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo**→**set\_logicalName()****YServo****servo**→**setLogicalName()**[servo setLogicalName: ]

Modifie le nom logique du servo.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YServo <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du servo.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**servo→set\_neutral()****YServo****servo→setNeutral() [servo setNeutral: ]**

Modifie la durée de l'impulsion correspondant à la position neutre du servo.

js	function <b>set_neutral</b> ( <b>newval</b> )
nodejs	function <b>set_neutral</b> ( <b>newval</b> )
php	function <b>set_neutral</b> ( <b>\$newval</b> )
cpp	int <b>set_neutral</b> ( int <b>newval</b> )
m	-(int) setNeutral : (int) <b>newval</b>
pas	function <b>set_neutral</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_neutral</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_neutral</b> ( int <b>newval</b> )
java	int <b>set_neutral</b> ( int <b>newval</b> )
py	def <b>set_neutral</b> ( <b>newval</b> )
cmd	YServo <b>target set_neutral newval</b>

La durée est spécifiée en microsecondes, et la valeur standard est 1500 [us]. Ce réglage permet de décaler la plage d'utilisation du servo. Attention, l'utilisation d'une plage supérieure aux caractéristiques du servo risque fortement d'endommager le servo.

**Paramètres :**

**newval** un entier représentant la durée de l'impulsion correspondant à la position neutre du servo

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo**→**set\_position()****servo**→**setPosition()**[servo setPosition: ]

Modifie immédiatement la consigne de position du servo.

js	function <b>set_position</b> ( <b>newval</b> )
nodejs	function <b>set_position</b> ( <b>newval</b> )
php	function <b>set_position</b> ( <b>\$newval</b> )
cpp	int <b>set_position</b> ( int <b>newval</b> )
m	-(int) setPosition : (int) <b>newval</b>
pas	function <b>set_position</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_position</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_position</b> ( int <b>newval</b> )
java	int <b>set_position</b> ( int <b>newval</b> )
py	def <b>set_position</b> ( <b>newval</b> )
cmd	YServo <b>target set_position newval</b>

**Paramètres :**

**newval** un entier représentant immédiatement la consigne de position du servo

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo→set\_positionAtPowerOn()**  
**servo→setPositionAtPowerOn()[servo**  
**setPositionAtPowerOn: ]**

**YServo**

Configure la position du servo au démarrage du module.

js	function <b>set_positionAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_positionAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_positionAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_positionAtPowerOn</b> ( int <b>newval</b> )
m	-(int) setPositionAtPowerOn : (int) <b>newval</b>
pas	function <b>set_positionAtPowerOn</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_positionAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_positionAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_positionAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_positionAtPowerOn</b> ( <b>newval</b> )
cmd	YServo <b>target</b> <b>set_positionAtPowerOn</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module sinon la modification n'aura aucun effet.

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo**→**set\_range()****servo**→**setRange()**[servo setRange: ]

Modifie la plage d'utilisation du servo, en pourcents.

js	function <b>set_range</b> ( <b>newval</b> )
nodejs	function <b>set_range</b> ( <b>newval</b> )
php	function <b>set_range</b> ( <b>\$newval</b> )
cpp	int <b>set_range</b> ( int <b>newval</b> )
m	-(int) setRange : (int) <b>newval</b>
pas	function <b>set_range</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_range</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_range</b> ( int <b>newval</b> )
java	int <b>set_range</b> ( int <b>newval</b> )
py	def <b>set_range</b> ( <b>newval</b> )
cmd	YServo <b>target set_range newval</b>

La valeur 100% correspond à un signal de commande standard, variant de 1 [ms] à 2 [ms]. Pour les servos supportent une plage double, de 0.5 [ms] à 2.5 [ms], vous pouvez utiliser une valeur allant jusqu'à 200%. Attention, l'utilisation d'une plage supérieure aux caractéristiques du servo risque fortement d'endommager le servo.

**Paramètres :**

**newval** un entier représentant la plage d'utilisation du servo, en pourcents

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**servo→set\_userdata()****YServo****servo→setUserData()[servo setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**servo→wait\_async()****YServo**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.38. Interface de la fonction Temperature

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_temperature.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YTemperature = yoctolib.YTemperature;</code>
php	<code>require_once('yocto_temperature.php');</code>
c++	<code>#include "yocto_temperature.h"</code>
m	<code>#import "yocto_temperature.h"</code>
pas	<code>uses yocto_temperature;</code>
vb	<code>yocto_temperature.vb</code>
cs	<code>yocto_temperature.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YTemperature;</code>
py	<code>from yocto_temperature import *</code>

### Fonction globales

#### **yFindTemperature(func)**

Permet de retrouver un capteur de température d'après un identifiant donné.

#### **yFirstTemperature()**

Commence l'énumération des capteurs de température accessibles par la librairie.

### Méthodes des objets YTemperature

#### **temperature→calibrateFromPoints(rawValues, refValues)**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### **temperature→describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de température au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### **temperature→get\_advertisedValue()**

Retourne la valeur courante du capteur de température (pas plus de 6 caractères).

#### **temperature→get\_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### **temperature→get\_currentValue()**

Retourne la valeur actuelle de la température.

#### **temperature→get\_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

#### **temperature→get\_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

#### **temperature→get\_friendlyName()**

Retourne un identifiant global du capteur de température au format `NOM_MODULE . NOM_FONCTION`.

#### **temperature→get\_functionDescriptor()**

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### **temperature→get\_functionId()**

Retourne l'identifiant matériel du capteur de température, sans référence au module.

#### **temperature→get\_hardwareId()**

	Retourne l'identifiant matériel unique du capteur de température au format SERIAL . FUNCTIONID.
<b>temperature</b> → <b>get_highestValue()</b>	Retourne la valeur maximale observée pour la température depuis le démarrage du module.
<b>temperature</b> → <b>get_logFrequency()</b>	Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.
<b>temperature</b> → <b>get_logicalName()</b>	Retourne le nom logique du capteur de température.
<b>temperature</b> → <b>get_lowestValue()</b>	Retourne la valeur minimale observée pour la température depuis le démarrage du module.
<b>temperature</b> → <b>get_module()</b>	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
<b>temperature</b> → <b>get_module_async(callback, context)</b>	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
<b>temperature</b> → <b>get_recordedData(startTime, endTime)</b>	Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.
<b>temperature</b> → <b>get_reportFrequency()</b>	Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.
<b>temperature</b> → <b>get_resolution()</b>	Retourne la résolution des valeurs mesurées.
<b>temperature</b> → <b>get_sensorType()</b>	Retourne le type de capteur de température utilisé par le module
<b>temperature</b> → <b>get_unit()</b>	Retourne l'unité dans laquelle la température est exprimée.
<b>temperature</b> → <b>get_userData()</b>	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.
<b>temperature</b> → <b>isOnline()</b>	Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.
<b>temperature</b> → <b>isOnline_async(callback, context)</b>	Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.
<b>temperature</b> → <b>load(msValidity)</b>	Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.
<b>temperature</b> → <b>loadCalibrationPoints(rawValues, refValues)</b>	Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.
<b>temperature</b> → <b>load_async(msValidity, callback, context)</b>	Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.
<b>temperature</b> → <b>nextTemperature()</b>	Continue l'énumération des capteurs de température commencée à l'aide de yFirstTemperature ( ).
<b>temperature</b> → <b>registerTimedReportCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque notification périodique.
<b>temperature</b> → <b>registerValueCallback(callback)</b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b>temperature</b> → <b>set_highestValue(newval)</b>	



Modifie la mémoire de valeur maximale observée.

**temperature**→**set\_logFrequency**(newval)

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**temperature**→**set\_logicalName**(newval)

Modifie le nom logique du capteur de température.

**temperature**→**set\_lowestValue**(newval)

Modifie la mémoire de valeur minimale observée.

**temperature**→**set\_reportFrequency**(newval)

Modifie la fréquence de notification périodique des valeurs mesurées.

**temperature**→**set\_resolution**(newval)

Modifie la résolution des valeurs physique mesurées.

**temperature**→**set\_sensorType**(newval)

Change le type de senseur utilisé par le module.

**temperature**→**set\_userData**(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**temperature**→**wait\_async**(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YTemperature.FindTemperature() yFindTemperature()yFindTemperature()

## YTemperature

Permet de retrouver un capteur de température d'après un identifiant donné.

js	function <b>yFindTemperature</b> ( <b>func</b> )
nodejs	function <b>FindTemperature</b> ( <b>func</b> )
php	function <b>yFindTemperature</b> ( <b>\$func</b> )
cpp	YTemperature* <b>yFindTemperature</b> ( const string& <b>func</b> )
m	YTemperature* <b>yFindTemperature</b> ( NSString* <b>func</b> )
pas	function <b>yFindTemperature</b> ( <b>func</b> : string): TYTemperature
vb	function <b>yFindTemperature</b> ( ByVal <b>func</b> As String) As YTemperature
cs	YTemperature <b>FindTemperature</b> ( string <b>func</b> )
java	YTemperature <b>FindTemperature</b> ( String <b>func</b> )
py	def <b>FindTemperature</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de température soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YTemperature.isOnline()` pour tester si le capteur de température est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de température sans ambiguïté

### Retourne :

un objet de classe `YTemperature` qui permet ensuite de contrôler le capteur de température.

## YTemperature.FirstTemperature() yFirstTemperature()yFirstTemperature()

## YTemperature

Commence l'énumération des capteurs de température accessibles par la librairie.

js	function <b>yFirstTemperature</b> ( )
nodejs	function <b>FirstTemperature</b> ( )
php	function <b>yFirstTemperature</b> ( )
cpp	YTemperature* <b>yFirstTemperature</b> ( )
m	YTemperature* <b>yFirstTemperature</b> ( )
pas	function <b>yFirstTemperature</b> ( ): TYTemperature
vb	function <b>yFirstTemperature</b> ( ) As YTemperature
cs	YTemperature <b>FirstTemperature</b> ( )
java	YTemperature <b>FirstTemperature</b> ( )
py	def <b>FirstTemperature</b> ( )

Utiliser la fonction `YTemperature.nextTemperature( )` pour itérer sur les autres capteurs de température.

### Retourne :

un pointeur sur un objet `YTemperature`, correspondant à le premier capteur de température accessible en ligne, ou `null` si il n'y a pas de capteurs de température disponibles.

## temperature→calibrateFromPoints()[temperature calibrateFromPoints: ]

YTemperature

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                   refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YTemperature target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature→describe()[temperature describe]****YTemperature**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de température au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de température (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**temperature**→**get\_advertisedValue()****YTemperature****temperature**→**advertisedValue()[temperature  
advertisedValue]**

Retourne la valeur courante du capteur de température (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YTemperature <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de température (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**temperature**→**get\_currentRawValue()**  
**temperature**→**currentRawValue()**[**temperature**  
**currentRawValue**]

**YTemperature**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YTemperature <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**temperature→get\_currentValue()**  
**temperature→currentValue()[temperature**  
**currentValue]**

**YTemperature**

Retourne la valeur actuelle de la température.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YTemperature <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de la température

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.



**temperature**→**get\_errorMessage()**  
**temperature**→**errorMessage()**[**temperature**  
**errorMessage**]

**YTemperature**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de température.

**temperature→get\_errorType()**

**YTemperature**

**temperature→errorType()**

---

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de température.

**temperature**→**get\_friendlyName()**  
**temperature**→**friendlyName()**[**temperature**  
**friendlyName**]

**YTemperature**

Retourne un identifiant global du capteur de température au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de température si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de température (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de température en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**temperature→get\_functionDescriptor()**  
**temperature→functionDescriptor()[temperature**  
**functionDescriptor]**

**YTemperature**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**temperature**→**get\_functionId()****YTemperature****temperature**→**functionId()**[**temperature functionId**]

Retourne l'identifiant matériel du capteur de température, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de température (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**temperature→get\_hardwareId()**

**YTemperature**

**temperature→hardwareId()[temperature hardwareId]**

Retourne l'identifiant matériel unique du capteur de température au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de température (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de température (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**temperature**→**get\_highestValue()**  
**temperature**→**highestValue()**[**temperature**  
**highestValue**]

**YTemperature**

Retourne la valeur maximale observée pour la température depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YTemperature <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la température depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**temperature→get\_logFrequency()**  
**temperature→logFrequency()[temperature**  
**logFrequency]**

**YTemperature**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YTemperature <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.



**temperature**→**get\_logicalName()**  
**temperature**→**logicalName()**[**temperature**  
**logicalName**]

**YTemperature**

Retourne le nom logique du capteur de température.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YTemperature <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de température. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**temperature**→**get\_lowestValue()**

**YTemperature**

**temperature**→**lowestValue()**[**temperature**  
**lowestValue**]

Retourne la valeur minimale observée pour la température depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YTemperature <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la température depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**temperature**→**get\_module()****YTemperature****temperature**→**module()**[temperature module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	<code>YModule *</code> <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

temperature→get\_module\_async()

YTemperature

temperature→module\_async()

---

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module_async( callback, context)
nodejs	function get_module_async( callback, context)

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**temperature**→**get\_recordedData()**  
**temperature**→**recordedData()**[**temperature**  
**recordedData: ]**

**YTemperature**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YTemperature <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

#### Paramètres :

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

#### Retourne :

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**temperature**→**get\_reportFrequency()****YTemperature****temperature**→**reportFrequency()**[**temperature**  
**reportFrequency**]

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YTemperature <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**temperature**→**get\_resolution()****YTemperature****temperature**→**resolution()**[**temperature resolution**]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YTemperature <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**temperature→get\_sensorType()****YTemperature****temperature→sensorType()[temperature sensorType]**

Retourne le type de capteur de température utilisé par le module

js	function <b>get_sensorType</b> ( )
nodejs	function <b>get_sensorType</b> ( )
php	function <b>get_sensorType</b> ( )
cpp	Y_SENSORTYPE_enum <b>get_sensorType</b> ( )
m	-(Y_SENSORTYPE_enum) sensorType
pas	function <b>get_sensorType</b> ( ): Integer
vb	function <b>get_sensorType</b> ( ) As Integer
cs	int <b>get_sensorType</b> ( )
java	int <b>get_sensorType</b> ( )
py	def <b>get_sensorType</b> ( )
cmd	YTemperature <b>target</b> <b>get_sensorType</b>

**Retourne :**

une valeur parmi Y\_SENSORTYPE\_DIGITAL, Y\_SENSORTYPE\_TYPE\_K, Y\_SENSORTYPE\_TYPE\_E, Y\_SENSORTYPE\_TYPE\_J, Y\_SENSORTYPE\_TYPE\_N, Y\_SENSORTYPE\_TYPE\_R, Y\_SENSORTYPE\_TYPE\_S, Y\_SENSORTYPE\_TYPE\_T, Y\_SENSORTYPE\_PT100\_4WIRES, Y\_SENSORTYPE\_PT100\_3WIRES et Y\_SENSORTYPE\_PT100\_2WIRES représentant le type de capteur de température utilisé par le module

En cas d'erreur, déclenche une exception ou retourne Y\_SENSORTYPE\_INVALID.



**temperature**→**get\_unit()****YTemperature****temperature**→**unit()**[temperature unit]

Retourne l'unité dans laquelle la température est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YTemperature <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la température est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**temperature→get\_userdata()**

**YTemperature**

**temperature→userData()[temperature userData]**

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**temperature→isOnline()[temperature isOnline]****YTemperature**

Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de température sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le capteur de température est joignable, `false` sinon

**temperature→isOnline\_async()****YTemperature**

Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de température sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**temperature→load()[temperature load: ]****YTemperature**

Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.

js	function load( msValidity)
nodejs	function load( msValidity)
php	function load( \$msValidity)
cpp	YRETCODE load( int msValidity)
m	-(YRETCODE) load : (int) msValidity
pas	function load( msValidity: integer): YRETCODE
vb	function load( ByVal msValidity As Integer) As YRETCODE
cs	YRETCODE load( int msValidity)
java	int load( long msValidity)
py	def load( msValidity)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## temperature→loadCalibrationPoints()[temperature loadCalibrationPoints: ]

YTemperature

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
node.js function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                    var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YTemperature target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature→load\_async()****YTemperature**

Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## temperature→nextTemperature()[temperature nextTemperature]

YTemperature

Continue l'énumération des capteurs de température commencée à l'aide de `yFirstTemperature()`.

js	function <b>nextTemperature</b> ( )
nodejs	function <b>nextTemperature</b> ( )
php	function <b>nextTemperature</b> ( )
cpp	YTemperature * <b>nextTemperature</b> ( )
m	-(YTemperature*) <b>nextTemperature</b>
pas	function <b>nextTemperature</b> ( ): TYTemperature
vb	function <b>nextTemperature</b> ( ) As YTemperature
cs	YTemperature <b>nextTemperature</b> ( )
java	YTemperature <b>nextTemperature</b> ( )
py	def <b>nextTemperature</b> ( )

### Retourne :

un pointeur sur un objet YTemperature accessible en ligne, ou `null` lorsque l'énumération est terminée.



## temperature→registerTimedReportCallback() [temperature registerTimedReportCallback: ]

## YTemperature

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YTemperatureTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YTemperatureTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYTemperatureTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## temperature→registerValueCallback()[temperature registerValueCallback: ]

YTemperature

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YTemperatureValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YTemperatureValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYTemperatureValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**temperature→set\_highestValue()**  
**temperature→setHighestValue()[temperature**  
**setHighestValue: ]**

**YTemperature**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_highestValue</b> <b>newval</b>

#### Paramètres :

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

temperature→set\_logFrequency()

YTemperature

temperature→setLogFrequency()[temperature  
setLogFrequency: ]

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function set_logFrequency( newval)
nodejs	function set_logFrequency( newval)
php	function set_logFrequency( \$newval)
cpp	int set_logFrequency( const string& newval)
m	-(int) setLogFrequency : (NSString*) newval
pas	function set_logFrequency( newval: string): integer
vb	function set_logFrequency( ByVal newval As String) As Integer
cs	int set_logFrequency( string newval)
java	int set_logFrequency( String newval)
py	def set_logFrequency( newval)
cmd	YTemperature target set_logFrequency newval

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature**→**set\_logicalName()**  
**temperature**→**setLogicalName()**[**temperature**  
**setLogicalName: ]**

**YTemperature**

Modifie le nom logique du capteur de température.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

#### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du capteur de température.

#### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature**→**set\_lowestValue()****YTemperature****temperature**→**setLowestValue()**[**temperature**  
**setLowestValue: ]**

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature→set\_reportFrequency()****YTemperature****temperature→setReportFrequency()[temperature  
setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature**→**set\_resolution()****YTemperature****temperature**→**setResolution()**[**temperature**  
**setResolution:** ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**temperature→set\_sensorType()****YTemperature****temperature→setSensorType()[temperature  
setSensorType: ]**

Change le type de senseur utilisé par le module.

js	function <b>set_sensorType</b> ( <b>newval</b> )
nodejs	function <b>set_sensorType</b> ( <b>newval</b> )
php	function <b>set_sensorType</b> ( <b>\$newval</b> )
cpp	int <b>set_sensorType</b> ( Y_SENSORTYPE_enum <b>newval</b> )
m	-(int) setSensorType : (Y_SENSORTYPE_enum) <b>newval</b>
pas	function <b>set_sensorType</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_sensorType</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_sensorType</b> ( int <b>newval</b> )
java	int <b>set_sensorType</b> ( int <b>newval</b> )
py	def <b>set_sensorType</b> ( <b>newval</b> )
cmd	YTemperature <b>target</b> <b>set_sensorType</b> <b>newval</b>

Cette fonction sert à spécifier le type de thermocouple (K,E, etc..) raccordé au module. Cette fonction n'aura pas d'effet si le module utilise un capteur digital. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une valeur parmi Y\_SENSORTYPE\_DIGITAL, Y\_SENSORTYPE\_TYPE\_K, Y\_SENSORTYPE\_TYPE\_E, Y\_SENSORTYPE\_TYPE\_J, Y\_SENSORTYPE\_TYPE\_N, Y\_SENSORTYPE\_TYPE\_R, Y\_SENSORTYPE\_TYPE\_S, Y\_SENSORTYPE\_TYPE\_T, Y\_SENSORTYPE\_PT100\_4WIRES, Y\_SENSORTYPE\_PT100\_3WIRES et Y\_SENSORTYPE\_PT100\_2WIRES

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**temperature**→**set\_userdata()****YTemperature****temperature**→**setUserData()**[**temperature**  
**setUserData:** ]

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :****data** objet quelconque à mémoriser

**temperature→wait\_async()****YTemperature**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

### 3.39. Interface de la fonction Tilt

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrémas atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_tilt.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YTilt = yoctolib.YTilt;
php	require_once('yocto_tilt.php');
c++	#include "yocto_tilt.h"
m	#import "yocto_tilt.h"
pas	uses yocto_tilt;
vb	yocto_tilt.vb
cs	yocto_tilt.cs
java	import com.yoctopuce.YoctoAPI.YTilt;
py	from yocto_tilt import *

Fonction globales
<b>yFindTilt(func)</b> Permet de retrouver un inclinomètre d'après un identifiant donné.
<b>yFirstTilt()</b> Commence l'énumération des inclinomètres accessibles par la librairie.
Méthodes des objets YTilt
<b>tilt→calibrateFromPoints(rawValues, refValues)</b> Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.
<b>tilt→describe()</b> Retourne un court texte décrivant de manière non-ambigüe l'instance de l'inclinomètre au format TYPE ( NAME ) = SERIAL . FUNCTIONID.
<b>tilt→get_advertisedValue()</b> Retourne la valeur courante de l'inclinomètre (pas plus de 6 caractères).
<b>tilt→get_currentRawValue()</b> Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).
<b>tilt→get_currentValue()</b> Retourne la valeur actuelle de l'inclinaison.
<b>tilt→get_errorMessage()</b> Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.
<b>tilt→get_errorType()</b> Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.
<b>tilt→get_friendlyName()</b> Retourne un identifiant global de l'inclinomètre au format NOM_MODULE . NOM_FONCTION.
<b>tilt→get_functionDescriptor()</b> Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
<b>tilt→get_functionId()</b> Retourne l'identifiant matériel de l'inclinomètre, sans référence au module.
<b>tilt→get_hardwareId()</b> Retourne l'identifiant matériel unique de l'inclinomètre au format SERIAL . FUNCTIONID.

**tilt→get\_highestValue()**

Retourne la valeur maximale observée pour l'inclinaison depuis le démarrage du module.

**tilt→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**tilt→get\_logicalName()**

Retourne le nom logique de l'inclinomètre.

**tilt→get\_lowestValue()**

Retourne la valeur minimale observée pour l'inclinaison depuis le démarrage du module.

**tilt→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**tilt→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**tilt→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**tilt→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**tilt→get\_resolution()**

Retourne la résolution des valeurs mesurées.

**tilt→get\_unit()**

Retourne l'unité dans laquelle l'inclinaison est exprimée.

**tilt→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**tilt→isOnline()**

Vérifie si le module hébergeant l'inclinomètre est joignable, sans déclencher d'erreur.

**tilt→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'inclinomètre est joignable, sans déclencher d'erreur.

**tilt→load(msValidity)**

Met en cache les valeurs courantes de l'inclinomètre, avec une durée de validité spécifiée.

**tilt→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

**tilt→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'inclinomètre, avec une durée de validité spécifiée.

**tilt→nextTilt()**

Continue l'énumération des inclinomètres commencée à l'aide de `yFirstTilt()`.

**tilt→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**tilt→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**tilt→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée.

**tilt→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

### 3. Reference

#### **tilt→set\_logicalName(newval)**

Modifie le nom logique de l'inclinomètre.

#### **tilt→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée.

#### **tilt→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

#### **tilt→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

#### **tilt→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

#### **tilt→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

**YTilt.FindTilt()****YTilt****yFindTilt()yFindTilt()**

Permet de retrouver un inclinomètre d'après un identifiant donné.

js	function <b>yFindTilt</b> ( <b>func</b> )
nodejs	function <b>FindTilt</b> ( <b>func</b> )
php	function <b>yFindTilt</b> ( <b>\$func</b> )
cpp	YTilt* <b>yFindTilt</b> ( const string& <b>func</b> )
m	YTilt* <b>yFindTilt</b> ( NSString* <b>func</b> )
pas	function <b>yFindTilt</b> ( <b>func</b> : string): YTilt
vb	function <b>yFindTilt</b> ( ByVal <b>func</b> As String) As YTilt
cs	YTilt <b>FindTilt</b> ( string <b>func</b> )
java	YTilt <b>FindTilt</b> ( String <b>func</b> )
py	def <b>FindTilt</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'inclinomètre soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YTilt.isOnline()` pour tester si l'inclinomètre est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

**Paramètres :**

**func** une chaîne de caractères qui référence l'inclinomètre sans ambiguïté

**Retourne :**

un objet de classe `YTilt` qui permet ensuite de contrôler l'inclinomètre.

## YTilt.FirstTilt() yFirstTilt()yFirstTilt()

YTilt

Commence l'énumération des inclinomètres accessibles par la librairie.

js	function <b>yFirstTilt</b> ( )
nodejs	function <b>FirstTilt</b> ( )
php	function <b>yFirstTilt</b> ( )
cpp	YTilt* <b>yFirstTilt</b> ( )
m	YTilt* <b>yFirstTilt</b> ( )
pas	function <b>yFirstTilt</b> ( ): TYTilt
vb	function <b>yFirstTilt</b> ( ) As YTilt
cs	YTilt <b>FirstTilt</b> ( )
java	YTilt <b>FirstTilt</b> ( )
py	def <b>FirstTilt</b> ( )

Utiliser la fonction `YTilt.nextTilt()` pour itérer sur les autres inclinomètres.

### Retourne :

un pointeur sur un objet `YTilt`, correspondant à le premier inclinomètre accessible en ligne, ou `null` si il n'y a pas de inclinomètres disponibles.



**tilt→calibrateFromPoints()[tilt calibrateFromPoints: ]****YTilt**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                           vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
                           : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                           List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                             ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YTilt target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

**Paramètres :**

**rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.

**refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt→describe()[tilt describe]****YTilt**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'inclinomètre au format  
 TYPE ( NAME ) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.relay1)=unresolved si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant l'inclinomètre (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

**tilt→get\_advertisedValue()****YTilt****tilt→advertisedValue()[tilt advertisedValue]**

Retourne la valeur courante de l'inclinomètre (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YTilt <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de l'inclinomètre (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**tilt→get\_currentRawValue()****YTilt****tilt→currentRawValue()[tilt currentRawValue]**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YTilt <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**tilt**→**get\_currentValue()****YTilt****tilt**→**currentValue()**[tilt currentValue]

Retourne la valeur actuelle de l'inclinaison.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YTilt <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur actuelle de l'inclinaison

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**tilt**→**get\_errorMessage()****tilt**→**errorMessage()**[tilt errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'inclinomètre.

**tilt→get\_errorType()****YTilt****tilt→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'inclinomètre.

**tilt→get\_friendlyName()****tilt→friendlyName()[tilt friendlyName]**

Retourne un identifiant global de l'inclinomètre au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
c++	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'inclinomètre si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'inclinomètre (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'inclinomètre en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.



**tilt→get\_functionDescriptor()****YTilt****tilt→functionDescriptor()[tilt functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**tilt**→**get\_functionId()****tilt**→**functionId()**[**tilt functionId**]

Retourne l'identifiant matériel de l'inclinomètre, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'inclinomètre (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**tilt→get\_hardwareId()****YTilt****tilt→hardwareId()[tilt hardwareId]**

Retourne l'identifiant matériel unique de l'inclinomètre au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'inclinomètre (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'inclinomètre (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**tilt→get\_highestValue()****tilt→highestValue()[tilt highestValue]**

Retourne la valeur maximale observée pour l'inclinaison depuis le démarrage du module.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YTilt <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour l'inclinaison depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**tilt→get\_logFrequency()****YTilt****tilt→logFrequency()[tilt logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YTilt <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**tilt→get\_logicalName()****tilt→logicalName()[tilt logicalName]**

Retourne le nom logique de l'inclinomètre.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YTilt <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'inclinomètre. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**tilt→get\_lowestValue()****YTilt****tilt→lowestValue()[tilt lowestValue]**

Retourne la valeur minimale observée pour l'inclinaison depuis le démarrage du module.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YTilt <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour l'inclinaison depuis le démarrage du module

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**tilt**→**get\_module()****tilt**→**module()**[tilt module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`



**tilt**→**get\_module\_async()****YTilt****tilt**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**tilt→get\_recordedData()****tilt→recordedData()[tilt recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YTilt <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**tilt→get\_reportFrequency()****YTilt****tilt→reportFrequency()[tilt reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YTilt <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**tilt**→**get\_resolution()****tilt**→**resolution()**[tilt resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YTilt <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**tilt**→**get\_unit()****YTilt****tilt**→**unit()**[tilt unit]

Retourne l'unité dans laquelle l'inclinaison est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YTilt <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle l'inclinaison est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**tilt→get\_userdata()****tilt→userdata()[tilt userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**tilt→isOnline())[tilt isOnline]****YTilt**

Vérifie si le module hébergeant l'inclinomètre est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'inclinomètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'inclinomètre est joignable, `false` sinon

Vérifie si le module hébergeant l'inclinomètre est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'inclinomètre sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**tilt→load()[tilt load: ]****YTilt**

Met en cache les valeurs courantes de l'inclinomètre, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## tilt→loadCalibrationPoints()[tilt loadCalibrationPoints: ]

YTilt

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YTilt target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## tilt→load\_async()

YTilt

Met en cache les valeurs courantes de l'inclinomètre, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**tilt**→**nextTilt()**[**tilt nextTilt**]**YTilt**

Continue l'énumération des inclinomètres commencée à l'aide de `yFirstTilt()`.

js	function <b>nextTilt</b> ( )
nodejs	function <b>nextTilt</b> ( )
php	function <b>nextTilt</b> ( )
cpp	YTilt * <b>nextTilt</b> ( )
m	-(YTilt*) <b>nextTilt</b>
pas	function <b>nextTilt</b> ( ): TYTilt
vb	function <b>nextTilt</b> ( ) As YTilt
cs	YTilt <b>nextTilt</b> ( )
java	YTilt <b>nextTilt</b> ( )
py	def <b>nextTilt</b> ( )

**Retourne :**

un pointeur sur un objet `YTilt` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## tilt→registerTimedReportCallback()[tilt registerTimedReportCallback: ]

YTilt

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YTiltTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YTiltTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYTiltTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## tilt→registerValueCallback()[tilt registerValueCallback: ]

YTilt

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function registerValueCallback( callback)
nodejs	function registerValueCallback( callback)
php	function registerValueCallback( \$callback)
cpp	int registerValueCallback( YTiltValueCallback callback)
m	-(int) registerValueCallback : (YTiltValueCallback) callback
pas	function registerValueCallback( callback: TYTiltValueCallback): LongInt
vb	function registerValueCallback( ) As Integer
cs	int registerValueCallback( ValueCallback callback)
java	int registerValueCallback( UpdateCallback callback)
py	def registerValueCallback( callback)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**tilt→set\_highestValue()****YTilt****tilt→setHighestValue()[tilt setHighestValue: ]**

Modifie la mémoire de valeur maximale observée.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YTilt <b>target set_highestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt→set\_logFrequency()****tilt→setLogFrequency()[tilt setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YTilt <b>target set_logFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**tilt→set\_logicalName()****YTilt****tilt→setLogicalName()[tilt setLogicalName: ]**

Modifie le nom logique de l'inclinomètre.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YTilt <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'inclinomètre.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt**→**set\_lowestValue()****tilt**→**setLowestValue()**[tilt setLowestValue: ]

Modifie la mémoire de valeur minimale observée.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YTilt <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt→set\_reportFrequency()****YTilt****tilt→setReportFrequency()[tilt setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YTilt <b>target set_reportFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt**→**set\_resolution()****tilt**→**setResolution()**[tilt setResolution: ]

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YTilt <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**tilt→set\_userdata()****YTilt****tilt→setUserData()[tilt setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**tilt→wait\_async()**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.40. Interface de la fonction Voc

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<code>&lt;script type='text/javascript' src='yocto_voc.js'&gt;&lt;/script&gt;</code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YVoc = yoctolib.YVoc;</code>
php	<code>require_once('yocto_voc.php');</code>
c++	<code>#include "yocto_voc.h"</code>
m	<code>#import "yocto_voc.h"</code>
pas	<code>uses yocto_voc;</code>
vb	<code>yocto_voc.vb</code>
cs	<code>yocto_voc.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YVoc;</code>
py	<code>from yocto_voc import *</code>

### Fonction globales

#### yFindVoc(func)

Permet de retrouver un capteur de Composés Organiques Volatils d'après un identifiant donné.

#### yFirstVoc()

Commence l'énumération des capteurs de Composés Organiques Volatils accessibles par la librairie.

### Méthodes des objets YVoc

#### voc→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### voc→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de Composés Organiques Volatils au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### voc→get\_advertisedValue()

Retourne la valeur courante du capteur de Composés Organiques Volatils (pas plus de 6 caractères).

#### voc→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### voc→get\_currentValue()

Retourne la mesure actuelle du taux de VOC estimé.

#### voc→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

#### voc→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

#### voc→get\_friendlyName()

Retourne un identifiant global du capteur de Composés Organiques Volatils au format `NOM_MODULE . NOM_FONCTION`.

#### voc→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### voc→get\_functionId()

Retourne l'identifiant matériel du capteur de Composés Organiques Volatils, sans référence au module.

### 3. Reference

#### **voc→get\_hardwareId()**

Retourne l'identifiant matériel unique du capteur de Composés Organiques Volatils au format `SERIAL.FUNCTIONID`.

#### **voc→get\_highestValue()**

Retourne la valeur maximale observée pour le taux de VOC estimé.

#### **voc→get\_logFrequency()**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

#### **voc→get\_logicalName()**

Retourne le nom logique du capteur de Composés Organiques Volatils.

#### **voc→get\_lowestValue()**

Retourne la valeur minimale observée pour le taux de VOC estimé.

#### **voc→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **voc→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **voc→get\_recordedData(startTime, endTime)**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

#### **voc→get\_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

#### **voc→get\_resolution()**

Retourne la résolution des valeurs mesurées.

#### **voc→get\_unit()**

Retourne l'unité dans laquelle le taux de VOC estimé est exprimée.

#### **voc→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **voc→isOnline()**

Vérifie si le module hébergeant le capteur de Composés Organiques Volatils est joignable, sans déclencher d'erreur.

#### **voc→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le capteur de Composés Organiques Volatils est joignable, sans déclencher d'erreur.

#### **voc→load(msValidity)**

Met en cache les valeurs courantes du capteur de Composés Organiques Volatils, avec une durée de validité spécifiée.

#### **voc→loadCalibrationPoints(rawValues, refValues)**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

#### **voc→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du capteur de Composés Organiques Volatils, avec une durée de validité spécifiée.

#### **voc→nextVoc()**

Continue l'énumération des capteurs de Composés Organiques Volatils commencée à l'aide de `yFirstVoc()`.

#### **voc→registerTimedReportCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.



**voc→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**voc→set\_highestValue(newval)**

Modifie la mémoire de valeur maximale observée pour le taux de VOC estimé.

**voc→set\_logFrequency(newval)**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**voc→set\_logicalName(newval)**

Modifie le nom logique du capteur de Composés Organiques Volatils.

**voc→set\_lowestValue(newval)**

Modifie la mémoire de valeur minimale observée pour le taux de VOC estimé.

**voc→set\_reportFrequency(newval)**

Modifie la fréquence de notification périodique des valeurs mesurées.

**voc→set\_resolution(newval)**

Modifie la résolution des valeurs physique mesurées.

**voc→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**voc→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YVoc.FindVoc() yFindVoc()yFindVoc()

YVoc

Permet de retrouver un capteur de Composés Organiques Volatils d'après un identifiant donné.

js	function <b>yFindVoc</b> ( <b>func</b> )
nodejs	function <b>FindVoc</b> ( <b>func</b> )
php	function <b>yFindVoc</b> ( <b>\$func</b> )
cpp	YVoc* <b>yFindVoc</b> ( const string& <b>func</b> )
m	YVoc* <b>yFindVoc</b> ( NSString* <b>func</b> )
pas	function <b>yFindVoc</b> ( <b>func</b> : string): TYVoc
vb	function <b>yFindVoc</b> ( ByVal <b>func</b> As String) As YVoc
cs	YVoc <b>FindVoc</b> ( string <b>func</b> )
java	YVoc <b>FindVoc</b> ( String <b>func</b> )
py	def <b>FindVoc</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de Composés Organiques Volatils soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YVoc.isOnline()` pour tester si le capteur de Composés Organiques Volatils est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de Composés Organiques Volatils sans ambiguïté

### Retourne :

un objet de classe `YVoc` qui permet ensuite de contrôler le capteur de Composés Organiques Volatils.

## YVoc.FirstVoc() yFirstVoc()yFirstVoc()

## YVoc

Commence l'énumération des capteurs de Composés Organiques Volatils accessibles par la librairie.

js	function <b>yFirstVoc</b> ( )
nodejs	function <b>FirstVoc</b> ( )
php	function <b>yFirstVoc</b> ( )
cpp	YVoc* <b>yFirstVoc</b> ( )
m	YVoc* <b>yFirstVoc</b> ( )
pas	function <b>yFirstVoc</b> ( ): TYVoc
vb	function <b>yFirstVoc</b> ( ) As YVoc
cs	YVoc <b>FirstVoc</b> ( )
java	YVoc <b>FirstVoc</b> ( )
py	def <b>FirstVoc</b> ( )

Utiliser la fonction `YVoc.nextVoc( )` pour itérer sur les autres capteurs de Composés Organiques Volatils.

### Retourne :

un pointeur sur un objet `YVoc`, correspondant à le premier capteur de Composés Organiques Volatils accessible en ligne, ou `null` si il n'y a pas de capteurs de Composés Organiques Volatils disponibles.

**voc→calibrateFromPoints()[voc calibrateFromPoints:  
]****YVoc**

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js function calibrateFromPoints( rawValues, refValues)
nodejs function calibrateFromPoints( rawValues, refValues)
php function calibrateFromPoints( $rawValues, $refValues)
cpp int calibrateFromPoints( vector<double> rawValues,
                             vector<double> refValues)

m -(int) calibrateFromPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function calibrateFromPoints( rawValues: TDoubleArray,
                                  refValues: TDoubleArray): LongInt

vb procedure calibrateFromPoints( )
cs int calibrateFromPoints( List<double> rawValues,
                             List<double> refValues)

java int calibrateFromPoints( ArrayList<Double> rawValues,
                              ArrayList<Double> refValues)

py def calibrateFromPoints( rawValues, refValues)
cmd YVoc target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

**Paramètres :**

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc→describe()[voc describe]****YVoc**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de Composés Organiques Volatils au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de Composés Organiques Volatils (ex: `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**voc**→**get\_advertisedValue()****YVoc****voc**→**advertisedValue()**[**voc advertisedValue**]

Retourne la valeur courante du capteur de Composés Organiques Volatils (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YVoc <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de Composés Organiques Volatils (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**voc**→**get\_currentRawValue()****YVoc****voc**→**currentRawValue()**[**voc currentRawValue**]

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YVoc <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**voc**→**get\_currentValue()****YVoc****voc**→**currentValue()**[voc currentValue]

Retourne la mesure actuelle du taux de VOC estimé.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YVoc <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la mesure actuelle du taux de VOC estimé

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.



**voc→get\_errorMessage()****YVoc****voc→errorMessage()[voc errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de Composés Organiques Volatils.

**voc**→**get\_errorType()****YVoc****voc**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de Composés Organiques Volatils.

**voc**→**get\_friendlyName()****YVoc****voc**→**friendlyName()**[**voc friendlyName**]

Retourne un identifiant global du capteur de Composés Organiques Volatils au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de Composés Organiques Volatils si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de Composés Organiques Volatils (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de Composés Organiques Volatils en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**voc**→**get\_functionDescriptor()****YVoc****voc**→**functionDescriptor()[voc functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**voc**→**get\_functionId()****YVoc****voc**→**functionId()**[**voc functionId**]

Retourne l'identifiant matériel du capteur de Composés Organiques Volatils, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de Composés Organiques Volatils (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**voc**→**get\_hardwareId()****YVoc****voc**→**hardwareId()**[**voc hardwareId**]

Retourne l'identifiant matériel unique du capteur de Composés Organiques Volatils au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de Composés Organiques Volatils (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de Composés Organiques Volatils (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**voc**→**get\_highestValue()****YVoc****voc**→**highestValue()**[voc highestValue]

Retourne la valeur maximale observée pour le taux de VOC estimé.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YVoc <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour le taux de VOC estimé

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**voc**→**get\_logFrequency()****YVoc****voc**→**logFrequency()**[**voc logFrequency**]

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YVoc <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.



**voc**→**get\_logicalName()****YVoc****voc**→**logicalName()**[voc logicalName]

Retourne le nom logique du capteur de Composés Organiques Volatils.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YVoc <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de Composés Organiques Volatils. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**voc**→**get\_lowestValue()****YVoc****voc**→**lowestValue()**[voc lowestValue]

Retourne la valeur minimale observée pour le taux de VOC estimé.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YVoc <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour le taux de VOC estimé

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**voc**→**get\_module()****YVoc****voc**→**module()[voc module]**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	<code>YModule *</code> <b>get_module</b> ( )
m	-( <code>YModule*</code> ) module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**voc**→**get\_module\_async()****YVoc****voc**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voc→get\_recordedData()****YVoc****voc→recordedData()[voc recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
cpp	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YVoc <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intercalé de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.

**voc→get\_reportFrequency()****YVoc****voc→reportFrequency()[voc reportFrequency]**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
cpp	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YVoc <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**voc→get\_resolution()****YVoc****voc→resolution()[voc resolution]**

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YVoc <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**voc**→**get\_unit()****voc**→**unit()**[**voc unit**]

Retourne l'unité dans laquelle le taux de VOC estimé est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YVoc <b>target get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle le taux de VOC estimé est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.



**voc→get\_userdata()****YVoc****voc→userdata()[voc userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**voc**→**isOnline()**[voc isOnline]**YVoc**

Vérifie si le module hébergeant le capteur de Composés Organiques Volatils est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de Composés Organiques Volatils sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le capteur de Composés Organiques Volatils est joignable, false sinon

**voc→isOnline\_async()****YVoc**

Vérifie si le module hébergeant le capteur de Composés Organiques Volatils est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de Composés Organiques Volatils sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voc→load()[voc load: ]****YVoc**

Met en cache les valeurs courantes du capteur de Composés Organiques Volatils, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## voc→loadCalibrationPoints()[voc loadCalibrationPoints: ]

YVoc

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
nodejs function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
                             : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YVoc target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc**→**load\_async()****YVoc**

Met en cache les valeurs courantes du capteur de Composés Organiques Volatils, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voc→nextVoc()[voc nextVoc]****YVoc**

Continue l'énumération des capteurs de Composés Organiques Volatils commencée à l'aide de `yFirstVoc()`.

js	function <b>nextVoc</b> ( )
nodejs	function <b>nextVoc</b> ( )
php	function <b>nextVoc</b> ( )
cpp	YVoc * <b>nextVoc</b> ( )
m	-(YVoc*) <b>nextVoc</b>
pas	function <b>nextVoc</b> ( ): TYVoc
vb	function <b>nextVoc</b> ( ) As YVoc
cs	YVoc <b>nextVoc</b> ( )
java	YVoc <b>nextVoc</b> ( )
py	def <b>nextVoc</b> ( )

**Retourne :**

un pointeur sur un objet `YVoc` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**voc→registerTimedReportCallback()[voc  
registerTimedReportCallback: ]****YVoc**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YVocTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YVocTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYVocTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

**Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.



## voc→registerValueCallback()[voc registerValueCallback: ]

YVoc

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
c++	int <b>registerValueCallback</b> ( YVocValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YVocValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYVocValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**voc**→**set\_highestValue()****YVoc****voc**→**setHighestValue()**[**voc setHighestValue:** ]

Modifie la mémoire de valeur maximale observée pour le taux de VOC estimé.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YVoc <b>target set_highestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour le taux de VOC estimé

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc→set\_logFrequency()****YVoc****voc→setLogFrequency()[voc setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YVoc <b>target set_logFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc**→**set\_logicalName()****YVoc****voc**→**setLogicalName()**[**voc setLogicalName:** ]

Modifie le nom logique du capteur de Composés Organiques Volatils.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YVoc <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de Composés Organiques Volatils.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc**→**set\_lowestValue()****YVoc****voc**→**setLowestValue()**[**voc setLowestValue:** ]

Modifie la mémoire de valeur minimale observée pour le taux de VOC estimé.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YVoc <b>target set_lowestValue newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour le taux de VOC estimé

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc→set\_reportFrequency()**  
**voc→setReportFrequency()[voc**  
**setReportFrequency: ]**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YVoc <b>target set_reportFrequency newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc→set\_resolution()****YVoc****voc→setResolution()[voc setResolution: ]**

Modifie la résolution des valeurs physique mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YVoc <b>target set_resolution newval</b>

La résolution correspond à la précision de l'affichage des mesures. Elle ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs physique mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voc**→**set\_userdata()****voc**→**setUserData()**[**voc setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser



**voc→wait\_async()****YVoc**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.41. Interface de la fonction Voltage

La librairie de programmation Yoctopuce permet lire une valeur instantanée du capteur, ainsi que les extrêmes atteints.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_voltage.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YVoltage = yoctolib.YVoltage;
php	require_once('yocto_voltage.php');
c++	#include "yocto_voltage.h"
m	#import "yocto_voltage.h"
pas	uses yocto_voltage;
vb	yocto_voltage.vb
cs	yocto_voltage.cs
java	import com.yoctopuce.YoctoAPI.YVoltage;
py	from yocto_voltage import *

### Fonction globales

#### yFindVoltage(func)

Permet de retrouver un capteur de tension d'après un identifiant donné.

#### yFirstVoltage()

Commence l'énumération des capteurs de tension accessibles par la librairie.

### Méthodes des objets YVoltage

#### voltage→calibrateFromPoints(rawValues, refValues)

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

#### voltage→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de tension au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### voltage→get\_advertisedValue()

Retourne la valeur courante du capteur de tension (pas plus de 6 caractères).

#### voltage→get\_currentRawValue()

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

#### voltage→get\_currentValue()

Retourne la valeur instantanée de la tension.

#### voltage→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

#### voltage→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

#### voltage→get\_friendlyName()

Retourne un identifiant global du capteur de tension au format `NOM_MODULE . NOM_FONCTION`.

#### voltage→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### voltage→get\_functionId()

Retourne l'identifiant matériel du capteur de tension, sans référence au module.

#### voltage→get\_hardwareId()

Retourne l'identifiant matériel unique du capteur de tension au format `SERIAL.FUNCTIONID`.

**`voltage→get_highestValue()`**

Retourne la valeur maximale observée pour la tension.

**`voltage→get_logFrequency()`**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

**`voltage→get_logicalName()`**

Retourne le nom logique du capteur de tension.

**`voltage→get_lowestValue()`**

Retourne la valeur minimale observée pour la tension.

**`voltage→get_module()`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`voltage→get_module_async(callback, context)`**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**`voltage→get_recordedData(startTime, endTime)`**

Retourne un objet `DataSet` représentant des mesures de ce capteur précédemment enregistrées à l'aide du `DataLogger`, pour l'intervalle de temps spécifié.

**`voltage→get_reportFrequency()`**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

**`voltage→get_resolution()`**

Retourne la résolution des valeurs mesurées.

**`voltage→get_unit()`**

Retourne l'unité dans laquelle la tension est exprimée.

**`voltage→get_userData()`**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**`voltage→isOnline()`**

Vérifie si le module hébergeant le capteur de tension est joignable, sans déclencher d'erreur.

**`voltage→isOnline_async(callback, context)`**

Vérifie si le module hébergeant le capteur de tension est joignable, sans déclencher d'erreur.

**`voltage→load(msValidity)`**

Met en cache les valeurs courantes du capteur de tension, avec une durée de validité spécifiée.

**`voltage→loadCalibrationPoints(rawValues, refValues)`**

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

**`voltage→load_async(msValidity, callback, context)`**

Met en cache les valeurs courantes du capteur de tension, avec une durée de validité spécifiée.

**`voltage→nextVoltage()`**

Continue l'énumération des capteurs de tension commencée à l'aide de `yFirstVoltage()`.

**`voltage→registerTimedReportCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

**`voltage→registerValueCallback(callback)`**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**`voltage→set_highestValue(newval)`**

Modifie la mémoire de valeur maximale observée pour la tension.

**`voltage→set_logFrequency(newval)`**

### 3. Reference

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

**voltage**→**set\_logicalName**(**newval**)

Modifie le nom logique du capteur de tension.

**voltage**→**set\_lowestValue**(**newval**)

Modifie la mémoire de valeur minimale observée pour la tension.

**voltage**→**set\_reportFrequency**(**newval**)

Modifie la fréquence de notification périodique des valeurs mesurées.

**voltage**→**set\_resolution**(**newval**)

Modifie la résolution des valeurs mesurées.

**voltage**→**set\_userData**(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**voltage**→**wait\_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YVoltage.FindVoltage() yFindVoltage()yFindVoltage()

## YVoltage

Permet de retrouver un capteur de tension d'après un identifiant donné.

js	function <b>yFindVoltage</b> ( <b>func</b> )
nodejs	function <b>FindVoltage</b> ( <b>func</b> )
php	function <b>yFindVoltage</b> ( <b>\$func</b> )
cpp	YVoltage* <b>yFindVoltage</b> ( const string& <b>func</b> )
m	YVoltage* <b>yFindVoltage</b> ( NSString* <b>func</b> )
pas	function <b>yFindVoltage</b> ( <b>func</b> : string): TYVoltage
vb	function <b>yFindVoltage</b> ( ByVal <b>func</b> As String) As YVoltage
cs	YVoltage <b>FindVoltage</b> ( string <b>func</b> )
java	YVoltage <b>FindVoltage</b> ( String <b>func</b> )
py	def <b>FindVoltage</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le capteur de tension soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YVoltage.isOnline()` pour tester si le capteur de tension est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le capteur de tension sans ambiguïté

### Retourne :

un objet de classe `YVoltage` qui permet ensuite de contrôler le capteur de tension.

## YVoltage.FirstVoltage() yFirstVoltage()yFirstVoltage()

YVoltage

Commence l'énumération des capteurs de tension accessibles par la librairie.

js	function <b>yFirstVoltage</b> ( )
nodejs	function <b>FirstVoltage</b> ( )
php	function <b>yFirstVoltage</b> ( )
cpp	YVoltage* <b>yFirstVoltage</b> ( )
m	YVoltage* <b>yFirstVoltage</b> ( )
pas	function <b>yFirstVoltage</b> ( ): TYVoltage
vb	function <b>yFirstVoltage</b> ( ) As YVoltage
cs	YVoltage <b>FirstVoltage</b> ( )
java	YVoltage <b>FirstVoltage</b> ( )
py	def <b>FirstVoltage</b> ( )

Utiliser la fonction `YVoltage.nextVoltage( )` pour itérer sur les autres capteurs de tension.

### Retourne :

un pointeur sur un objet `YVoltage`, correspondant à le premier capteur de tension accessible en ligne, ou `null` si il n'y a pas de capteurs de tension disponibles.

## voltage→calibrateFromPoints()[voltage calibrateFromPoints: ]

YVoltage

Enregistre des points de correction de mesure, typiquement pour compenser l'effet d'un boîtier sur les mesures rendues par le capteur.

```

js      function calibrateFromPoints( rawValues, refValues)
nodejs  function calibrateFromPoints( rawValues, refValues)
php     function calibrateFromPoints( $rawValues, $refValues)
cpp     int calibrateFromPoints( vector<double> rawValues,
                                vector<double> refValues)

m       -(int) calibrateFromPoints : (NSMutableArray*) rawValues
                                : (NSMutableArray*) refValues

pas     function calibrateFromPoints( rawValues: TDoubleArray,
                                refValues: TDoubleArray): LongInt

vb      procedure calibrateFromPoints( )
cs      int calibrateFromPoints( List<double> rawValues,
                                List<double> refValues)

java    int calibrateFromPoints( ArrayList<Double> rawValues,
                                ArrayList<Double> refValues)

py      def calibrateFromPoints( rawValues, refValues)
cmd     YVoltage target calibrateFromPoints rawValues refValues

```

Il est possible d'enregistrer jusqu'à cinq points de correction. Les points de correction doivent être fournis en ordre croissant, et dans la plage valide du capteur. Le module effectue automatiquement une interpolation linéaire de l'erreur entre les points spécifiés. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Pour plus de plus amples possibilités d'appliquer une surcalibration aux capteurs, veuillez contacter [support@yoctopuce.com](mailto:support@yoctopuce.com).

### Paramètres :

- rawValues** tableau de nombres flottants, correspondant aux valeurs brutes rendues par le capteur pour les points de correction.
- refValues** tableau de nombres flottants, correspondant aux valeurs corrigées désirées pour les points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage→describe()[voltage describe]****YVoltage**

Retourne un court texte décrivant de manière non-ambigüe l'instance du capteur de tension au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le capteur de tension (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)



**voltage**→**get\_advertisedValue()****YVoltage****voltage**→**advertisedValue()[voltage advertisedValue]**

Retourne la valeur courante du capteur de tension (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YVoltage <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du capteur de tension (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**voltage→get\_currentRawValue()**  
**voltage→currentRawValue()[voltage**  
**currentRawValue]**

**YVoltage**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function <b>get_currentRawValue</b> ( )
nodejs	function <b>get_currentRawValue</b> ( )
php	function <b>get_currentRawValue</b> ( )
cpp	double <b>get_currentRawValue</b> ( )
m	-(double) currentRawValue
pas	function <b>get_currentRawValue</b> ( ): double
vb	function <b>get_currentRawValue</b> ( ) As Double
cs	double <b>get_currentRawValue</b> ( )
java	double <b>get_currentRawValue</b> ( )
py	def <b>get_currentRawValue</b> ( )
cmd	YVoltage <b>target</b> <b>get_currentRawValue</b>

**Retourne :**

une valeur numérique représentant la valeur brute retournée par le capteur (sans arrondi ni calibration)

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTRAWVALUE\_INVALID.

**voltage**→**get\_currentValue()****YVoltage****voltage**→**currentValue()**[voltage currentValue]

Retourne la valeur instantanée de la tension.

js	function <b>get_currentValue</b> ( )
nodejs	function <b>get_currentValue</b> ( )
php	function <b>get_currentValue</b> ( )
cpp	double <b>get_currentValue</b> ( )
m	-(double) currentValue
pas	function <b>get_currentValue</b> ( ): double
vb	function <b>get_currentValue</b> ( ) As Double
cs	double <b>get_currentValue</b> ( )
java	double <b>get_currentValue</b> ( )
py	def <b>get_currentValue</b> ( )
cmd	YVoltage <b>target</b> <b>get_currentValue</b>

**Retourne :**

une valeur numérique représentant la valeur instantanée de la tension

En cas d'erreur, déclenche une exception ou retourne Y\_CURRENTVALUE\_INVALID.

**voltage**→**get\_errorMessage()****YVoltage****voltage**→**errorMessage()**[voltage errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du capteur de tension.

**voltage**→**get\_errorType()**  
**voltage**→**errorType()**

**YVoltage**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du capteur de tension.

**voltage**→**get\_friendlyName()****YVoltage****voltage**→**friendlyName()**[**voltage friendlyName**]

Retourne un identifiant global du capteur de tension au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du capteur de tension si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du capteur de tension (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le capteur de tension en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**voltage**→**get\_functionDescriptor()**  
**voltage**→**functionDescriptor()[voltage**  
**functionDescriptor]**

**YVoltage**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**voltage**→**get\_functionId()**

**YVoltage**

**voltage**→**functionId()**[**voltage functionId**]

Retourne l'identifiant matériel du capteur de tension, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le capteur de tension (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.



**voltage**→**get\_hardwareId()****YVoltage****voltage**→**hardwareId()**[**voltage hardwareId**]

Retourne l'identifiant matériel unique du capteur de tension au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du capteur de tension (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le capteur de tension (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**voltage**→**get\_highestValue()****YVoltage****voltage**→**highestValue()**[**voltage** **highestValue**]

Retourne la valeur maximale observée pour la tension.

js	function <b>get_highestValue</b> ( )
nodejs	function <b>get_highestValue</b> ( )
php	function <b>get_highestValue</b> ( )
cpp	double <b>get_highestValue</b> ( )
m	-(double) highestValue
pas	function <b>get_highestValue</b> ( ): double
vb	function <b>get_highestValue</b> ( ) As Double
cs	double <b>get_highestValue</b> ( )
java	double <b>get_highestValue</b> ( )
py	def <b>get_highestValue</b> ( )
cmd	YVoltage <b>target</b> <b>get_highestValue</b>

**Retourne :**

une valeur numérique représentant la valeur maximale observée pour la tension

En cas d'erreur, déclenche une exception ou retourne Y\_HIGHESTVALUE\_INVALID.

**voltage→get\_logFrequency()****YVoltage****voltage→logFrequency()[voltage logFrequency]**

Retourne la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données.

js	function <b>get_logFrequency</b> ( )
nodejs	function <b>get_logFrequency</b> ( )
php	function <b>get_logFrequency</b> ( )
cpp	string <b>get_logFrequency</b> ( )
m	-(NSString*) logFrequency
pas	function <b>get_logFrequency</b> ( ): string
vb	function <b>get_logFrequency</b> ( ) As String
cs	string <b>get_logFrequency</b> ( )
java	String <b>get_logFrequency</b> ( )
py	def <b>get_logFrequency</b> ( )
cmd	YVoltage <b>target</b> <b>get_logFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger, ou "OFF" si les mesures ne sont pas stockées dans la mémoire de l'enregistreur de données

En cas d'erreur, déclenche une exception ou retourne Y\_LOGFREQUENCY\_INVALID.

**voltage**→**get\_logicalName()****YVoltage****voltage**→**logicalName()**[voltage logicalName]

Retourne le nom logique du capteur de tension.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YVoltage <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du capteur de tension. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**voltage**→**get\_lowestValue()****YVoltage****voltage**→**lowestValue()**[voltage lowestValue]

Retourne la valeur minimale observée pour la tension.

js	function <b>get_lowestValue</b> ( )
nodejs	function <b>get_lowestValue</b> ( )
php	function <b>get_lowestValue</b> ( )
cpp	double <b>get_lowestValue</b> ( )
m	-(double) lowestValue
pas	function <b>get_lowestValue</b> ( ): double
vb	function <b>get_lowestValue</b> ( ) As Double
cs	double <b>get_lowestValue</b> ( )
java	double <b>get_lowestValue</b> ( )
py	def <b>get_lowestValue</b> ( )
cmd	YVoltage <b>target</b> <b>get_lowestValue</b>

**Retourne :**

une valeur numérique représentant la valeur minimale observée pour la tension

En cas d'erreur, déclenche une exception ou retourne Y\_LOWESTVALUE\_INVALID.

**voltage**→**get\_module()****YVoltage****voltage**→**module()**[voltage module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**voltage**→**get\_module\_async()**  
**voltage**→**module\_async()**

**YVoltage**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voltage→get\_recordedData()****YVoltage****voltage→recordedData()[voltage recordedData: ]**

Retourne un objet DataSet représentant des mesures de ce capteur précédemment enregistrées à l'aide du DataLogger, pour l'intervalle de temps spécifié.

js	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
nodejs	function <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
php	function <b>get_recordedData</b> ( <b>\$startTime</b> , <b>\$endTime</b> )
c++	YDataSet <b>get_recordedData</b> ( s64 <b>startTime</b> , s64 <b>endTime</b> )
m	-(YDataSet*) recordedData : (s64) <b>startTime</b> : (s64) <b>endTime</b>
pas	function <b>get_recordedData</b> ( <b>startTime</b> : int64, <b>endTime</b> : int64): TYDataSet
vb	function <b>get_recordedData</b> ( ) As YDataSet
cs	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
java	YDataSet <b>get_recordedData</b> ( long <b>startTime</b> , long <b>endTime</b> )
py	def <b>get_recordedData</b> ( <b>startTime</b> , <b>endTime</b> )
cmd	YVoltage <b>target</b> <b>get_recordedData</b> <b>startTime</b> <b>endTime</b>

Veuillez vous référer à la documentation de la classe DataSet pour plus plus d'informations sur la manière d'obtenir un aperçu des mesures pour la période, et comment charger progressivement une grande quantité de mesures depuis le dataLogger.

Cette méthode ne fonctionne que si le module utilise un firmware récent, car les objets DataSet ne sont pas supportés par les firmwares antérieurs à la révision 13000.

**Paramètres :**

**startTime** le début de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite sur le début des mesures.

**endTime** la fin de l'intervalle de mesure désiré, c'est à dire en nombre de secondes depuis le 1er janvier 1970 UTC. La valeur 0 peut être utilisée pour ne poser aucune limite de fin.

**Retourne :**

une instance de YDataSet, dont les méthodes permettent de d'accéder aux données historiques souhaitées.



**voltage→get\_reportFrequency()**  
**voltage→reportFrequency()[voltage**  
**reportFrequency]**

**YVoltage**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function <b>get_reportFrequency</b> ( )
nodejs	function <b>get_reportFrequency</b> ( )
php	function <b>get_reportFrequency</b> ( )
c++	string <b>get_reportFrequency</b> ( )
m	-(NSString*) reportFrequency
pas	function <b>get_reportFrequency</b> ( ): string
vb	function <b>get_reportFrequency</b> ( ) As String
cs	string <b>get_reportFrequency</b> ( )
java	String <b>get_reportFrequency</b> ( )
py	def <b>get_reportFrequency</b> ( )
cmd	YVoltage <b>target</b> <b>get_reportFrequency</b>

**Retourne :**

une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction

En cas d'erreur, déclenche une exception ou retourne Y\_REPORTFREQUENCY\_INVALID.

**voltage**→**get\_resolution()****YVoltage****voltage**→**resolution()**[voltage resolution]

Retourne la résolution des valeurs mesurées.

js	function <b>get_resolution</b> ( )
nodejs	function <b>get_resolution</b> ( )
php	function <b>get_resolution</b> ( )
cpp	double <b>get_resolution</b> ( )
m	-(double) resolution
pas	function <b>get_resolution</b> ( ): double
vb	function <b>get_resolution</b> ( ) As Double
cs	double <b>get_resolution</b> ( )
java	double <b>get_resolution</b> ( )
py	def <b>get_resolution</b> ( )
cmd	YVoltage <b>target</b> <b>get_resolution</b>

La résolution correspond à la précision numérique de la représentation des mesures. Elle n'est pas forcément identique à la précision réelle du capteur.

**Retourne :**

une valeur numérique représentant la résolution des valeurs mesurées

En cas d'erreur, déclenche une exception ou retourne Y\_RESOLUTION\_INVALID.

**voltage**→**get\_unit()****YVoltage****voltage**→**unit()**[voltage unit]

Retourne l'unité dans laquelle la tension est exprimée.

js	function <b>get_unit</b> ( )
nodejs	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YVoltage <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la tension est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**voltage→get\_userdata()**

**YVoltage**

**voltage→userdata()[voltage userData]**

---

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

<code>js</code>	<code>function get_userdata( )</code>
<code>nodejs</code>	<code>function get_userdata( )</code>
<code>php</code>	<code>function get_userdata( )</code>
<code>cpp</code>	<code>void * get_userdata( )</code>
<code>m</code>	<code>-(void*) userData</code>
<code>pas</code>	<code>function get_userdata( ): Tobject</code>
<code>vb</code>	<code>function get_userdata( ) As Object</code>
<code>cs</code>	<code>object get_userdata( )</code>
<code>java</code>	<code>Object get_userdata( )</code>
<code>py</code>	<code>def get_userdata( )</code>

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**voltage→isOnline()[voltage isOnline]****YVoltage**

Vérifie si le module hébergeant le capteur de tension est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du capteur de tension sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le capteur de tension est joignable, `false` sinon

**voltage→isOnline\_async()****YVoltage**

Vérifie si le module hébergeant le capteur de tension est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du capteur de tension sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voltage→load()[voltage load: ]****YVoltage**

Met en cache les valeurs courantes du capteur de tension, avec une durée de validité spécifiée.

js	function load( msValidity)
node.js	function load( msValidity)
php	function load( \$msValidity)
cpp	YRETCODE load( int msValidity)
m	-(YRETCODE) load : (int) msValidity
pas	function load( msValidity: integer): YRETCODE
vb	function load( ByVal msValidity As Integer) As YRETCODE
cs	YRETCODE load( int msValidity)
java	int load( long msValidity)
py	def load( msValidity)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## voltage→loadCalibrationPoints()[voltage loadCalibrationPoints: ]

YVoltage

Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode `calibrateFromPoints`.

```

js function loadCalibrationPoints( rawValues, refValues)
node.js function loadCalibrationPoints( rawValues, refValues)
php function loadCalibrationPoints( &$rawValues, &$refValues)
cpp int loadCalibrationPoints( vector<double>& rawValues,
                             vector<double>& refValues)

m -(int) loadCalibrationPoints : (NSMutableArray*) rawValues
   : (NSMutableArray*) refValues

pas function loadCalibrationPoints( var rawValues: TDoubleArray,
                                   var refValues: TDoubleArray): LongInt

vb procedure loadCalibrationPoints( )
cs int loadCalibrationPoints( List<double> rawValues,
                             List<double> refValues)

java int loadCalibrationPoints( ArrayList<Double> rawValues,
                               ArrayList<Double> refValues)

py def loadCalibrationPoints( rawValues, refValues)
cmd YVoltage target loadCalibrationPoints rawValues refValues

```

### Paramètres :

- rawValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs brutes des points de correction.
- refValues** tableau de nombres flottants, qui sera rempli par la fonction avec les valeurs désirées des points de correction.

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**voltage→load\_async()****YVoltage**

Met en cache les valeurs courantes du capteur de tension, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**voltage→nextVoltage()[voltage nextVoltage]****YVoltage**

Continue l'énumération des capteurs de tension commencée à l'aide de `yFirstVoltage()`.

js	function <b>nextVoltage</b> ( )
nodejs	function <b>nextVoltage</b> ( )
php	function <b>nextVoltage</b> ( )
cpp	YVoltage * <b>nextVoltage</b> ( )
m	-(YVoltage*) <b>nextVoltage</b>
pas	function <b>nextVoltage</b> ( ): TYVoltage
vb	function <b>nextVoltage</b> ( ) As YVoltage
cs	YVoltage <b>nextVoltage</b> ( )
java	YVoltage <b>nextVoltage</b> ( )
py	def <b>nextVoltage</b> ( )

**Retourne :**

un pointeur sur un objet `YVoltage` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## voltage→registerTimedReportCallback()[voltage registerTimedReportCallback: ]

YVoltage

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
nodejs	function <b>registerTimedReportCallback</b> ( <b>callback</b> )
php	function <b>registerTimedReportCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerTimedReportCallback</b> ( YVoltageTimedReportCallback <b>callback</b> )
m	-(int) <b>registerTimedReportCallback</b> : (YVoltageTimedReportCallback) <b>callback</b>
pas	function <b>registerTimedReportCallback</b> ( <b>callback</b> : TYVoltageTimedReportCallback): LongInt
vb	function <b>registerTimedReportCallback</b> ( ) As Integer
cs	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
java	int <b>registerTimedReportCallback</b> ( TimedReportCallback <b>callback</b> )
py	def <b>registerTimedReportCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callbacks peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callbacks ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

## voltage→registerValueCallback()[voltage registerValueCallback: ]

YVoltage

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YVoltageValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YVoltageValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYVoltageValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**voltage**→**set\_highestValue()****YVoltage****voltage**→**setHighestValue()**[**voltage** **setHighestValue**:  
**]**

Modifie la mémoire de valeur maximale observée pour la tension.

js	function <b>set_highestValue</b> ( <b>newval</b> )
nodejs	function <b>set_highestValue</b> ( <b>newval</b> )
php	function <b>set_highestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_highestValue</b> ( double <b>newval</b> )
m	-(int) setHighestValue : (double) <b>newval</b>
pas	function <b>set_highestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_highestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_highestValue</b> ( double <b>newval</b> )
java	int <b>set_highestValue</b> ( double <b>newval</b> )
py	def <b>set_highestValue</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_highestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur maximale observée pour la tension

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage→set\_logFrequency()****YVoltage****voltage→setLogFrequency()[voltage  
setLogFrequency: ]**

Modifie la fréquence d'enregistrement des mesures dans le datalogger.

js	function <b>set_logFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_logFrequency</b> ( <b>newval</b> )
php	function <b>set_logFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_logFrequency</b> ( const string& <b>newval</b> )
m	-(int) setLogFrequency : (NSString*) <b>newval</b>
pas	function <b>set_logFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_logFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logFrequency</b> ( string <b>newval</b> )
java	int <b>set_logFrequency</b> ( String <b>newval</b> )
py	def <b>set_logFrequency</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_logFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver l'enregistrement des mesures de cette fonction, utilisez la valeur "OFF".

**Paramètres :**

**newval** une chaîne de caractères représentant la fréquence d'enregistrement des mesures dans le datalogger

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage**→**set\_logicalName()****YVoltage****voltage**→**setLogicalName()**[**voltage setLogicalName:**  
**]**

Modifie le nom logique du capteur de tension.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du capteur de tension.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage**→**set\_lowestValue()****YVoltage****voltage**→**setLowestValue()**[**voltage** **setLowestValue:** ]

Modifie la mémoire de valeur minimale observée pour la tension.

js	function <b>set_lowestValue</b> ( <b>newval</b> )
nodejs	function <b>set_lowestValue</b> ( <b>newval</b> )
php	function <b>set_lowestValue</b> ( <b>\$newval</b> )
cpp	int <b>set_lowestValue</b> ( double <b>newval</b> )
m	-(int) setLowestValue : (double) <b>newval</b>
pas	function <b>set_lowestValue</b> ( <b>newval</b> : double): integer
vb	function <b>set_lowestValue</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_lowestValue</b> ( double <b>newval</b> )
java	int <b>set_lowestValue</b> ( double <b>newval</b> )
py	def <b>set_lowestValue</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_lowestValue</b> <b>newval</b>

**Paramètres :**

**newval** une valeur numérique représentant la mémoire de valeur minimale observée pour la tension

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**voltage→set\_reportFrequency()**  
**voltage→setReportFrequency()[voltage**  
**setReportFrequency: ]**

**YVoltage**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function <b>set_reportFrequency</b> ( <b>newval</b> )
nodejs	function <b>set_reportFrequency</b> ( <b>newval</b> )
php	function <b>set_reportFrequency</b> ( <b>\$newval</b> )
cpp	int <b>set_reportFrequency</b> ( const string& <b>newval</b> )
m	-(int) setReportFrequency : (NSString*) <b>newval</b>
pas	function <b>set_reportFrequency</b> ( <b>newval</b> : string): integer
vb	function <b>set_reportFrequency</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_reportFrequency</b> ( string <b>newval</b> )
java	int <b>set_reportFrequency</b> ( String <b>newval</b> )
py	def <b>set_reportFrequency</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_reportFrequency</b> <b>newval</b>

La fréquence peut être spécifiée en mesures par secondes, en mesures par minutes (par exemple "15/m") ou en mesures par heure (par exemple "4/h"). Pour désactiver les notifications périodiques pour cette fonction, utilisez la valeur "OFF".

#### Paramètres :

**newval** une chaîne de caractères représentant la fréquence de notification périodique des valeurs mesurées

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage**→**set\_resolution()****YVoltage****voltage**→**setResolution()**[**voltage** **setResolution:** ]

Modifie la résolution des valeurs mesurées.

js	function <b>set_resolution</b> ( <b>newval</b> )
nodejs	function <b>set_resolution</b> ( <b>newval</b> )
php	function <b>set_resolution</b> ( <b>\$newval</b> )
cpp	int <b>set_resolution</b> ( double <b>newval</b> )
m	-(int) setResolution : (double) <b>newval</b>
pas	function <b>set_resolution</b> ( <b>newval</b> : double): integer
vb	function <b>set_resolution</b> ( ByVal <b>newval</b> As Double) As Integer
cs	int <b>set_resolution</b> ( double <b>newval</b> )
java	int <b>set_resolution</b> ( double <b>newval</b> )
py	def <b>set_resolution</b> ( <b>newval</b> )
cmd	YVoltage <b>target</b> <b>set_resolution</b> <b>newval</b>

La résolution correspond à la précision de la représentation numérique des mesures. Changer la résolution ne change pas la précision de la mesure elle-même.

**Paramètres :**

**newval** une valeur numérique représentant la résolution des valeurs mesurées

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**voltage**→**set\_userdata()****YVoltage****voltage**→**setUserData()**[**voltage setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**voltage→wait\_async()****YVoltage**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.42. Interface de la fonction Source de tension

La librairie de programmation Yoctopuce permet de commander la tension de sortie du module. Vous pouvez affecter une valeur fixe, ou faire des transitions de voltage.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_vsource.js'></script>
php	require_once('yocto_vsource.php');
cpp	#include "yocto_vsource.h"
m	#import "yocto_vsource.h"
pas	uses yocto_vsource;
vb	yocto_vsource.vb
cs	yocto_vsource.cs
java	import com.yoctopuce.YoctoAPI.YVSource;
py	from yocto_vsource import *

Fonction globales
<b>yFindVSource(func)</b> Permet de retrouver une source de tension d'après un identifiant donné.
<b>yFirstVSource()</b> Commence l'énumération des sources de tension accessibles par la librairie.
Méthodes des objets YVSource
<b>vsource→describe()</b> Retourne un court texte décrivant la fonction au format TYPE ( NAME ) = SERIAL . FUNCTIONID.
<b>vsource→get_advertisedValue()</b> Retourne la valeur courante de la source de tension (pas plus de 6 caractères).
<b>vsource→get_errorMessage()</b> Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.
<b>vsource→get_errorType()</b> Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.
<b>vsource→get_extPowerFailure()</b> Rend TRUE si le voltage de l'alimentation externe est trop bas.
<b>vsource→get_failure()</b> Indique si le module est en condition d'erreur.
<b>vsource→get_friendlyName()</b> Retourne un identifiant global de la fonction au format NOM_MODULE . NOM_FONCTION.
<b>vsource→get_functionDescriptor()</b> Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
<b>vsource→get_functionId()</b> Retourne l'identifiant matériel de la fonction, sans référence au module.
<b>vsource→get_hardwareId()</b> Retourne l'identifiant matériel unique de la fonction au format SERIAL . FUNCTIONID.
<b>vsource→get_logicalName()</b> Retourne le nom logique de la source de tension.
<b>vsource→get_module()</b> Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
<b>vsource→get_module_async(callback, context)</b>

### 3. Reference

	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<b><code>vsource→get_overCurrent()</code></b>	Rend TRUE si l'appareil connecté à la sortie du module consomme trop de courant.
<b><code>vsource→get_overHeat()</code></b>	Rend TRUE si le module est en surchauffe.
<b><code>vsource→get_overLoad()</code></b>	Rend TRUE si le module n'est pas capable de tenir la tension de sortie demandée.
<b><code>vsource→get_regulationFailure()</code></b>	Rend TRUE si le voltage de sortie de trop élevé par report à la tension demandée demandée.
<b><code>vsource→get_unit()</code></b>	Retourne l'unité dans laquelle la tension est exprimée.
<b><code>vsource→get_userData()</code></b>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<b><code>vsource→get_voltage()</code></b>	Retourne la valeur de la commande de tension de sortie en mV
<b><code>vsource→isOnline()</code></b>	Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.
<b><code>vsource→isOnline_async(callback, context)</code></b>	Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.
<b><code>vsource→load(msValidity)</code></b>	Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.
<b><code>vsource→load_async(msValidity, callback, context)</code></b>	Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.
<b><code>vsource→nextVSource()</code></b>	Continue l'énumération des sources de tension commencée à l'aide de <code>yFirstVSource()</code> .
<b><code>vsource→pulse(voltage, ms_duration)</code></b>	Active la sortie à une tension donnée, et pour durée spécifiée, puis revient ensuite spontanément à zéro volt.
<b><code>vsource→registerValueCallback(callback)</code></b>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<b><code>vsource→set_logicalName(newval)</code></b>	Modifie le nom logique de la source de tension.
<b><code>vsource→set_userData(data)</code></b>	Enregistre un contexte libre dans l'attribut <code>userData</code> de la fonction, afin de le retrouver plus tard à l'aide de la méthode <code>get_userData</code> .
<b><code>vsource→set_voltage(newval)</code></b>	Règle la tension de sortie du module (en milliVolts).
<b><code>vsource→voltageMove(target, ms_duration)</code></b>	Déclenche une variation constante de la sortie vers une valeur donnée.
<b><code>vsource→wait_async(callback, context)</code></b>	Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

**yFindVSource()** —**YVSource****YVSource.FindVSource()****yFindVSource()**

Permet de retrouver une source de tension d'après un identifiant donné.

js	function <b>yFindVSource</b> ( <b>func</b> )
php	function <b>yFindVSource</b> ( <b>\$func</b> )
cpp	YVSource* <b>yFindVSource</b> ( const string& <b>func</b> )
m	YVSource* <b>yFindVSource</b> ( NSString* <b>func</b> )
pas	function <b>yFindVSource</b> ( <b>func</b> : string): TYVSource
vb	function <b>yFindVSource</b> ( ByVal <b>func</b> As String) As YVSource
cs	YVSource <b>FindVSource</b> ( string <b>func</b> )
java	YVSource <b>FindVSource</b> ( String <b>func</b> )
py	def <b>FindVSource</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que la source de tension soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YVSource.isOnline()` pour tester si la source de tension est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

**Paramètres :**

**func** une chaîne de caractères qui référence la source de tension sans ambiguïté

**Retourne :**

un objet de classe `YVSource` qui permet ensuite de contrôler la source de tension.

**yFirstVSource()** —**YVSource****YVSource.FirstVSource()****yFirstVSource()**

Commence l'énumération des sources de tension accessibles par la librairie.

js	function <b>yFirstVSource</b> ( )
php	function <b>yFirstVSource</b> ( )
cpp	YVSource* <b>yFirstVSource</b> ( )
m	YVSource* <b>yFirstVSource</b> ( )
pas	function <b>yFirstVSource</b> ( ): TYVSource
vb	function <b>yFirstVSource</b> ( ) As YVSource
cs	YVSource <b>FirstVSource</b> ( )
java	YVSource <b>FirstVSource</b> ( )
py	def <b>FirstVSource</b> ( )

Utiliser la fonction `YVSource.nextVSource()` pour itérer sur les autres sources de tension.

**Retourne :**

un pointeur sur un objet `YVSource`, correspondant à la première source de tension accessible en ligne, ou `null` si il n'y a pas de sources de tension disponibles.



**vsource→describe()[vsource describe]****YVSource**

Retourne un court texte décrivant la fonction au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant la fonction (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**vsourc**→**get\_advertisedValue()****YVSource****vsourc**→**advertisedValue()**[**vsourc**  
**advertisedValue**]

Retourne la valeur courante de la source de tension (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YVSource <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante de la source de tension (pas plus de 6 caractères)

En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**vsource**→**get\_errorMessage()****YVSource****vsource**→**errorMessage()**[vsource errorMessage]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.

js	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de la fonction

**vsSource**→**get\_errorType()****YVSource****vsSource**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.

js	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de la fonction

**vsources**→**get\_extPowerFailure()**  
**vsources**→**extPowerFailure()**[**vsources**  
**extPowerFailure**]

YVSource

Rend TRUE si le voltage de l'alimentation externe est trop bas.

js	function <b>get_extPowerFailure</b> ( )
php	function <b>get_extPowerFailure</b> ( )
cpp	Y_EXTPOWERFAILURE_enum <b>get_extPowerFailure</b> ( )
m	-(Y_EXTPOWERFAILURE_enum) extPowerFailure
pas	function <b>get_extPowerFailure</b> ( ): Integer
vb	function <b>get_extPowerFailure</b> ( ) As Integer
cs	int <b>get_extPowerFailure</b> ( )
java	int <b>get_extPowerFailure</b> ( )
py	def <b>get_extPowerFailure</b> ( )
cmd	YVSource <b>target</b> <b>get_extPowerFailure</b>

#### Retourne :

soit Y\_EXTPOWERFAILURE\_FALSE, soit Y\_EXTPOWERFAILURE\_TRUE

En cas d'erreur, déclenche une exception ou retourne Y\_EXTPOWERFAILURE\_INVALID.

**vsources**→**get\_failure()****YVSource****vsources**→**failure()**[vsources failure]

Indique si le module est en condition d'erreur.

js	function <b>get_failure</b> ( )
php	function <b>get_failure</b> ( )
cpp	Y_FAILURE_enum <b>get_failure</b> ( )
m	-(Y_FAILURE_enum) failure
pas	function <b>get_failure</b> ( ): Integer
vb	function <b>get_failure</b> ( ) As Integer
cs	int <b>get_failure</b> ( )
java	int <b>get_failure</b> ( )
py	def <b>get_failure</b> ( )
cmd	YVSource <b>target</b> <b>get_failure</b>

Il est possible de savoir de quelle erreur il s'agit en testant `get_overheat`, `get_overcurrent` etc... Lorsqu'une condition d'erreur est rencontrée, la tension de sortie est mise à zéro et ne peut pas être changée tant la fonction `reset()` n'aura pas été appelée.

**Retourne :**

soit `Y_FAILURE_FALSE`, soit `Y_FAILURE_TRUE`

En cas d'erreur, déclenche une exception ou retourne `Y_FAILURE_INVALID`.

**vsource**→**get\_friendlyName()****YVSource****vsource**→**friendlyName()**[vsource friendlyName]

Retourne un identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	virtual string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	override string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de la fonction (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant la fonction en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**vsSource**→**get\_functionDescriptor()**  
**vsSource**→**functionDescriptor()[vsSource**  
**vsSourceDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID



**vsource**→**get\_functionId()****YVSource****vsource**→**functionId()**[**vsource vsourceId**]

Retourne l'identifiant matériel de la fonction, sans référence au module.

js	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant la fonction (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**vsourc**→**get\_hardwareId()****vsourc**→**hardwareId()**[vsourc hardwareId]

Retourne l'identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant la fonction (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**vsource**→**get\_logicalName()****YVSource****vsource**→**logicalName()**[vsource logicalName]

Retourne le nom logique de la source de tension.

js	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YVSource <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de la source de tension

En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**vsourc**→**get\_module()****YVSource****vsourc**→**module()**[vsourc module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

**vsources**→**get\_module\_async()**  
**vsources**→**module\_async()**

**YVSource**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

```
js function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

#### Paramètres :

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

#### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**vsources**→**get\_overCurrent()****YVSource****vsources**→**overCurrent()**[**vsources** **overCurrent**]

Rend TRUE si l'appareil connecté à la sortie du module consomme trop de courant.

js	function <b>get_overCurrent</b> ( )
php	function <b>get_overCurrent</b> ( )
cpp	Y_OVERCURRENT_enum <b>get_overCurrent</b> ( )
m	-(Y_OVERCURRENT_enum) overCurrent
pas	function <b>get_overCurrent</b> ( ): Integer
vb	function <b>get_overCurrent</b> ( ) As Integer
cs	int <b>get_overCurrent</b> ( )
java	int <b>get_overCurrent</b> ( )
py	def <b>get_overCurrent</b> ( )
cmd	YVSource <b>target</b> <b>get_overCurrent</b>

**Retourne :**

soit Y\_OVERCURRENT\_FALSE, soit Y\_OVERCURRENT\_TRUE

En cas d'erreur, déclenche une exception ou retourne Y\_OVERCURRENT\_INVALID.

**vsource**→**get\_overHeat()****YVSource****vsource**→**overHeat()**[vsource overHeat]

Rend TRUE si le module est en surchauffe.

js	function <b>get_overHeat</b> ( )
php	function <b>get_overHeat</b> ( )
cpp	Y_OVERHEAT_enum <b>get_overHeat</b> ( )
m	-(Y_OVERHEAT_enum) overHeat
pas	function <b>get_overHeat</b> ( ): Integer
vb	function <b>get_overHeat</b> ( ) As Integer
cs	int <b>get_overHeat</b> ( )
java	int <b>get_overHeat</b> ( )
py	def <b>get_overHeat</b> ( )
cmd	YVSource <b>target</b> <b>get_overHeat</b>

**Retourne :**

soit Y\_OVERHEAT\_FALSE, soit Y\_OVERHEAT\_TRUE

En cas d'erreur, déclenche une exception ou retourne Y\_OVERHEAT\_INVALID.

**vs**source→**get\_overLoad()****YVSource****vs**source→**overLoad()**[vs**source overLoad**]

Rend TRUE si le module n'est pas capable de tenir la tension de sortie demandée.

js	function <b>get_overLoad</b> ( )
php	function <b>get_overLoad</b> ( )
cpp	Y_OVERLOAD_enum <b>get_overLoad</b> ( )
m	-(Y_OVERLOAD_enum) overLoad
pas	function <b>get_overLoad</b> ( ): Integer
vb	function <b>get_overLoad</b> ( ) As Integer
cs	int <b>get_overLoad</b> ( )
java	int <b>get_overLoad</b> ( )
py	def <b>get_overLoad</b> ( )
cmd	YVSource <b>target</b> <b>get_overLoad</b>

**Retourne :**

soit Y\_OVERLOAD\_FALSE, soit Y\_OVERLOAD\_TRUE

En cas d'erreur, déclenche une exception ou retourne Y\_OVERLOAD\_INVALID.



**vsources**→**get\_regulationFailure()**  
**vsources**→**regulationFailure()**[**vsources**  
**regulationFailure**]

YVSource

Rend TRUE si le voltage de sortie de trop élevé par rapport à la tension demandée demandée.

js	function <b>get_regulationFailure</b> ( )
php	function <b>get_regulationFailure</b> ( )
cpp	Y_REGULATIONFAILURE_enum <b>get_regulationFailure</b> ( )
m	-(Y_REGULATIONFAILURE_enum) regulationFailure
pas	function <b>get_regulationFailure</b> ( ): Integer
vb	function <b>get_regulationFailure</b> ( ) As Integer
cs	int <b>get_regulationFailure</b> ( )
java	int <b>get_regulationFailure</b> ( )
py	def <b>get_regulationFailure</b> ( )
cmd	YVSource <b>target</b> <b>get_regulationFailure</b>

#### Retourne :

soit Y\_REGULATIONFAILURE\_FALSE, soit Y\_REGULATIONFAILURE\_TRUE

En cas d'erreur, déclenche une exception ou retourne Y\_REGULATIONFAILURE\_INVALID.

**vsources**→**get\_unit()****YVSource****vsources**→**unit()**[vsources unit]

Retourne l'unité dans laquelle la tension est exprimée.

js	function <b>get_unit</b> ( )
php	function <b>get_unit</b> ( )
cpp	string <b>get_unit</b> ( )
m	-(NSString*) unit
pas	function <b>get_unit</b> ( ): string
vb	function <b>get_unit</b> ( ) As String
cs	string <b>get_unit</b> ( )
java	String <b>get_unit</b> ( )
py	def <b>get_unit</b> ( )
cmd	YVSource <b>target</b> <b>get_unit</b>

**Retourne :**

une chaîne de caractères représentant l'unité dans laquelle la tension est exprimée

En cas d'erreur, déclenche une exception ou retourne Y\_UNIT\_INVALID.

**vsource**→**get\_userData()****YVSource****vsource**→**userData()**[vsource userData]

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function <b>get_userData</b> ( )
php	function <b>get_userData</b> ( )
cpp	void * <b>get_userData</b> ( )
m	-(void*) userData
pas	function <b>get_userData</b> ( ): Tobject
vb	function <b>get_userData</b> ( ) As Object
cs	object <b>get_userData</b> ( )
java	Object <b>get_userData</b> ( )
py	def <b>get_userData</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**vsources**→**get\_voltage()****YVSource****vsources**→**voltage()**[vsources voltage]

Retourne la valeur de la commande de tension de sortie en mV

js	function <b>get_voltage</b> ( )
php	function <b>get_voltage</b> ( )
cpp	int <b>get_voltage</b> ( )
m	-(int) voltage
pas	function <b>get_voltage</b> ( ): LongInt
vb	function <b>get_voltage</b> ( ) As Integer
cs	int <b>get_voltage</b> ( )
java	int <b>get_voltage</b> ( )
py	def <b>get_voltage</b> ( )

**Retourne :**

un entier représentant la valeur de la commande de tension de sortie en mV

En cas d'erreur, déclenche une exception ou retourne Y\_VOLTAGE\_INVALID.

**vsource**→**isOnline()**[vsource isOnline]**YVSource**

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si la fonction est joignable, `false` sinon

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

#### Paramètres :

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

#### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**vsource→load()[vsource load: ]****YVSource**

Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

#### Paramètres :

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

#### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**vsource**→**nextVSource()**[**vsource** **nextVSource**]**YVSource**

Continue l'énumération des sources de tension commencée à l'aide de `yFirstVSource()`.

js	function <b>nextVSource</b> ( )
php	function <b>nextVSource</b> ( )
cpp	YVSource * <b>nextVSource</b> ( )
m	-(YVSource*) <b>nextVSource</b>
pas	function <b>nextVSource</b> ( ): TYVSource
vb	function <b>nextVSource</b> ( ) As YVSource
cs	YVSource <b>nextVSource</b> ( )
java	YVSource <b>nextVSource</b> ( )
py	def <b>nextVSource</b> ( )

**Retourne :**

un pointeur sur un objet `YVSource` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**vsources→pulse()[vsources pulse: ]****YVSource**

Active la sortie à une tension donnée, et pour durée spécifiée, puis revient ensuite spontanément à zéro volt.

js	function <b>pulse</b> ( <b>voltage</b> , <b>ms_duration</b> )
php	function <b>pulse</b> ( <b>\$voltage</b> , <b>\$ms_duration</b> )
cpp	int <b>pulse</b> ( int <b>voltage</b> , int <b>ms_duration</b> )
m	-(int) <b>pulse</b> : (int) <b>voltage</b> : (int) <b>ms_duration</b>
pas	function <b>pulse</b> ( <b>voltage</b> : integer, <b>ms_duration</b> : integer): integer
vb	function <b>pulse</b> ( ByVal <b>voltage</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>pulse</b> ( int <b>voltage</b> , int <b>ms_duration</b> )
java	int <b>pulse</b> ( int <b>voltage</b> , int <b>ms_duration</b> )
py	def <b>pulse</b> ( <b>voltage</b> , <b>ms_duration</b> )
cmd	YVSource <b>target pulse voltage ms_duration</b>

**Paramètres :**

**voltage**            tension demandée, en millivolts  
**ms\_duration**    durée de l'impulsion, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## **vsource→registerValueCallback()[vsource registerValueCallback: ]**

YVSource

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	void <b>registerValueCallback</b> ( YDisplayUpdateCallback <b>callback</b> )
pas	procedure <b>registerValueCallback</b> ( <b>callback</b> : TGenericUpdateCallback)
vb	procedure <b>registerValueCallback</b> ( ByVal <b>callback</b> As GenericUpdateCallback)
cs	void <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
java	void <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )
m	-(void) <b>registerValueCallback</b> : (YFunctionUpdateCallback) <b>callback</b>

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### **Paramètres :**

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**vsSource**→**set\_logicalName()****YVSource****vsSource**→**setLogicalName()**[**vsSource**  
**setLogicalName:** ]

Modifie le nom logique de la source de tension.

js	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YVSource <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de la source de tension

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**vsource**→**set\_userdata()****YVSource****vsource**→**setUserData()**[**vsource setUserData:** ]

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**vsourceset\_voltage()****YVSource****vsourcesetVoltage() [vsourcesetVoltage: ]**

Règle la tension de sortie du module (en milliVolts).

js	function <b>set_voltage</b> ( <b>newval</b> )
php	function <b>set_voltage</b> ( <b>\$newval</b> )
cpp	int <b>set_voltage</b> ( int <b>newval</b> )
m	-(int) setVoltage : (int) <b>newval</b>
pas	function <b>set_voltage</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_voltage</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_voltage</b> ( int <b>newval</b> )
java	int <b>set_voltage</b> ( int <b>newval</b> )
py	def <b>set_voltage</b> ( <b>newval</b> )
cmd	YVSource <b>target set_voltage newval</b>

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**vsource→voltageMove()**[vsource voltageMove: ]**YVSource**

Déclenche une variation constante de la sortie vers une valeur donnée.

js	function <b>voltageMove</b> ( <b>target</b> , <b>ms_duration</b> )
php	function <b>voltageMove</b> ( <b>\$target</b> , <b>\$ms_duration</b> )
cpp	int <b>voltageMove</b> ( int <b>target</b> , int <b>ms_duration</b> )
m	-(int) <b>voltageMove</b> : (int) <b>target</b> : (int) <b>ms_duration</b>
pas	function <b>voltageMove</b> ( <b>target</b> : integer, <b>ms_duration</b> : integer): integer
vb	function <b>voltageMove</b> ( ByVal <b>target</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>voltageMove</b> ( int <b>target</b> , int <b>ms_duration</b> )
java	int <b>voltageMove</b> ( int <b>target</b> , int <b>ms_duration</b> )
py	def <b>voltageMove</b> ( <b>target</b> , <b>ms_duration</b> )
cmd	YVSource <b>target voltageMove target ms_duration</b>

**Paramètres :**

**target** nouvelle valeur de sortie à la fin de la transition, en milliVolts.  
**ms\_duration** durée de la transition, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**vs**source→wait\_async()**YVSource**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la VM Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :



## 3.43. Interface de la fonction WakeUpMonitor

La fonction WakeUpMonitor prend en charge le contrôle global de toutes les sources de réveil possibles ainsi que les mises en sommeil automatiques.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_wakeupmonitor.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YWakeUpMonitor = yoctolib.YWakeUpMonitor;
php	require_once('yocto_wakeupmonitor.php');
c++	#include "yocto_wakeupmonitor.h"
m	#import "yocto_wakeupmonitor.h"
pas	uses yocto_wakeupmonitor;
vb	yocto_wakeupmonitor.vb
cs	yocto_wakeupmonitor.cs
java	import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
py	from yocto_wakeupmonitor import *

### Fonction globales

#### yFindWakeUpMonitor(func)

Permet de retrouver un moniteur d'après un identifiant donné.

#### yFirstWakeUpMonitor()

Commence l'énumération des Moniteurs accessibles par la librairie.

### Méthodes des objets YWakeUpMonitor

#### wakeupmonitor→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

#### wakeupmonitor→get\_advertisedValue()

Retourne la valeur courante du moniteur (pas plus de 6 caractères).

#### wakeupmonitor→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

#### wakeupmonitor→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

#### wakeupmonitor→get\_friendlyName()

Retourne un identifiant global du moniteur au format `NOM_MODULE . NOM_FONCTION`.

#### wakeupmonitor→get\_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### wakeupmonitor→get\_functionId()

Retourne l'identifiant matériel du moniteur, sans référence au module.

#### wakeupmonitor→get\_hardwareId()

Retourne l'identifiant matériel unique du moniteur au format `SERIAL . FUNCTIONID`.

#### wakeupmonitor→get\_logicalName()

Retourne le nom logique du moniteur.

#### wakeupmonitor→get\_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### wakeupmonitor→get\_module\_async(callback, context)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**wakeupmonitor→get\_nextWakeUp()**

Retourne la prochaine date/heure de réveil agendée (format UNIX)

**wakeupmonitor→get\_powerDuration()**

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

**wakeupmonitor→get\_sleepCountdown()**

Retourne le temps avant le prochain sommeil.

**wakeupmonitor→get\_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

**wakeupmonitor→get\_wakeUpReason()**

Renvoie la raison du dernier réveil.

**wakeupmonitor→get\_wakeUpState()**

Revoie l'état actuel du moniteur

**wakeupmonitor→isOnline()**

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

**wakeupmonitor→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

**wakeupmonitor→load(msValidity)**

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

**wakeupmonitor→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

**wakeupmonitor→nextWakeUpMonitor()**

Continue l'énumération des Moniteurs commencée à l'aide de yFirstWakeUpMonitor().

**wakeupmonitor→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**wakeupmonitor→resetSleepCountDown()**

Réinitialise le compteur de mise en sommeil.

**wakeupmonitor→set\_logicalName(newval)**

Modifie le nom logique du moniteur.

**wakeupmonitor→set\_nextWakeUp(newval)**

Modifie les jours de la semaine où un réveil doit avoir lieu.

**wakeupmonitor→set\_powerDuration(newval)**

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

**wakeupmonitor→set\_sleepCountdown(newval)**

Modifie le temps avant le prochain sommeil .

**wakeupmonitor→set\_userdata(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

**wakeupmonitor→sleep(secBeforeSleep)**

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

**wakeupmonitor→sleepFor(secUntilWakeUp, secBeforeSleep)**

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

**wakeupmonitor→sleepUntil(wakeUpTime, secBeforeSleep)**

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

**wakeupmonitor→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

**wakeupmonitor**→**wakeUp()**

Force un réveil.

## YWakeUpMonitor.FindWakeUpMonitor() yFindWakeUpMonitor()yFindWakeUpMonitor()

## YWakeupMonitor

Permet de retrouver un moniteur d'après un identifiant donné.

js	function <b>yFindWakeUpMonitor</b> ( <b>func</b> )
nodejs	function <b>FindWakeUpMonitor</b> ( <b>func</b> )
php	function <b>yFindWakeUpMonitor</b> ( <b>\$func</b> )
c++	YWakeupMonitor* <b>yFindWakeUpMonitor</b> ( const string& <b>func</b> )
m	YWakeupMonitor* <b>yFindWakeUpMonitor</b> ( NSString* <b>func</b> )
pas	function <b>yFindWakeUpMonitor</b> ( <b>func</b> : string): TYWakeUpMonitor
vb	function <b>yFindWakeUpMonitor</b> ( ByVal <b>func</b> As String) As YWakeupMonitor
cs	YWakeupMonitor <b>FindWakeUpMonitor</b> ( string <b>func</b> )
java	YWakeupMonitor <b>FindWakeUpMonitor</b> ( String <b>func</b> )
py	def <b>FindWakeUpMonitor</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le moniteur soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeupMonitor.isOnline()` pour tester si le moniteur est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le moniteur sans ambiguïté

### Retourne :

un objet de classe `YWakeupMonitor` qui permet ensuite de contrôler le moniteur.

## YWakeUpMonitor.FirstWakeUpMonitor() yFirstWakeUpMonitor()yFirstWakeUpMonitor()

## YWakeUpMonitor

Commence l'énumération des Moniteurs accessibles par la librairie.

js	function <b>yFirstWakeUpMonitor</b> ( )
nodejs	function <b>FirstWakeUpMonitor</b> ( )
php	function <b>yFirstWakeUpMonitor</b> ( )
cpp	YWakeupMonitor* <b>yFirstWakeUpMonitor</b> ( )
m	YWakeupMonitor* <b>yFirstWakeUpMonitor</b> ( )
pas	function <b>yFirstWakeUpMonitor</b> ( ): TYWakeUpMonitor
vb	function <b>yFirstWakeUpMonitor</b> ( ) As YWakeUpMonitor
cs	YWakeupMonitor <b>FirstWakeUpMonitor</b> ( )
java	YWakeupMonitor <b>FirstWakeUpMonitor</b> ( )
py	def <b>FirstWakeUpMonitor</b> ( )

Utiliser la fonction `YWakeupMonitor.nextWakeUpMonitor()` pour itérer sur les autres Moniteurs.

### Retourne :

un pointeur sur un objet `YWakeupMonitor`, correspondant à le premier moniteur accessible en ligne, ou `null` si il n'y a pas de Moniteurs disponibles.

**wakeupmonitor→describe()[wakeupmonitor  
describe]****YWakeUpMonitor**

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur au format  
 TYPE (NAME) = SERIAL . FUNCTIONID.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

**Retourne :**

une chaîne de caractères décrivant le moniteur (ex:  
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wakeupmonitor→get\_advertisedValue()

YWakeUpMonitor

wakeupmonitor→advertisedValue()[wakeupmonitor  
advertisedValue]

Retourne la valeur courante du moniteur (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>get_advertisedValue</b>

#### Retourne :

une chaîne de caractères représentant la valeur courante du moniteur (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**wakeupmonitor**→**get\_errorMessage()****YWakeUpMonitor****wakeupmonitor**→**errorMessage()**[wakeupmonitor  
**errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du moniteur.



**wakeupmonitor→get\_errorType()**  
**wakeupmonitor→errorType()**

**YWakeUpMonitor**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du moniteur.

**wakeupmonitor**→**get\_friendlyName()****YWakeUpMonitor****wakeupmonitor**→**friendlyName()**[**wakeupmonitor**  
**friendlyName**]

Retourne un identifiant global du moniteur au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du moniteur si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du moniteur (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le moniteur en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**wakeupmonitor**→**get\_functionDescriptor()****YWakeUpMonitor****wakeupmonitor**→**functionDescriptor()****[wakeupmonitor functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**wakeupmonitor**→**get\_functionId()**

**YWakeUpMonitor**

**wakeupmonitor**→**functionId()**[wakeupmonitor  
**functionId**]

Retourne l'identifiant matériel du moniteur, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple relay1.

**Retourne :**

une chaîne de caractères identifiant le moniteur (ex: relay1) En cas d'erreur, déclenche une exception ou retourne Y\_FUNCTIONID\_INVALID.

**wakeupmonitor**→**get\_hardwareId()****YWakeUpMonitor****wakeupmonitor**→**hardwareId()**[wakeupmonitor  
**hardwareId**]

Retourne l'identifiant matériel unique du moniteur au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du moniteur (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le moniteur (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**wakeupmonitor**→**get\_logicalName()****YWakeUpMonitor****wakeupmonitor**→**logicalName()**[**wakeupmonitor**  
**logicalName**]

Retourne le nom logique du moniteur.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du moniteur. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**wakeupmonitor**→**get\_module()****YWakeUpMonitor****wakeupmonitor**→**module()**[wakeupmonitor module]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

wakeupmonitor→get\_module\_async()

YWakeUpMonitor

wakeupmonitor→module\_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js function get_module_async( callback, context)
nodejs function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

#### Paramètres :

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

#### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**wakeupmonitor**→**get\_nextWakeUp()****YWakeUpMonitor****wakeupmonitor**→**nextWakeUp()**[wakeupmonitor  
**nextWakeUp]**

Retourne la prochaine date/heure de réveil agendée (format UNIX)

js	function <b>get_nextWakeUp</b> ( )
nodejs	function <b>get_nextWakeUp</b> ( )
php	function <b>get_nextWakeUp</b> ( )
cpp	s64 <b>get_nextWakeUp</b> ( )
m	-(s64) nextWakeUp
pas	function <b>get_nextWakeUp</b> ( ): int64
vb	function <b>get_nextWakeUp</b> ( ) As Long
cs	long <b>get_nextWakeUp</b> ( )
java	long <b>get_nextWakeUp</b> ( )
py	def <b>get_nextWakeUp</b> ( )

**Retourne :**

un entier représentant la prochaine date/heure de réveil agendée (format UNIX)

En cas d'erreur, déclenche une exception ou retourne `Y_NEXTWAKEUP_INVALID`.

**wakeupmonitor**→**get\_powerDuration()****YWakeUpMonitor****wakeupmonitor**→**powerDuration()**[wakeupmonitor  
**powerDuration**]

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

js	function <b>get_powerDuration</b> ( )
nodejs	function <b>get_powerDuration</b> ( )
php	function <b>get_powerDuration</b> ( )
cpp	int <b>get_powerDuration</b> ( )
m	-(int) powerDuration
pas	function <b>get_powerDuration</b> ( ): LongInt
vb	function <b>get_powerDuration</b> ( ) As Integer
cs	int <b>get_powerDuration</b> ( )
java	int <b>get_powerDuration</b> ( )
py	def <b>get_powerDuration</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>get_powerDuration</b>

**Retourne :**

un entier représentant le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement

En cas d'erreur, déclenche une exception ou retourne Y\_POWERDURATION\_INVALID.

wakeupmonitor→get\_sleepCountdown()

YWakeUpMonitor

wakeupmonitor→sleepCountdown()[wakeupmonitor  
sleepCountdown]

Retourne le temps avant le prochain sommeil.

js	function <b>get_sleepCountdown</b> ( )
nodejs	function <b>get_sleepCountdown</b> ( )
php	function <b>get_sleepCountdown</b> ( )
cpp	int <b>get_sleepCountdown</b> ( )
m	-(int) sleepCountdown
pas	function <b>get_sleepCountdown</b> ( ): LongInt
vb	function <b>get_sleepCountdown</b> ( ) As Integer
cs	int <b>get_sleepCountdown</b> ( )
java	int <b>get_sleepCountdown</b> ( )
py	def <b>get_sleepCountdown</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>get_sleepCountdown</b>

#### Retourne :

un entier représentant le temps avant le prochain sommeil

En cas d'erreur, déclenche une exception ou retourne Y\_SLEEP\_COUNTDOWN\_INVALID.

**wakeupmonitor**→**get\_userdata()****YWakeUpMonitor****wakeupmonitor**→**userData()**[wakeupmonitor  
**userData]**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) <code>userData</code>
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**wakeupmonitor**→**get\_wakeUpReason()****YWakeUpMonitor****wakeupmonitor**→**wakeUpReason()**[**wakeupmonitor**  
**wakeUpReason**]

Renvoie la raison du dernier réveil.

js	function <b>get_wakeUpReason</b> ( )
nodejs	function <b>get_wakeUpReason</b> ( )
php	function <b>get_wakeUpReason</b> ( )
cpp	Y_WAKEUPREASON_enum <b>get_wakeUpReason</b> ( )
m	-(Y_WAKEUPREASON_enum) wakeUpReason
pas	function <b>get_wakeUpReason</b> ( ): Integer
vb	function <b>get_wakeUpReason</b> ( ) As Integer
cs	int <b>get_wakeUpReason</b> ( )
java	int <b>get_wakeUpReason</b> ( )
py	def <b>get_wakeUpReason</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>get_wakeUpReason</b>

**Retourne :**

```

une valeur parmi Y_WAKEUPREASON_USBPOWER, Y_WAKEUPREASON_EXTPOWER,
Y_WAKEUPREASON_ENDOFSLEEP,                Y_WAKEUPREASON_EXTSIG1,
Y_WAKEUPREASON_EXTSIG2,                    Y_WAKEUPREASON_EXTSIG3,
Y_WAKEUPREASON_EXTSIG4,                    Y_WAKEUPREASON_SCHEDULE1,
Y_WAKEUPREASON_SCHEDULE2,                  Y_WAKEUPREASON_SCHEDULE3,
Y_WAKEUPREASON_SCHEDULE4,                  Y_WAKEUPREASON_SCHEDULE5    et
Y_WAKEUPREASON_SCHEDULE6

```

En cas d'erreur, déclenche une exception ou retourne Y\_WAKEUPREASON\_INVALID.

wakeupmonitor→get\_wakeUpState()

YWakeUpMonitor

wakeupmonitor→wakeUpState()[wakeupmonitor  
wakeUpState]

Revoie l'état actuel du moniteur

js	function <b>get_wakeUpState</b> ( )
nodejs	function <b>get_wakeUpState</b> ( )
php	function <b>get_wakeUpState</b> ( )
cpp	Y_WAKEUPSTATE_enum <b>get_wakeUpState</b> ( )
m	-(Y_WAKEUPSTATE_enum) wakeUpState
pas	function <b>get_wakeUpState</b> ( ): Integer
vb	function <b>get_wakeUpState</b> ( ) As Integer
cs	int <b>get_wakeUpState</b> ( )
java	int <b>get_wakeUpState</b> ( )
py	def <b>get_wakeUpState</b> ( )

#### Retourne :

soit Y\_WAKEUPSTATE\_SLEEPING, soit Y\_WAKEUPSTATE\_AWAKE

En cas d'erreur, déclenche une exception ou retourne Y\_WAKEUPSTATE\_INVALID.

**wakeupmonitor**→**isOnline()**[wakeupmonitor isOnline]**YWakeUpMonitor**

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du moniteur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si le moniteur est joignable, `false` sinon

**wakeupmonitor→isOnline\_async()****YWakeUpMonitor**

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du moniteur sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.



**wakeupmonitor→load()[wakeupmonitor load: ]****YWakeUpMonitor**

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupmonitor→load\_async()****YWakeUpMonitor**

Met en cache les valeurs courantes du moniteur, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-qu'el à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## wakeupmonitor→nextWakeUpMonitor() [wakeupmonitor nextWakeUpMonitor]

## YWakeUpMonitor

Continue l'énumération des Moniteurs commencée à l'aide de `yFirstWakeUpMonitor()`.

js	function <b>nextWakeUpMonitor</b> ( )
nodejs	function <b>nextWakeUpMonitor</b> ( )
php	function <b>nextWakeUpMonitor</b> ( )
cpp	YWakeupMonitor * <b>nextWakeUpMonitor</b> ( )
m	-(YWakeupMonitor*) <b>nextWakeUpMonitor</b>
pas	function <b>nextWakeUpMonitor</b> ( ): TYWakeUpMonitor
vb	function <b>nextWakeUpMonitor</b> ( ) As YWakeUpMonitor
cs	YWakeupMonitor <b>nextWakeUpMonitor</b> ( )
java	YWakeupMonitor <b>nextWakeUpMonitor</b> ( )
py	def <b>nextWakeUpMonitor</b> ( )

### Retourne :

un pointeur sur un objet `YWakeupMonitor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## wakeupmonitor→registerValueCallback() [wakeupmonitor registerValueCallback: ]

YWakeUpMonitor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YWakeUpMonitorValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YWakeUpMonitorValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYWakeUpMonitorValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## wakeupmonitor→resetSleepCountDown() [wakeupmonitor resetSleepCountDown]

## YWakeUpMonitor

Réinitialise le compteur de mise en sommeil.

js	function <b>resetSleepCountDown</b> ( )
nodejs	function <b>resetSleepCountDown</b> ( )
php	function <b>resetSleepCountDown</b> ( )
cpp	int <b>resetSleepCountDown</b> ( )
m	-(int) <b>resetSleepCountDown</b>
pas	function <b>resetSleepCountDown</b> ( ): LongInt
vb	function <b>resetSleepCountDown</b> ( ) As Integer
cs	int <b>resetSleepCountDown</b> ( )
java	int <b>resetSleepCountDown</b> ( )
py	def <b>resetSleepCountDown</b> ( )
cmd	YWakeUpMonitor <b>target</b> <b>resetSleepCountDown</b>

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set\_logicalName()

YWakeUpMonitor

wakeupmonitor→setLogicalName()[wakeupmonitor  
setLogicalName: ]

Modifie le nom logique du moniteur.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YWakeUpMonitor <b>target set_logicalName newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

#### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du moniteur.

#### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set\_nextWakeUp()

YWakeUpMonitor

wakeupmonitor→setNextWakeUp()[wakeupmonitor  
setNextWakeUp: ]

Modifie les jours de la semaine où un réveil doit avoir lieu.

js	function <b>set_nextWakeUp</b> ( <b>newval</b> )
nodejs	function <b>set_nextWakeUp</b> ( <b>newval</b> )
php	function <b>set_nextWakeUp</b> ( <b>\$newval</b> )
cpp	int <b>set_nextWakeUp</b> ( s64 <b>newval</b> )
m	-(int) setNextWakeUp : (s64) <b>newval</b>
pas	function <b>set_nextWakeUp</b> ( <b>newval</b> : int64): integer
vb	function <b>set_nextWakeUp</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_nextWakeUp</b> ( long <b>newval</b> )
java	int <b>set_nextWakeUp</b> ( long <b>newval</b> )
py	def <b>set_nextWakeUp</b> ( <b>newval</b> )
cmd	YWakeUpMonitor <b>target</b> <b>set_nextWakeUp</b> <b>newval</b>

#### Paramètres :

**newval** un entier représentant les jours de la semaine où un réveil doit avoir lieu

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set\_powerDuration()

YWakeUpMonitor

wakeupmonitor→setPowerDuration()[wakeupmonitor  
setPowerDuration: ]

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

js	function set_powerDuration( newval)
nodejs	function set_powerDuration( newval)
php	function set_powerDuration( \$newval)
cpp	int set_powerDuration( int newval)
m	-(int) setPowerDuration : (int) newval
pas	function set_powerDuration( newval: LongInt): integer
vb	function set_powerDuration( ByVal newval As Integer) As Integer
cs	int set_powerDuration( int newval)
java	int set_powerDuration( int newval)
py	def set_powerDuration( newval)
cmd	YWakeUpMonitor target set_powerDuration newval

#### Paramètres :

**newval** un entier représentant le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**wakeupmonitor→set\_sleepCountdown()**  
**wakeupmonitor→setSleepCountdown()**  
**[wakeupmonitor setSleepCountdown: ]**

**YWakeUpMonitor**

Modifie le temps avant le prochain sommeil .

js	function <b>set_sleepCountdown</b> ( <b>newval</b> )
nodejs	function <b>set_sleepCountdown</b> ( <b>newval</b> )
php	function <b>set_sleepCountdown</b> ( <b>\$newval</b> )
cpp	int <b>set_sleepCountdown</b> ( int <b>newval</b> )
m	-(int) setSleepCountdown : (int) <b>newval</b>
pas	function <b>set_sleepCountdown</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_sleepCountdown</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_sleepCountdown</b> ( int <b>newval</b> )
java	int <b>set_sleepCountdown</b> ( int <b>newval</b> )
py	def <b>set_sleepCountdown</b> ( <b>newval</b> )
cmd	YWakeUpMonitor <b>target</b> <b>set_sleepCountdown</b> <b>newval</b>

#### Paramètres :

**newval** un entier représentant le temps avant le prochain sommeil

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupmonitor**→**set\_userdata()****YWakeUpMonitor****wakeupmonitor**→**setUserData()**[wakeupmonitor**setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
c++	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**wakeupmonitor→sleep()[wakeupmonitor sleep: ]****YWakeUpMonitor**

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

js	function <b>sleep</b> ( <b>secBeforeSleep</b> )
nodejs	function <b>sleep</b> ( <b>secBeforeSleep</b> )
php	function <b>sleep</b> ( <b>\$secBeforeSleep</b> )
cpp	int <b>sleep</b> ( int <b>secBeforeSleep</b> )
m	-(int) <b>sleep</b> : (int) <b>secBeforeSleep</b>
pas	function <b>sleep</b> ( <b>secBeforeSleep</b> : LongInt): LongInt
vb	function <b>sleep</b> ( ) As Integer
cs	int <b>sleep</b> ( int <b>secBeforeSleep</b> )
java	int <b>sleep</b> ( int <b>secBeforeSleep</b> )
py	def <b>sleep</b> ( <b>secBeforeSleep</b> )
cmd	YWakeUpMonitor <b>target sleep secBeforeSleep</b>

**Paramètres :**

**secBeforeSleep** nombre de seconde avant la mise en sommeil

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## wakeupmonitor→sleepFor()[wakeupmonitor sleepFor: ]

YWakeUpMonitor

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

js	function <b>sleepFor</b> ( <b>secUntilWakeUp</b> , <b>secBeforeSleep</b> )
nodejs	function <b>sleepFor</b> ( <b>secUntilWakeUp</b> , <b>secBeforeSleep</b> )
php	function <b>sleepFor</b> ( <b>\$secUntilWakeUp</b> , <b>\$secBeforeSleep</b> )
c++	int <b>sleepFor</b> ( int <b>secUntilWakeUp</b> , int <b>secBeforeSleep</b> )
m	-(int) <b>sleepFor</b> : (int) <b>secUntilWakeUp</b> : (int) <b>secBeforeSleep</b>
pas	function <b>sleepFor</b> ( <b>secUntilWakeUp</b> : LongInt, <b>secBeforeSleep</b> : LongInt): LongInt
vb	function <b>sleepFor</b> ( ) As Integer
cs	int <b>sleepFor</b> ( int <b>secUntilWakeUp</b> , int <b>secBeforeSleep</b> )
java	int <b>sleepFor</b> ( int <b>secUntilWakeUp</b> , int <b>secBeforeSleep</b> )
py	def <b>sleepFor</b> ( <b>secUntilWakeUp</b> , <b>secBeforeSleep</b> )
cmd	YWakeUpMonitor <b>target</b> <b>sleepFor</b> <b>secUntilWakeUp</b> <b>secBeforeSleep</b>

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

### Paramètres :

- secUntilWakeUp** durée de la mise en sommeil, en secondes
- secBeforeSleep** nombre de secondes avant la mise en sommeil

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## wakeupmonitor→sleepUntil()[wakeupmonitor sleepUntil: ]

## YWakeUpMonitor

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

js	function <b>sleepUntil</b> ( <b>wakeUpTime</b> , <b>secBeforeSleep</b> )
nodejs	function <b>sleepUntil</b> ( <b>wakeUpTime</b> , <b>secBeforeSleep</b> )
php	function <b>sleepUntil</b> ( <b>\$wakeUpTime</b> , <b>\$secBeforeSleep</b> )
cpp	int <b>sleepUntil</b> ( int <b>wakeUpTime</b> , int <b>secBeforeSleep</b> )
m	-(int) <b>sleepUntil</b> : (int) <b>wakeUpTime</b> : (int) <b>secBeforeSleep</b>
pas	function <b>sleepUntil</b> ( <b>wakeUpTime</b> : LongInt, <b>secBeforeSleep</b> : LongInt): LongInt
vb	function <b>sleepUntil</b> ( ) As Integer
cs	int <b>sleepUntil</b> ( int <b>wakeUpTime</b> , int <b>secBeforeSleep</b> )
java	int <b>sleepUntil</b> ( int <b>wakeUpTime</b> , int <b>secBeforeSleep</b> )
py	def <b>sleepUntil</b> ( <b>wakeUpTime</b> , <b>secBeforeSleep</b> )
cmd	YWakeUpMonitor <b>target</b> <b>sleepUntil</b> <b>wakeUpTime</b> <b>secBeforeSleep</b>

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

### Paramètres :

- wakeUpTime** date/heure du réveil (format UNIX)
- secBeforeSleep** nombre de secondes avant la mise en sommeil

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupmonitor**→**wait\_async()****YWakeUpMonitor**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

---

**wakeupmonitor**→**wakeUp()**[wakeupmonitor wakeUp]**YWakeUpMonitor**

---

Force un réveil.

js	function <b>wakeUp</b> ( )
nodejs	function <b>wakeUp</b> ( )
php	function <b>wakeUp</b> ( )
cpp	int <b>wakeUp</b> ( )
m	-(int) <b>wakeUp</b>
pas	function <b>wakeUp</b> ( ): LongInt
vb	function <b>wakeUp</b> ( ) As Integer
cs	int <b>wakeUp</b> ( )
java	int <b>wakeUp</b> ( )
py	def <b>wakeUp</b> ( )
cmd	YWakeUpMonitor <b>target wakeUp</b>

### 3.44. Interface de la fonction WakeUpSchedule

La fonction WakeUpSchedule implémente une condition de réveil. Le réveil est spécifiée par un ensemble de mois et/ou jours et/ou heures et/ou minutes où il doit se produire.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_wakeupschedule.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YWakeUpSchedule = yoctolib.YWakeUpSchedule;
php	require_once('yocto_wakeupschedule.php');
c++	#include "yocto_wakeupschedule.h"
m	#import "yocto_wakeupschedule.h"
pas	uses yocto_wakeupschedule;
vb	yocto_wakeupschedule.vb
cs	yocto_wakeupschedule.cs
java	import com.yoctopuce.YoctoAPI.YWakeUpSchedule;
py	from yocto_wakeupschedule import *

Fonction globales
<b>yFindWakeUpSchedule(func)</b> Permet de retrouver un réveil agendé d'après un identifiant donné.
<b>yFirstWakeUpSchedule()</b> Commence l'énumération des réveils agendés accessibles par la librairie.
Méthodes des objets YWakeUpSchedule
<b>wakeupschedule→describe()</b> Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE ( NAME ) = SERIAL . FUNCTIONID.
<b>wakeupschedule→get_advertisedValue()</b> Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).
<b>wakeupschedule→get_errorMessage()</b> Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
<b>wakeupschedule→get_errorType()</b> Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
<b>wakeupschedule→get_friendlyName()</b> Retourne un identifiant global du réveil agendé au format NOM_MODULE . NOM_FONCTION.
<b>wakeupschedule→get_functionDescriptor()</b> Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
<b>wakeupschedule→get_functionId()</b> Retourne l'identifiant matériel du réveil agendé, sans référence au module.
<b>wakeupschedule→get_hardwareId()</b> Retourne l'identifiant matériel unique du réveil agendé au format SERIAL . FUNCTIONID.
<b>wakeupschedule→get_hours()</b> Retourne les heures où le réveil est actif..
<b>wakeupschedule→get_logicalName()</b> Retourne le nom logique du réveil agendé.
<b>wakeupschedule→get_minutes()</b> Retourne toutes les minutes de chaque heure où le réveil est actif.
<b>wakeupschedule→get_minutesA()</b>



Retourne les minutes de l'intervall 00-29 de chaque heures où le réveil est actif.

#### **wakeupschedule→get\_minutesB()**

Retourne les minutes de l'intervall 30-59 de chaque heure où le réveil est actif.

#### **wakeupschedule→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **wakeupschedule→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **wakeupschedule→get\_monthDays()**

Retourne les jours du mois où le réveil est actif..

#### **wakeupschedule→get\_months()**

Retourne les mois où le réveil est actif..

#### **wakeupschedule→get\_nextOccurence()**

Retourne la date/heure de la prochaine occurrence de réveil

#### **wakeupschedule→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **wakeupschedule→get\_weekDays()**

Retourne les jours de la semaine où le réveil est actif..

#### **wakeupschedule→isOnline()**

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

#### **wakeupschedule→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

#### **wakeupschedule→load(msValidity)**

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

#### **wakeupschedule→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

#### **wakeupschedule→nextWakeUpSchedule()**

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`.

#### **wakeupschedule→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

#### **wakeupschedule→set\_hours(newval, newval)**

Modifie les heures où un réveil doit avoir lieu

#### **wakeupschedule→set\_logicalName(newval)**

Modifie le nom logique du réveil agendé.

#### **wakeupschedule→set\_minutes(bitmap)**

Modifie toutes les minutes où un réveil doit avoir lieu

#### **wakeupschedule→set\_minutesA(newval, newval)**

Modifie les minutes de l'intervall 00-29 où un réveil doit avoir lieu

#### **wakeupschedule→set\_minutesB(newval)**

Modifie les minutes de l'intervall 30-59 où un réveil doit avoir lieu.

#### **wakeupschedule→set\_monthDays(newval, newval)**

Modifie les jours du mois où un réveil doit avoir lieu

#### **wakeupschedule→set\_months(newval, newval)**

Modifie les mois où un réveil doit avoir lieu

#### **wakeupschedule→set\_userData(data)**

### 3. Reference

---

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

**wakeupschedule**→**set\_weekDays**(**newval**, **newval**)

Modifie les jours de la semaine où un réveil doit avoir lieu

**wakeupschedule**→**wait\_async**(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YWakeUpSchedule.FindWakeUpSchedule() yFindWakeUpSchedule()yFindWakeUpSchedule()

## YWakeupSchedule

Permet de retrouver un réveil agendé d'après un identifiant donné.

js	function <b>yFindWakeUpSchedule</b> ( <b>func</b> )
nodejs	function <b>FindWakeUpSchedule</b> ( <b>func</b> )
php	function <b>yFindWakeUpSchedule</b> ( <b>\$func</b> )
cpp	YWakeupSchedule* <b>yFindWakeUpSchedule</b> ( const string& <b>func</b> )
m	YWakeupSchedule* <b>yFindWakeUpSchedule</b> ( NSString* <b>func</b> )
pas	function <b>yFindWakeUpSchedule</b> ( <b>func</b> : string): TYWakeUpSchedule
vb	function <b>yFindWakeUpSchedule</b> ( ByVal <b>func</b> As String) As YWakeUpSchedule
cs	YWakeupSchedule <b>FindWakeUpSchedule</b> ( string <b>func</b> )
java	YWakeupSchedule <b>FindWakeUpSchedule</b> ( String <b>func</b> )
py	def <b>FindWakeUpSchedule</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le réveil agendé soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeupSchedule.isOnline()` pour tester si le réveil agendé est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le réveil agendé sans ambiguïté

### Retourne :

un objet de classe `YWakeupSchedule` qui permet ensuite de contrôler le réveil agendé.

## YWakeUpSchedule.FirstWakeUpSchedule() yFirstWakeUpSchedule()yFirstWakeUpSchedule()

## YWakeupSchedule

Commence l'énumération des réveils agendés accessibles par la librairie.

js	function <b>yFirstWakeUpSchedule</b> ( )
nodejs	function <b>FirstWakeUpSchedule</b> ( )
php	function <b>yFirstWakeUpSchedule</b> ( )
cpp	YWakeupSchedule* <b>yFirstWakeUpSchedule</b> ( )
m	YWakeupSchedule* <b>yFirstWakeUpSchedule</b> ( )
pas	function <b>yFirstWakeUpSchedule</b> ( ): TYWakeUpSchedule
vb	function <b>yFirstWakeUpSchedule</b> ( ) As YWakeupSchedule
cs	YWakeupSchedule <b>FirstWakeUpSchedule</b> ( )
java	YWakeupSchedule <b>FirstWakeUpSchedule</b> ( )
py	def <b>FirstWakeUpSchedule</b> ( )

Utiliser la fonction `YWakeupSchedule.nextWakeUpSchedule( )` pour itérer sur les autres réveils agendés.

### Retourne :

un pointeur sur un objet `YWakeupSchedule`, correspondant à le premier réveil agendé accessible en ligne, ou `null` si il n'y a pas de réveils agendés disponibles.

## wakeupschedule→describe()[wakeupschedule describe]

## YWakeUpSchedule

Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debugueur.

### Retourne :

une chaîne de caractères décrivant le réveil agendé (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

**wakeupschedule**→**get\_advertisedValue()****YWakeUpSchedule****wakeupschedule**→**advertisedValue()****[wakeupschedule advertisedValue]**

Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_advertisedValue</b>

**Retourne :**

une chaîne de caractères représentant la valeur courante du réveil agendé (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.

**wakeupschedule→get\_errorMessage()**  
**wakeupschedule→errorMessage()[wakeupschedule**  
**errorMessage]**

**YWakeUpSchedule**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.

**wakeupschedule→get\_errorType()**

**YWakeUpSchedule**

**wakeupschedule→errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.



**wakeupschedule→get\_friendlyName()****YWakeUpSchedule****wakeupschedule→friendlyName()[wakeupschedule  
friendlyName]**

Retourne un identifiant global du réveil agendé au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du réveil agendé si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du réveil agendé (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le réveil agendé en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

**wakeupschedule**→**get\_functionDescriptor()**  
**wakeupschedule**→**functionDescriptor()**  
**[wakeupschedule functionDescriptor]**

**YWakeUpSchedule**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

**Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**wakeupschedule→get\_functionId()****YWakeUpSchedule****wakeupschedule→functionId()[wakeupschedule  
functionId]**

Retourne l'identifiant matériel du réveil agendé, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) <b>functionId</b>
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le réveil agendé (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**wakeupschedule**→**get\_hardwareId()**

**YWakeUpSchedule**

**wakeupschedule**→**hardwareId()[wakeupschedule hardwareId]**

Retourne l'identifiant matériel unique du réveil agendé au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du réveil agendé (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le réveil agendé (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.

**wakeupschedule→get\_hours()****YWakeUpSchedule****wakeupschedule→hours()[wakeupschedule hours]**

Retourne les heures où le réveil est actif..

js	function <b>get_hours</b> ( )
nodejs	function <b>get_hours</b> ( )
php	function <b>get_hours</b> ( )
cpp	int <b>get_hours</b> ( )
m	-(int) hours
pas	function <b>get_hours</b> ( ): LongInt
vb	function <b>get_hours</b> ( ) As Integer
cs	int <b>get_hours</b> ( )
java	int <b>get_hours</b> ( )
py	def <b>get_hours</b> ( )
cmd	YWakeUpSchedule <b>target get_hours</b>

**Retourne :**

un entier représentant les heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_HOURS\_INVALID.

**wakeupschedule**→**get\_logicalName()****YWakeUpSchedule****wakeupschedule**→**logicalName()**[wakeupschedule  
**logicalName**]

Retourne le nom logique du réveil agendé.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du réveil agendé. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**wakeupschedule→get\_minutes()****YWakeUpSchedule****wakeupschedule→minutes()[wakeupschedule  
minutes]**

Retourne toutes les minutes de chaque heure où le réveil est actif.

js	function <b>get_minutes</b> ( )
nodejs	function <b>get_minutes</b> ( )
php	function <b>get_minutes</b> ( )
cpp	s64 <b>get_minutes</b> ( )
m	-(s64) minutes
pas	function <b>get_minutes</b> ( ): int64
vb	function <b>get_minutes</b> ( ) As Long
cs	long <b>get_minutes</b> ( )
java	long <b>get_minutes</b> ( )
py	def <b>get_minutes</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_minutes</b>

**wakeupschedule**→**get\_minutesA()****YWakeUpSchedule****wakeupschedule**→**minutesA()**[**wakeupschedule**  
**minutesA**]

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

js	function <b>get_minutesA</b> ( )
nodejs	function <b>get_minutesA</b> ( )
php	function <b>get_minutesA</b> ( )
cpp	int <b>get_minutesA</b> ( )
m	-(int) minutesA
pas	function <b>get_minutesA</b> ( ): LongInt
vb	function <b>get_minutesA</b> ( ) As Integer
cs	int <b>get_minutesA</b> ( )
java	int <b>get_minutesA</b> ( )
py	def <b>get_minutesA</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_minutesA</b>

**Retourne :**

un entier représentant les minutes de l'interval 00-29 de chaque heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_MINUTESA\_INVALID.



**wakeupschedule→get\_minutesB()****YWakeUpSchedule****wakeupschedule→minutesB()[wakeupschedule  
minutesB]**

Retourne les minutes de l'intervall 30-59 de chaque heure où le réveil est actif.

js	function <b>get_minutesB</b> ( )
nodejs	function <b>get_minutesB</b> ( )
php	function <b>get_minutesB</b> ( )
cpp	int <b>get_minutesB</b> ( )
m	-(int) minutesB
pas	function <b>get_minutesB</b> ( ): LongInt
vb	function <b>get_minutesB</b> ( ) As Integer
cs	int <b>get_minutesB</b> ( )
java	int <b>get_minutesB</b> ( )
py	def <b>get_minutesB</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_minutesB</b>

**Retourne :**

un entier représentant les minutes de l'intervall 30-59 de chaque heure où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_MINUTESB\_INVALID.

**wakeupschedule**→**get\_module()****YWakeUpSchedule****wakeupschedule**→**module()[wakeupschedule module]**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
cpp	<code>YModule *</code> <b>get_module</b> ( )
m	<code>-(YModule*)</code> module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

---

**wakeupschedule→get\_module\_async()****YWakeUpSchedule****wakeupschedule→module\_async()**

---

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**wakeupschedule**→**get\_monthDays()****YWakeUpSchedule****wakeupschedule**→**monthDays()[wakeupschedule  
monthDays]**

Retourne les jours du mois où le réveil est actif..

js	function <b>get_monthDays</b> ( )
nodejs	function <b>get_monthDays</b> ( )
php	function <b>get_monthDays</b> ( )
cpp	int <b>get_monthDays</b> ( )
m	-(int) monthDays
pas	function <b>get_monthDays</b> ( ): LongInt
vb	function <b>get_monthDays</b> ( ) As Integer
cs	int <b>get_monthDays</b> ( )
java	int <b>get_monthDays</b> ( )
py	def <b>get_monthDays</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_monthDays</b>

**Retourne :**

un entier représentant les jours du mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_MONTHDAYS\_INVALID.

**wakeupschedule→get\_months()**  
**wakeupschedule→months()[wakeupschedule**  
**months]**

**YWakeUpSchedule**

Retourne les mois où le réveil est actif..

js	function <b>get_months</b> ( )
nodejs	function <b>get_months</b> ( )
php	function <b>get_months</b> ( )
cpp	int <b>get_months</b> ( )
m	-(int) months
pas	function <b>get_months</b> ( ): LongInt
vb	function <b>get_months</b> ( ) As Integer
cs	int <b>get_months</b> ( )
java	int <b>get_months</b> ( )
py	def <b>get_months</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_months</b>

**Retourne :**

un entier représentant les mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_MONTHS\_INVALID.

**wakeupschedule**→**get\_nextOccurence()**

**YWakeUpSchedule**

**wakeupschedule**→**nextOccurence()**[**wakeupschedule**  
**nextOccurence**]

Retourne la date/heure de la prochaine occurrence de réveil

js	function <b>get_nextOccurence</b> ( )
nodejs	function <b>get_nextOccurence</b> ( )
php	function <b>get_nextOccurence</b> ( )
cpp	s64 <b>get_nextOccurence</b> ( )
m	-(s64) nextOccurence
pas	function <b>get_nextOccurence</b> ( ): int64
vb	function <b>get_nextOccurence</b> ( ) As Long
cs	long <b>get_nextOccurence</b> ( )
java	long <b>get_nextOccurence</b> ( )
py	def <b>get_nextOccurence</b> ( )

**Retourne :**

un entier représentant la date/heure de la prochaine occurrence de réveil

En cas d'erreur, déclenche une exception ou retourne Y\_NEXTOCCURENCE\_INVALID.

**wakeupschedule→get\_userdata()****YWakeUpSchedule****wakeupschedule→userData()[wakeupschedule  
userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**wakeupschedule**→**get\_weekDays()****YWakeUpSchedule****wakeupschedule**→**weekDays()[wakeupschedule weekDays]**

Retourne les jours de la semaine où le réveil est actif..

js	function <b>get_weekDays</b> ( )
nodejs	function <b>get_weekDays</b> ( )
php	function <b>get_weekDays</b> ( )
cpp	int <b>get_weekDays</b> ( )
m	-(int) weekDays
pas	function <b>get_weekDays</b> ( ): LongInt
vb	function <b>get_weekDays</b> ( ) As Integer
cs	int <b>get_weekDays</b> ( )
java	int <b>get_weekDays</b> ( )
py	def <b>get_weekDays</b> ( )
cmd	YWakeUpSchedule <b>target</b> <b>get_weekDays</b>

**Retourne :**

un entier représentant les jours de la semaine où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne Y\_WEEKDAYS\_INVALID.



## wakeupschedule→isOnline()[wakeupschedule isOnline]

## YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

### Retourne :

true si le réveil agendé est joignable, false sinon

## wakeupschedule→isOnline\_async()

## YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**wakeupschedule→load()[wakeupschedule load: ]****YWakeUpSchedule**

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
node.js	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule→load\_async()****YWakeUpSchedule**

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## wakeupschedule→nextWakeUpSchedule() [wakeupschedule nextWakeUpSchedule]

## YWakeUpSchedule

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`.

js	function <b>nextWakeUpSchedule</b> ( )
nodejs	function <b>nextWakeUpSchedule</b> ( )
php	function <b>nextWakeUpSchedule</b> ( )
cpp	YWakeupSchedule * <b>nextWakeUpSchedule</b> ( )
m	-(YWakeupSchedule*) <b>nextWakeUpSchedule</b>
pas	function <b>nextWakeUpSchedule</b> ( ): TYWakeUpSchedule
vb	function <b>nextWakeUpSchedule</b> ( ) As YWakeUpSchedule
cs	YWakeupSchedule <b>nextWakeUpSchedule</b> ( )
java	YWakeupSchedule <b>nextWakeUpSchedule</b> ( )
py	def <b>nextWakeUpSchedule</b> ( )

### Retourne :

un pointeur sur un objet `YWakeupSchedule` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## wakeupschedule→registerValueCallback() [wakeupschedule registerValueCallback: ]

## YWakeUpSchedule

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YWakeUpScheduleValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YWakeUpScheduleValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYWakeUpScheduleValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**wakeupschedule→set\_hours()****YWakeUpSchedule****wakeupschedule→setHours()[wakeupschedule  
setHours: ]**

Modifie les heures où un réveil doit avoir lieu

js	function <b>set_hours</b> ( <b>newval</b> )
nodejs	function <b>set_hours</b> ( <b>newval</b> )
php	function <b>set_hours</b> ( <b>\$newval</b> )
cpp	int <b>set_hours</b> ( int <b>newval</b> )
m	-(int) setHours : (int) <b>newval</b>
pas	function <b>set_hours</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_hours</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_hours</b> ( int <b>newval</b> )
java	int <b>set_hours</b> ( int <b>newval</b> )
py	def <b>set_hours</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target set_hours newval</b>

**Paramètres :****newval** un entier représentant les heures où un réveil doit avoir lieu**newval** un entier**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule→set\_logicalName()****YWakeUpSchedule****wakeupschedule→setLogicalName()****[wakeupschedule setLogicalName: ]**

Modifie le nom logique du réveil agendé.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique du réveil agendé.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**wakeupschedule→set\_minutes()****YWakeUpSchedule****wakeupschedule→setMinutes()[wakeupschedule  
setMinutes: ]**

Modifie toutes les minutes où un réveil doit avoir lieu

js	function <b>set_minutes</b> ( <b>bitmap</b> )
nodejs	function <b>set_minutes</b> ( <b>bitmap</b> )
php	function <b>set_minutes</b> ( <b>\$bitmap</b> )
cpp	int <b>set_minutes</b> ( s64 <b>bitmap</b> )
m	-(int) setMinutes : (s64) <b>bitmap</b>
pas	function <b>set_minutes</b> ( <b>bitmap</b> : int64): LongInt
vb	function <b>set_minutes</b> ( ) As Integer
cs	int <b>set_minutes</b> ( long <b>bitmap</b> )
java	int <b>set_minutes</b> ( long <b>bitmap</b> )
py	def <b>set_minutes</b> ( <b>bitmap</b> )
cmd	YWakeUpSchedule <b>target set_minutes bitmap</b>

**Paramètres :****bitmap** Minutes 00-59 de chaque heure où le réveil est actif.**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule**→**set\_minutesA()**

**YWakeUpSchedule**

**wakeupschedule**→**setMinutesA()**[wakeupschedule  
**setMinutesA: ]**

Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu

js	function <b>set_minutesA</b> ( <b>newval</b> )
nodejs	function <b>set_minutesA</b> ( <b>newval</b> )
php	function <b>set_minutesA</b> ( <b>\$newval</b> )
cpp	int <b>set_minutesA</b> ( int <b>newval</b> )
m	-(int) setMinutesA : (int) <b>newval</b>
pas	function <b>set_minutesA</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_minutesA</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_minutesA</b> ( int <b>newval</b> )
java	int <b>set_minutesA</b> ( int <b>newval</b> )
py	def <b>set_minutesA</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target set_minutesA newval</b>

#### Paramètres :

**newval** un entier représentant les minutes de l'interval 00-29 où un réveil doit avoir lieu

**newval** un entier

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule→set\_minutesB()****YWakeUpSchedule****wakeupschedule→setMinutesB()[wakeupschedule  
setMinutesB: ]**

Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

js	function <b>set_minutesB</b> ( <b>newval</b> )
nodejs	function <b>set_minutesB</b> ( <b>newval</b> )
php	function <b>set_minutesB</b> ( <b>\$newval</b> )
cpp	int <b>set_minutesB</b> ( int <b>newval</b> )
m	-(int) setMinutesB : (int) <b>newval</b>
pas	function <b>set_minutesB</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_minutesB</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_minutesB</b> ( int <b>newval</b> )
java	int <b>set_minutesB</b> ( int <b>newval</b> )
py	def <b>set_minutesB</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target set_minutesB newval</b>

**Paramètres :**

**newval** un entier représentant les minutes de l'interval 30-59 où un réveil doit avoir lieu

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule**→**set\_monthDays()****YWakeUpSchedule****wakeupschedule**→**setMonthDays()[wakeupschedule  
setMonthDays: ]**

Modifie les jours du mois où un réveil doit avoir lieu

js	function <b>set_monthDays</b> ( <b>newval</b> )
nodejs	function <b>set_monthDays</b> ( <b>newval</b> )
php	function <b>set_monthDays</b> ( <b>\$newval</b> )
cpp	int <b>set_monthDays</b> ( int <b>newval</b> )
m	-(int) setMonthDays : (int) <b>newval</b>
pas	function <b>set_monthDays</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_monthDays</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_monthDays</b> ( int <b>newval</b> )
java	int <b>set_monthDays</b> ( int <b>newval</b> )
py	def <b>set_monthDays</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target set_monthDays newval</b>

**Paramètres :****newval** un entier représentant les jours du mois où un réveil doit avoir lieu**newval** un entier**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule→set\_months()****YWakeUpSchedule****wakeupschedule→setMonths()[wakeupschedule  
setMonths: ]**

Modifie les mois où un réveil doit avoir lieu

js	function <b>set_months</b> ( <b>newval</b> )
nodejs	function <b>set_months</b> ( <b>newval</b> )
php	function <b>set_months</b> ( <b>\$newval</b> )
cpp	int <b>set_months</b> ( int <b>newval</b> )
m	-(int) setMonths : (int) <b>newval</b>
pas	function <b>set_months</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_months</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_months</b> ( int <b>newval</b> )
java	int <b>set_months</b> ( int <b>newval</b> )
py	def <b>set_months</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target set_months newval</b>

**Paramètres :****newval** un entier représentant les mois où un réveil doit avoir lieu**newval** un entier**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wakeupschedule**→**set\_userdata()****YWakeUpSchedule****wakeupschedule**→**setUserData()**[**wakeupschedule**  
**setUserData:** ]

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**wakeupschedule→set\_weekDays()****YWakeUpSchedule****wakeupschedule→setWeekDays()[wakeupschedule  
setWeekDays: ]**

Modifie les jours de la semaine où un réveil doit avoir lieu

js	function <b>set_weekDays</b> ( <b>newval</b> )
nodejs	function <b>set_weekDays</b> ( <b>newval</b> )
php	function <b>set_weekDays</b> ( <b>\$newval</b> )
cpp	int <b>set_weekDays</b> ( int <b>newval</b> )
m	-(int) setWeekDays : (int) <b>newval</b>
pas	function <b>set_weekDays</b> ( <b>newval</b> : LongInt): integer
vb	function <b>set_weekDays</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_weekDays</b> ( int <b>newval</b> )
java	int <b>set_weekDays</b> ( int <b>newval</b> )
py	def <b>set_weekDays</b> ( <b>newval</b> )
cmd	YWakeUpSchedule <b>target</b> <b>set_weekDays</b> <b>newval</b>

**Paramètres :****newval** un entier représentant les jours de la semaine où un réveil doit avoir lieu**newval** un entier**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## wakeupschedule→wait\_async()

## YWakeUpSchedule

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout :



## 3.45. Interface de la fonction Watchdog

La fonction WatchDog est gérée comme un relais qui couperait brièvement l'alimentation d'un appareil après un d'attente temps donné afin de provoquer une réinitialisation complète de cet appareil. Il suffit d'appeler le watchdog à intervalle régulier pour l'empêcher de provoquer la réinitialisation. Le watchdog peut aussi être piloté directement à l'aide des méthode *pulse* et *delayedpulse* pour éteindre un appareil pendant un temps donné.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_watchdog.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YWatchdog = yoctolib.YWatchdog;
php	require_once('yocto_watchdog.php');
c++	#include "yocto_watchdog.h"
m	#import "yocto_watchdog.h"
pas	uses yocto_watchdog;
vb	yocto_watchdog.vb
cs	yocto_watchdog.cs
java	import com.yoctopuce.YoctoAPI.YWatchdog;
py	from yocto_watchdog import *

### Fonction globales

#### yFindWatchdog(**func**)

Permet de retrouver un watchdog d'après un identifiant donné.

#### yFirstWatchdog()

Commence l'énumération des watchdog accessibles par la librairie.

### Méthodes des objets YWatchdog

#### watchdog→**delayedPulse**(**ms\_delay**, **ms\_duration**)

Pré-programme une impulsion

#### watchdog→**describe**()

Retourne un court texte décrivant de manière non-ambigüe l'instance du watchdog au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### watchdog→**get\_advertisedValue**()

Retourne la valeur courante du watchdog (pas plus de 6 caractères).

#### watchdog→**get\_autoStart**()

Retourne l'état du watchdog à la mise sous tension du module.

#### watchdog→**get\_countdown**()

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`.

#### watchdog→**get\_errorMessage**()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

#### watchdog→**get\_errorType**()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

#### watchdog→**get\_friendlyName**()

Retourne un identifiant global du watchdog au format `NOM_MODULE . NOM_FONCTION`.

#### watchdog→**get\_functionDescriptor**()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

#### watchdog→**get\_functionId**()

Retourne l'identifiant matériel du watchdog, sans référence au module.

#### **watchdog→get\_hardwareId()**

Retourne l'identifiant matériel unique du watchdog au format `SERIAL.FUNCTIONID`.

#### **watchdog→get\_logicalName()**

Retourne le nom logique du watchdog.

#### **watchdog→get\_maxTimeOnStateA()**

Retourne le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état A avant de basculer automatiquement dans l'état B.

#### **watchdog→get\_maxTimeOnStateB()**

Retourne le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état B avant de basculer automatiquement dans l'état A.

#### **watchdog→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **watchdog→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

#### **watchdog→get\_output()**

Retourne l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

#### **watchdog→get\_pulseTimer()**

Retourne le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée.

#### **watchdog→get\_running()**

Retourne l'état du watchdog.

#### **watchdog→get\_state()**

Retourne l'état du watchdog (A pour la position de repos, B pour l'état actif).

#### **watchdog→get\_stateAtPowerOn()**

Retourne l'état du watchdog au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

#### **watchdog→get\_triggerDelay()**

Retourne le délai d'attente avant qu'un reset ne soit automatiquement généré par le watchdog, en millisecondes.

#### **watchdog→get\_triggerDuration()**

Retourne la durée d'un reset généré par le watchdog, en millisecondes.

#### **watchdog→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

#### **watchdog→isOnline()**

Vérifie si le module hébergeant le watchdog est joignable, sans déclencher d'erreur.

#### **watchdog→isOnline\_async(callback, context)**

Vérifie si le module hébergeant le watchdog est joignable, sans déclencher d'erreur.

#### **watchdog→load(msValidity)**

Met en cache les valeurs courantes du watchdog, avec une durée de validité spécifiée.

#### **watchdog→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes du watchdog, avec une durée de validité spécifiée.

#### **watchdog→nextWatchdog()**

Continue l'énumération des watchdog commencée à l'aide de `yFirstWatchdog()`.

#### **watchdog→pulse(ms\_duration)**

Commute le relais à l'état B (actif) pour une durée spécifiée, puis revient ensuite spontanément vers l'état A (état de repos).

**watchdog→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**watchdog→resetWatchdog()**

Réinitialise le WatchDog.

**watchdog→set\_autoStart(newval)**

Modifie l'état du watching au démarrage du module.

**watchdog→set\_logicalName(newval)**

Modifie le nom logique du watchdog.

**watchdog→set\_maxTimeOnStateA(newval)**

Règle le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état A avant de basculer automatiquement dans l'état B.

**watchdog→set\_maxTimeOnStateB(newval)**

Règle le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état B avant de basculer automatiquement dans l'état A.

**watchdog→set\_output(newval)**

Modifie l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

**watchdog→set\_running(newval)**

Modifie manuellement l'état de fonctionnement du watchdog.

**watchdog→set\_state(newval)**

Modifie l'état du watchdog (A pour la position de repos, B pour l'état actif).

**watchdog→set\_stateAtPowerOn(newval)**

Pré-programme l'état du watchdog au démarrage du module(A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

**watchdog→set\_triggerDelay(newval)**

Modifie le délai d'attente avant qu'un reset ne soit généré par le watchdog, en millisecondes.

**watchdog→set\_triggerDuration(newval)**

Modifie la durée des resets générés par le watchdog, en millisecondes.

**watchdog→set\_userData(data)**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

**watchdog→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YWatchdog.FindWatchdog() yFindWatchdog()yFindWatchdog()

YWatchdog

Permet de retrouver un watchdog d'après un identifiant donné.

js	function <b>yFindWatchdog</b> ( <b>func</b> )
nodejs	function <b>FindWatchdog</b> ( <b>func</b> )
php	function <b>yFindWatchdog</b> ( <b>\$func</b> )
cpp	YWatchdog* <b>yFindWatchdog</b> ( const string& <b>func</b> )
m	YWatchdog* <b>yFindWatchdog</b> ( NSString* <b>func</b> )
pas	function <b>yFindWatchdog</b> ( <b>func</b> : string): TYWatchdog
vb	function <b>yFindWatchdog</b> ( ByVal <b>func</b> As String) As YWatchdog
cs	YWatchdog <b>FindWatchdog</b> ( string <b>func</b> )
java	YWatchdog <b>FindWatchdog</b> ( String <b>func</b> )
py	def <b>FindWatchdog</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le watchdog soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWatchdog.isOnline()` pour tester si le watchdog est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence le watchdog sans ambiguïté

### Retourne :

un objet de classe `YWatchdog` qui permet ensuite de contrôler le watchdog.

## YWatchdog.FirstWatchdog() yFirstWatchdog()yFirstWatchdog()

## YWatchdog

Commence l'énumération des watchdog accessibles par la librairie.

js	function <b>yFirstWatchdog</b> ( )
nodejs	function <b>FirstWatchdog</b> ( )
php	function <b>yFirstWatchdog</b> ( )
cpp	YWatchdog* <b>yFirstWatchdog</b> ( )
m	YWatchdog* <b>yFirstWatchdog</b> ( )
pas	function <b>yFirstWatchdog</b> ( ): TYWatchdog
vb	function <b>yFirstWatchdog</b> ( ) As YWatchdog
cs	YWatchdog <b>FirstWatchdog</b> ( )
java	YWatchdog <b>FirstWatchdog</b> ( )
py	def <b>FirstWatchdog</b> ( )

Utiliser la fonction `YWatchdog.nextWatchdog( )` pour itérer sur les autres watchdog.

### Retourne :

un pointeur sur un objet `YWatchdog`, correspondant à le premier watchdog accessible en ligne, ou `null` si il n'y a pas de watchdog disponibles.

**watchdog→delayedPulse()[watchdog delayedPulse: ]****YWatchdog**

Pré-programme une impulsion

js	function <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
nodejs	function <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
php	function <b>delayedPulse</b> ( <b>\$ms_delay</b> , <b>\$ms_duration</b> )
cpp	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
m	-(int) <b>delayedPulse</b> : (int) <b>ms_delay</b> : (int) <b>ms_duration</b>
pas	function <b>delayedPulse</b> ( <b>ms_delay</b> : LongInt, <b>ms_duration</b> : LongInt): integer
vb	function <b>delayedPulse</b> ( ByVal <b>ms_delay</b> As Integer, ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
java	int <b>delayedPulse</b> ( int <b>ms_delay</b> , int <b>ms_duration</b> )
py	def <b>delayedPulse</b> ( <b>ms_delay</b> , <b>ms_duration</b> )
cmd	YWatchdog <b>target</b> <b>delayedPulse</b> <b>ms_delay</b> <b>ms_duration</b>

**Paramètres :****ms\_delay**    delai d'attente avant l'impulsion, en millisecondes**ms\_duration**    durée de l'impulsion, en millisecondes**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→describe()[watchdog describe]****YWatchdog**

Retourne un court texte décrivant de manière non-ambigüe l'instance du watchdog au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant le watchdog (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## watchdog→get\_advertisedValue() watchdog→advertisedValue()[watchdog advertisedValue]

YWatchdog

Retourne la valeur courante du watchdog (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YWatchdog <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante du watchdog (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.



**watchdog→get\_autoStart()****YWatchdog****watchdog→autoStart()[watchdog autoStart]**

Retourne l'état du watchdog à la mise sous tension du module.

js	function <b>get_autoStart</b> ( )
nodejs	function <b>get_autoStart</b> ( )
php	function <b>get_autoStart</b> ( )
cpp	Y_AUTOSTART_enum <b>get_autoStart</b> ( )
m	-(Y_AUTOSTART_enum) autoStart
pas	function <b>get_autoStart</b> ( ): Integer
vb	function <b>get_autoStart</b> ( ) As Integer
cs	int <b>get_autoStart</b> ( )
java	int <b>get_autoStart</b> ( )
py	def <b>get_autoStart</b> ( )
cmd	YWatchdog <b>target</b> <b>get_autoStart</b>

**Retourne :**

soit Y\_AUTOSTART\_OFF, soit Y\_AUTOSTART\_ON, selon l'état du watchdog à la mise sous tension du module

En cas d'erreur, déclenche une exception ou retourne Y\_AUTOSTART\_INVALID.

**watchdog→get\_countdown()****YWatchdog****watchdog→countdown()[watchdog countdown]**

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`.

js	function <b>get_countdown</b> ( )
nodejs	function <b>get_countdown</b> ( )
php	function <b>get_countdown</b> ( )
cpp	s64 <b>get_countdown</b> ( )
m	-(s64) countdown
pas	function <b>get_countdown</b> ( ): int64
vb	function <b>get_countdown</b> ( ) As Long
cs	long <b>get_countdown</b> ( )
java	long <b>get_countdown</b> ( )
py	def <b>get_countdown</b> ( )
cmd	YWatchdog <b>target</b> <b>get_countdown</b>

Si aucune impulsion n'est programmée, retourne zéro.

**Retourne :**

un entier représentant le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`

En cas d'erreur, déclenche une exception ou retourne `Y_COUNTDOWN_INVALID`.

**watchdog→get\_errorMessage()****YWatchdog****watchdog→errorMessage()[watchdog errorMessage]**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du watchdog.

**watchdog**→**get\_errorType()****YWatchdog****watchdog**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du watchdog.

**watchdog→get\_friendlyName()****YWatchdog****watchdog→friendlyName()[watchdog friendlyName]**

Retourne un identifiant global du watchdog au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et du watchdog si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel du watchdog (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant le watchdog en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## **watchdog→get\_functionDescriptor() watchdog→functionDescriptor()[watchdog functionDescriptor]**

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### **Retourne :**

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**watchdog→get\_functionId()****YWatchdog****watchdog→functionId()[watchdog functionId]**

Retourne l'identifiant matériel du watchdog, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant le watchdog (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**watchdog**→**get\_hardwareId()****YWatchdog****watchdog**→**hardwareId()**[**watchdog hardwareId**]

Retourne l'identifiant matériel unique du watchdog au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
c++	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du watchdog (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant le watchdog (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**watchdog→get\_logicalName()****YWatchdog****watchdog→logicalName()[watchdog logicalName]**

Retourne le nom logique du watchdog.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YWatchdog <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique du watchdog. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

## watchdog→get\_maxTimeOnStateA() watchdog→maxTimeOnStateA()[watchdog maxTimeOnStateA]

YWatchdog

Retourne le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état A avant de basculer automatiquement dans l'état B.

js	function <b>get_maxTimeOnStateA</b> ( )
nodejs	function <b>get_maxTimeOnStateA</b> ( )
php	function <b>get_maxTimeOnStateA</b> ( )
cpp	s64 <b>get_maxTimeOnStateA</b> ( )
m	-(s64) maxTimeOnStateA
pas	function <b>get_maxTimeOnStateA</b> ( ): int64
vb	function <b>get_maxTimeOnStateA</b> ( ) As Long
cs	long <b>get_maxTimeOnStateA</b> ( )
java	long <b>get_maxTimeOnStateA</b> ( )
py	def <b>get_maxTimeOnStateA</b> ( )
cmd	YWatchdog <b>target</b> <b>get_maxTimeOnStateA</b>

Zéro signifie qu'il n'y a pas de limitation

### Retourne :

un entier représentant le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état A avant de basculer automatiquement dans l'état B

En cas d'erreur, déclenche une exception ou retourne Y\_MAXTIMEONSTATEA\_INVALID.

**watchdog→get\_maxTimeOnStateB()****YWatchdog****watchdog→maxTimeOnStateB()[watchdog  
maxTimeOnStateB]**

Retourne le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état B avant de basculer automatiquement dans l'état A.

js	function <b>get_maxTimeOnStateB</b> ( )
nodejs	function <b>get_maxTimeOnStateB</b> ( )
php	function <b>get_maxTimeOnStateB</b> ( )
cpp	s64 <b>get_maxTimeOnStateB</b> ( )
m	-(s64) maxTimeOnStateB
pas	function <b>get_maxTimeOnStateB</b> ( ): int64
vb	function <b>get_maxTimeOnStateB</b> ( ) As Long
cs	long <b>get_maxTimeOnStateB</b> ( )
java	long <b>get_maxTimeOnStateB</b> ( )
py	def <b>get_maxTimeOnStateB</b> ( )
cmd	YWatchdog <b>target</b> <b>get_maxTimeOnStateB</b>

Zéro signifie qu'il n'y a pas de limitation

**Retourne :**

un entier représentant le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état B avant de basculer automatiquement dans l'état A

En cas d'erreur, déclenche une exception ou retourne Y\_MAXTIMEONSTATEB\_INVALID.

**watchdog**→**get\_module()****YWatchdog****watchdog**→**module()**[**watchdog module**]

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	<code>YModule *</code> <b>get_module</b> ( )
m	-( <code>YModule*</code> ) module
pas	function <b>get_module</b> ( ): <code>TYModule</code>
vb	function <b>get_module</b> ( ) As <code>YModule</code>
cs	<code>YModule</code> <b>get_module</b> ( )
java	<code>YModule</code> <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

**Retourne :**

une instance de `YModule`

## watchdog→get\_module\_async() watchdog→module\_async()

## YWatchdog

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

### Paramètres :

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**watchdog**→**get\_output()****YWatchdog****watchdog**→**output()**[**watchdog output**]

Retourne l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

js	function <b>get_output</b> ( )
nodejs	function <b>get_output</b> ( )
php	function <b>get_output</b> ( )
cpp	Y_OUTPUT_enum <b>get_output</b> ( )
m	-(Y_OUTPUT_enum) output
pas	function <b>get_output</b> ( ): Integer
vb	function <b>get_output</b> ( ) As Integer
cs	int <b>get_output</b> ( )
java	int <b>get_output</b> ( )
py	def <b>get_output</b> ( )
cmd	YWatchdog <b>target</b> <b>get_output</b>

**Retourne :**

soit Y\_OUTPUT\_OFF, soit Y\_OUTPUT\_ON, selon l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur

En cas d'erreur, déclenche une exception ou retourne Y\_OUTPUT\_INVALID.

**watchdog→get\_pulseTimer()****YWatchdog****watchdog→pulseTimer()[watchdog pulseTimer]**

Retourne le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée.

js	function <b>get_pulseTimer</b> ( )
nodejs	function <b>get_pulseTimer</b> ( )
php	function <b>get_pulseTimer</b> ( )
cpp	s64 <b>get_pulseTimer</b> ( )
m	-(s64) pulseTimer
pas	function <b>get_pulseTimer</b> ( ): int64
vb	function <b>get_pulseTimer</b> ( ) As Long
cs	long <b>get_pulseTimer</b> ( )
java	long <b>get_pulseTimer</b> ( )
py	def <b>get_pulseTimer</b> ( )
cmd	YWatchdog <b>target</b> <b>get_pulseTimer</b>

Si aucune impulsion n'est en cours, retourne zéro.

**Retourne :**

un entier représentant le nombre de millisecondes restantes avant le retour à la position de repos (état A), durant la génération d'une impulsion mesurée

En cas d'erreur, déclenche une exception ou retourne Y\_PULSETIMER\_INVALID.

**watchdog**→**get\_running()****YWatchdog****watchdog**→**running()**[**watchdog running**]

Retourne l'état du watchdog.

js	function <b>get_running</b> ( )
nodejs	function <b>get_running</b> ( )
php	function <b>get_running</b> ( )
cpp	Y_RUNNING_enum <b>get_running</b> ( )
m	-(Y_RUNNING_enum) running
pas	function <b>get_running</b> ( ): Integer
vb	function <b>get_running</b> ( ) As Integer
cs	int <b>get_running</b> ( )
java	int <b>get_running</b> ( )
py	def <b>get_running</b> ( )
cmd	YWatchdog <b>target</b> <b>get_running</b>

**Retourne :**

soit Y\_RUNNING\_OFF, soit Y\_RUNNING\_ON, selon l'état du watchdog

En cas d'erreur, déclenche une exception ou retourne Y\_RUNNING\_INVALID.



**watchdog→get\_state()****YWatchdog****watchdog→state()[watchdog state]**

Retourne l'état du watchdog (A pour la position de repos, B pour l'état actif).

js	function <b>get_state</b> ( )
nodejs	function <b>get_state</b> ( )
php	function <b>get_state</b> ( )
cpp	Y_STATE_enum <b>get_state</b> ( )
m	-(Y_STATE_enum) state
pas	function <b>get_state</b> ( ): Integer
vb	function <b>get_state</b> ( ) As Integer
cs	int <b>get_state</b> ( )
java	int <b>get_state</b> ( )
py	def <b>get_state</b> ( )
cmd	YWatchdog <b>target</b> <b>get_state</b>

**Retourne :**

soit Y\_STATE\_A, soit Y\_STATE\_B, selon l'état du watchdog (A pour la position de repos, B pour l'état actif)

En cas d'erreur, déclenche une exception ou retourne Y\_STATE\_INVALID.

**watchdog→get\_stateAtPowerOn()****YWatchdog****watchdog→stateAtPowerOn()[watchdog  
stateAtPowerOn]**

Retourne l'état du watchdog au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

js	function <b>get_stateAtPowerOn</b> ( )
nodejs	function <b>get_stateAtPowerOn</b> ( )
php	function <b>get_stateAtPowerOn</b> ( )
cpp	Y_STATEATPOWERON_enum <b>get_stateAtPowerOn</b> ( )
m	-(Y_STATEATPOWERON_enum) stateAtPowerOn
pas	function <b>get_stateAtPowerOn</b> ( ): Integer
vb	function <b>get_stateAtPowerOn</b> ( ) As Integer
cs	int <b>get_stateAtPowerOn</b> ( )
java	int <b>get_stateAtPowerOn</b> ( )
py	def <b>get_stateAtPowerOn</b> ( )
cmd	YWatchdog <b>target</b> <b>get_stateAtPowerOn</b>

**Retourne :**

une valeur parmi Y\_STATEATPOWERON\_UNCHANGED, Y\_STATEATPOWERON\_A et Y\_STATEATPOWERON\_B représentant l'état du watchdog au démarrage du module (A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement)

En cas d'erreur, déclenche une exception ou retourne Y\_STATEATPOWERON\_INVALID.

**watchdog→get\_triggerDelay()****YWatchdog****watchdog→triggerDelay()[watchdog triggerDelay]**

Retourne le délai d'attente avant qu'un reset ne soit automatiquement généré par le watchdog, en millisecondes.

js	function <b>get_triggerDelay</b> ( )
nodejs	function <b>get_triggerDelay</b> ( )
php	function <b>get_triggerDelay</b> ( )
cpp	s64 <b>get_triggerDelay</b> ( )
m	-(s64) triggerDelay
pas	function <b>get_triggerDelay</b> ( ): int64
vb	function <b>get_triggerDelay</b> ( ) As Long
cs	long <b>get_triggerDelay</b> ( )
java	long <b>get_triggerDelay</b> ( )
py	def <b>get_triggerDelay</b> ( )
cmd	YWatchdog <b>target</b> <b>get_triggerDelay</b>

**Retourne :**

un entier représentant le délai d'attente avant qu'un reset ne soit automatiquement généré par le watchdog, en millisecondes

En cas d'erreur, déclenche une exception ou retourne Y\_TRIGGERDELAY\_INVALID.

## watchdog→get\_triggerDuration() watchdog→triggerDuration()[watchdog triggerDuration]

YWatchdog

Retourne la durée d'un reset généré par le watchdog, en millisecondes.

js	function <b>get_triggerDuration</b> ( )
nodejs	function <b>get_triggerDuration</b> ( )
php	function <b>get_triggerDuration</b> ( )
cpp	s64 <b>get_triggerDuration</b> ( )
m	-(s64) triggerDuration
pas	function <b>get_triggerDuration</b> ( ): int64
vb	function <b>get_triggerDuration</b> ( ) As Long
cs	long <b>get_triggerDuration</b> ( )
java	long <b>get_triggerDuration</b> ( )
py	def <b>get_triggerDuration</b> ( )
cmd	YWatchdog <b>target</b> <b>get_triggerDuration</b>

### Retourne :

un entier représentant la durée d'un reset généré par le watchdog, en millisecondes

En cas d'erreur, déclenche une exception ou retourne Y\_TRIGGERDURATION\_INVALID.

**watchdog→get\_userdata()****YWatchdog****watchdog→userdata()[watchdog userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
c++	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.

**watchdog**→**isOnline()**[**watchdog isOnline**]**YWatchdog**

Vérifie si le module hébergeant le watchdog est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache du watchdog sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

true si le watchdog est joignable, false sinon

## watchdog→isOnline\_async()

## YWatchdog

Vérifie si le module hébergeant le watchdog est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du watchdog sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

### Paramètres :

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

### Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**watchdog→load()[watchdog load: ]****YWatchdog**

Met en cache les valeurs courantes du watchdog, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**watchdog→load\_async()****YWatchdog**

Met en cache les valeurs courantes du watchdog, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou `YAPI_SUCCESS`)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## watchdog→nextWatchdog()[watchdog nextWatchdog]

YWatchdog

Continue l'énumération des watchdog commencée à l'aide de `yFirstWatchdog()`.

js	function <b>nextWatchdog</b> ( )
nodejs	function <b>nextWatchdog</b> ( )
php	function <b>nextWatchdog</b> ( )
c++	YWatchdog * <b>nextWatchdog</b> ( )
m	-(YWatchdog*) <b>nextWatchdog</b>
pas	function <b>nextWatchdog</b> ( ): TYWatchdog
vb	function <b>nextWatchdog</b> ( ) As YWatchdog
cs	YWatchdog <b>nextWatchdog</b> ( )
java	YWatchdog <b>nextWatchdog</b> ( )
py	def <b>nextWatchdog</b> ( )

### Retourne :

un pointeur sur un objet `YWatchdog` accessible en ligne, ou `null` lorsque l'énumération est terminée.

**watchdog→pulse()[watchdog pulse: ]****YWatchdog**

Commute le relais à l'état B (actif) pour une durée spécifiée, puis revient ensuite spontanément vers l'état A (état de repos).

js	function <b>pulse</b> ( <b>ms_duration</b> )
nodejs	function <b>pulse</b> ( <b>ms_duration</b> )
php	function <b>pulse</b> ( <b>\$ms_duration</b> )
cpp	int <b>pulse</b> ( int <b>ms_duration</b> )
m	-(int) <b>pulse</b> : (int) <b>ms_duration</b>
pas	function <b>pulse</b> ( <b>ms_duration</b> : LongInt): integer
vb	function <b>pulse</b> ( ByVal <b>ms_duration</b> As Integer) As Integer
cs	int <b>pulse</b> ( int <b>ms_duration</b> )
java	int <b>pulse</b> ( int <b>ms_duration</b> )
py	def <b>pulse</b> ( <b>ms_duration</b> )
cmd	YWatchdog <b>target pulse ms_duration</b>

**Paramètres :**

**ms\_duration** durée de l'impulsion, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## watchdog→registerValueCallback()[watchdog registerValueCallback: ]

YWatchdog

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YWatchdogValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YWatchdogValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYWatchdogValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

## watchdog→resetWatchdog()[watchdog resetWatchdog]

## YWatchdog

Réinitialise le WatchDog.

js	function <b>resetWatchdog</b> ( )
nodejs	function <b>resetWatchdog</b> ( )
php	function <b>resetWatchdog</b> ( )
cpp	int <b>resetWatchdog</b> ( )
m	-(int) <b>resetWatchdog</b>
pas	function <b>resetWatchdog</b> ( ): integer
vb	function <b>resetWatchdog</b> ( ) As Integer
cs	int <b>resetWatchdog</b> ( )
java	int <b>resetWatchdog</b> ( )
py	def <b>resetWatchdog</b> ( )
cmd	YWatchdog <b>target</b> <b>resetWatchdog</b>

Quand le watchdog est en fonctionnement cette fonction doit être appelée à interval régulier, pour empêcher que le watchdog ne se déclenche

### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_autoStart()****YWatchdog****watchdog→setAutoStart()[watchdog setAutoStart: ]**

Modifie l'état du watching au démarrage du module.

js	function <b>set_autoStart</b> ( <b>newval</b> )
nodejs	function <b>set_autoStart</b> ( <b>newval</b> )
php	function <b>set_autoStart</b> ( <b>\$newval</b> )
cpp	int <b>set_autoStart</b> ( Y_AUTOSTART_enum <b>newval</b> )
m	-(int) setAutoStart : (Y_AUTOSTART_enum) <b>newval</b>
pas	function <b>set_autoStart</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_autoStart</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_autoStart</b> ( int <b>newval</b> )
java	int <b>set_autoStart</b> ( int <b>newval</b> )
py	def <b>set_autoStart</b> ( <b>newval</b> )
cmd	YWatchdog <b>target set_autoStart newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**newval** soit Y\_AUTOSTART\_OFF, soit Y\_AUTOSTART\_ON, selon l'état du watching au démarrage du module

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## watchdog→set\_logicalName() watchdog→setLogicalName()[watchdog setLogicalName: ]

YWatchdog

Modifie le nom logique du watchdog.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YWatchdog <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName( )` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash( )` du module si le réglage doit être préservé.

### Paramètres :

**newval** une chaîne de caractères représentant le nom logique du watchdog.

### Retourne :

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_maxTimeOnStateA()****YWatchdog****watchdog→setMaxTimeOnStateA()[watchdog  
setMaxTimeOnStateA: ]**

Règle le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état A avant de basculer automatiquement dans l'état B.

js	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
nodejs	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
php	function <b>set_maxTimeOnStateA</b> ( <b>\$newval</b> )
cpp	int <b>set_maxTimeOnStateA</b> ( s64 <b>newval</b> )
m	-(int) setMaxTimeOnStateA : (s64) <b>newval</b>
pas	function <b>set_maxTimeOnStateA</b> ( <b>newval</b> : int64): integer
vb	function <b>set_maxTimeOnStateA</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_maxTimeOnStateA</b> ( long <b>newval</b> )
java	int <b>set_maxTimeOnStateA</b> ( long <b>newval</b> )
py	def <b>set_maxTimeOnStateA</b> ( <b>newval</b> )
cmd	YWatchdog <b>target set_maxTimeOnStateA newval</b>

Zéro signifie qu'il n'y a pas de limitation

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**watchdog→set\_maxTimeOnStateB()****YWatchdog****watchdog→setMaxTimeOnStateB()[watchdog  
setMaxTimeOnStateB: ]**

Règle le temps maximal (en ms) pendant lequel le watchdog peut rester dans l'état B avant de basculer automatiquement dans l'état A.

js	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
nodejs	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
php	function <b>set_maxTimeOnStateB</b> ( <b>\$newval</b> )
cpp	int <b>set_maxTimeOnStateB</b> ( s64 <b>newval</b> )
m	-(int) setMaxTimeOnStateB : (s64) <b>newval</b>
pas	function <b>set_maxTimeOnStateB</b> ( <b>newval</b> : int64): integer
vb	function <b>set_maxTimeOnStateB</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_maxTimeOnStateB</b> ( long <b>newval</b> )
java	int <b>set_maxTimeOnStateB</b> ( long <b>newval</b> )
py	def <b>set_maxTimeOnStateB</b> ( <b>newval</b> )
cmd	YWatchdog <b>target</b> <b>set_maxTimeOnStateB</b> <b>newval</b>

Zéro signifie qu'il n'y a pas de limitation

**Paramètres :**

**newval** un entier

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_output()**

YWatchdog

**watchdog→setOutput()[watchdog setOutput: ]**

Modifie l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

js	function <b>set_output</b> ( <b>newval</b> )
nodejs	function <b>set_output</b> ( <b>newval</b> )
php	function <b>set_output</b> ( <b>\$newval</b> )
cpp	int <b>set_output</b> ( Y_OUTPUT_enum <b>newval</b> )
m	-(int) setOutput : (Y_OUTPUT_enum) <b>newval</b>
pas	function <b>set_output</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_output</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_output</b> ( int <b>newval</b> )
java	int <b>set_output</b> ( int <b>newval</b> )
py	def <b>set_output</b> ( <b>newval</b> )
cmd	YWatchdog <b>target set_output newval</b>

**Paramètres :**

**newval** soit Y\_OUTPUT\_OFF, soit Y\_OUTPUT\_ON, selon l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_running()****YWatchdog****watchdog→setRunning()[watchdog setRunning: ]**

Modifie manuellement l'état de fonctionnement du watchdog.

js	function <b>set_running</b> ( <b>newval</b> )
nodejs	function <b>set_running</b> ( <b>newval</b> )
php	function <b>set_running</b> ( <b>\$newval</b> )
cpp	int <b>set_running</b> ( Y_RUNNING_enum <b>newval</b> )
m	-(int) setRunning : (Y_RUNNING_enum) <b>newval</b>
pas	function <b>set_running</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_running</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_running</b> ( int <b>newval</b> )
java	int <b>set_running</b> ( int <b>newval</b> )
py	def <b>set_running</b> ( <b>newval</b> )
cmd	YWatchdog <b>target set_running newval</b>

**Paramètres :**

**newval** soit Y\_RUNNING\_OFF, soit Y\_RUNNING\_ON, selon manuellement l'état de fonctionnement du watchdog

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_state()****watchdog→setState()[watchdog setState: ]**

Modifie l'état du watchdog (A pour la position de repos, B pour l'état actif).

js	function <b>set_state</b> ( <b>newval</b> )
nodejs	function <b>set_state</b> ( <b>newval</b> )
php	function <b>set_state</b> ( <b>\$newval</b> )
cpp	int <b>set_state</b> ( Y_STATE_enum <b>newval</b> )
m	-(int) <b>setState</b> : (Y_STATE_enum) <b>newval</b>
pas	function <b>set_state</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_state</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_state</b> ( int <b>newval</b> )
java	int <b>set_state</b> ( int <b>newval</b> )
py	def <b>set_state</b> ( <b>newval</b> )
cmd	YWatchdog <b>target set_state newval</b>

**Paramètres :**

**newval** soit Y\_STATE\_A, soit Y\_STATE\_B, selon l'état du watchdog (A pour la position de repos, B pour l'état actif)

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_stateAtPowerOn()****YWatchdog****watchdog→setStateAtPowerOn()[watchdog  
setStateAtPowerOn: ]**

Pré-programme l'état du watchdog au démarrage du module(A pour la position de repos, B pour l'état actif, UNCHANGED pour aucun changement).

js	function <b>set_stateAtPowerOn</b> ( <b>newval</b> )
nodejs	function <b>set_stateAtPowerOn</b> ( <b>newval</b> )
php	function <b>set_stateAtPowerOn</b> ( <b>\$newval</b> )
cpp	int <b>set_stateAtPowerOn</b> ( Y_STATEATPOWERON_enum <b>newval</b> )
m	-(int) <b>setStateAtPowerOn</b> : (Y_STATEATPOWERON_enum) <b>newval</b>
pas	function <b>set_stateAtPowerOn</b> ( <b>newval</b> : Integer): integer
vb	function <b>set_stateAtPowerOn</b> ( ByVal <b>newval</b> As Integer) As Integer
cs	int <b>set_stateAtPowerOn</b> ( int <b>newval</b> )
java	int <b>set_stateAtPowerOn</b> ( int <b>newval</b> )
py	def <b>set_stateAtPowerOn</b> ( <b>newval</b> )
cmd	YWatchdog <b>target</b> <b>set_stateAtPowerOn</b> <b>newval</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` du module sinon la modification n'aura aucun effet.

**Paramètres :**

**newval** une valeur parmi Y\_STATEATPOWERON\_UNCHANGED, Y\_STATEATPOWERON\_A et Y\_STATEATPOWERON\_B

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_triggerDelay()****YWatchdog****watchdog→setTriggerDelay()[watchdog  
setTriggerDelay: ]**

Modifie le délai d'attente avant qu'un reset ne soit généré par le watchdog, en millisecondes.

js	function <b>set_triggerDelay</b> ( <b>newval</b> )
nodejs	function <b>set_triggerDelay</b> ( <b>newval</b> )
php	function <b>set_triggerDelay</b> ( <b>\$newval</b> )
c++	int <b>set_triggerDelay</b> ( s64 <b>newval</b> )
m	-(int) setTriggerDelay : (s64) <b>newval</b>
pas	function <b>set_triggerDelay</b> ( <b>newval</b> : int64): integer
vb	function <b>set_triggerDelay</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_triggerDelay</b> ( long <b>newval</b> )
java	int <b>set_triggerDelay</b> ( long <b>newval</b> )
py	def <b>set_triggerDelay</b> ( <b>newval</b> )
cmd	YWatchdog <b>target</b> <b>set_triggerDelay</b> <b>newval</b>

**Paramètres :**

**newval** un entier représentant le délai d'attente avant qu'un reset ne soit généré par le watchdog, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

## watchdog→set\_triggerDuration()

### watchdog→setTriggerDuration()[watchdog setTriggerDuration: ]

YWatchdog

Modifie la durée des resets générés par le watchdog, en millisecondes.

js	function <b>set_triggerDuration</b> ( <b>newval</b> )
nodejs	function <b>set_triggerDuration</b> ( <b>newval</b> )
php	function <b>set_triggerDuration</b> ( <b>\$newval</b> )
cpp	int <b>set_triggerDuration</b> ( s64 <b>newval</b> )
m	-(int) setTriggerDuration : (s64) <b>newval</b>
pas	function <b>set_triggerDuration</b> ( <b>newval</b> : int64): integer
vb	function <b>set_triggerDuration</b> ( ByVal <b>newval</b> As Long) As Integer
cs	int <b>set_triggerDuration</b> ( long <b>newval</b> )
java	int <b>set_triggerDuration</b> ( long <b>newval</b> )
py	def <b>set_triggerDuration</b> ( <b>newval</b> )
cmd	YWatchdog <b>target</b> <b>set_triggerDuration</b> <b>newval</b>

#### Paramètres :

**newval** un entier représentant la durée des resets générés par le watchdog, en millisecondes

#### Retourne :

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**watchdog→set\_userdata()**

YWatchdog

**watchdog→setUserData()[watchdog setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode get\_userdata.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser



**watchdog→wait\_async()****YWatchdog**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

## 3.46. Interface de la fonction Wireless

La fonction YWireless permet de configurer et de contrôler la configuration du réseau sans fil sur les modules Yoctopuce qui en sont dotés.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_wireless.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YWireless = yoctolib.YWireless;
php	require_once('yocto_wireless.php');
c++	#include "yocto_wireless.h"
m	#import "yocto_wireless.h"
pas	uses yocto_wireless;
vb	yocto_wireless.vb
cs	yocto_wireless.cs
java	import com.yoctopuce.YoctoAPI.YWireless;
py	from yocto_wireless import *

### Fonction globales

#### yFindWireless(func)

Permet de retrouver une interface réseau sans fil d'après un identifiant donné.

#### yFirstWireless()

Commence l'énumération des interfaces réseau sans fil accessibles par la librairie.

### Méthodes des objets YWireless

#### wireless→adhocNetwork(ssid, securityKey)

Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".

#### wireless→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau sans fil au format `TYPE ( NAME ) = SERIAL . FUNCTIONID`.

#### wireless→get\_advertisedValue()

Retourne la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

#### wireless→get\_channel()

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

#### wireless→get\_detectedWlans()

Retourne une liste d'objets objet YFileRecord qui décrivent les réseaux sans fils détectés.

#### wireless→get\_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

#### wireless→get\_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

#### wireless→get\_friendlyName()

Retourne un identifiant global de l'interface réseau sans fil au format `NOM_MODULE . NOM_FONCTION`.

#### wireless→get\_functionDescriptor()

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

#### wireless→get\_functionId()

Retourne l'identifiant matériel de l'interface réseau sans fil, sans référence au module.

#### wireless→get\_hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau sans fil au format `SERIAL.FUNCTIONID`.

**wireless→get\_linkQuality()**

Retourne la qualité de la connection, exprimée en pourcents.

**wireless→get\_logicalName()**

Retourne le nom logique de l'interface réseau sans fil.

**wireless→get\_message()**

Retourne le dernier message de diagnostic de l'interface au réseau sans fil.

**wireless→get\_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**wireless→get\_module\_async(callback, context)**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

**wireless→get\_security()**

Retourne l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné.

**wireless→get\_ssid()**

Retourne le nom (SSID) du réseau sans-fil sélectionné.

**wireless→get\_userData()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userData`.

**wireless→isOnline()**

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

**wireless→isOnline\_async(callback, context)**

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

**wireless→joinNetwork(ssid, securityKey)**

Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").

**wireless→load(msValidity)**

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

**wireless→load\_async(msValidity, callback, context)**

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

**wireless→nextWireless()**

Continue l'énumération des interfaces réseau sans fil commencée à l'aide de `yFirstWireless()`.

**wireless→registerValueCallback(callback)**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

**wireless→set\_logicalName(newval)**

Modifie le nom logique de l'interface réseau sans fil.

**wireless→set\_userData(data)**

Enregistre un contexte libre dans l'attribut `userData` de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

**wireless→wait\_async(callback, context)**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

## YWireless.FindWireless() yFindWireless()yFindWireless()

YWireless

Permet de retrouver une interface réseau sans fil d'après un identifiant donné.

js	function <b>yFindWireless</b> ( <b>func</b> )
nodejs	function <b>FindWireless</b> ( <b>func</b> )
php	function <b>yFindWireless</b> ( <b>\$func</b> )
c++	YWireless* <b>yFindWireless</b> ( string <b>func</b> )
m	+(YWireless*) <b>yFindWireless</b> : (NSString*) <b>func</b>
pas	function <b>yFindWireless</b> ( <b>func</b> : string): TYWireless
vb	function <b>yFindWireless</b> ( ByVal <b>func</b> As String) As YWireless
cs	YWireless <b>FindWireless</b> ( string <b>func</b> )
java	YWireless <b>FindWireless</b> ( String <b>func</b> )
py	def <b>FindWireless</b> ( <b>func</b> )

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau sans fil soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWireless.isOnline()` pour tester si l'interface réseau sans fil est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

### Paramètres :

**func** une chaîne de caractères qui référence l'interface réseau sans fil sans ambiguïté

### Retourne :

un objet de classe `YWireless` qui permet ensuite de contrôler l'interface réseau sans fil.

## YWireless.FirstWireless() yFirstWireless()yFirstWireless()

## YWireless

Commence l'énumération des interfaces réseau sans fil accessibles par la librairie.

js	function <b>yFirstWireless</b> ( )
nodejs	function <b>FirstWireless</b> ( )
php	function <b>yFirstWireless</b> ( )
cpp	YWireless* <b>yFirstWireless</b> ( )
m	YWireless* <b>yFirstWireless</b> ( )
pas	function <b>yFirstWireless</b> ( ): TYWireless
vb	function <b>yFirstWireless</b> ( ) As YWireless
cs	YWireless <b>FirstWireless</b> ( )
java	YWireless <b>FirstWireless</b> ( )
py	def <b>FirstWireless</b> ( )

Utiliser la fonction `YWireless.nextWireless()` pour itérer sur les autres interfaces réseau sans fil.

### Retourne :

un pointeur sur un objet `YWireless`, correspondant à la première interface réseau sans fil accessible en ligne, ou `null` si il n'y a pas de interfaces réseau sans fil disponibles.

**wireless→adhocNetwork() [wireless adhocNetwork: ]****YWireless**

Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".

js	function <b>adhocNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
nodejs	function <b>adhocNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
php	function <b>adhocNetwork</b> ( <b>\$ssid</b> , <b>\$securityKey</b> )
c++	int <b>adhocNetwork</b> ( string <b>ssid</b> , string <b>securityKey</b> )
m	-(int) <b>adhocNetwork</b> : (NSString*) <b>ssid</b> : (NSString*) <b>securityKey</b>
pas	function <b>adhocNetwork</b> ( <b>ssid</b> : string, <b>securityKey</b> : string): integer
vb	function <b>adhocNetwork</b> ( ByVal <b>ssid</b> As String, ByVal <b>securityKey</b> As String) As Integer
cs	int <b>adhocNetwork</b> ( string <b>ssid</b> , string <b>securityKey</b> )
java	int <b>adhocNetwork</b> ( String <b>ssid</b> , String <b>securityKey</b> )
py	def <b>adhocNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
cmd	YWireless <b>target</b> <b>adhocNetwork</b> <b>ssid</b> <b>securityKey</b>

Si une clef d'accès est spécifiée, le réseau sera protégé par une sécurité WEP128 (l'utilisation de WPA n'est pas standardisée en mode ad-hoc). N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**ssid** nom du réseau sans fil à créer  
**securityKey** clé d'accès de réseau, sous forme de chaîne de caractères

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wireless→describe()[wireless describe]****YWireless**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau sans fil au format `TYPE (NAME) = SERIAL . FUNCTIONID`.

js	function <b>describe</b> ( )
nodejs	function <b>describe</b> ( )
php	function <b>describe</b> ( )
cpp	string <b>describe</b> ( )
m	-(NSString*) <b>describe</b>
pas	function <b>describe</b> ( ): string
vb	function <b>describe</b> ( ) As String
cs	string <b>describe</b> ( )
java	String <b>describe</b> ( )
py	def <b>describe</b> ( )

Plus précisément, `TYPE` correspond au type de fonction, `NAME` correspond au nom utilisé lors du premier accès à la fonction, `SERIAL` correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et `FUNCTIONID` correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La méthode va retourner `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

**Retourne :**

une chaîne de caractères décrivant l'interface réseau sans fil (ex:  
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

## wireless→get\_advertisedValue() wireless→advertisedValue()[wireless advertisedValue]

YWireless

Retourne la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

js	function <b>get_advertisedValue</b> ( )
nodejs	function <b>get_advertisedValue</b> ( )
php	function <b>get_advertisedValue</b> ( )
cpp	string <b>get_advertisedValue</b> ( )
m	-(NSString*) advertisedValue
pas	function <b>get_advertisedValue</b> ( ): string
vb	function <b>get_advertisedValue</b> ( ) As String
cs	string <b>get_advertisedValue</b> ( )
java	String <b>get_advertisedValue</b> ( )
py	def <b>get_advertisedValue</b> ( )
cmd	YWireless <b>target</b> <b>get_advertisedValue</b>

### Retourne :

une chaîne de caractères représentant la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères). En cas d'erreur, déclenche une exception ou retourne Y\_ADVERTISEDVALUE\_INVALID.



**wireless**→**get\_channel()****YWireless****wireless**→**channel()**[wireless channel]

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

js	function <b>get_channel</b> ( )
nodejs	function <b>get_channel</b> ( )
php	function <b>get_channel</b> ( )
cpp	int <b>get_channel</b> ( )
m	-(int) channel
pas	function <b>get_channel</b> ( ): LongInt
vb	function <b>get_channel</b> ( ) As Integer
cs	int <b>get_channel</b> ( )
java	int <b>get_channel</b> ( )
py	def <b>get_channel</b> ( )
cmd	YWireless <b>target</b> <b>get_channel</b>

**Retourne :**

un entier représentant le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé

En cas d'erreur, déclenche une exception ou retourne Y\_CHANNEL\_INVALID.

**wireless**→**get\_detectedWlans()****YWireless****wireless**→**detectedWlans()[wireless detectedWlans]**

Retourne une liste d'objets objet YFileRecord qui décrivent les réseaux sans fils détectés.

js	function <b>get_detectedWlans</b> ( )
nodejs	function <b>get_detectedWlans</b> ( )
php	function <b>get_detectedWlans</b> ( )
cpp	vector<YWlanRecord> <b>get_detectedWlans</b> ( )
m	-(NSMutableArray*) detectedWlans
pas	function <b>get_detectedWlans</b> ( ): TYWlanRecordArray
vb	function <b>get_detectedWlans</b> ( ) As List
cs	List<YWlanRecord> <b>get_detectedWlans</b> ( )
java	ArrayList<YWlanRecord> <b>get_detectedWlans</b> ( )
py	def <b>get_detectedWlans</b> ( )
cmd	YWireless <b>target</b> <b>get_detectedWlans</b>

La liste n'est pas mise à jour quand le module est déjà connecté à un accès sans fil (mode "infrastructure"). Pour forcer la détection des réseaux sans fil, il faut appeler `addhocNetwork( )` pour se déconnecter du réseau actuel. L'appelant est responsable de la désallocation de la liste retournée.

**Retourne :**

une liste d'objets YWlanRecord, contenant le SSID, le canal, la qualité du signal, et l'algorithme de sécurité utilisé par le réseau sans-fil

En cas d'erreur, déclenche une exception ou retourne une liste vide.

**wireless**→**get\_errorMessage()****YWireless****wireless**→**errorMessage()**[**wireless errorMessage**]

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

js	function <b>get_errorMessage</b> ( )
nodejs	function <b>get_errorMessage</b> ( )
php	function <b>get_errorMessage</b> ( )
cpp	string <b>get_errorMessage</b> ( )
m	-(NSString*) errorMessage
pas	function <b>get_errorMessage</b> ( ): string
vb	function <b>get_errorMessage</b> ( ) As String
cs	string <b>get_errorMessage</b> ( )
java	String <b>get_errorMessage</b> ( )
py	def <b>get_errorMessage</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau sans fil.

**wireless**→**get\_errorType()****YWireless****wireless**→**errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

js	function <b>get_errorType</b> ( )
nodejs	function <b>get_errorType</b> ( )
php	function <b>get_errorType</b> ( )
cpp	YRETCODE <b>get_errorType</b> ( )
pas	function <b>get_errorType</b> ( ): YRETCODE
vb	function <b>get_errorType</b> ( ) As YRETCODE
cs	YRETCODE <b>get_errorType</b> ( )
java	int <b>get_errorType</b> ( )
py	def <b>get_errorType</b> ( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

**Retourne :**

un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau sans fil.

**wireless**→**get\_friendlyName()****YWireless****wireless**→**friendlyName()**[wireless friendlyName]

Retourne un identifiant global de l'interface réseau sans fil au format `NOM_MODULE.NOM_FONCTION`.

js	function <b>get_friendlyName</b> ( )
nodejs	function <b>get_friendlyName</b> ( )
php	function <b>get_friendlyName</b> ( )
cpp	string <b>get_friendlyName</b> ( )
m	-(NSString*) friendlyName
cs	string <b>get_friendlyName</b> ( )
java	String <b>get_friendlyName</b> ( )
py	def <b>get_friendlyName</b> ( )

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau sans fil si ils sont définis, soit respectivement le numéro de série du module et l'identifiant matériel de l'interface réseau sans fil (par exemple: `MyCustomName.relay1`)

**Retourne :**

une chaîne de caractères identifiant l'interface réseau sans fil en utilisant les noms logiques (ex: `MyCustomName.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FRIENDLYNAME_INVALID`.

## wireless→get\_functionDescriptor() wireless→functionDescriptor()[wireless functionDescriptor]

Retourne un identifiant unique de type YFUN\_DESCR correspondant à la fonction.

js	function <b>get_functionDescriptor</b> ( )
nodejs	function <b>get_functionDescriptor</b> ( )
php	function <b>get_functionDescriptor</b> ( )
cpp	YFUN_DESCR <b>get_functionDescriptor</b> ( )
m	-(YFUN_DESCR) functionDescriptor
pas	function <b>get_functionDescriptor</b> ( ): YFUN_DESCR
vb	function <b>get_functionDescriptor</b> ( ) As YFUN_DESCR
cs	YFUN_DESCR <b>get_functionDescriptor</b> ( )
java	String <b>get_functionDescriptor</b> ( )
py	def <b>get_functionDescriptor</b> ( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

### Retourne :

un identifiant de type YFUN\_DESCR. Si la fonction n'a jamais été contactée, la valeur retournée sera Y\_FUNCTIONDESCRIPTOR\_INVALID

**wireless**→**get\_functionId()****YWireless****wireless**→**functionId()**[wireless functionId]

Retourne l'identifiant matériel de l'interface réseau sans fil, sans référence au module.

js	function <b>get_functionId</b> ( )
nodejs	function <b>get_functionId</b> ( )
php	function <b>get_functionId</b> ( )
cpp	string <b>get_functionId</b> ( )
m	-(NSString*) functionId
vb	function <b>get_functionId</b> ( ) As String
cs	string <b>get_functionId</b> ( )
java	String <b>get_functionId</b> ( )
py	def <b>get_functionId</b> ( )

Par exemple `relay1`.

**Retourne :**

une chaîne de caractères identifiant l'interface réseau sans fil (ex: `relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

**wireless**→**get\_hardwareId()****YWireless****wireless**→**hardwareId()**[**wireless hardwareId**]

Retourne l'identifiant matériel unique de l'interface réseau sans fil au format `SERIAL.FUNCTIONID`.

js	function <b>get_hardwareId</b> ( )
nodejs	function <b>get_hardwareId</b> ( )
php	function <b>get_hardwareId</b> ( )
cpp	string <b>get_hardwareId</b> ( )
m	-(NSString*) hardwareId
vb	function <b>get_hardwareId</b> ( ) As String
cs	string <b>get_hardwareId</b> ( )
java	String <b>get_hardwareId</b> ( )
py	def <b>get_hardwareId</b> ( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau sans fil (par exemple `RELAYLO1-123456.relay1`).

**Retourne :**

une chaîne de caractères identifiant l'interface réseau sans fil (ex: `RELAYLO1-123456.relay1`) En cas d'erreur, déclenche une exception ou retourne `Y_HARDWAREID_INVALID`.



**wireless→get\_linkQuality()****YWireless****wireless→linkQuality()[wireless linkQuality]**

Retourne la qualité de la connection, exprimée en pourcents.

js	function <b>get_linkQuality</b> ( )
nodejs	function <b>get_linkQuality</b> ( )
php	function <b>get_linkQuality</b> ( )
cpp	int <b>get_linkQuality</b> ( )
m	-(int) linkQuality
pas	function <b>get_linkQuality</b> ( ): LongInt
vb	function <b>get_linkQuality</b> ( ) As Integer
cs	int <b>get_linkQuality</b> ( )
java	int <b>get_linkQuality</b> ( )
py	def <b>get_linkQuality</b> ( )
cmd	YWireless <b>target</b> <b>get_linkQuality</b>

**Retourne :**

un entier représentant la qualité de la connection, exprimée en pourcents

En cas d'erreur, déclenche une exception ou retourne Y\_LINKQUALITY\_INVALID.

**wireless**→**get\_logicalName()****YWireless****wireless**→**logicalName()**[wireless logicalName]

Retourne le nom logique de l'interface réseau sans fil.

js	function <b>get_logicalName</b> ( )
nodejs	function <b>get_logicalName</b> ( )
php	function <b>get_logicalName</b> ( )
cpp	string <b>get_logicalName</b> ( )
m	-(NSString*) logicalName
pas	function <b>get_logicalName</b> ( ): string
vb	function <b>get_logicalName</b> ( ) As String
cs	string <b>get_logicalName</b> ( )
java	String <b>get_logicalName</b> ( )
py	def <b>get_logicalName</b> ( )
cmd	YWireless <b>target</b> <b>get_logicalName</b>

**Retourne :**

une chaîne de caractères représentant le nom logique de l'interface réseau sans fil. En cas d'erreur, déclenche une exception ou retourne Y\_LOGICALNAME\_INVALID.

**wireless→get\_message()****YWireless****wireless→message()[wireless message]**

Retourne le dernier message de diagnostic de l'interface au réseau sans fil.

js	function <b>get_message</b> ( )
nodejs	function <b>get_message</b> ( )
php	function <b>get_message</b> ( )
cpp	string <b>get_message</b> ( )
m	-(NSString*) message
pas	function <b>get_message</b> ( ): string
vb	function <b>get_message</b> ( ) As String
cs	string <b>get_message</b> ( )
java	String <b>get_message</b> ( )
py	def <b>get_message</b> ( )
cmd	YWireless <b>target</b> <b>get_message</b>

**Retourne :**

une chaîne de caractères représentant le dernier message de diagnostic de l'interface au réseau sans fil

En cas d'erreur, déclenche une exception ou retourne Y\_MESSAGE\_INVALID.

**wireless**→**get\_module()****YWireless****wireless**→**module()**[wireless module]

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module</b> ( )
nodejs	function <b>get_module</b> ( )
php	function <b>get_module</b> ( )
c++	YModule * <b>get_module</b> ( )
m	-(YModule*) module
pas	function <b>get_module</b> ( ): TYModule
vb	function <b>get_module</b> ( ) As YModule
cs	YModule <b>get_module</b> ( )
java	YModule <b>get_module</b> ( )
py	def <b>get_module</b> ( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

**Retourne :**

une instance de YModule

**wireless**→**get\_module\_async()****YWireless****wireless**→**module\_async()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )
nodejs	function <b>get_module_async</b> ( <b>callback</b> , <b>context</b> )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de `YModule`

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**wireless**→**get\_security()****wireless**→**security()**[wireless security]

Retourne l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné.

js	function <b>get_security</b> ( )
nodejs	function <b>get_security</b> ( )
php	function <b>get_security</b> ( )
cpp	Y_SECURITY_enum <b>get_security</b> ( )
m	-(Y_SECURITY_enum) security
pas	function <b>get_security</b> ( ): Integer
vb	function <b>get_security</b> ( ) As Integer
cs	int <b>get_security</b> ( )
java	int <b>get_security</b> ( )
py	def <b>get_security</b> ( )
cmd	YWireless <b>target</b> <b>get_security</b>

**Retourne :**

une valeur parmi Y\_SECURITY\_UNKNOWN, Y\_SECURITY\_OPEN, Y\_SECURITY\_WEP, Y\_SECURITY\_WPA et Y\_SECURITY\_WPA2 représentant l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné

En cas d'erreur, déclenche une exception ou retourne Y\_SECURITY\_INVALID.

**wireless**→**get\_ssid()****YWireless****wireless**→**ssid()**[**wireless ssid**]

Retourne le nom (SSID) du réseau sans-fil sélectionné.

js	function <b>get_ssid</b> ( )
nodejs	function <b>get_ssid</b> ( )
php	function <b>get_ssid</b> ( )
cpp	string <b>get_ssid</b> ( )
m	-(NSString*) ssid
pas	function <b>get_ssid</b> ( ): string
vb	function <b>get_ssid</b> ( ) As String
cs	string <b>get_ssid</b> ( )
java	String <b>get_ssid</b> ( )
py	def <b>get_ssid</b> ( )
cmd	YWireless <b>target</b> <b>get_ssid</b>

**Retourne :**

une chaîne de caractères représentant le nom (SSID) du réseau sans-fil sélectionné

En cas d'erreur, déclenche une exception ou retourne Y\_SSID\_INVALID.

**wireless→get\_userdata()****YWireless****wireless→userdata()[wireless userData]**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set\_userdata.

js	function <b>get_userdata</b> ( )
nodejs	function <b>get_userdata</b> ( )
php	function <b>get_userdata</b> ( )
cpp	void * <b>get_userdata</b> ( )
m	-(void*) userData
pas	function <b>get_userdata</b> ( ): Tobject
vb	function <b>get_userdata</b> ( ) As Object
cs	object <b>get_userdata</b> ( )
java	Object <b>get_userdata</b> ( )
py	def <b>get_userdata</b> ( )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Retourne :**

l'objet stocké précédemment par l'appelant.



**wireless**→**isOnline()**[wireless isOnline]**YWireless**

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

js	function <b>isOnline</b> ( )
nodejs	function <b>isOnline</b> ( )
php	function <b>isOnline</b> ( )
cpp	bool <b>isOnline</b> ( )
m	-(BOOL) <b>isOnline</b>
pas	function <b>isOnline</b> ( ): boolean
vb	function <b>isOnline</b> ( ) As Boolean
cs	bool <b>isOnline</b> ( )
java	boolean <b>isOnline</b> ( )
py	def <b>isOnline</b> ( )

Si les valeurs des attributs en cache de l'interface réseau sans fil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

**Retourne :**

`true` si l'interface réseau sans fil est joignable, `false` sinon

Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.

```
js function isOnline_async( callback, context)
nodejs function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sans fil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

**wireless→joinNetwork()[wireless joinNetwork: ]****YWireless**

Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").

js	function <b>joinNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
nodejs	function <b>joinNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
php	function <b>joinNetwork</b> ( <b>\$ssid</b> , <b>\$securityKey</b> )
c++	int <b>joinNetwork</b> ( string <b>ssid</b> , string <b>securityKey</b> )
m	-(int) <b>joinNetwork</b> : (NSString*) <b>ssid</b> : (NSString*) <b>securityKey</b>
pas	function <b>joinNetwork</b> ( <b>ssid</b> : string, <b>securityKey</b> : string): integer
vb	function <b>joinNetwork</b> ( ByVal <b>ssid</b> As String, ByVal <b>securityKey</b> As String) As Integer
cs	int <b>joinNetwork</b> ( string <b>ssid</b> , string <b>securityKey</b> )
java	int <b>joinNetwork</b> ( String <b>ssid</b> , String <b>securityKey</b> )
py	def <b>joinNetwork</b> ( <b>ssid</b> , <b>securityKey</b> )
cmd	YWireless <b>target joinNetwork ssid securityKey</b>

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

**Paramètres :**

**ssid** nom du réseau sans fil à utiliser  
**securityKey** clé d'accès au réseau, sous forme de chaîne de caractères

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wireless→load()[wireless load: ]****YWireless**

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

js	function <b>load</b> ( <b>msValidity</b> )
nodejs	function <b>load</b> ( <b>msValidity</b> )
php	function <b>load</b> ( <b>\$msValidity</b> )
cpp	YRETCODE <b>load</b> ( int <b>msValidity</b> )
m	-(YRETCODE) <b>load</b> : (int) <b>msValidity</b>
pas	function <b>load</b> ( <b>msValidity</b> : integer): YRETCODE
vb	function <b>load</b> ( ByVal <b>msValidity</b> As Integer) As YRETCODE
cs	YRETCODE <b>load</b> ( int <b>msValidity</b> )
java	int <b>load</b> ( long <b>msValidity</b> )
py	def <b>load</b> ( <b>msValidity</b> )

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

**Paramètres :**

**msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

**Retourne :**

YAPI\_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

**wireless→load\_async()****YWireless**

Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.

```
js function load_async( msValidity, callback, context)
```

```
nodejs function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionnellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

**Paramètres :**

- msValidity** un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
- callback** fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI\_SUCCESS)
- context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

## wireless→nextWireless()[wireless nextWireless]

## YWireless

Continue l'énumération des interfaces réseau sans fil commencée à l'aide de `yFirstWireless()`.

js	function <b>nextWireless</b> ( )
nodejs	function <b>nextWireless</b> ( )
php	function <b>nextWireless</b> ( )
cpp	YWireless * <b>nextWireless</b> ( )
m	-(YWireless*) <b>nextWireless</b>
pas	function <b>nextWireless</b> ( ): TYWireless
vb	function <b>nextWireless</b> ( ) As YWireless
cs	YWireless <b>nextWireless</b> ( )
java	YWireless <b>nextWireless</b> ( )
py	def <b>nextWireless</b> ( )

### Retourne :

un pointeur sur un objet `YWireless` accessible en ligne, ou `null` lorsque l'énumération est terminée.

## wireless→registerValueCallback()[wireless registerValueCallback: ]

YWireless

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function <b>registerValueCallback</b> ( <b>callback</b> )
nodejs	function <b>registerValueCallback</b> ( <b>callback</b> )
php	function <b>registerValueCallback</b> ( <b>\$callback</b> )
cpp	int <b>registerValueCallback</b> ( YWirelessValueCallback <b>callback</b> )
m	-(int) <b>registerValueCallback</b> : (YWirelessValueCallback) <b>callback</b>
pas	function <b>registerValueCallback</b> ( <b>callback</b> : TYWirelessValueCallback): LongInt
vb	function <b>registerValueCallback</b> ( ) As Integer
cs	int <b>registerValueCallback</b> ( ValueCallback <b>callback</b> )
java	int <b>registerValueCallback</b> ( UpdateCallback <b>callback</b> )
py	def <b>registerValueCallback</b> ( <b>callback</b> )

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

### Paramètres :

**callback** la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

**wireless→set\_logicalName()****YWireless****wireless→setLogicalName()[wireless  
setLogicalName: ]**

Modifie le nom logique de l'interface réseau sans fil.

js	function <b>set_logicalName</b> ( <b>newval</b> )
nodejs	function <b>set_logicalName</b> ( <b>newval</b> )
php	function <b>set_logicalName</b> ( <b>\$newval</b> )
cpp	int <b>set_logicalName</b> ( const string& <b>newval</b> )
m	-(int) setLogicalName : (NSString*) <b>newval</b>
pas	function <b>set_logicalName</b> ( <b>newval</b> : string): integer
vb	function <b>set_logicalName</b> ( ByVal <b>newval</b> As String) As Integer
cs	int <b>set_logicalName</b> ( string <b>newval</b> )
java	int <b>set_logicalName</b> ( String <b>newval</b> )
py	def <b>set_logicalName</b> ( <b>newval</b> )
cmd	YWireless <b>target</b> <b>set_logicalName</b> <b>newval</b>

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

**Paramètres :**

**newval** une chaîne de caractères représentant le nom logique de l'interface réseau sans fil.

**Retourne :**

YAPI\_SUCCESS si l'appel se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.



**wireless→set\_userdata()****YWireless****wireless→setUserData()[wireless setUserData: ]**

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userdata`.

js	function <b>set_userdata</b> ( <b>data</b> )
nodejs	function <b>set_userdata</b> ( <b>data</b> )
php	function <b>set_userdata</b> ( <b>\$data</b> )
cpp	void <b>set_userdata</b> ( void* <b>data</b> )
m	-(void) setUserData : (void*) <b>data</b>
pas	procedure <b>set_userdata</b> ( <b>data</b> : Tobject)
vb	procedure <b>set_userdata</b> ( ByVal <b>data</b> As Object)
cs	void <b>set_userdata</b> ( object <b>data</b> )
java	void <b>set_userdata</b> ( Object <b>data</b> )
py	def <b>set_userdata</b> ( <b>data</b> )

Cet attribut n'est pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

**Paramètres :**

**data** objet quelconque à mémoriser

**wireless→wait\_async()****YWireless**

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js function wait_async( callback, context)
```

```
nodejs function wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

**Paramètres :**

**callback** fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

**context** contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

**Retourne :**

rien du tout :

# Index

## A

Accelerometer 37  
adhocNetwork, YWireless 1729  
Alimentation 492  
AnButton 83

## B

Blueprint 10  
Brute 346

## C

calibrate, YLightSensor 766  
calibrateFromPoints, YAccelerometer 41  
calibrateFromPoints, YCarbonDioxide 129  
calibrateFromPoints, YCompass 205  
calibrateFromPoints, YCurrent 249  
calibrateFromPoints, YGenericSensor 558  
calibrateFromPoints, YGyro 608  
calibrateFromPoints, YHumidity 692  
calibrateFromPoints, YLightSensor 767  
calibrateFromPoints, YMagnetometer 810  
calibrateFromPoints, YPower 996  
calibrateFromPoints, YPressure 1043  
calibrateFromPoints, YQt 1155  
calibrateFromPoints, YSensor 1309  
calibrateFromPoints, YTemperature 1391  
calibrateFromPoints, YTilt 1436  
calibrateFromPoints, YVoc 1479  
calibrateFromPoints, YVoltage 1522  
callbackLogin, YNetwork 909  
cancel3DCalibration, YRefFrame 1229  
CarbonDioxide 125  
CheckLogicalName, YAPI 12  
clear, YDisplayLayer 461  
clearConsole, YDisplayLayer 462  
ColorLed 168  
Compass 201  
Configuration 1225  
consoleOut, YDisplayLayer 463  
Contrôle 3, 5, 492, 858, 965  
copyLayerContent, YDisplay 413  
Current 245

## D

DataLogger 288  
delayedPulse, YDigitalIO 365  
delayedPulse, YRelay 1269  
delayedPulse, YWatchdog 1681  
describe, YAccelerometer 42  
describe, YAnButton 87  
describe, YCarbonDioxide 130  
describe, YColorLed 171

describe, YCompass 206  
describe, YCurrent 250  
describe, YDataLogger 291  
describe, YDigitalIO 366  
describe, YDisplay 414  
describe, YDualPower 495  
describe, YFiles 524  
describe, YGenericSensor 559  
describe, YGyro 609  
describe, YHubPort 662  
describe, YHumidity 693  
describe, YLed 734  
describe, YLightSensor 768  
describe, YMagnetometer 811  
describe, YModule 862  
describe, YNetwork 910  
describe, YOsControl 968  
describe, YPower 997  
describe, YPressure 1044  
describe, YPwmOutput 1086  
describe, YPwmPowerSource 1127  
describe, YQt 1156  
describe, YRealTimeClock 1197  
describe, YRefFrame 1230  
describe, YRelay 1270  
describe, YSensor 1310  
describe, YServo 1352  
describe, YTemperature 1392  
describe, YTilt 1437  
describe, YVoc 1480  
describe, YVoltage 1523  
describe, YVSource 1564  
describe, YWakeUpMonitor 1601  
describe, YWakeUpSchedule 1640  
describe, YWatchdog 1682  
describe, YWireless 1730  
DigitalIO 361  
DisableExceptions, YAPI 13  
Display 409  
DisplayLayer 460  
Données 323, 333, 346  
download, YFiles 525  
download, YModule 863  
download\_async, YFiles 526  
drawBar, YDisplayLayer 464  
drawBitmap, YDisplayLayer 465  
drawCircle, YDisplayLayer 466  
drawDisc, YDisplayLayer 467  
drawImage, YDisplayLayer 468  
drawPixel, YDisplayLayer 469  
drawRect, YDisplayLayer 470  
drawText, YDisplayLayer 471  
dutyCycleMove, YPwmOutput 1087

## E

EnableExceptions, YAPI 14  
EnableUSBHost, YAPI 15  
Enregistrées 333, 346  
Erreurs 7

## F

fade, YDisplay 415  
Files 521  
FindAccelerometer, YAccelerometer 39  
FindAnButton, YAnButton 85  
FindCarbonDioxide, YCarbonDioxide 127  
FindColorLed, YColorLed 169  
FindCompass, YCompass 203  
FindCurrent, YCurrent 247  
FindDataLogger, YDataLogger 289  
FindDigitalIO, YDigitalIO 363  
FindDisplay, YDisplay 411  
FindDualPower, YDualPower 493  
FindFiles, YFiles 522  
FindGenericSensor, YGenericSensor 556  
FindGyro, YGyro 606  
FindHubPort, YHubPort 660  
FindHumidity, YHumidity 690  
FindLed, YLed 732  
FindLightSensor, YLightSensor 764  
FindMagnetometer, YMagnetometer 809  
FindModule, YModule 860  
FindNetwork, YNetwork 908  
FindOsControl, YOsControl 966  
FindPower, YPower 994  
FindPressure, YPressure 1041  
FindPwmOutput, YPwmOutput 1084  
FindPwmPowerSource, YPwmPowerSource 1125  
FindQt, YQt 1153  
FindRealTimeClock, YRealTimeClock 1195  
FindRefFrame, YRefFrame 1227  
FindRelay, YRelay 1267  
FindSensor, YSensor 1307  
FindServo, YServo 1350  
FindTemperature, YTemperature 1389  
FindTilt, YTilt 1434  
FindVoc, YVoc 1477  
FindVoltage, YVoltage 1520  
FindVSource, YVSource 1562  
FindWakeUpMonitor, YWakeUpMonitor 1599  
FindWakeUpSchedule, YWakeUpSchedule 1638  
FindWatchdog, YWatchdog 1679  
FindWireless, YWireless 1727  
FirstAccelerometer, YAccelerometer 40  
FirstAnButton, YAnButton 86  
FirstCarbonDioxide, YCarbonDioxide 128  
FirstColorLed, YColorLed 170  
FirstCompass, YCompass 204  
FirstCurrent, YCurrent 248  
FirstDataLogger, YDataLogger 290  
FirstDigitalIO, YDigitalIO 364

FirstDisplay, YDisplay 412  
FirstDualPower, YDualPower 494  
FirstFiles, YFiles 523  
FirstGenericSensor, YGenericSensor 557  
FirstGyro, YGyro 607  
FirstHubPort, YHubPort 661  
FirstHumidity, YHumidity 691  
FirstLed, YLed 733  
FirstLightSensor, YLightSensor 765  
FirstMagnetometer, YMagnetometer 809  
FirstModule, YModule 861  
FirstNetwork, YNetwork 908  
FirstOsControl, YOsControl 967  
FirstPower, YPower 995  
FirstPressure, YPressure 1042  
FirstPwmOutput, YPwmOutput 1085  
FirstPwmPowerSource, YPwmPowerSource 1126  
FirstQt, YQt 1154  
FirstRealTimeClock, YRealTimeClock 1196  
FirstRefFrame, YRefFrame 1228  
FirstRelay, YRelay 1268  
FirstSensor, YSensor 1308  
FirstServo, YServo 1351  
FirstTemperature, YTemperature 1390  
FirstTilt, YTilt 1435  
FirstVoc, YVoc 1478  
FirstVoltage, YVoltage 1521  
FirstVSource, YVSource 1563  
FirstWakeUpMonitor, YWakeUpMonitor 1600  
FirstWakeUpSchedule, YWakeUpSchedule 1639  
FirstWatchdog, YWatchdog 1680  
FirstWireless, YWireless 1728  
Fonctions 11, 1305  
forgetAllDataStreams, YDataLogger 292  
format\_fs, YFiles 527  
Forme 323  
FreeAPI, YAPI 16  
functionCount, YModule 864  
functionId, YModule 865  
functionName, YModule 866  
functionValue, YModule 867

## G

GenericSensor 554  
get\_3DCalibrationHint, YRefFrame 1231  
get\_3DCalibrationLogMsg, YRefFrame 1232  
get\_3DCalibrationProgress, YRefFrame 1233  
get\_3DCalibrationStage, YRefFrame 1234  
get\_3DCalibrationStageProgress, YRefFrame 1235  
get\_adminPassword, YNetwork 911  
get\_advertisedValue, YAccelerometer 43  
get\_advertisedValue, YAnButton 88  
get\_advertisedValue, YCarbonDioxide 131  
get\_advertisedValue, YColorLed 172  
get\_advertisedValue, YCompass 207  
get\_advertisedValue, YCurrent 251  
get\_advertisedValue, YDataLogger 293

get\_advertisedValue, YDigitalIO 367  
 get\_advertisedValue, YDisplay 416  
 get\_advertisedValue, YDualPower 496  
 get\_advertisedValue, YFiles 528  
 get\_advertisedValue, YGenericSensor 560  
 get\_advertisedValue, YGyro 610  
 get\_advertisedValue, YHubPort 663  
 get\_advertisedValue, YHumidity 694  
 get\_advertisedValue, YLed 735  
 get\_advertisedValue, YLightSensor 769  
 get\_advertisedValue, YMagnetometer 812  
 get\_advertisedValue, YNetwork 912  
 get\_advertisedValue, YOsControl 969  
 get\_advertisedValue, YPower 998  
 get\_advertisedValue, YPressure 1045  
 get\_advertisedValue, YPwmOutput 1088  
 get\_advertisedValue, YPwmPowerSource 1128  
 get\_advertisedValue, YQt 1157  
 get\_advertisedValue, YRealTimeClock 1198  
 get\_advertisedValue, YRefFrame 1236  
 get\_advertisedValue, YRelay 1271  
 get\_advertisedValue, YSensor 1311  
 get\_advertisedValue, YServo 1353  
 get\_advertisedValue, YTemperature 1393  
 get\_advertisedValue, YTilt 1438  
 get\_advertisedValue, YVoc 1481  
 get\_advertisedValue, YVoltage 1524  
 get\_advertisedValue, YVSource 1565  
 get\_advertisedValue, YWakeUpMonitor 1602  
 get\_advertisedValue, YWakeUpSchedule 1641  
 get\_advertisedValue, YWatchdog 1683  
 get\_advertisedValue, YWireless 1731  
 get\_analogCalibration, YAnButton 89  
 get\_autoStart, YDataLogger 294  
 get\_autoStart, YWatchdog 1684  
 get\_averageValue, YDataRun 323  
 get\_averageValue, YDataStream 347  
 get\_averageValue, YMeasure 852  
 get\_baudRate, YHubPort 664  
 get\_beacon, YModule 868  
 get\_bearing, YRefFrame 1237  
 get\_bitDirection, YDigitalIO 368  
 get\_bitOpenDrain, YDigitalIO 369  
 get\_bitPolarity, YDigitalIO 370  
 get\_bitState, YDigitalIO 371  
 get\_blinking, YLed 736  
 get\_brightness, YDisplay 417  
 get\_calibratedValue, YAnButton 90  
 get\_calibrationMax, YAnButton 91  
 get\_calibrationMin, YAnButton 92  
 get\_callbackCredentials, YNetwork 913  
 get\_callbackEncoding, YNetwork 914  
 get\_callbackMaxDelay, YNetwork 915  
 get\_callbackMethod, YNetwork 916  
 get\_callbackMinDelay, YNetwork 917  
 get\_callbackUrl, YNetwork 918  
 get\_channel, YWireless 1732  
 get\_columnCount, YDataStream 348  
 get\_columnNames, YDataStream 349  
 get\_cosPhi, YPower 999  
 get\_countdown, YRelay 1272  
 get\_countdown, YWatchdog 1685  
 get\_currentRawValue, YAccelerometer 44  
 get\_currentRawValue, YCarbonDioxide 132  
 get\_currentRawValue, YCompass 208  
 get\_currentRawValue, YCurrent 252  
 get\_currentRawValue, YGenericSensor 561  
 get\_currentRawValue, YGyro 611  
 get\_currentRawValue, YHumidity 695  
 get\_currentRawValue, YLightSensor 770  
 get\_currentRawValue, YMagnetometer 813  
 get\_currentRawValue, YPower 1000  
 get\_currentRawValue, YPressure 1046  
 get\_currentRawValue, YQt 1158  
 get\_currentRawValue, YSensor 1312  
 get\_currentRawValue, YTemperature 1394  
 get\_currentRawValue, YTilt 1439  
 get\_currentRawValue, YVoc 1482  
 get\_currentRawValue, YVoltage 1525  
 get\_currentRunIndex, YDataLogger 295  
 get\_currentValue, YAccelerometer 45  
 get\_currentValue, YCarbonDioxide 133  
 get\_currentValue, YCompass 209  
 get\_currentValue, YCurrent 253  
 get\_currentValue, YGenericSensor 562  
 get\_currentValue, YGyro 612  
 get\_currentValue, YHumidity 696  
 get\_currentValue, YLightSensor 771  
 get\_currentValue, YMagnetometer 814  
 get\_currentValue, YPower 1001  
 get\_currentValue, YPressure 1047  
 get\_currentValue, YQt 1159  
 get\_currentValue, YSensor 1313  
 get\_currentValue, YTemperature 1395  
 get\_currentValue, YTilt 1440  
 get\_currentValue, YVoc 1483  
 get\_currentValue, YVoltage 1526  
 get\_data, YDataStream 350  
 get\_dataRows, YDataStream 351  
 get\_dataSamplesIntervalMs, YDataStream 352  
 get\_dataSets, YDataLogger 296  
 get\_dataStreams, YDataLogger 297  
 get\_dateTime, YRealTimeClock 1199  
 get\_detectedWlans, YWireless 1733  
 get\_discoverable, YNetwork 919  
 get\_display, YDisplayLayer 472  
 get\_displayHeight, YDisplay 418  
 get\_displayHeight, YDisplayLayer 473  
 get\_displayLayer, YDisplay 419  
 get\_displayType, YDisplay 420  
 get\_displayWidth, YDisplay 421  
 get\_displayWidth, YDisplayLayer 474  
 get\_duration, YDataRun 324  
 get\_duration, YDataStream 353  
 get\_dutyCycle, YPwmOutput 1089  
 get\_dutyCycleAtPowerOn, YPwmOutput 1090  
 get\_enabled, YDisplay 422  
 get\_enabled, YHubPort 665

get\_enabled, YPwmOutput 1091  
get\_enabled, YServo 1354  
get\_enabledAtPowerOn, YPwmOutput 1092  
get\_enabledAtPowerOn, YServo 1355  
get\_endTimeUTC, YDataSet 334  
get\_endTimeUTC, YMeasure 853  
get\_errorMessage, YAccelerometer 46  
get\_errorMessage, YAnButton 93  
get\_errorMessage, YCarbonDioxide 134  
get\_errorMessage, YColorLed 173  
get\_errorMessage, YCompass 210  
get\_errorMessage, YCurrent 254  
get\_errorMessage, YDataLogger 298  
get\_errorMessage, YDigitalIO 372  
get\_errorMessage, YDisplay 423  
get\_errorMessage, YDualPower 497  
get\_errorMessage, YFiles 529  
get\_errorMessage, YGenericSensor 563  
get\_errorMessage, YGyro 613  
get\_errorMessage, YHubPort 666  
get\_errorMessage, YHumidity 697  
get\_errorMessage, YLed 737  
get\_errorMessage, YLightSensor 772  
get\_errorMessage, YMagnetometer 815  
get\_errorMessage, YModule 869  
get\_errorMessage, YNetwork 920  
get\_errorMessage, YOsControl 970  
get\_errorMessage, YPower 1002  
get\_errorMessage, YPressure 1048  
get\_errorMessage, YPwmOutput 1093  
get\_errorMessage, YPwmPowerSource 1129  
get\_errorMessage, YQt 1160  
get\_errorMessage, YRealTimeClock 1200  
get\_errorMessage, YRefFrame 1238  
get\_errorMessage, YRelay 1273  
get\_errorMessage, YSensor 1314  
get\_errorMessage, YServo 1356  
get\_errorMessage, YTemperature 1396  
get\_errorMessage, YTilt 1441  
get\_errorMessage, YVoc 1484  
get\_errorMessage, YVoltage 1527  
get\_errorMessage, YVSource 1566  
get\_errorMessage, YWakeUpMonitor 1603  
get\_errorMessage, YWakeUpSchedule 1642  
get\_errorMessage, YWatchdog 1686  
get\_errorMessage, YWireless 1734  
get\_errorType, YAccelerometer 47  
get\_errorType, YAnButton 94  
get\_errorType, YCarbonDioxide 135  
get\_errorType, YColorLed 174  
get\_errorType, YCompass 211  
get\_errorType, YCurrent 255  
get\_errorType, YDataLogger 299  
get\_errorType, YDigitalIO 373  
get\_errorType, YDisplay 424  
get\_errorType, YDualPower 498  
get\_errorType, YFiles 530  
get\_errorType, YGenericSensor 564  
get\_errorType, YGyro 614

get\_errorType, YHubPort 667  
get\_errorType, YHumidity 698  
get\_errorType, YLed 738  
get\_errorType, YLightSensor 773  
get\_errorType, YMagnetometer 816  
get\_errorType, YModule 870  
get\_errorType, YNetwork 921  
get\_errorType, YOsControl 971  
get\_errorType, YPower 1003  
get\_errorType, YPressure 1049  
get\_errorType, YPwmOutput 1094  
get\_errorType, YPwmPowerSource 1130  
get\_errorType, YQt 1161  
get\_errorType, YRealTimeClock 1201  
get\_errorType, YRefFrame 1239  
get\_errorType, YRelay 1274  
get\_errorType, YSensor 1315  
get\_errorType, YServo 1357  
get\_errorType, YTemperature 1397  
get\_errorType, YTilt 1442  
get\_errorType, YVoc 1485  
get\_errorType, YVoltage 1528  
get\_errorType, YVSource 1567  
get\_errorType, YWakeUpMonitor 1604  
get\_errorType, YWakeUpSchedule 1643  
get\_errorType, YWatchdog 1687  
get\_errorType, YWireless 1735  
get\_extPowerFailure, YVSource 1568  
get\_extVoltage, YDualPower 499  
get\_failure, YVSource 1569  
get\_filesCount, YFiles 531  
get\_firmwareRelease, YModule 871  
get\_freeSpace, YFiles 532  
get\_frequency, YPwmOutput 1095  
get\_friendlyName, YAccelerometer 48  
get\_friendlyName, YAnButton 95  
get\_friendlyName, YCarbonDioxide 136  
get\_friendlyName, YColorLed 175  
get\_friendlyName, YCompass 212  
get\_friendlyName, YCurrent 256  
get\_friendlyName, YDataLogger 300  
get\_friendlyName, YDigitalIO 374  
get\_friendlyName, YDisplay 425  
get\_friendlyName, YDualPower 500  
get\_friendlyName, YFiles 533  
get\_friendlyName, YGenericSensor 565  
get\_friendlyName, YGyro 615  
get\_friendlyName, YHubPort 668  
get\_friendlyName, YHumidity 699  
get\_friendlyName, YLed 739  
get\_friendlyName, YLightSensor 774  
get\_friendlyName, YMagnetometer 817  
get\_friendlyName, YNetwork 922  
get\_friendlyName, YOsControl 972  
get\_friendlyName, YPower 1004  
get\_friendlyName, YPressure 1050  
get\_friendlyName, YPwmOutput 1096  
get\_friendlyName, YPwmPowerSource 1131  
get\_friendlyName, YQt 1162

get\_friendlyName, YRealTimeClock 1202  
get\_friendlyName, YRefFrame 1240  
get\_friendlyName, YRelay 1275  
get\_friendlyName, YSensor 1316  
get\_friendlyName, YServo 1358  
get\_friendlyName, YTemperature 1398  
get\_friendlyName, YTilt 1443  
get\_friendlyName, YVoc 1486  
get\_friendlyName, YVoltage 1529  
get\_friendlyName, YVSource 1570  
get\_friendlyName, YWakeUpMonitor 1605  
get\_friendlyName, YWakeUpSchedule 1644  
get\_friendlyName, YWatchdog 1688  
get\_friendlyName, YWireless 1736  
get\_functionDescriptor, YAccelerometer 49  
get\_functionDescriptor, YAnButton 96  
get\_functionDescriptor, YCarbonDioxide 137  
get\_functionDescriptor, YColorLed 176  
get\_functionDescriptor, YCompass 213  
get\_functionDescriptor, YCurrent 257  
get\_functionDescriptor, YDataLogger 301  
get\_functionDescriptor, YDigitalIO 375  
get\_functionDescriptor, YDisplay 426  
get\_functionDescriptor, YDualPower 501  
get\_functionDescriptor, YFiles 534  
get\_functionDescriptor, YGenericSensor 566  
get\_functionDescriptor, YGyro 616  
get\_functionDescriptor, YHubPort 669  
get\_functionDescriptor, YHumidity 700  
get\_functionDescriptor, YLed 740  
get\_functionDescriptor, YLightSensor 775  
get\_functionDescriptor, YMagnetometer 818  
get\_functionDescriptor, YNetwork 923  
get\_functionDescriptor, YOsControl 973  
get\_functionDescriptor, YPower 1005  
get\_functionDescriptor, YPressure 1051  
get\_functionDescriptor, YPwmOutput 1097  
get\_functionDescriptor, YPwmPowerSource 1132  
get\_functionDescriptor, YQt 1163  
get\_functionDescriptor, YRealTimeClock 1203  
get\_functionDescriptor, YRefFrame 1241  
get\_functionDescriptor, YRelay 1276  
get\_functionDescriptor, YSensor 1317  
get\_functionDescriptor, YServo 1359  
get\_functionDescriptor, YTemperature 1399  
get\_functionDescriptor, YTilt 1444  
get\_functionDescriptor, YVoc 1487  
get\_functionDescriptor, YVoltage 1530  
get\_functionDescriptor, YVSource 1571  
get\_functionDescriptor, YWakeUpMonitor 1606  
get\_functionDescriptor, YWakeUpSchedule 1645  
get\_functionDescriptor, YWatchdog 1689  
get\_functionDescriptor, YWireless 1737  
get\_functionId, YAccelerometer 50  
get\_functionId, YAnButton 97  
get\_functionId, YCarbonDioxide 138  
get\_functionId, YColorLed 177  
get\_functionId, YCompass 214  
get\_functionId, YCurrent 258

get\_functionId, YDataLogger 302  
get\_functionId, YDataSet 335  
get\_functionId, YDigitalIO 376  
get\_functionId, YDisplay 427  
get\_functionId, YDualPower 502  
get\_functionId, YFiles 535  
get\_functionId, YGenericSensor 567  
get\_functionId, YGyro 617  
get\_functionId, YHubPort 670  
get\_functionId, YHumidity 701  
get\_functionId, YLed 741  
get\_functionId, YLightSensor 776  
get\_functionId, YMagnetometer 819  
get\_functionId, YNetwork 924  
get\_functionId, YOsControl 974  
get\_functionId, YPower 1006  
get\_functionId, YPressure 1052  
get\_functionId, YPwmOutput 1098  
get\_functionId, YPwmPowerSource 1133  
get\_functionId, YQt 1164  
get\_functionId, YRealTimeClock 1204  
get\_functionId, YRefFrame 1242  
get\_functionId, YRelay 1277  
get\_functionId, YSensor 1318  
get\_functionId, YServo 1360  
get\_functionId, YTemperature 1400  
get\_functionId, YTilt 1445  
get\_functionId, YVoc 1488  
get\_functionId, YVoltage 1531  
get\_functionId, YVSource 1572  
get\_functionId, YWakeUpMonitor 1607  
get\_functionId, YWakeUpSchedule 1646  
get\_functionId, YWatchdog 1690  
get\_functionId, YWireless 1738  
get\_hardwareId, YAccelerometer 51  
get\_hardwareId, YAnButton 98  
get\_hardwareId, YCarbonDioxide 139  
get\_hardwareId, YColorLed 178  
get\_hardwareId, YCompass 215  
get\_hardwareId, YCurrent 259  
get\_hardwareId, YDataLogger 303  
get\_hardwareId, YDataSet 336  
get\_hardwareId, YDigitalIO 377  
get\_hardwareId, YDisplay 428  
get\_hardwareId, YDualPower 503  
get\_hardwareId, YFiles 536  
get\_hardwareId, YGenericSensor 568  
get\_hardwareId, YGyro 618  
get\_hardwareId, YHubPort 671  
get\_hardwareId, YHumidity 702  
get\_hardwareId, YLed 742  
get\_hardwareId, YLightSensor 777  
get\_hardwareId, YMagnetometer 820  
get\_hardwareId, YModule 872  
get\_hardwareId, YNetwork 925  
get\_hardwareId, YOsControl 975  
get\_hardwareId, YPower 1007  
get\_hardwareId, YPressure 1053  
get\_hardwareId, YPwmOutput 1099

get\_hardwareId, YPwmPowerSource 1134  
get\_hardwareId, YQt 1165  
get\_hardwareId, YRealTimeClock 1205  
get\_hardwareId, YRefFrame 1243  
get\_hardwareId, YRelay 1278  
get\_hardwareId, YSensor 1319  
get\_hardwareId, YServo 1361  
get\_hardwareId, YTemperature 1401  
get\_hardwareId, YTilt 1446  
get\_hardwareId, YVoc 1489  
get\_hardwareId, YVoltage 1532  
get\_hardwareId, YVSource 1573  
get\_hardwareId, YWakeUpMonitor 1608  
get\_hardwareId, YWakeUpSchedule 1647  
get\_hardwareId, YWatchdog 1691  
get\_hardwareId, YWireless 1739  
get\_heading, YGyro 619  
get\_highestValue, YAccelerometer 52  
get\_highestValue, YCarbonDioxide 140  
get\_highestValue, YCompass 216  
get\_highestValue, YCurrent 260  
get\_highestValue, YGenericSensor 569  
get\_highestValue, YGyro 620  
get\_highestValue, YHumidity 703  
get\_highestValue, YLightSensor 778  
get\_highestValue, YMagnetometer 821  
get\_highestValue, YPower 1008  
get\_highestValue, YPressure 1054  
get\_highestValue, YQt 1166  
get\_highestValue, YSensor 1320  
get\_highestValue, YTemperature 1402  
get\_highestValue, YTilt 1447  
get\_highestValue, YVoc 1490  
get\_highestValue, YVoltage 1533  
get\_hours, YWakeUpSchedule 1648  
get\_hslColor, YColorLed 179  
get\_icon2d, YModule 873  
get\_ipAddress, YNetwork 926  
get\_isPressed, YAnButton 99  
get\_lastLogs, YModule 874  
get\_lastTimePressed, YAnButton 100  
get\_lastTimeReleased, YAnButton 101  
get\_layerCount, YDisplay 429  
get\_layerHeight, YDisplay 430  
get\_layerHeight, YDisplayLayer 475  
get\_layerWidth, YDisplay 431  
get\_layerWidth, YDisplayLayer 476  
get\_linkQuality, YWireless 1740  
get\_list, YFiles 537  
get\_logFrequency, YAccelerometer 53  
get\_logFrequency, YCarbonDioxide 141  
get\_logFrequency, YCompass 217  
get\_logFrequency, YCurrent 261  
get\_logFrequency, YGenericSensor 570  
get\_logFrequency, YGyro 621  
get\_logFrequency, YHumidity 704  
get\_logFrequency, YLightSensor 779  
get\_logFrequency, YMagnetometer 822  
get\_logFrequency, YPower 1009

get\_logFrequency, YPressure 1055  
get\_logFrequency, YQt 1167  
get\_logFrequency, YSensor 1321  
get\_logFrequency, YTemperature 1403  
get\_logFrequency, YTilt 1448  
get\_logFrequency, YVoc 1491  
get\_logFrequency, YVoltage 1534  
get\_logicalName, YAccelerometer 54  
get\_logicalName, YAnButton 102  
get\_logicalName, YCarbonDioxide 142  
get\_logicalName, YColorLed 180  
get\_logicalName, YCompass 218  
get\_logicalName, YCurrent 262  
get\_logicalName, YDataLogger 304  
get\_logicalName, YDigitalIO 378  
get\_logicalName, YDisplay 432  
get\_logicalName, YDualPower 504  
get\_logicalName, YFiles 538  
get\_logicalName, YGenericSensor 571  
get\_logicalName, YGyro 622  
get\_logicalName, YHubPort 672  
get\_logicalName, YHumidity 705  
get\_logicalName, YLed 743  
get\_logicalName, YLightSensor 780  
get\_logicalName, YMagnetometer 823  
get\_logicalName, YModule 875  
get\_logicalName, YNetwork 927  
get\_logicalName, YOsControl 976  
get\_logicalName, YPower 1010  
get\_logicalName, YPressure 1056  
get\_logicalName, YPwmOutput 1100  
get\_logicalName, YPwmPowerSource 1135  
get\_logicalName, YQt 1168  
get\_logicalName, YRealTimeClock 1206  
get\_logicalName, YRefFrame 1244  
get\_logicalName, YRelay 1279  
get\_logicalName, YSensor 1322  
get\_logicalName, YServo 1362  
get\_logicalName, YTemperature 1404  
get\_logicalName, YTilt 1449  
get\_logicalName, YVoc 1492  
get\_logicalName, YVoltage 1535  
get\_logicalName, YVSource 1574  
get\_logicalName, YWakeUpMonitor 1609  
get\_logicalName, YWakeUpSchedule 1649  
get\_logicalName, YWatchdog 1692  
get\_logicalName, YWireless 1741  
get\_lowestValue, YAccelerometer 55  
get\_lowestValue, YCarbonDioxide 143  
get\_lowestValue, YCompass 219  
get\_lowestValue, YCurrent 263  
get\_lowestValue, YGenericSensor 572  
get\_lowestValue, YGyro 623  
get\_lowestValue, YHumidity 706  
get\_lowestValue, YLightSensor 781  
get\_lowestValue, YMagnetometer 824  
get\_lowestValue, YPower 1011  
get\_lowestValue, YPressure 1057  
get\_lowestValue, YQt 1169



get\_lowestValue, YSensor 1323  
get\_lowestValue, YTemperature 1405  
get\_lowestValue, YTilt 1450  
get\_lowestValue, YVoc 1493  
get\_lowestValue, YVoltage 1536  
get\_luminosity, YLed 744  
get\_luminosity, YModule 876  
get\_macAddress, YNetwork 928  
get\_magneticHeading, YCompass 220  
get\_maxTimeOnStateA, YRelay 1280  
get\_maxTimeOnStateA, YWatchdog 1693  
get\_maxTimeOnStateB, YRelay 1281  
get\_maxTimeOnStateB, YWatchdog 1694  
get\_maxValue, YDataRun 325  
get\_maxValue, YDataStream 354  
get\_maxValue, YMeasure 854  
get\_measureNames, YDataRun 326  
get\_measures, YDataSet 337  
get\_message, YWireless 1742  
get\_meter, YPower 1012  
get\_meterTimer, YPower 1013  
get\_minutes, YWakeUpSchedule 1650  
get\_minutesA, YWakeUpSchedule 1651  
get\_minutesB, YWakeUpSchedule 1652  
get\_minValue, YDataRun 327  
get\_minValue, YDataStream 355  
get\_minValue, YMeasure 855  
get\_module, YAccelerometer 56  
get\_module, YAnButton 103  
get\_module, YCarbonDioxide 144  
get\_module, YColorLed 181  
get\_module, YCompass 221  
get\_module, YCurrent 264  
get\_module, YDataLogger 305  
get\_module, YDigitalIO 379  
get\_module, YDisplay 433  
get\_module, YDualPower 505  
get\_module, YFiles 539  
get\_module, YGenericSensor 573  
get\_module, YGyro 624  
get\_module, YHubPort 673  
get\_module, YHumidity 707  
get\_module, YLed 745  
get\_module, YLightSensor 782  
get\_module, YMagnetometer 825  
get\_module, YNetwork 930  
get\_module, YOsControl 978  
get\_module, YPower 1015  
get\_module, YPressure 1059  
get\_module, YPwmOutput 1101  
get\_module, YPwmPowerSource 1136  
get\_module, YQt 1170  
get\_module, YRealTimeClock 1207  
get\_module, YRefFrame 1245  
get\_module, YRelay 1282  
get\_module, YSensor 1325  
get\_module, YServo 1364  
get\_module, YTemperature 1407  
get\_module, YTilt 1452

get\_module, YVoc 1494  
get\_module, YVoltage 1537  
get\_module, YVSource 1575  
get\_module, YWakeUpMonitor 1610  
get\_module, YWakeUpSchedule 1653  
get\_module, YWatchdog 1695  
get\_module, YWireless 1743  
get\_module\_async, YAccelerometer 57  
get\_module\_async, YAnButton 104  
get\_module\_async, YCarbonDioxide 145  
get\_module\_async, YColorLed 182  
get\_module\_async, YCompass 222  
get\_module\_async, YCurrent 265  
get\_module\_async, YDataLogger 306  
get\_module\_async, YDigitalIO 380  
get\_module\_async, YDisplay 434  
get\_module\_async, YDualPower 506  
get\_module\_async, YFiles 540  
get\_module\_async, YGenericSensor 574  
get\_module\_async, YGyro 625  
get\_module\_async, YHubPort 674  
get\_module\_async, YHumidity 708  
get\_module\_async, YLed 746  
get\_module\_async, YLightSensor 783  
get\_module\_async, YMagnetometer 826  
get\_module\_async, YNetwork 930  
get\_module\_async, YOsControl 978  
get\_module\_async, YPower 1015  
get\_module\_async, YPressure 1059  
get\_module\_async, YPwmOutput 1102  
get\_module\_async, YPwmPowerSource 1137  
get\_module\_async, YQt 1171  
get\_module\_async, YRealTimeClock 1208  
get\_module\_async, YRefFrame 1246  
get\_module\_async, YRelay 1283  
get\_module\_async, YSensor 1325  
get\_module\_async, YServo 1364  
get\_module\_async, YTemperature 1407  
get\_module\_async, YTilt 1452  
get\_module\_async, YVoc 1495  
get\_module\_async, YVoltage 1538  
get\_module\_async, YVSource 1576  
get\_module\_async, YWakeUpMonitor 1611  
get\_module\_async, YWakeUpSchedule 1654  
get\_module\_async, YWatchdog 1696  
get\_module\_async, YWireless 1744  
get\_monthDays, YWakeUpSchedule 1655  
get\_months, YWakeUpSchedule 1656  
get\_mountOrientation, YRefFrame 1247  
get\_mountPosition, YRefFrame 1248  
get\_neutral, YServo 1365  
get\_nextOccurrence, YWakeUpSchedule 1657  
get\_nextWakeUp, YWakeUpMonitor 1612  
get\_orientation, YDisplay 435  
get\_output, YRelay 1284  
get\_output, YWatchdog 1697  
get\_outputVoltage, YDigitalIO 381  
get\_overCurrent, YVSource 1577  
get\_overHeat, YVSource 1578

get\_overLoad, YVSource 1579  
get\_period, YPwmOutput 1103  
get\_persistentSettings, YModule 877  
get\_pitch, YGyro 626  
get\_poeCurrent, YNetwork 931  
get\_portDirection, YDigitalIO 382  
get\_portOpenDrain, YDigitalIO 383  
get\_portPolarity, YDigitalIO 384  
get\_portSize, YDigitalIO 385  
get\_portState, YDigitalIO 386  
get\_portState, YHubPort 675  
get\_position, YServo 1366  
get\_positionAtPowerOn, YServo 1367  
get\_power, YLed 747  
get\_powerControl, YDualPower 507  
get\_powerDuration, YWakeUpMonitor 1613  
get\_powerMode, YPwmPowerSource 1138  
get\_powerState, YDualPower 508  
get\_preview, YDataSet 338  
get\_primaryDNS, YNetwork 932  
get\_productId, YModule 878  
get\_productName, YModule 879  
get\_productRelease, YModule 880  
get\_progress, YDataSet 339  
get\_pulseCounter, YAnButton 105  
get\_pulseDuration, YPwmOutput 1104  
get\_pulseTimer, YAnButton 106  
get\_pulseTimer, YRelay 1285  
get\_pulseTimer, YWatchdog 1698  
get\_quaternionW, YGyro 627  
get\_quaternionX, YGyro 628  
get\_quaternionY, YGyro 629  
get\_quaternionZ, YGyro 630  
get\_range, YServo 1368  
get\_rawValue, YAnButton 107  
get\_readiness, YNetwork 933  
get\_rebootCountdown, YModule 881  
get\_recordedData, YAccelerometer 58  
get\_recordedData, YCarbonDioxide 146  
get\_recordedData, YCompass 223  
get\_recordedData, YCurrent 266  
get\_recordedData, YGenericSensor 575  
get\_recordedData, YGyro 631  
get\_recordedData, YHumidity 709  
get\_recordedData, YLightSensor 784  
get\_recordedData, YMagnetometer 827  
get\_recordedData, YPower 1016  
get\_recordedData, YPressure 1060  
get\_recordedData, YQt 1172  
get\_recordedData, YSensor 1326  
get\_recordedData, YTemperature 1408  
get\_recordedData, YTilt 1453  
get\_recordedData, YVoc 1496  
get\_recordedData, YVoltage 1539  
get\_recording, YDataLogger 307  
get\_regulationFailure, YVSource 1580  
get\_reportFrequency, YAccelerometer 59  
get\_reportFrequency, YCarbonDioxide 147  
get\_reportFrequency, YCompass 224

get\_reportFrequency, YCurrent 267  
get\_reportFrequency, YGenericSensor 576  
get\_reportFrequency, YGyro 632  
get\_reportFrequency, YHumidity 710  
get\_reportFrequency, YLightSensor 785  
get\_reportFrequency, YMagnetometer 828  
get\_reportFrequency, YPower 1017  
get\_reportFrequency, YPressure 1061  
get\_reportFrequency, YQt 1173  
get\_reportFrequency, YSensor 1327  
get\_reportFrequency, YTemperature 1409  
get\_reportFrequency, YTilt 1454  
get\_reportFrequency, YVoc 1497  
get\_reportFrequency, YVoltage 1540  
get\_resolution, YAccelerometer 60  
get\_resolution, YCarbonDioxide 148  
get\_resolution, YCompass 225  
get\_resolution, YCurrent 268  
get\_resolution, YGenericSensor 577  
get\_resolution, YGyro 633  
get\_resolution, YHumidity 711  
get\_resolution, YLightSensor 786  
get\_resolution, YMagnetometer 829  
get\_resolution, YPower 1018  
get\_resolution, YPressure 1062  
get\_resolution, YQt 1174  
get\_resolution, YSensor 1328  
get\_resolution, YTemperature 1410  
get\_resolution, YTilt 1455  
get\_resolution, YVoc 1498  
get\_resolution, YVoltage 1541  
get\_rgbColor, YColorLed 183  
get\_rgbColorAtPowerOn, YColorLed 184  
get\_roll, YGyro 634  
get\_router, YNetwork 934  
get\_rowCount, YDataStream 356  
get\_runIndex, YDataStream 357  
get\_running, YWatchdog 1699  
get\_secondaryDNS, YNetwork 935  
get\_security, YWireless 1745  
get\_sensitivity, YAnButton 108  
get\_sensorType, YTemperature 1411  
get\_serialNumber, YModule 882  
get\_shutdownCountdown, YOsControl 979  
get\_signalRange, YGenericSensor 578  
get\_signalUnit, YGenericSensor 579  
get\_signalValue, YGenericSensor 580  
get\_sleepCountdown, YWakeUpMonitor 1614  
get\_ssid, YWireless 1746  
get\_startTime, YDataStream 358  
get\_startTimeUTC, YDataRun 328  
get\_startTimeUTC, YDataSet 340  
get\_startTimeUTC, YDataStream 359  
get\_startTimeUTC, YMeasure 856  
get\_startupSeq, YDisplay 436  
get\_state, YRelay 1286  
get\_state, YWatchdog 1700  
get\_stateAtPowerOn, YRelay 1287  
get\_stateAtPowerOn, YWatchdog 1701

- get\_subnetMask, YNetwork 936
- get\_summary, YDataSet 341
- get\_timeSet, YRealTimeClock 1209
- get\_timeUTC, YDataLogger 308
- get\_triggerDelay, YWatchdog 1702
- get\_triggerDuration, YWatchdog 1703
- get\_unit, YAccelerometer 61
- get\_unit, YCarbonDioxide 149
- get\_unit, YCompass 226
- get\_unit, YCurrent 269
- get\_unit, YDataSet 342
- get\_unit, YGenericSensor 581
- get\_unit, YGyro 635
- get\_unit, YHumidity 712
- get\_unit, YLightSensor 787
- get\_unit, YMagnetometer 830
- get\_unit, YPower 1019
- get\_unit, YPressure 1063
- get\_unit, YQt 1175
- get\_unit, YSensor 1329
- get\_unit, YTemperature 1412
- get\_unit, YTilt 1456
- get\_unit, YVoc 1499
- get\_unit, YVoltage 1542
- get\_unit, YVSource 1581
- get\_unixTime, YRealTimeClock 1210
- get\_upTime, YModule 883
- get\_usbBandwidth, YModule 884
- get\_usbCurrent, YModule 885
- get\_userData, YAccelerometer 62
- get\_userData, YAnButton 109
- get\_userData, YCarbonDioxide 150
- get\_userData, YColorLed 185
- get\_userData, YCompass 227
- get\_userData, YCurrent 270
- get\_userData, YDataLogger 309
- get\_userData, YDigitalIO 387
- get\_userData, YDisplay 437
- get\_userData, YDualPower 509
- get\_userData, YFiles 541
- get\_userData, YGenericSensor 582
- get\_userData, YGyro 636
- get\_userData, YHubPort 676
- get\_userData, YHumidity 713
- get\_userData, YLed 748
- get\_userData, YLightSensor 788
- get\_userData, YMagnetometer 831
- get\_userData, YModule 886
- get\_userData, YNetwork 937
- get\_userData, YOsControl 980
- get\_userData, YPower 1020
- get\_userData, YPressure 1064
- get\_userData, YPwmOutput 1105
- get\_userData, YPwmPowerSource 1139
- get\_userData, YQt 1176
- get\_userData, YRealTimeClock 1211
- get\_userData, YRefFrame 1249
- get\_userData, YRelay 1288
- get\_userData, YSensor 1330

- get\_userData, YServo 1369
- get\_userData, YTemperature 1413
- get\_userData, YTilt 1457
- get\_userData, YVoc 1500
- get\_userData, YVoltage 1543
- get\_userData, YVSource 1582
- get\_userData, YWakeUpMonitor 1615
- get\_userData, YWakeUpSchedule 1658
- get\_userData, YWatchdog 1704
- get\_userData, YWireless 1747
- get\_userPassword, YNetwork 938
- get\_utcOffset, YRealTimeClock 1212
- get\_valueCount, YDataRun 329
- get\_valueInterval, YDataRun 330
- get\_valueRange, YGenericSensor 583
- get\_voltage, YVSource 1583
- get\_wakeUpReason, YWakeUpMonitor 1616
- get\_wakeUpState, YWakeUpMonitor 1617
- get\_weekDays, YWakeUpSchedule 1659
- get\_wwwWatchdogDelay, YNetwork 939
- get\_xValue, YAccelerometer 63
- get\_xValue, YGyro 637
- get\_xValue, YMagnetometer 832
- get\_yValue, YAccelerometer 64
- get\_yValue, YGyro 638
- get\_yValue, YMagnetometer 833
- get\_zValue, YAccelerometer 65
- get\_zValue, YGyro 639
- get\_zValue, YMagnetometer 834
- GetAPIVersion, YAPI 17
- GetTickCount, YAPI 18
- Gyro 604

## H

- HandleEvents, YAPI 19
- hide, YDisplayLayer 477
- Horloge 1194
- hslMove, YColorLed 186
- Humidity 688

## I

- InitAPI, YAPI 20
- Interface 37, 83, 125, 168, 201, 245, 288, 361, 409, 460, 492, 521, 554, 604, 659, 688, 731, 762, 806, 858, 904, 992, 1039, 1082, 1124, 1151, 1194, 1265, 1305, 1348, 1387, 1432, 1475, 1518, 1561, 1597, 1636, 1677, 1726
- Introduction 1
- isOnline, YAccelerometer 66
- isOnline, YAnButton 110
- isOnline, YCarbonDioxide 151
- isOnline, YColorLed 187
- isOnline, YCompass 228
- isOnline, YCurrent 271
- isOnline, YDataLogger 310
- isOnline, YDigitalIO 388
- isOnline, YDisplay 438
- isOnline, YDualPower 510

isOnline, YFiles 542  
isOnline, YGenericSensor 584  
isOnline, YGyro 640  
isOnline, YHubPort 677  
isOnline, YHumidity 714  
isOnline, YLed 749  
isOnline, YLightSensor 789  
isOnline, YMagnetometer 835  
isOnline, YModule 887  
isOnline, YNetwork 940  
isOnline, YOsControl 981  
isOnline, YPower 1021  
isOnline, YPressure 1065  
isOnline, YPwmOutput 1106  
isOnline, YPwmPowerSource 1140  
isOnline, YQt 1177  
isOnline, YRealTimeClock 1213  
isOnline, YRefFrame 1250  
isOnline, YRelay 1289  
isOnline, YSensor 1331  
isOnline, YServo 1370  
isOnline, YTemperature 1414  
isOnline, YTilt 1458  
isOnline, YVoc 1501  
isOnline, YVoltage 1544  
isOnline, YVSource 1584  
isOnline, YWakeUpMonitor 1618  
isOnline, YWakeUpSchedule 1660  
isOnline, YWatchdog 1705  
isOnline, YWireless 1748  
isOnline\_async, YAccelerometer 67  
isOnline\_async, YAnButton 111  
isOnline\_async, YCarbonDioxide 152  
isOnline\_async, YColorLed 188  
isOnline\_async, YCompass 229  
isOnline\_async, YCurrent 272  
isOnline\_async, YDataLogger 311  
isOnline\_async, YDigitalIO 389  
isOnline\_async, YDisplay 439  
isOnline\_async, YDualPower 511  
isOnline\_async, YFiles 543  
isOnline\_async, YGenericSensor 585  
isOnline\_async, YGyro 641  
isOnline\_async, YHubPort 678  
isOnline\_async, YHumidity 715  
isOnline\_async, YLed 750  
isOnline\_async, YLightSensor 790  
isOnline\_async, YMagnetometer 836  
isOnline\_async, YModule 888  
isOnline\_async, YNetwork 941  
isOnline\_async, YOsControl 982  
isOnline\_async, YPower 1022  
isOnline\_async, YPressure 1066  
isOnline\_async, YPwmOutput 1107  
isOnline\_async, YPwmPowerSource 1141  
isOnline\_async, YQt 1178  
isOnline\_async, YRealTimeClock 1214  
isOnline\_async, YRefFrame 1251  
isOnline\_async, YRelay 1290

isOnline\_async, YSensor 1332  
isOnline\_async, YServo 1371  
isOnline\_async, YTemperature 1415  
isOnline\_async, YTilt 1459  
isOnline\_async, YVoc 1502  
isOnline\_async, YVoltage 1545  
isOnline\_async, YVSource 1585  
isOnline\_async, YWakeUpMonitor 1619  
isOnline\_async, YWakeUpSchedule 1661  
isOnline\_async, YWatchdog 1706  
isOnline\_async, YWireless 1749

## J

joinNetwork, YWireless 1750

## L

LightSensor 762  
lineTo, YDisplayLayer 478  
load, YAccelerometer 68  
load, YAnButton 112  
load, YCarbonDioxide 153  
load, YColorLed 189  
load, YCompass 230  
load, YCurrent 273  
load, YDataLogger 312  
load, YDigitalIO 390  
load, YDisplay 440  
load, YDualPower 512  
load, YFiles 544  
load, YGenericSensor 586  
load, YGyro 642  
load, YHubPort 679  
load, YHumidity 716  
load, YLed 751  
load, YLightSensor 791  
load, YMagnetometer 837  
load, YModule 889  
load, YNetwork 942  
load, YOsControl 983  
load, YPower 1023  
load, YPressure 1067  
load, YPwmOutput 1108  
load, YPwmPowerSource 1142  
load, YQt 1179  
load, YRealTimeClock 1215  
load, YRefFrame 1252  
load, YRelay 1291  
load, YSensor 1333  
load, YServo 1372  
load, YTemperature 1416  
load, YTilt 1460  
load, YVoc 1503  
load, YVoltage 1546  
load, YVSource 1586  
load, YWakeUpMonitor 1620  
load, YWakeUpSchedule 1662  
load, YWatchdog 1707  
load, YWireless 1751

- load\_async, YAccelerometer 70
- load\_async, YAnButton 113
- load\_async, YCarbonDioxide 155
- load\_async, YColorLed 190
- load\_async, YCompass 232
- load\_async, YCurrent 275
- load\_async, YDataLogger 313
- load\_async, YDigitalIO 391
- load\_async, YDisplay 441
- load\_async, YDualPower 513
- load\_async, YFiles 545
- load\_async, YGenericSensor 588
- load\_async, YGyro 644
- load\_async, YHubPort 680
- load\_async, YHumidity 718
- load\_async, YLed 752
- load\_async, YLightSensor 793
- load\_async, YMagnetometer 839
- load\_async, YModule 890
- load\_async, YNetwork 943
- load\_async, YOsControl 984
- load\_async, YPower 1025
- load\_async, YPressure 1069
- load\_async, YPwmOutput 1109
- load\_async, YPwmPowerSource 1143
- load\_async, YQt 1181
- load\_async, YRealTimeClock 1216
- load\_async, YRefFrame 1253
- load\_async, YRelay 1292
- load\_async, YSensor 1335
- load\_async, YServo 1373
- load\_async, YTemperature 1418
- load\_async, YTilt 1462
- load\_async, YVoc 1505
- load\_async, YVoltage 1548
- load\_async, YVSource 1587
- load\_async, YWakeUpMonitor 1621
- load\_async, YWakeUpSchedule 1663
- load\_async, YWatchdog 1708
- load\_async, YWireless 1752
- loadCalibrationPoints, YAccelerometer 69
- loadCalibrationPoints, YCarbonDioxide 154
- loadCalibrationPoints, YCompass 231
- loadCalibrationPoints, YCurrent 274
- loadCalibrationPoints, YGenericSensor 587
- loadCalibrationPoints, YGyro 643
- loadCalibrationPoints, YHumidity 717
- loadCalibrationPoints, YLightSensor 792
- loadCalibrationPoints, YMagnetometer 838
- loadCalibrationPoints, YPower 1024
- loadCalibrationPoints, YPressure 1068
- loadCalibrationPoints, YQt 1180
- loadCalibrationPoints, YSensor 1334
- loadCalibrationPoints, YTemperature 1417
- loadCalibrationPoints, YTilt 1461
- loadCalibrationPoints, YVoc 1504
- loadCalibrationPoints, YVoltage 1547
- loadMore, YDataSet 343
- loadMore\_async, YDataSet 344

## M

- Magnetometer 806
- Mesurée 852
- Mise 323
- Module 5, 858
- more3DCalibration, YRefFrame 1254
- move, YServo 1374
- moveTo, YDisplayLayer 479

## N

- Network 904
- newSequence, YDisplay 442
- nextAccelerometer, YAccelerometer 71
- nextAnButton, YAnButton 114
- nextCarbonDioxide, YCarbonDioxide 156
- nextColorLed, YColorLed 191
- nextCompass, YCompass 233
- nextCurrent, YCurrent 276
- nextDataLogger, YDataLogger 314
- nextDigitalIO, YDigitalIO 392
- nextDisplay, YDisplay 443
- nextDualPower, YDualPower 514
- nextFiles, YFiles 546
- nextGenericSensor, YGenericSensor 589
- nextGyro, YGyro 645
- nextHubPort, YHubPort 681
- nextHumidity, YHumidity 719
- nextLed, YLed 753
- nextLightSensor, YLightSensor 794
- nextMagnetometer, YMagnetometer 840
- nextModule, YModule 891
- nextNetwork, YNetwork 944
- nextOsControl, YOsControl 985
- nextPower, YPower 1026
- nextPressure, YPressure 1070
- nextPwmOutput, YPwmOutput 1110
- nextPwmPowerSource, YPwmPowerSource 1144
- nextQt, YQt 1182
- nextRealTimeClock, YRealTimeClock 1217
- nextRefFrame, YRefFrame 1255
- nextRelay, YRelay 1293
- nextSensor, YSensor 1336
- nextServo, YServo 1375
- nextTemperature, YTemperature 1419
- nextTilt, YTilt 1463
- nextVoc, YVoc 1506
- nextVoltage, YVoltage 1549
- nextVSource, YVSource 1588
- nextWakeUpMonitor, YWakeUpMonitor 1622
- nextWakeUpSchedule, YWakeUpSchedule 1664
- nextWatchdog, YWatchdog 1709
- nextWireless, YWireless 1753

## O

- Objective-C 3

Objets 460

## P

pauseSequence, YDisplay 444  
ping, YNetwork 945  
playSequence, YDisplay 445  
Port 659  
Power 992  
PreregisterHub, YAPI 21  
Pressure 1039  
pulse, YDigitalIO 393  
pulse, YRelay 1294  
pulse, YVSource 1589  
pulse, YWatchdog 1710  
pulseDurationMove, YPwmOutput 1111  
PwmPowerSource 1124

## Q

Quaternion 1151

## R

Real 1194  
reboot, YModule 892  
Reference 10  
Référentiel 1225  
registerAnglesCallback, YGyro 646  
RegisterDeviceArrivalCallback, YAPI 22  
RegisterDeviceRemovalCallback, YAPI 23  
RegisterHub, YAPI 24  
RegisterHubDiscoveryCallback, YAPI 26  
registerLogCallback, YModule 893  
RegisterLogFunction, YAPI 27  
registerQuaternionCallback, YGyro 647  
registerTimedReportCallback, YAccelerometer 72  
registerTimedReportCallback, YCarbonDioxide 157  
registerTimedReportCallback, YCompass 234  
registerTimedReportCallback, YCurrent 277  
registerTimedReportCallback, YGenericSensor 590  
registerTimedReportCallback, YGyro 648  
registerTimedReportCallback, YHumidity 720  
registerTimedReportCallback, YLightSensor 795  
registerTimedReportCallback, YMagnetometer 841  
registerTimedReportCallback, YPower 1027  
registerTimedReportCallback, YPressure 1071  
registerTimedReportCallback, YQt 1183  
registerTimedReportCallback, YSensor 1337  
registerTimedReportCallback, YTemperature 1420  
registerTimedReportCallback, YTilt 1464  
registerTimedReportCallback, YVoc 1507  
registerTimedReportCallback, YVoltage 1550  
registerValueCallback, YAccelerometer 73  
registerValueCallback, YAnButton 115

registerValueCallback, YCarbonDioxide 158  
registerValueCallback, YColorLed 192  
registerValueCallback, YCompass 235  
registerValueCallback, YCurrent 278  
registerValueCallback, YDataLogger 315  
registerValueCallback, YDigitalIO 394  
registerValueCallback, YDisplay 446  
registerValueCallback, YDualPower 515  
registerValueCallback, YFiles 547  
registerValueCallback, YGenericSensor 591  
registerValueCallback, YGyro 649  
registerValueCallback, YHubPort 682  
registerValueCallback, YHumidity 721  
registerValueCallback, YLed 754  
registerValueCallback, YLightSensor 796  
registerValueCallback, YMagnetometer 842  
registerValueCallback, YNetwork 946  
registerValueCallback, YOsControl 986  
registerValueCallback, YPower 1028  
registerValueCallback, YPressure 1072  
registerValueCallback, YPwmOutput 1112  
registerValueCallback, YPwmPowerSource 1145  
registerValueCallback, YQt 1184  
registerValueCallback, YRealTimeClock 1218  
registerValueCallback, YRefFrame 1256  
registerValueCallback, YRelay 1295  
registerValueCallback, YSensor 1338  
registerValueCallback, YServo 1376  
registerValueCallback, YTemperature 1421  
registerValueCallback, YTilt 1465  
registerValueCallback, YVoc 1508  
registerValueCallback, YVoltage 1551  
registerValueCallback, YVSource 1590  
registerValueCallback, YWakeUpMonitor 1623  
registerValueCallback, YWakeUpSchedule 1665  
registerValueCallback, YWatchdog 1711  
registerValueCallback, YWireless 1754  
Relay 1265  
remove, YFiles 548  
reset, YDisplayLayer 480  
reset, YPower 1029  
resetAll, YDisplay 447  
resetCounter, YAnButton 116  
resetSleepCountDown, YWakeUpMonitor 1624  
resetWatchdog, YWatchdog 1712  
revertFromFlash, YModule 894  
rgbMove, YColorLed 193

## S

save3DCalibration, YRefFrame 1257  
saveSequence, YDisplay 448  
saveToFlash, YModule 895  
SelectArchitecture, YAPI 28  
selectColorPen, YDisplayLayer 481  
selectEraser, YDisplayLayer 482  
selectFont, YDisplayLayer 483  
selectGrayPen, YDisplayLayer 484  
Senseur 1305  
Séquence 323, 333, 346

Servo 1348  
set\_adminPassword, YNetwork 947  
set\_analogCalibration, YAnButton 117  
set\_autoStart, YDataLogger 316  
set\_autoStart, YWatchdog 1713  
set\_beacon, YModule 896  
set\_bearing, YRefFrame 1258  
set\_bitDirection, YDigitalIO 395  
set\_bitOpenDrain, YDigitalIO 396  
set\_bitPolarity, YDigitalIO 397  
set\_bitState, YDigitalIO 398  
set\_blinking, YLed 755  
set\_brightness, YDisplay 449  
set\_calibrationMax, YAnButton 118  
set\_calibrationMin, YAnButton 119  
set\_callbackCredentials, YNetwork 948  
set\_callbackEncoding, YNetwork 949  
set\_callbackMaxDelay, YNetwork 950  
set\_callbackMethod, YNetwork 951  
set\_callbackMinDelay, YNetwork 952  
set\_callbackUrl, YNetwork 953  
set\_discoverable, YNetwork 954  
set\_dutyCycle, YPwmOutput 1113  
set\_dutyCycleAtPowerOn, YPwmOutput 1114  
set\_enabled, YDisplay 450  
set\_enabled, YHubPort 683  
set\_enabled, YPwmOutput 1115  
set\_enabled, YServo 1377  
set\_enabledAtPowerOn, YPwmOutput 1116  
set\_enabledAtPowerOn, YServo 1378  
set\_frequency, YPwmOutput 1117  
set\_highestValue, YAccelerometer 74  
set\_highestValue, YCarbonDioxide 159  
set\_highestValue, YCompass 236  
set\_highestValue, YCurrent 279  
set\_highestValue, YGenericSensor 592  
set\_highestValue, YGyro 650  
set\_highestValue, YHumidity 722  
set\_highestValue, YLightSensor 797  
set\_highestValue, YMagnetometer 843  
set\_highestValue, YPower 1030  
set\_highestValue, YPressure 1073  
set\_highestValue, YQt 1185  
set\_highestValue, YSensor 1339  
set\_highestValue, YTemperature 1422  
set\_highestValue, YTilt 1466  
set\_highestValue, YVoc 1509  
set\_highestValue, YVoltage 1552  
set\_hours, YWakeUpSchedule 1666  
set\_hslColor, YColorLed 194  
set\_logFrequency, YAccelerometer 75  
set\_logFrequency, YCarbonDioxide 160  
set\_logFrequency, YCompass 237  
set\_logFrequency, YCurrent 280  
set\_logFrequency, YGenericSensor 593  
set\_logFrequency, YGyro 651  
set\_logFrequency, YHumidity 723  
set\_logFrequency, YLightSensor 798  
set\_logFrequency, YMagnetometer 844

set\_logFrequency, YPower 1031  
set\_logFrequency, YPressure 1074  
set\_logFrequency, YQt 1186  
set\_logFrequency, YSensor 1340  
set\_logFrequency, YTemperature 1423  
set\_logFrequency, YTilt 1467  
set\_logFrequency, YVoc 1510  
set\_logFrequency, YVoltage 1553  
set\_logicalName, YAccelerometer 76  
set\_logicalName, YAnButton 120  
set\_logicalName, YCarbonDioxide 161  
set\_logicalName, YColorLed 195  
set\_logicalName, YCompass 238  
set\_logicalName, YCurrent 281  
set\_logicalName, YDataLogger 317  
set\_logicalName, YDigitalIO 399  
set\_logicalName, YDisplay 451  
set\_logicalName, YDualPower 516  
set\_logicalName, YFiles 549  
set\_logicalName, YGenericSensor 594  
set\_logicalName, YGyro 652  
set\_logicalName, YHubPort 684  
set\_logicalName, YHumidity 724  
set\_logicalName, YLed 756  
set\_logicalName, YLightSensor 799  
set\_logicalName, YMagnetometer 845  
set\_logicalName, YModule 897  
set\_logicalName, YNetwork 955  
set\_logicalName, YOsControl 987  
set\_logicalName, YPower 1032  
set\_logicalName, YPressure 1075  
set\_logicalName, YPwmOutput 1118  
set\_logicalName, YPwmPowerSource 1146  
set\_logicalName, YQt 1187  
set\_logicalName, YRealTimeClock 1219  
set\_logicalName, YRefFrame 1259  
set\_logicalName, YRelay 1296  
set\_logicalName, YSensor 1341  
set\_logicalName, YServo 1379  
set\_logicalName, YTemperature 1424  
set\_logicalName, YTilt 1468  
set\_logicalName, YVoc 1511  
set\_logicalName, YVoltage 1554  
set\_logicalName, YVSource 1591  
set\_logicalName, YWakeUpMonitor 1625  
set\_logicalName, YWakeUpSchedule 1667  
set\_logicalName, YWatchdog 1714  
set\_logicalName, YWireless 1755  
set\_lowestValue, YAccelerometer 77  
set\_lowestValue, YCarbonDioxide 162  
set\_lowestValue, YCompass 239  
set\_lowestValue, YCurrent 282  
set\_lowestValue, YGenericSensor 595  
set\_lowestValue, YGyro 653  
set\_lowestValue, YHumidity 725  
set\_lowestValue, YLightSensor 800  
set\_lowestValue, YMagnetometer 846  
set\_lowestValue, YPower 1033  
set\_lowestValue, YPressure 1076

set_lowestValue, YQt	1188	set_resolution, YCurrent	284
set_lowestValue, YSensor	1342	set_resolution, YGenericSensor	597
set_lowestValue, YTemperature	1425	set_resolution, YGyro	655
set_lowestValue, YTilt	1469	set_resolution, YHumidity	727
set_lowestValue, YVoc	1512	set_resolution, YLightSensor	802
set_lowestValue, YVoltage	1555	set_resolution, YMagnetometer	848
set_luminosity, YLed	757	set_resolution, YPower	1035
set_luminosity, YModule	898	set_resolution, YPressure	1078
set_maxTimeOnStateA, YRelay	1297	set_resolution, YQt	1190
set_maxTimeOnStateA, YWatchdog	1715	set_resolution, YSensor	1344
set_maxTimeOnStateB, YRelay	1298	set_resolution, YTemperature	1427
set_maxTimeOnStateB, YWatchdog	1716	set_resolution, YTilt	1471
set_minutes, YWakeUpSchedule	1668	set_resolution, YVoc	1514
set_minutesA, YWakeUpSchedule	1669	set_resolution, YVoltage	1557
set_minutesB, YWakeUpSchedule	1670	set_rgbColor, YColorLed	196
set_monthDays, YWakeUpSchedule	1671	set_rgbColorAtPowerOn, YColorLed	197
set_months, YWakeUpSchedule	1672	set_running, YWatchdog	1718
set_mountPosition, YRefFrame	1260	set_secondaryDNS, YNetwork	957
set_neutral, YServo	1380	set_sensitivity, YAnButton	121
set_nextWakeUp, YWakeUpMonitor	1626	set_sensorType, YTemperature	1428
set_orientation, YDisplay	452	set_signalRange, YGenericSensor	598
set_output, YRelay	1299	set_sleepCountdown, YWakeUpMonitor	1628
set_output, YWatchdog	1717	set_startupSeq, YDisplay	453
set_outputVoltage, YDigitalIO	400	set_state, YRelay	1300
set_period, YPwmOutput	1119	set_state, YWatchdog	1719
set_portDirection, YDigitalIO	401	set_stateAtPowerOn, YRelay	1301
set_portOpenDrain, YDigitalIO	402	set_stateAtPowerOn, YWatchdog	1720
set_portPolarity, YDigitalIO	403	set_timeUTC, YDataLogger	319
set_portState, YDigitalIO	404	set_triggerDelay, YWatchdog	1721
set_position, YServo	1381	set_triggerDuration, YWatchdog	1722
set_positionAtPowerOn, YServo	1382	set_unit, YGenericSensor	599
set_power, YLed	758	set_unixTime, YRealTimeClock	1220
set_powerControl, YDualPower	517	set_usbBandwidth, YModule	899
set_powerDuration, YWakeUpMonitor	1627	set_userData, YAccelerometer	80
set_powerMode, YPwmPowerSource	1147	set_userData, YAnButton	122
set_primaryDNS, YNetwork	956	set_userData, YCarbonDioxide	165
set_pulseDuration, YPwmOutput	1120	set_userData, YColorLed	198
set_range, YServo	1383	set_userData, YCompass	242
set_recording, YDataLogger	318	set_userData, YCurrent	285
set_reportFrequency, YAccelerometer	78	set_userData, YDataLogger	320
set_reportFrequency, YCarbonDioxide	163	set_userData, YDigitalIO	405
set_reportFrequency, YCompass	240	set_userData, YDisplay	454
set_reportFrequency, YCurrent	283	set_userData, YDualPower	518
set_reportFrequency, YGenericSensor	596	set_userData, YFiles	550
set_reportFrequency, YGyro	654	set_userData, YGenericSensor	600
set_reportFrequency, YHumidity	726	set_userData, YGyro	656
set_reportFrequency, YLightSensor	801	set_userData, YHubPort	685
set_reportFrequency, YMagnetometer	847	set_userData, YHumidity	728
set_reportFrequency, YPower	1034	set_userData, YLed	759
set_reportFrequency, YPressure	1077	set_userData, YLightSensor	803
set_reportFrequency, YQt	1189	set_userData, YMagnetometer	849
set_reportFrequency, YSensor	1343	set_userData, YModule	900
set_reportFrequency, YTemperature	1426	set_userData, YNetwork	958
set_reportFrequency, YTilt	1470	set_userData, YOsControl	988
set_reportFrequency, YVoc	1513	set_userData, YPower	1036
set_reportFrequency, YVoltage	1556	set_userData, YPressure	1079
set_resolution, YAccelerometer	79	set_userData, YPwmOutput	1121
set_resolution, YCarbonDioxide	164	set_userData, YPwmPowerSource	1148
set_resolution, YCompass	241	set_userData, YQt	1191



- set\_userdata, YRealTimeClock 1221
- set\_userdata, YRefFrame 1261
- set\_userdata, YRelay 1302
- set\_userdata, YSensor 1345
- set\_userdata, YServo 1384
- set\_userdata, YTemperature 1429
- set\_userdata, YTilt 1472
- set\_userdata, YVoc 1515
- set\_userdata, YVoltage 1558
- set\_userdata, YVSource 1592
- set\_userdata, YWakeUpMonitor 1629
- set\_userdata, YWakeUpSchedule 1673
- set\_userdata, YWatchdog 1723
- set\_userdata, YWireless 1756
- set\_userPassword, YNetwork 959
- set\_utcOffset, YRealTimeClock 1222
- set\_valueInterval, YDataRun 331
- set\_valueRange, YGenericSensor 601
- set\_voltage, YVSource 1593
- set\_weekDays, YWakeUpSchedule 1674
- set\_wwwWatchdogDelay, YNetwork 960
- setAntialiasingMode, YDisplayLayer 485
- setConsoleBackground, YDisplayLayer 486
- setConsoleMargins, YDisplayLayer 487
- setConsoleWordWrap, YDisplayLayer 488
- SetDelegate, YAPI 29
- setLayerPosition, YDisplayLayer 489
- SetTimeout, YAPI 30
- shutdown, YOsControl 989
- Sleep, YAPI 31
- sleep, YWakeUpMonitor 1630
- sleepFor, YWakeUpMonitor 1631
- sleepUntil, YWakeUpMonitor 1632
- Source 1561
- start3DCalibration, YRefFrame 1262
- stopSequence, YDisplay 455
- swapLayerContent, YDisplay 456

## T

- Temperature 1387
- Temps 1194
- Tension 1561
- Tilt 1432
- toggle\_bitState, YDigitalIO 406
- triggerFirmwareUpdate, YModule 901
- TriggerHubDiscovery, YAPI 32
- Type 1305

## U

- unhide, YDisplayLayer 490
- UnregisterHub, YAPI 33
- UpdateDeviceList, YAPI 34
- UpdateDeviceList\_async, YAPI 35
- upload, YDisplay 457
- upload, YFiles 551
- useDHCP, YNetwork 961
- useStaticIP, YNetwork 962

## V

- Valeur 852
- Voltage 1518
- voltageMove, YVSource 1594

## W

- wait\_async, YAccelerometer 81
- wait\_async, YAnButton 123
- wait\_async, YCarbonDioxide 166
- wait\_async, YColorLed 199
- wait\_async, YCompass 243
- wait\_async, YCurrent 286
- wait\_async, YDataLogger 321
- wait\_async, YDigitalIO 407
- wait\_async, YDisplay 458
- wait\_async, YDualPower 519
- wait\_async, YFiles 552
- wait\_async, YGenericSensor 602
- wait\_async, YGyro 657
- wait\_async, YHubPort 686
- wait\_async, YHumidity 729
- wait\_async, YLed 760
- wait\_async, YLightSensor 804
- wait\_async, YMagnetometer 850
- wait\_async, YModule 902
- wait\_async, YNetwork 963
- wait\_async, YOsControl 990
- wait\_async, YPower 1037
- wait\_async, YPressure 1080
- wait\_async, YPwmOutput 1122
- wait\_async, YPwmPowerSource 1149
- wait\_async, YQt 1192
- wait\_async, YRealTimeClock 1223
- wait\_async, YRefFrame 1263
- wait\_async, YRelay 1303
- wait\_async, YSensor 1346
- wait\_async, YServo 1385
- wait\_async, YTemperature 1430
- wait\_async, YTilt 1473
- wait\_async, YVoc 1516
- wait\_async, YVoltage 1559
- wait\_async, YVSource 1595
- wait\_async, YWakeUpMonitor 1633
- wait\_async, YWakeUpSchedule 1675
- wait\_async, YWatchdog 1724
- wait\_async, YWireless 1757
- wakeUp, YWakeUpMonitor 1634
- WakeUpMonitor 1597
- WakeUpSchedule 1636
- Watchdog 1677
- Wireless 1726

## Y

- YAccelerometer 39-81
- YAnButton 85-123
- YAPI 26-35

YCarbonDioxide 127-166	yFirstCarbonDioxide 128
yCheckLogicalName 12	yFirstColorLed 170
YColorLed 169-199	yFirstCompass 204
YCompass 203-243	yFirstCurrent 248
YCurrent 247-286	yFirstDataLogger 290
YDataLogger 289-321	yFirstDigitalIO 364
YDataRun 323-331	yFirstDisplay 412
YDataSet 334-344	yFirstDualPower 494
YDataStream 347-359	yFirstFiles 523
YDigitalIO 363-407	yFirstGenericSensor 557
yDisableExceptions 13	yFirstGyro 607
YDisplay 411-458	yFirstHubPort 661
YDisplayLayer 461-490	yFirstHumidity 691
YDualPower 493-519	yFirstLed 733
yEnableExceptions 14	yFirstLightSensor 765
yEnableUSBHost 15	yFirstMagnetometer 809
YFiles 522-552	yFirstModule 861
yFindAccelerometer 39	yFirstNetwork 908
yFindAnButton 85	yFirstOsControl 967
yFindCarbonDioxide 127	yFirstPower 995
yFindColorLed 169	yFirstPressure 1042
yFindCompass 203	yFirstPwmOutput 1085
yFindCurrent 247	yFirstPwmPowerSource 1126
yFindDataLogger 289	yFirstQt 1154
yFindDigitalIO 363	yFirstRealTimeClock 1196
yFindDisplay 411	yFirstRefFrame 1228
yFindDualPower 493	yFirstRelay 1268
yFindFiles 522	yFirstSensor 1308
yFindGenericSensor 556	yFirstServo 1351
yFindGyro 606	yFirstTemperature 1390
yFindHubPort 660	yFirstTilt 1435
yFindHumidity 690	yFirstVoc 1478
yFindLed 732	yFirstVoltage 1521
yFindLightSensor 764	yFirstVSource 1563
yFindMagnetometer 808	yFirstWakeUpMonitor 1600
yFindModule 860	yFirstWakeUpSchedule 1639
yFindNetwork 907	yFirstWatchdog 1680
yFindOsControl 966	yFirstWireless 1728
yFindPower 994	yFreeAPI 16
yFindPressure 1041	YGenericSensor 556-602
yFindPwmOutput 1084	yGetAPIVersion 17
yFindPwmPowerSource 1125	yGetTickCount 18
yFindQt 1153	YGyro 606-657
yFindRealTimeClock 1195	yHandleEvents 19
yFindRefFrame 1227	YHubPort 660-686
yFindRelay 1267	YHumidity 690-729
yFindSensor 1307	yInitAPI 20
yFindServo 1350	YLed 732-760
yFindTemperature 1389	YLightSensor 764-804
yFindTilt 1434	YMagnetometer 808-850
yFindVoc 1477	YMeasure 852-856
yFindVoltage 1520	YModule 860-902
yFindVSource 1562	YNetwork 907-963
yFindWakeUpMonitor 1599	Yocto-Demo 3
yFindWakeUpSchedule 1638	Yocto-hub 659
yFindWatchdog 1679	YOsControl 966-990
yFindWireless 1727	YPower 994-1037
yFirstAccelerometer 40	yPreregisterHub 21
yFirstAnButton 86	YPressure 1041-1080

YPwmOutput 1084-1122  
YPwmPowerSource 1125-1149  
YQt 1153-1192  
YRealTimeClock 1195-1223  
YRefFrame 1227-1263  
yRegisterDeviceArrivalCallback 22  
yRegisterDeviceRemovalCallback 23  
yRegisterHub 24  
yRegisterHubDiscoveryCallback 26  
yRegisterLogFunction 27  
YRelay 1267-1303  
ySelectArchitecture 28  
YSensor 1307-1346  
YServo 1350-1385  
ySetDelegate 29

ySetTimeout 30  
ySleep 31  
YTemperature 1389-1430  
YTilt 1434-1473  
yTriggerHubDiscovery 32  
yUnregisterHub 33  
yUpdateDeviceList 34  
yUpdateDeviceList\_async 35  
YVoc 1477-1516  
YVoltage 1520-1559  
YVSource 1562-1595  
YWakeUpMonitor 1599-1634  
YWakeUpSchedule 1638-1675  
YWatchdog 1679-1724  
YWireless 1727-1757