



Référence de l'API JAVA

Table des matières

1. Introduction	1
2. Utilisation du Yocto-Demo en Java	3
2.1. Préparation	3
2.2. Contrôle de la fonction Led	3
2.3. Contrôle de la partie module	5
2.4. Gestion des erreurs	7
Blueprint	10
3. Reference	10
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
Index	1759

1. Introduction

Ce manuel est votre référence pour l'utilisation de la librairie Java de Yoctopuce pour interfacer 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 Java

Java est un langage orienté objet développé par Sun Microsystem. Son principal avantage est la portabilité, mais cette portabilité a un coût. Java fait une telle abstraction des couches matérielles qu'il est très difficile d'interagir directement avec elles. C'est pourquoi l'API java standard de Yoctopuce ne fonctionne pas en natif: elle doit passer par l'intermédiaire d'un VirtualHub pour pouvoir communiquer avec les modules Yoctopuce.

2.1. Préparation

Connectez vous sur le site de Yoctopuce et téléchargez les éléments suivants:

- La librairie de programmation pour Java¹
- Le programme VirtualHub² pour Windows, Mac OS X ou Linux selon l'OS que vous utilisez

La librairie est disponible en fichier sources, mais elle aussi disponible sous la forme d'un fichier jar. Branchez vos modules, Décompressez les fichiers de la librairie dans un répertoire de votre choix. Lancez le programme VirtualHub, et vous pouvez commencer vos premiers test. Vous n'avez pas besoin d'installer de driver.

Afin de les garder simples, tous les exemples fournis dans cette documentation sont des applications consoles. Il va de soit que que le fonctionnement des librairies est strictement identiques si vous les intégrez dans une application dotée d'une interface graphique.

2.2. Contrôle de la fonction Led

Il suffit de quelques lignes de code pour piloter un Yocto-Demo. Voici le squelette d'un fragment de code Java qui utilise la fonction Led.

```
[...]  
  
// On récupère l'objet représentant le module (ici connecté en local sur USB)  
YAPI.RegisterHub("127.0.0.1");  
led = YLed.FindLed("YCTOPOC1-123456.led");  
  
//Pour gérer le hot-plug, on vérifie que le module est là  
if (led.isOnline())  
{ //Use led.set_power()  
    ...  
}
```

¹ www.yoctopuce.com/FR/libraries.php

² www.yoctopuce.com/FR/virtualhub.php

```
}
[...]
```

Voyons maintenant en détail ce que font ces quelques lignes.

YAPI.RegisterHub

La fonction `YAPI.RegisterHub` initialise l'API de Yoctopuce en indiquant où les modules doivent être recherchés. Le paramètre est l'adresse du virtual hub capable de voir les modules. Si l'initialisation se passe mal, une exception sera générée.

YLed.FindLed

La fonction `YLed.FindLed`, 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):

```
led = YLed.FindLed("YCTOPOC1-123456.led")
led = YLed.FindLed("YCTOPOC1-123456.MaFonction")
led = YLed.FindLed("MonModule.led")
led = YLed.FindLed("MonModule.MaFonction")
led = YLed.FindLed("MaFonction")
```

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

isOnline

La méthode `YLed.isOnline()` de l'objet renvoyé par `FindLed` 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 `YLed.FindLed` permet d'allumer et d'éteindre la led. L'argument est `YLed.POWER_ON` ou `YLed.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.

Un exemple réel

Lancez votre environnement java et ouvrez le projet correspondant, fourni dans le répertoire **Exemples/Doc-GettingStarted-Yocto-Demo** de la librairie Yoctopuce.

Vous reconnaîtrez dans cet exemple l'utilisation des fonctions expliquées ci-dessus, cette fois utilisées avec le décorum nécessaire à en faire un petit programme d'exemple concret.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import com.yoctopuce.YoctoAPI.*;

/**
 *
 * @author yocto
 */
public class Demo {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try {
            // setup the API to use local VirtualHub
            YAPI.RegisterHub("127.0.0.1");
        } catch (YAPI_Exception ex) {
            System.out.println("Cannot contact VirtualHub on 127.0.0.1 (" +

```



```

ex.getLocalizedMessage() + "));
    System.out.println("Ensure that the VirtualHub application is running");
    System.exit(1);
}

YLed led;
if (args.length > 0) {
    led = YLed.FindLed(args[0]);
} else {
    led = YLed.FirstLed();
    if (led == null) {
        System.out.println("No module connected (check USB cable)");
        System.exit(1);
    }
}

try {
    System.out.println("Switch led ON");
    led.set_power(YLed.POWER_ON);
    YAPI.Sleep(1000);
    System.out.println("Switch led OFF");
    led.set_power(YLed.POWER_OFF);
} catch (YAPI_Exception ex) {
    System.out.println("Module "+led.describe()+" not connected (check
identification and USB cable)");
}

YAPI.FreeAPI();
}
}

```

2.3. 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 com.yoctopuce.YoctoAPI.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Demo {

    public static void main(String[] args)
    {
        try {
            // setup the API to use local VirtualHub
            YAPI.RegisterHub("127.0.0.1");
        } catch (YAPI_Exception ex) {
            System.out.println("Cannot contact VirtualHub on 127.0.0.1 (" +
ex.getLocalizedMessage() + "));
            System.out.println("Ensure that the VirtualHub application is running");
            System.exit(1);
        }
        System.out.println("usage: demo [serial or logical name] [ON/OFF]");

        YModule module;
        if (args.length == 0) {
            module = YModule.FirstModule();
            if (module == null) {
                System.out.println("No module connected (check USB cable)");
                System.exit(1);
            }
        } else {
            module = YModule.FindModule(args[0]); // use serial or logical name
        }

        try {
            if (args.length > 1) {
                if (args[1].equalsIgnoreCase("ON")) {
                    module.setBeacon(YModule.BEACON_ON);
                } else {

```

```

        module.setBeacon(YModule.BEACON_OFF);
    }
}
System.out.println("serial:      " + module.get_serialNumber());
System.out.println("logical name: " + module.get_logicalName());
System.out.println("luminosity:   " + module.get_luminosity());
if (module.get_beacon() == YModule.BEACON_ON) {
    System.out.println("beacon:      ON");
} else {
    System.out.println("beacon:      OFF");
}
System.out.println("upTime:      " + module.get_upTime() / 1000 + " sec");
System.out.println("USB current:  " + module.get_usbCurrent() + " mA");
System.out.println("logs:\n" + module.get_lastLogs());
} catch (YAPI_Exception ex) {
    System.out.println(args[1] + " not connected (check identification and USB
cable)");
}
YAPI.FreeAPI();
}
}

```

Chaque propriété xxx du module peut être lue grâce à une méthode du type `YModule.get_xxxx()`, et les propriétés qui se sont pas en lecture seule peuvent être modifiées à l'aide de la méthode `YModule.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 `YModule.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 `YModule.saveToFlash()`. Inversement il est possible de forcer le module à oublier ses réglages courants en utilisant la méthode `YModule.revertFromFlash()`. Ce petit exemple ci-dessous vous permet changer le nom logique d'un module.

```

import com.yoctopuce.YoctoAPI.*;

public class Demo {

    public static void main(String[] args)
    {
        try {
            // setup the API to use local VirtualHub
            YAPI.RegisterHub("127.0.0.1");
        } catch (YAPI_Exception ex) {
            System.out.println("Cannot contact VirtualHub on 127.0.0.1 (" +
ex.getLocalizedMessage() + ")");
            System.out.println("Ensure that the VirtualHub application is running");
            System.exit(1);
        }

        if (args.length != 2) {
            System.out.println("usage: demo <serial or logical name> <new logical name>");
            System.exit(1);
        }

        YModule m;
        String newname;

        m = YModule.FindModule(args[0]); // use serial or logical name

        try {
            newname = args[1];
            if (!YAPI.CheckLogicalName(newname))
            {
                System.out.println("Invalid name (" + newname + ")");
                System.exit(1);
            }

            m.set_logicalName(newname);
        }
    }
}

```

```

        m.saveToFlash(); // do not forget this

        System.out.println("Module: serial= " + m.get_serialNumber());
        System.out.println(" / name= " + m.get_logicalName());
    } catch (YAPI_Exception ex) {
        System.out.println("Module " + args[0] + "not connected (check identification
and USB cable)");
        System.out.println(ex.getMessage());
        System.exit(1);
    }

    YAPI.FreeAPI();
}
}

```

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 `YModule.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 `YModule.yFirstModule()` qui renvoie le premier module trouvé, il suffit ensuite d'appeler la méthode `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 modules connectés

```

import com.yoctopuce.YoctoAPI.*;

public class Demo {

    public static void main(String[] args)
    {
        try {
            // setup the API to use local VirtualHub
            YAPI.RegisterHub("127.0.0.1");
        } catch (YAPI_Exception ex) {
            System.out.println("Cannot contact VirtualHub on 127.0.0.1 (" +
ex.getLocalizedMessage() + ")");
            System.out.println("Ensure that the VirtualHub application is running");
            System.exit(1);
        }

        System.out.println("Device list");
        YModule module = YModule.FirstModule();
        while (module != null) {
            try {
                System.out.println(module.get_serialNumber() + " (" +
module.get_productName() + ")");
            } catch (YAPI_Exception ex) {
                break;
            }
            module = module.nextModule();
        }

        YAPI.FreeAPI();
    }
}

```

2.4. 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

vosre 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.

Dans l'API java, le traitement d'erreur est implémenté au moyen d'exceptions. Vous devrez donc intercepter et traiter correctement ces exceptions si vous souhaitez avoir un projet fiable qui ne crashera pas des que vous débrancherez un module.

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><script type='text/javascript' src='yocto_api.js'></script></code>
nodejs	<code>var yoctolib = require('yoctolib'); var YAPI = yoctolib.YAPI; 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()**YAPI****yCheckLogicalName()****YAPI.CheckLogicalName()**

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

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

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() yDisableExceptions()

YAPI

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

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

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()

YAPI

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

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

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()**YAPI****yEnableUSBHost()**`YAPI.EnableUSBHost ()`

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

Libère la mémoire dynamique utilisée par la librairie Yoctopuce.

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

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()**YAPI****yGetAPIVersion()** **YAPI.GetAPIVersion()**

Retourne la version de la librairie Yoctopuce utilisée.

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

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()**YAPI****yGetTickCount()****YAPI.GetTickCount()**

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

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

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()

YAPI

yHandleEvents()YAPI.HandleEvents()

Maintient la communication de la librairie avec les modules Yoctopuce.

js	function yHandleEvents (errmsg)
nodejs	function HandleEvents (errmsg)
php	function yHandleEvents (&\$errmsg)
cpp	YRETCODE yHandleEvents (string& errmsg)
m	YRETCODE yHandleEvents (NSError** errmsg)
pas	function yHandleEvents (var errmsg : string): integer
vb	function yHandleEvents (ByRef errmsg As String) As YRETCODE
cs	YRETCODE HandleEvents (ref string errmsg)
java	int HandleEvents ()
py	def HandleEvents (errmsg =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()**YAPI****yInitAPI()**`YAPI.InitAPI()`

Initialise la librairie de programmation de Yoctopuce explicitement.

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

Il n'est pas indispensable d'appeler `yInitAPI()`, 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()**YAPI****yPreregisterHub()**`YAPI.PreregisterHub()`

Alternative plus tolérante à RegisterHub().

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

Cette fonction a le même but et la même paramètres 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()****YAPI.RegisterDeviceArrivalCallback()**

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

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

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

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

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

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()**YAPI****yRegisterHub()** **YAPI.RegisterHub()**

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()**YAPI****yRegisterHubDiscoveryCallback()****YAPI.RegisterHubDiscoveryCallback()**

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

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

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

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

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

Utile pour déboguer le fonctionnement de l'API.

Paramètres :

logfun une procédure 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() ySetDelegate()

YAPI

(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.

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

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()** `YAPI.Sleep()`

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

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

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()

YAPI.TriggerHubDiscovery()

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()

YAPI

yUnregisterHub()YAPI.UnregisterHub()

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

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

Paramètres :

url une chaîne de caractères contenant "usb" ou

YAPI.UpdateDeviceList()**YAPI****yUpdateDeviceList()** **YAPI.UpdateDeviceList()**

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

js	function yUpdateDeviceList (errmsg)
nodejs	function UpdateDeviceList (errmsg)
php	function yUpdateDeviceList (&\$errmsg)
cpp	YRETCODE yUpdateDeviceList (string& errmsg)
m	YRETCODE yUpdateDeviceList (NSError** errmsg)
pas	function yUpdateDeviceList (var errmsg : string): integer
vb	function yUpdateDeviceList (ByRef errmsg As String) As YRETCODE
cs	YRETCODE UpdateDeviceList (ref string errmsg)
java	int UpdateDeviceList ()
py	def UpdateDeviceList (errmsg =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	<script type='text/javascript' src='yocto_accelerometer.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YAccelerometer = yoctolib.YAccelerometer;
php	require_once('yocto_accelerometer.php');
c++	#include "yocto_accelerometer.h"
m	#import "yocto_accelerometer.h"
pas	uses yocto_accelerometer;
vb	yocto_accelerometer.vb
cs	yocto_accelerometer.cs
java	import com.yoctopuce.YoctoAPI.YAccelerometer;
py	from yocto_accelerometer import *

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()**YAccelerometer****yFindAccelerometer()****YAccelerometer.FindAccelerometer()**

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

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

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()**YAccelerometer****yFirstAccelerometer()****YAccelerometer.FirstAccelerometer()**

Commence l'énumération des accéléromètres accessibles par la librairie.

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

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()**YAccelerometer****accelerometer.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 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.

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()**YAccelerometer****accelerometer.describe()**

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

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

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 l'accéléromètre (ex:
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

accelerometer→**get_advertisedValue()****YAccelerometer****accelerometer**→**advertisedValue()****accelerometer.get_advertisedValue()**

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

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

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.get_currentRawValue()**

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

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

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.get_currentValue()

Retourne la valeur actuelle de l'accélération.

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

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.get_errorMessage()**

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

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

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()**YAccelerometer****accelerometer→errorType()****accelerometer.get_errorType()**

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

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

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()**YAccelerometer****accelerometer→friendlyName()****accelerometer.get_friendlyName()**

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

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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()**
accelerometer→**hardwareId()**
accelerometer.get_hardwareId()

YAccelerometer

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

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

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.get_highestValue()**

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

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YAccelerometer target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique de l'accéléromètre.

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

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()**
accelerometer→**lowestValue()**
accelerometer.get_lowestValue()

YAccelerometer

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

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

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.get_module()**

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

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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

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.

accelerometer→get_reportFrequency()**YAccelerometer****accelerometer→reportFrequency()****accelerometer.get_reportFrequency()**

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

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

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()**YAccelerometer****accelerometer→resolution()****accelerometer.get_resolution()**

Retourne la résolution des valeurs mesurées.

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

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.get_unit()`

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

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

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.get_userdata()**

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

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

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.

accelerometer→**get_xValue()****YAccelerometer****accelerometer**→**xValue()****accelerometer.get_xValue()**

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

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

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.get_yValue()**

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

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

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.get_zValue()**

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

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

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()**YAccelerometer****accelerometer.isOnline()**

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

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

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 (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.

accelerometer→loadCalibrationPoints()**YAccelerometer****accelerometer.loadCalibrationPoints()**

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	YAccelerometer 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.

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()**

YAccelerometer

accelerometer.nextAccelerometer()

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

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

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

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'object fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

accelerometer→registerValueCallback()**YAccelerometer****accelerometer.registerValueCallback()**

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 (YAccelerometerValueCallback callback)
m	-(int) registerValueCallback : (YAccelerometerValueCallback) callback
pas	function registerValueCallback (callback : TYAccelerometerValueCallback): 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.

accelerometer→set_highestValue()**YAccelerometer****accelerometer→setHighestValue()****accelerometer.set_highestValue()**

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	YAccelerometer 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.

accelerometer→**set_logFrequency()****YAccelerometer****accelerometer**→**setLogFrequency()****accelerometer.set_logFrequency()**

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	YAccelerometer 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.

accelerometer→set_logicalName()**YAccelerometer****accelerometer→setLogicalName()****accelerometer.set_logicalName()**

Modifie le nom logique de l'accéléromètre.

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	YAccelerometer 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

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

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.set_reportFrequency()**

Modifie la fréquence de notification périodique des valeurs mesurées.

js	function set_reportFrequency (newval)
node.js	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	YAccelerometer 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.

accelerometer→**set_resolution()****YAccelerometer****accelerometer**→**setResolution()****accelerometer.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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

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><script type='text/javascript' src='yocto_anbutton.js'></script></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()**YAnButton****yFindAnButton()****YAnButton.FindAnButton()**

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

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

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()**YAnButton****yFirstAnButton()****YAnButton.FirstAnButton()**

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

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

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

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()****YAnButton****anbutton**→**advertisedValue()****anbutton.get_advertisedValue()**

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

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

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()****YAnButton****anbutton**→**analogCalibration()****anbutton.get_analogCalibration()**

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

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

Retourne :

soit Y_ANALOGCALIBRATION_OFF, soit Y_ANALOGCALIBRATION_ON

En cas d'erreur, déclenche une exception ou retourne Y_ANALOGCALIBRATION_INVALID.

anbutton→**get_calibratedValue()****YAnButton****anbutton**→**calibratedValue()****anbutton.get_calibratedValue()**

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

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

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.get_calibrationMax()**

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

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

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.get_calibrationMin()**

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

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

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.get_errorMessage()**

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

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

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()****YAnButton****anbutton**→**errorType()****anbutton.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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()**YAnButton****anbutton→functionDescriptor()****anbutton.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

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

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.get_isPressed()**

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

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

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()**YAnButton****anbutton→lastTimePressed()****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).

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

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()****YAnButton****anbutton**→**lastTimeReleased()****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).

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

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.get_logicalName()**

Retourne le nom logique de l'entrée analogique.

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

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.get_module()**

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

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

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 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.

anbutton→**get_pulseCounter()****YAnButton****anbutton**→**pulseCounter()****anbutton.get_pulseCounter()**

Retourne la valeur du compteur d'impulsions.

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

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.get_pulseTimer()**

Retourne le timer du compteur d'impulsions (ms)

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

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.getRawValue()**

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

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

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.get_sensitivity()**

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

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

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

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

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)
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.

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()**YAnButton****anbutton.nextAnButton()**

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

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

Retourne :

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

anbutton→registerValueCallback()**YAnButton****anbutton.registerValueCallback()**

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 (YAnButtonValueCallback callback)
m	-(int) registerValueCallback : (YAnButtonValueCallback) callback
pas	function registerValueCallback (callback : TYAnButtonValueCallback): 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.

anbutton→resetCounter()**YAnButton****anbutton.resetCounter()**

réinitialise le compteur d'impulsions et son timer

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

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.set_analogCalibration()**

Enclenche ou déclenche le procédure de calibration.

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

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.set_calibrationMax()**

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

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

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.set_calibrationMin()**

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

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

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()**YAnButton****anbutton→setLogicalName()****anbutton.set_logicalName()**

Modifie le nom logique de l'entrée analogique.

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	YAnButton 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 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()****anbutton**→**setSensitivity()****anbutton.set_sensitivity()**

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

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

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.set_userdata()**

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

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

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()**YCarbonDioxide****yFindCarbonDioxide()****YCarbonDioxide.FindCarbonDioxide()**

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

js	function yFindCarbonDioxide (func)
nodejs	function FindCarbonDioxide (func)
php	function yFindCarbonDioxide (\$func)
c++	YCarbonDioxide* yFindCarbonDioxide (const string& func)
m	YCarbonDioxide* yFindCarbonDioxide (NSString* func)
pas	function yFindCarbonDioxide (func : string): TYCarbonDioxide
vb	function yFindCarbonDioxide (ByVal func As String) As YCarbonDioxide
cs	YCarbonDioxide FindCarbonDioxide (string func)
java	YCarbonDioxide FindCarbonDioxide (String func)
py	def FindCarbonDioxide (func)

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()**YCarbonDioxide****yFirstCarbonDioxide()****YCarbonDioxide.FirstCarbonDioxide()**

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

js	function yFirstCarbonDioxide ()
nodejs	function FirstCarbonDioxide ()
php	function yFirstCarbonDioxide ()
cpp	YCarbonDioxide* yFirstCarbonDioxide ()
m	YCarbonDioxide* yFirstCarbonDioxide ()
pas	function yFirstCarbonDioxide (): TYCarbonDioxide
vb	function yFirstCarbonDioxide () As YCarbonDioxide
cs	YCarbonDioxide FirstCarbonDioxide ()
java	YCarbonDioxide FirstCarbonDioxide ()
py	def FirstCarbonDioxide ()

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()**YCarbonDioxide****carbondioxide.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 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.

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()**YCarbonDioxide****carbondioxide.describe()**

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

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

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(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 capteur de CO2 (ex:
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

carbondioxide→get_advertisedValue()

YCarbonDioxide

carbondioxide→advertisedValue()

carbondioxide.get_advertisedValue()

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

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

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()**YCarbonDioxide****carbondioxide→currentRawValue()****carbondioxide.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YCarbonDioxide target get_currentRawValue

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()**
carbondioxide→**currentValue()**
carbondioxide.get_currentValue()

YCarbonDioxide

Retourne la valeur actuelle du taux de CO2.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YCarbonDioxide target get_currentValue

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()**YCarbonDioxide****carbondioxide→errorMessage()****carbondioxide.get_errorMessage()**

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

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

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()

YCarbonDioxide

carbondioxide→errorType()

carbondioxide.get_errorType()

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

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

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()**YCarbonDioxide****carbondioxide→friendlyName()****carbondioxide.get_friendlyName()**

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

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

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.get_functionDescriptor()**

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

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

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()****YCarbonDioxide****carbondioxide**→**functionId()****carbondioxide.get_functionId()**

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

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

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.get_hardwareId()

YCarbonDioxide

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

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

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()**YCarbonDioxide****carbondioxide→highestValue()****carbondioxide.get_highestValue()**

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

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YCarbonDioxide target get_highestValue

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()**YCarbonDioxide****carbondioxide→logFrequency()****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.

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

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.get_logicalName()

YCarbonDioxide

Retourne le nom logique du capteur de CO2.

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

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.get_lowestValue()**

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

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YCarbonDioxide target get_lowestValue

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.get_module()**

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

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

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()**YCarbonDioxide****carbondioxide→recordedData()****carbondioxide.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YCarbonDioxide target get_recordedData startTime endTime

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.

carbondioxide→get_reportFrequency()**YCarbonDioxide****carbondioxide→reportFrequency()****carbondioxide.get_reportFrequency()**

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

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

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()****YCarbonDioxide****carbondioxide**→**resolution()****carbondioxide.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YCarbonDioxide target get_resolution

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.get_unit()**

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

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

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.get_userdata()**

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

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

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.

carbondioxide→isOnline() carbondioxide.isOnline()

YCarbonDioxide

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

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

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 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.

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()**YCarbonDioxide****carbondioxide.nextCarbonDioxide()**

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

js	function nextCarbonDioxide ()
nodejs	function nextCarbonDioxide ()
php	function nextCarbonDioxide ()
cpp	YCarbonDioxide * nextCarbonDioxide ()
m	-(YCarbonDioxide*) nextCarbonDioxide
pas	function nextCarbonDioxide (): TYCarbonDioxide
vb	function nextCarbonDioxide () As YCarbonDioxide
cs	YCarbonDioxide nextCarbonDioxide ()
java	YCarbonDioxide nextCarbonDioxide ()
py	def nextCarbonDioxide ()

Retourne :

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

carbondioxide→registerTimedReportCallback()**YCarbonDioxide****carbondioxide.registerTimedReportCallback(
)**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YCarbonDioxideTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YCarbonDioxideTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYCarbonDioxideTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YCarbonDioxide****carbondioxide.registerValueCallback()**

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 (YCarbonDioxideValueCallback callback)
m	-(int) registerValueCallback : (YCarbonDioxideValueCallback) callback
pas	function registerValueCallback (callback : TYCarbonDioxideValueCallback): 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.

carbondioxide→**set_highestValue()****YCarbonDioxide****carbondioxide**→**setHighestValue()****carbondioxide.set_highestValue()**

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	YCarbonDioxide 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.

carbondioxide→**set_logFrequency()****YCarbonDioxide****carbondioxide**→**setLogFrequency()****carbondioxide.set_logFrequency()**

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	YCarbonDioxide 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.

carbondioxide→**set_logicalName()**
carbondioxide→**setLogicalName()**
carbondioxide.set_logicalName()

YCarbonDioxide

Modifie le nom logique du capteur de CO2.

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	YCarbonDioxide 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 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()
 carbondioxide→setLowestValue()
 carbondioxide.set_lowestValue()

YCarbonDioxide

Modifie la mémoire de valeur minimale observée.

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

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()
carbondioxide→setReportFrequency()
carbondioxide.set_reportFrequency()

YCarbonDioxide

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()****YCarbonDioxide****carbondioxide**→**setResolution()****carbondioxide.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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()****carbondioxide**→**setUserData()****carbondioxide.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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><script type='text/javascript' src='yocto_colorled.js'></script></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()**YColorLed****yFindColorLed()****YColorLed.FindColorLed()**

Permet de retrouver une led RGB d'après un identifiant donné.

js	function yFindColorLed (func)
nodejs	function FindColorLed (func)
php	function yFindColorLed (\$func)
c++	YColorLed* yFindColorLed (const string& func)
m	YColorLed* yFindColorLed (NSString* func)
pas	function yFindColorLed (func : string): TYColorLed
vb	function yFindColorLed (ByVal func As String) As YColorLed
cs	YColorLed FindColorLed (string func)
java	YColorLed FindColorLed (String func)
py	def FindColorLed (func)

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()**YColorLed****yFirstColorLed()****YColorLed.FirstColorLed()**

Commence l'énumération des leds RGB accessibles par la librairie.

js	function yFirstColorLed ()
nodejs	function FirstColorLed ()
php	function yFirstColorLed ()
cpp	YColorLed* yFirstColorLed ()
m	YColorLed* yFirstColorLed ()
pas	function yFirstColorLed (): TYColorLed
vb	function yFirstColorLed () As YColorLed
cs	YColorLed FirstColorLed ()
java	YColorLed FirstColorLed ()
py	def FirstColorLed ()

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.

colorled→**describe()****colorled.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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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()****YColorLed****colorled**→**advertisedValue()****colorled.get_advertisedValue()**

Retourne la valeur courante de la led RGB (pas plus de 6 caractères).

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

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.

colorled→**get_errorMessage()****YColorLed****colorled**→**errorMessage()****colorled.errorMessage()**

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

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

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()****YColorLed****colorled**→**errorType()****colorled.errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global de la led RGB au format `NOM_MODULE.NOM_FONCTION`.

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

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()****YColorLed****colorled**→**functionDescriptor()****colorled.get_functionDescriptor()**

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

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

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

colorled→**get_functionId()**

YColorLed

colorled→**functionId()****colorled.get_functionId()**

Retourne l'identifiant matériel de la led RGB, sans référence au module.

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

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**.**get_hardwareId()**

Retourne l'identifiant matériel unique de la led RGB au format `SERIAL.FUNCTIONID`.

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

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.get_hslColor()**

Retourne la couleur HSL courante de la led.

js	function get_hslColor ()
nodejs	function get_hslColor ()
php	function get_hslColor ()
cpp	int get_hslColor ()
m	-(int) hslColor
pas	function get_hslColor (): LongInt
vb	function get_hslColor () As Integer
cs	int get_hslColor ()
java	int get_hslColor ()
py	def get_hslColor ()
cmd	YColorLed target get_hslColor

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()**
colorled→**logicalName()**
colorled.get_logicalName()

YColorLed

Retourne le nom logique de la led RGB.

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

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.get_module()**

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

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

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()**
colorled→**module_async()**

YColorLed

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.

colorled→**get_rgbColor()****YColorLed****colorled**→**rgbColor()****colorled.get_rgbColor()**

Retourne la couleur RGB courante de la led.

js	function get_rgbColor ()
nodejs	function get_rgbColor ()
php	function get_rgbColor ()
cpp	int get_rgbColor ()
m	-(int) rgbColor
pas	function get_rgbColor (): LongInt
vb	function get_rgbColor () As Integer
cs	int get_rgbColor ()
java	int get_rgbColor ()
py	def get_rgbColor ()
cmd	YColorLed target get_rgbColor

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()****YColorLed****colorled**→**rgbColorAtPowerOn()****colorled.get_rgbColorAtPowerOn()**

Retourne la couleur configurée pour être affichage à l'allumage du module.

js	function get_rgbColorAtPowerOn ()
nodejs	function get_rgbColorAtPowerOn ()
php	function get_rgbColorAtPowerOn ()
cpp	int get_rgbColorAtPowerOn ()
m	-(int) rgbColorAtPowerOn
pas	function get_rgbColorAtPowerOn (): LongInt
vb	function get_rgbColorAtPowerOn () As Integer
cs	int get_rgbColorAtPowerOn ()
java	int get_rgbColorAtPowerOn ()
py	def get_rgbColorAtPowerOn ()
cmd	YColorLed target get_rgbColorAtPowerOn

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.

colored→**get_userdata()**

YColorLed

colored→**userData()****colored.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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 hslMove (hsl_target , ms_duration)
nodejs	function hslMove (hsl_target , ms_duration)
php	function hslMove (\$hsl_target , \$ms_duration)
cpp	int hslMove (int hsl_target , int ms_duration)
m	-(int) hslMove : (int) hsl_target : (int) ms_duration
pas	function hslMove (hsl_target : LongInt, ms_duration : LongInt): integer
vb	function hslMove (ByVal hsl_target As Integer, ByVal ms_duration As Integer) As Integer
cs	int hslMove (int hsl_target , int ms_duration)
java	int hslMove (int hsl_target , int ms_duration)
py	def hslMove (hsl_target , ms_duration)
cmd	YColorLed target hslMove hsl_target ms_duration

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

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-qu'el 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 load (msValidity)
nodejs	function load (msValidity)
php	function load (\$msValidity)
c++	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.

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-quel à la fonction de callback

Retourne :

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

colorled→**nextColorLed()****YColorLed****colorled.nextColorLed()**

Continue l'énumération des leds RGB commencée à l'aide de `yFirstColorLed()`.

js	function nextColorLed ()
nodejs	function nextColorLed ()
php	function nextColorLed ()
cpp	YColorLed * nextColorLed ()
m	-(YColorLed*) nextColorLed
pas	function nextColorLed (): TYColorLed
vb	function nextColorLed () As YColorLed
cs	YColorLed nextColorLed ()
java	YColorLed nextColorLed ()
py	def nextColorLed ()

Retourne :

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

colorled→registerValueCallback()**YColorLed****colorled.registerValueCallback()**

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 (YColorLedValueCallback callback)
m	-(int) registerValueCallback : (YColorLedValueCallback) callback
pas	function registerValueCallback (callback : TYColorLedValueCallback): 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.

colorled→**rgbMove()****colorled.rgbMove()****YColorLed**

Effectue une transition continue dans l'espace RGB entre la couleur courante et une nouvelle couleur.

js	function rgbMove (rgb_target , ms_duration)
nodejs	function rgbMove (rgb_target , ms_duration)
php	function rgbMove (\$rgb_target , \$ms_duration)
cpp	int rgbMove (int rgb_target , int ms_duration)
m	-(int) rgbMove : (int) rgb_target : (int) ms_duration
pas	function rgbMove (rgb_target : LongInt, ms_duration : LongInt): integer
vb	function rgbMove (ByVal rgb_target As Integer, ByVal ms_duration As Integer) As Integer
cs	int rgbMove (int rgb_target , int ms_duration)
java	int rgbMove (int rgb_target , int ms_duration)
py	def rgbMove (rgb_target , ms_duration)
cmd	YColorLed target rgbMove rgb_target ms_duration

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.set_hslColor()**

Modifie la couleur courante de la led, en utilisant une couleur HSL spécifiée.

js	function set_hslColor (newval)
nodejs	function set_hslColor (newval)
php	function set_hslColor (\$newval)
cpp	int set_hslColor (int newval)
m	-(int) setHslColor : (int) newval
pas	function set_hslColor (newval : LongInt): integer
vb	function set_hslColor (ByVal newval As Integer) As Integer
cs	int set_hslColor (int newval)
java	int set_hslColor (int newval)
py	def set_hslColor (newval)
cmd	YColorLed target set_hslColor newval

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.set_logicalName()**

Modifie le nom logique de la led RGB.

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	YColorLed 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 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.set_rgbColor()**

Modifie la couleur courante de la led, en utilisant une couleur RGB (Rouge Vert Bleu).

js	function set_rgbColor (newval)
nodejs	function set_rgbColor (newval)
php	function set_rgbColor (\$newval)
cpp	int set_rgbColor (int newval)
m	-(int) setRgbColor : (int) newval
pas	function set_rgbColor (newval : LongInt): integer
vb	function set_rgbColor (ByVal newval As Integer) As Integer
cs	int set_rgbColor (int newval)
java	int set_rgbColor (int newval)
py	def set_rgbColor (newval)
cmd	YColorLed target set_rgbColor newval

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.set_rgbColorAtPowerOn()**

Modifie la couleur que la led va afficher spontanément à l'allumage du module.

js	function set_rgbColorAtPowerOn (newval)
nodejs	function set_rgbColorAtPowerOn (newval)
php	function set_rgbColorAtPowerOn (\$newval)
cpp	int set_rgbColorAtPowerOn (int newval)
m	-(int) setRgbColorAtPowerOn : (int) newval
pas	function set_rgbColorAtPowerOn (newval : LongInt): integer
vb	function set_rgbColorAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_rgbColorAtPowerOn (int newval)
java	int set_rgbColorAtPowerOn (int newval)
py	def set_rgbColorAtPowerOn (newval)
cmd	YColorLed target set_rgbColorAtPowerOn newval

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.set_userdata()**

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

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_DESCRIPTOR` 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()**YCompass****yFindCompass()****YCompass.FindCompass()**

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

js	function yFindCompass (func)
nodejs	function FindCompass (func)
php	function yFindCompass (\$func)
cpp	YCompass* yFindCompass (const string& func)
m	YCompass* yFindCompass (NSString* func)
pas	function yFindCompass (func : string): TYCompass
vb	function yFindCompass (ByVal func As String) As YCompass
cs	YCompass FindCompass (string func)
java	YCompass FindCompass (String func)
py	def FindCompass (func)

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()**YCompass****yFirstCompass()****YCompass.FirstCompass()**

Commence l'énumération des compas accessibles par la librairie.

js	function yFirstCompass ()
nodejs	function FirstCompass ()
php	function yFirstCompass ()
cpp	YCompass* yFirstCompass ()
m	YCompass* yFirstCompass ()
pas	function yFirstCompass (): TYCompass
vb	function yFirstCompass () As YCompass
cs	YCompass FirstCompass ()
java	YCompass FirstCompass ()
py	def FirstCompass ()

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()**YCompass****compass.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 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.

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

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()****YCompass****compass**→**advertisedValue()****compass.get_advertisedValue()**

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

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

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()****YCompass****compass**→**currentRawValue()****compass.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YCompass target get_currentRawValue

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.get_currentValue()**

Retourne la valeur actuelle du cap relatif.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YCompass target get_currentValue

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

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()****compass.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du compas au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du compas, sans référence au module.

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

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()****compass**→**hardwareId()****compass.hardwareId()**

Retourne l'identifiant matériel unique du compas au format SERIAL.FUNCTIONID.

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

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.get_highestValue()**

Retourne la valeur maximale observée pour le cap relatif depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YCompass target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du compas.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour le cap relatif depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YCompass target get_lowestValue

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()****YCompass****compass**→**magneticHeading()****compass.get_magneticHeading()**

Retourne la direction du nord magnétique, indépendamment du cap configuré.

js	function get_magneticHeading ()
nodejs	function get_magneticHeading ()
php	function get_magneticHeading ()
cpp	double get_magneticHeading ()
m	-(double) magneticHeading
pas	function get_magneticHeading (): double
vb	function get_magneticHeading () As Double
cs	double get_magneticHeading ()
java	double get_magneticHeading ()
py	def get_magneticHeading ()
cmd	YCompass target get_magneticHeading

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.get_module()`

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

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

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 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-qu'el 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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
c++	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YCompass target get_recordedData startTime endTime

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()****YCompass****compass**→**reportFrequency()****compass.get_reportFrequency()**

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

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
c++	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YCompass target get_reportFrequency

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.get_resolution()`

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YCompass target get_resolution

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.get_unit()**

Retourne l'unité dans laquelle le cap relatif est exprimée.

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

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()****YCompass****compass**→**userData()****compass.get_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.

compass→**isOnline()**`compass.isOnline()`**YCompass**

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

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

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 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.

compass→loadCalibrationPoints()**YCompass****compass.loadCalibrationPoints()**

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 nextCompass ()
nodejs	function nextCompass ()
php	function nextCompass ()
cpp	YCompass * nextCompass ()
m	-(YCompass*) nextCompass
pas	function nextCompass (): TYCompass
vb	function nextCompass () As YCompass
cs	YCompass nextCompass ()
java	YCompass nextCompass ()
py	def nextCompass ()

Retourne :

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

compass→registerTimedReportCallback()**YCompass****compass.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YCompassTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YCompassTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYCompassTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YCompass****compass.registerValueCallback()**

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 (YCompassValueCallback callback)
m	-(int) registerValueCallback : (YCompassValueCallback) callback
pas	function registerValueCallback (callback : TYCompassValueCallback): 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.

compass→**set_highestValue()****YCompass****compass**→**setHighestValue()****compass.set_highestValue()**

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	YCompass 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.

compass→**set_logFrequency()****compass**→**setLogFrequency()****compass.set_logFrequency()**

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	YCompass 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.

compass→**set_logicalName()**
compass→**setLogicalName()**
compass.set_logicalName()

YCompass

Modifie le nom logique du compas.

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	YCompass 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

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

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.set_reportFrequency()

YCompass

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	YCompass 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.

compass→**set_resolution()****compass**→**setResolution()****compass.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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 `SERIAL.FUNCTIONID`.

current→**get_highestValue()**

Retourne la valeur maximale observée pour le courant.

current→**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.

current→**get_logicalName()**

Retourne le nom logique du capteur de courant.

current→**get_lowestValue()**

Retourne la valeur minimale observée pour le courant.

current→**get_module()**

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

current→**get_module_async(callback, context)**

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

current→**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.

current→**get_reportFrequency()**

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

current→**get_resolution()**

Retourne la résolution des valeurs mesurées.

current→**get_unit()**

Retourne l'unité dans laquelle le courant est exprimée.

current→**get_userData()**

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

current→**isOnline()**

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

current→**isOnline_async(callback, context)**

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

current→**load(msValidity)**

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

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

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

current→**load_async(msValidity, callback, context)**

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

current→**nextCurrent()**

Continue l'énumération des capteurs de courant commencée à l'aide de `yFirstCurrent()`.

current→**registerTimedReportCallback(callback)**

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

current→**registerValueCallback(callback)**

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

current→**set_highestValue(newval)**

Modifie la mémoire de valeur maximale observée pour le courant.

current→**set_logFrequency(newval)**

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()**YCurrent****yFindCurrent()****YCurrent.FindCurrent()**

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

js	function yFindCurrent (func)
nodejs	function FindCurrent (func)
php	function yFindCurrent (\$func)
cpp	YCurrent* yFindCurrent (const string& func)
m	YCurrent* yFindCurrent (NSString* func)
pas	function yFindCurrent (func : string): TYCurrent
vb	function yFindCurrent (ByVal func As String) As YCurrent
cs	YCurrent FindCurrent (string func)
java	YCurrent FindCurrent (String func)
py	def FindCurrent (func)

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()**YCurrent****yFirstCurrent()**`YCurrent.FirstCurrent()`

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

js	function yFirstCurrent ()
nodejs	function FirstCurrent ()
php	function yFirstCurrent ()
cpp	YCurrent* yFirstCurrent ()
m	YCurrent* yFirstCurrent ()
pas	function yFirstCurrent (): TYCurrent
vb	function yFirstCurrent () As YCurrent
cs	YCurrent FirstCurrent ()
java	YCurrent FirstCurrent ()
py	def FirstCurrent ()

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()**YCurrent****current.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 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.

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

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.get_advertisedValue()**

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

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

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()****YCurrent****current**→**currentRawValue()****current.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YCurrent target get_currentRawValue

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.get_currentValue()**

Retourne la valeur instantanée du courant.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YCurrent target get_currentValue

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**.**get_errorMessage()**

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

js	function get_errorMessage()
nodejs	function get_errorMessage()
php	function get_errorMessage()
c++	string get_errorMessage()
m	-(NSString*) errorMessage
pas	function get_errorMessage() : string
vb	function get_errorMessage() As String
cs	string get_errorMessage()
java	String get_errorMessage()
py	def get_errorMessage()

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()****current.get_errorType()**

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

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

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()**
current→**friendlyName()**
current.get_friendlyName()

YCurrent

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

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

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()****YCurrent****current**→**functionDescriptor()****current.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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()**
current→**highestValue()**
current.get_highestValue()

YCurrent

Retourne la valeur maximale observée pour le courant.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YCurrent target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du capteur de courant.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour le courant.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YCurrent target get_lowestValue

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.get_module()`

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

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

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YCurrent target get_recordedData startTime endTime

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.

current→**get_reportFrequency()****YCurrent****current**→**reportFrequency()****current.get_reportFrequency()**

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

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

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()****YCurrent****current**→**resolution()****current.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YCurrent target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle le courant est exprimée.

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

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()****YCurrent****current**→**userData()****current.get_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'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.

current→**isOnline()****current.isOnline()****YCurrent**

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

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

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 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.

current→loadCalibrationPoints()**YCurrent****current.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(&\$rawValues, &\$refValues)</code>
cpp	<code>int loadCalibrationPoints(vector<double>& rawValues, vector<double>& 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<double> rawValues, List<double> refValues)</code>
java	<code>int loadCalibrationPoints(ArrayList<Double> rawValues, ArrayList<Double> refValues)</code>
py	<code>def loadCalibrationPoints(rawValues, refValues)</code>
cmd	<code>YCurrent 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.

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-qu'el 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()`.

js	function nextCurrent ()
nodejs	function nextCurrent ()
php	function nextCurrent ()
cpp	YCurrent * nextCurrent ()
m	-(YCurrent*) nextCurrent
pas	function nextCurrent (): TYCurrent
vb	function nextCurrent () As YCurrent
cs	YCurrent nextCurrent ()
java	YCurrent nextCurrent ()
py	def nextCurrent ()

Retourne :

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

current→registerTimedReportCallback()**YCurrent****current.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YCurrentTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YCurrentTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYCurrentTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YCurrent****current.registerValueCallback()**

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 (YCurrentValueCallback callback)
m	-(int) registerValueCallback : (YCurrentValueCallback) callback
pas	function registerValueCallback (callback : TYCurrentValueCallback): 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.

current→**set_highestValue()****YCurrent****current**→**setHighestValue()****current.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour le courant.

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	YCurrent target set_highestValue newval

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()****YCurrent****current**→**setLogFrequency()****current.set_logFrequency()**

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	YCurrent 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.

current→**set_logicalName()****YCurrent****current**→**setLogicalName()****current.set_logicalName()**

Modifie le nom logique du capteur de courant.

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	YCurrent 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour le courant.

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

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()****YCurrent****current**→**setReportFrequency()****current.set_reportFrequency()**

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	YCurrent 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.

current→**set_resolution()****YCurrent****current**→**setResolution()****current.set_resolution()**

Modifie la résolution des valeurs mesurées.

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

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.set_userdata()**

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

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

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()**YDataLogger****yFindDataLogger()****YDataLogger.FindDataLogger()**

Permet de retrouver un enregistreur de données d'après un identifiant donné.

js	function yFindDataLogger (func)
nodejs	function FindDataLogger (func)
php	function yFindDataLogger (\$func)
c++	YDataLogger* yFindDataLogger (string func)
m	+(YDataLogger*) yFindDataLogger : (NSString*) func
pas	function yFindDataLogger (func : string): TYDataLogger
vb	function yFindDataLogger (ByVal func As String) As YDataLogger
cs	YDataLogger FindDataLogger (string func)
java	YDataLogger FindDataLogger (String func)
py	def FindDataLogger (func)

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()**YDataLogger****yFirstDataLogger()****YDataLogger.FirstDataLogger()**

Commence l'énumération des enregistreurs de données accessibles par la librairie.

js	function yFirstDataLogger ()
nodejs	function FirstDataLogger ()
php	function yFirstDataLogger ()
cpp	YDataLogger* yFirstDataLogger ()
m	YDataLogger* yFirstDataLogger ()
pas	function yFirstDataLogger (): TYDataLogger
vb	function yFirstDataLogger () As YDataLogger
cs	YDataLogger FirstDataLogger ()
java	YDataLogger FirstDataLogger ()
py	def FirstDataLogger ()

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

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()**YDataLogger****datalogger.forgetAllDataStreams()**

Efface tout l'historique des mesures de l'enregistreur de données.

js	function forgetAllDataStreams ()
nodejs	function forgetAllDataStreams ()
php	function forgetAllDataStreams ()
cpp	int forgetAllDataStreams ()
m	-(int) forgetAllDataStreams
pas	function forgetAllDataStreams (): LongInt
vb	function forgetAllDataStreams () As Integer
cs	int forgetAllDataStreams ()
java	int forgetAllDataStreams ()
py	def forgetAllDataStreams ()
cmd	YDataLogger target forgetAllDataStreams

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()****YDataLogger****datalogger**→**advertisedValue()****datalogger.get_advertisedValue()**

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

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

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.get_autoStart()**

Retourne le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

js	function get_autoStart ()
nodejs	function get_autoStart ()
php	function get_autoStart ()
cpp	Y_AUTOSTART_enum get_autoStart ()
m	-(Y_AUTOSTART_enum) autoStart
pas	function get_autoStart (): Integer
vb	function get_autoStart () As Integer
cs	int get_autoStart ()
java	int get_autoStart ()
py	def get_autoStart ()
cmd	YDataLogger target get_autoStart

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()****YDataLogger****datalogger**→**currentRunIndex()****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.

js	function get_currentRunIndex ()
nodejs	function get_currentRunIndex ()
php	function get_currentRunIndex ()
cpp	int get_currentRunIndex ()
m	-(int) currentRunIndex
pas	function get_currentRunIndex (): LongInt
vb	function get_currentRunIndex () As Integer
cs	int get_currentRunIndex ()
java	int get_currentRunIndex ()
py	def get_currentRunIndex ()
cmd	YDataLogger target get_currentRunIndex

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.get_dataSets()**

Retourne une liste d'objets YDataSet permettant de récupérer toutes les mesures stockées par l'enregistreur de données.

js	function get_dataSets ()
nodejs	function get_dataSets ()
php	function get_dataSets ()
cpp	vector<YDataSet> get_dataSets ()
m	-(NSMutableArray*) dataSets
pas	function get_dataSets (): TYDataSetArray
vb	function get_dataSets () As List
cs	List<YDataSet> get_dataSets ()
java	ArrayList<YDataSet> get_dataSets ()
py	def get_dataSets ()
cmd	YDataLogger target get_dataSets

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.get_dataStreams()**

Construit une liste de toutes les séquences de mesures mémorisées par l'enregistreur (ancienne méthode).

js	function get_dataStreams (v)
nodejs	function get_dataStreams (v)
php	function get_dataStreams (&\$v)
cpp	int get_dataStreams ()
m	-(int) dataStreams : (NSArray**) v
pas	function get_dataStreams (v : Tlist): integer
vb	procedure get_dataStreams (ByVal v As List)
cs	int get_dataStreams (List<YDataStream> v)
java	int get_dataStreams (ArrayList<YDataStream> v)
py	def get_dataStreams (v)

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()**YDataLogger****datalogger→errorMessage()****datalogger.get_errorMessage()**

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

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

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()****datalogger.get_errorType()**

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

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

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()**YDataLogger****datalogger→friendlyName()****datalogger.get_friendlyName()**

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

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

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()****YDataLogger****datalogger**→**functionDescriptor()****datalogger.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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()**datalogger→hardwareId()****datalogger.get_hardwareId()**

Retourne l'identifiant matériel unique de l'enregistreur de données au format SERIAL.FUNCTIONID.

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

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.get_logicalName()**

Retourne le nom logique de l'enregistreur de données.

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

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.get_module()**

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

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

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 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-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.get_recording()**

Retourne l'état d'activation de l'enregistreur de données.

js	function get_recording ()
nodejs	function get_recording ()
php	function get_recording ()
cpp	Y_RECORDING_enum get_recording ()
m	-(Y_RECORDING_enum) recording
pas	function get_recording (): Integer
vb	function get_recording () As Integer
cs	int get_recording ()
java	int get_recording ()
py	def get_recording ()
cmd	YDataLogger target get_recording

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.get_timeUTC()**

Retourne le timestamp Unix de l'heure UTC actuelle, lorsqu'elle est connue.

js	function get_timeUTC ()
nodejs	function get_timeUTC ()
php	function get_timeUTC ()
cpp	s64 get_timeUTC ()
m	-(s64) timeUTC
pas	function get_timeUTC (): int64
vb	function get_timeUTC () As Long
cs	long get_timeUTC ()
java	long get_timeUTC ()
py	def get_timeUTC ()
cmd	YDataLogger target get_timeUTC

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

datalogger→isOnline()`datalogger.isOnline()`**YDataLogger**

Vérifie si le module hébergeant l'enregistreur de données est joignable, sans déclencher d'erreur.

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

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.

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

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-quel à la fonction de callback

Retourne :

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

datalogger→nextDataLogger()**YDataLogger****datalogger.nextDataLogger()**

Continue l'énumération des enregistreurs de données commencée à l'aide de `yFirstDataLogger()`.

js	function nextDataLogger ()
nodejs	function nextDataLogger ()
php	function nextDataLogger ()
cpp	YDataLogger * nextDataLogger ()
m	-(YDataLogger*) nextDataLogger
pas	function nextDataLogger (): TYDataLogger
vb	function nextDataLogger () As YDataLogger
cs	YDataLogger nextDataLogger ()
java	YDataLogger nextDataLogger ()
py	def nextDataLogger ()

Retourne :

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

datalogger→**registerValueCallback()****YDataLogger****datalogger.registerValueCallback()**

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)
c++	int registerValueCallback (YDataLoggerValueCallback callback)
m	-(int) registerValueCallback : (YDataLoggerValueCallback) callback
pas	function registerValueCallback (callback : TYDataLoggerValueCallback): 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.

datalogger→**set_autoStart()****YDataLogger****datalogger**→**setAutoStart()****datalogger.set_autoStart()**

Modifie le mode d'activation automatique de l'enregistreur de données à la mise sous tension.

js	function set_autoStart (newval)
nodejs	function set_autoStart (newval)
php	function set_autoStart (\$newval)
cpp	int set_autoStart (Y_AUTOSTART_enum newval)
m	-(int) setAutoStart : (Y_AUTOSTART_enum) newval
pas	function set_autoStart (newval : Integer): integer
vb	function set_autoStart (ByVal newval As Integer) As Integer
cs	int set_autoStart (int newval)
java	int set_autoStart (int newval)
py	def set_autoStart (newval)
cmd	YDataLogger target set_autoStart newval

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()****datalogger**→**setLogicalName()****datalogger.set_logicalName()**

Modifie le nom logique de l'enregistreur de données.

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	YDataLogger 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 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.set_recording()

YDataLogger

Modifie l'état d'activation de l'enregistreur de données.

js	function set_recording (newval)
nodejs	function set_recording (newval)
php	function set_recording (\$newval)
cpp	int set_recording (Y_RECORDING_enum newval)
m	-(int) setRecording : (Y_RECORDING_enum) newval
pas	function set_recording (newval : Integer): integer
vb	function set_recording (ByVal newval As Integer) As Integer
cs	int set_recording (int newval)
java	int set_recording (int newval)
py	def set_recording (newval)
cmd	YDataLogger target set_recording newval

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()**datalogger→setTimeUTC()****datalogger.set_timeUTC()**

Modifie la référence de temps UTC, afin de l'attacher aux données enregistrées.

js	function set_timeUTC (newval)
nodejs	function set_timeUTC (newval)
php	function set_timeUTC (\$newval)
c++	int set_timeUTC (s64 newval)
m	-(int) setTimeUTC : (s64) newval
pas	function set_timeUTC (newval : int64): integer
vb	function set_timeUTC (ByVal newval As Long) As Integer
cs	int set_timeUTC (long newval)
java	int set_timeUTC (long newval)
py	def set_timeUTC (newval)
cmd	YDataLogger target set_timeUTC newval

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.set_userdata()**

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

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	
datarun→get_averageValue(measureName, pos)	Retourne la valeur moyenne des mesures observées au moment choisi.
datarun→get_duration()	Retourne la durée (en secondes) du Run.
datarun→get_maxValue(measureName, pos)	Retourne la valeur maximale des mesures observées au moment choisi.
datarun→get_measureNames()	Retourne les noms des valeurs mesurées par l'enregistreur de données.
datarun→get_minValue(measureName, pos)	Retourne la valeur minimale des mesures observées au moment choisi.
datarun→get_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).
datarun→get_valueCount()	Retourne le nombre de valeurs accessibles dans ce Run, étant donné l'intervalle de temps choisi entre les valeurs.
datarun→get_valueInterval()	Retourne l'intervalle de temps représenté par chaque valeur de ce run.
datarun→set_valueInterval(valueInterval)	Change l'intervalle de temps représenté par chaque valeur de ce run.

datarun→**get_averageValue()****datarun**→**averageValue()****datarun.get_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()**YDataRun****datarun→duration()**`datarun.get_duration()`

Retourne la durée (en secondes) du Run.

js	function get_duration ()
nodejs	function get_duration ()
php	function get_duration ()
java	long get_duration ()
py	def get_duration ()

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()datarun.get_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()****datarun.get_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()****datarun.get_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()****datarun.get_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()
datarun→valueInterval()
datarun.get_valueInterval()

YDataRun

Retourne l'intervalle de temps représenté par chaque valeur de ce run.

```
js function get_valueInterval( )  
nodejs function get_valueInterval( )  
php function get_valueInterval( )  
java int get_valueInterval( )  
py def get_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.

Retourne :

un entier positif correspondant au nombre de secondes couvertes par chaque valeur représentée dans le Run.

datarun→**set_valueInterval()**

YDataRun

datarun→**setValueInterval()**

datarun.set_valueInterval()

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><script type='text/javascript' src='yocto_api.js'></script></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.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).

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

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.get_functionId()**

Retourne l'identifiant matériel de la fonction qui a effectué les mesures, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
pas	function get_functionId (): string
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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`.

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

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.get_measures()`

Retourne toutes les mesures déjà disponibles pour le DataSet, sous forme d'une liste d'objets YMeasure.

js	function get_measures ()
nodejs	function get_measures ()
php	function get_measures ()
cpp	vector<YMeasure> get_measures ()
m	-(NSMutableArray*) measures
pas	function get_measures (): TYMeasureArray
vb	function get_measures () As List
cs	List<YMeasure> get_measures ()
java	ArrayList<YMeasure> get_measures ()
py	def get_measures ()

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.get_preview()**

Retourne une version résumée des mesures qui pourront être obtenues de ce YDataSet, sous forme d'une liste d'objets YMeasure.

js	function get_preview()
nodejs	function get_preview()
php	function get_preview()
cpp	vector<YMeasure> get_preview()
m	-(NSMutableArray*) preview
pas	function get_preview() : TYMeasureArray
vb	function get_preview() As List
cs	List<YMeasure> get_preview()
java	ArrayList<YMeasure> get_preview()
py	def get_preview()

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.get_progress()**

Retourne l'état d'avancement du chargement des données, sur une échelle de 0 à 100.

js	function get_progress ()
nodejs	function get_progress ()
php	function get_progress ()
cpp	int get_progress ()
m	-(int) progress
pas	function get_progress (): LongInt
vb	function get_progress () As Integer
cs	int get_progress ()
java	int get_progress ()
py	def get_progress ()

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.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).

js	function get_startTimeUTC ()
nodejs	function get_startTimeUTC ()
php	function get_startTimeUTC ()
cpp	s64 get_startTimeUTC ()
m	-(s64) startTimeUTC
pas	function get_startTimeUTC (): int64
vb	function get_startTimeUTC () As Long
cs	long get_startTimeUTC ()
java	long get_startTimeUTC ()
py	def get_startTimeUTC ()

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.get_summary()**

Retourne un objet YMeasure résumant tout le YDataSet.

js	function get_summary ()
nodejs	function get_summary ()
php	function get_summary ()
cpp	YMeasure get_summary ()
m	-(YMeasure*) summary
pas	function get_summary (): TYMeasure
vb	function get_summary () As YMeasure
cs	YMeasure get_summary ()
java	YMeasure get_summary ()
py	def get_summary ()

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.get_unit()**

Retourne l'unité dans laquelle la valeur mesurée est exprimée.

js	function get_unit ()
nodejs	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()

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 loadMore ()
nodejs	function loadMore ()
php	function loadMore ()
cpp	int loadMore ()
m	-(int) loadMore
pas	function loadMore (): LongInt
vb	function loadMore () As Integer
cs	int loadMore ()
java	int loadMore ()
py	def loadMore ()

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><script type='text/javascript' src='yocto_api.js'></script></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.get_averageValue()**

Retourne la moyenne des valeurs observées durant cette séquence.

js	function get_averageValue ()
nodejs	function get_averageValue ()
php	function get_averageValue ()
cpp	double get_averageValue ()
m	-(double) averageValue
pas	function get_averageValue (): double
vb	function get_averageValue () As Double
cs	double get_averageValue ()
java	double get_averageValue ()
py	def get_averageValue ()

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()**YDataStream****datastream→columnCount()****datastream.get_columnCount()**

Retourne le nombre de colonnes de données contenus dans la séquence.

js	function get_columnCount ()
nodejs	function get_columnCount ()
php	function get_columnCount ()
cpp	int get_columnCount ()
m	-(int) columnCount
pas	function get_columnCount (): LongInt
vb	function get_columnCount () As Integer
cs	int get_columnCount ()
java	int get_columnCount ()
py	def get_columnCount ()

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()**YDataStream****datastream→columnNames()****datastream.get_columnNames()**

Retourne le nom (la sémantique) des colonnes de données contenus dans la séquence.

js	function get_columnNames ()
nodejs	function get_columnNames ()
php	function get_columnNames ()
cpp	vector<string> get_columnNames ()
m	-(NSMutableArray*) columnNames
pas	function get_columnNames (): TStringArray
vb	function get_columnNames () As List
cs	List<string> get_columnNames ()
java	ArrayList<String> get_columnNames ()
py	def get_columnNames ()

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_data()

YDataStream

datastream→data()datastream.get_data()

Retourne une mesure unique de la séquence, spécifiée par l'index de l'enregistrement (ligne) et de la mesure (colonne).

js	function get_data (row , col)
nodejs	function get_data (row , col)
php	function get_data (\$row , \$col)
c++	double get_data (int row , int col)
m	-(double) data : (int) row : (int) col
pas	function get_data (row : LongInt, col : LongInt): double
vb	function get_data () As Double
cs	double get_data (int row , int col)
java	double get_data (int row , int col)
py	def get_data (row , col)

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.

Paramètres :

row index de l'enregistrement (ligne)
col index de la mesure (colonne)

Retourne :

un nombre décimal

En cas d'erreur, déclenche une exception ou retourne Y_DATA_INVALID.

datastream→get_dataRows()**YDataStream****datastream→dataRows()****datastream.get_dataRows()**

Retourne toutes les données mesurées contenues dans la séquence, sous forme d'une liste de vecteurs (table bidimensionnelle).

js	function get_dataRows ()
nodejs	function get_dataRows ()
php	function get_dataRows ()
cpp	vector< vector<double> > get_dataRows ()
m	-(NSMutableArray*) dataRows
pas	function get_dataRows (): TDoubleArrayArray
vb	function get_dataRows () As List
cs	List<List<double>> get_dataRows ()
java	ArrayList<ArrayList<Double>> get_dataRows ()
py	def get_dataRows ()

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.get_dataSamplesIntervalMs()**

Retourne le nombre de millisecondes entre chaque mesure de la séquence.

js	function get_dataSamplesIntervalMs ()
nodejs	function get_dataSamplesIntervalMs ()
php	function get_dataSamplesIntervalMs ()
cpp	int get_dataSamplesIntervalMs ()
m	-(int) dataSamplesIntervalMs
pas	function get_dataSamplesIntervalMs (): LongInt
vb	function get_dataSamplesIntervalMs () As Integer
cs	int get_dataSamplesIntervalMs ()
java	int get_dataSamplesIntervalMs ()
py	def get_dataSamplesIntervalMs ()

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.get_duration()**

Retourne la durée approximative de cette séquence, en secondes.

js	function get_duration ()
nodejs	function get_duration ()
php	function get_duration ()
cpp	int get_duration ()
m	-(int) duration
pas	function get_duration (): LongInt
vb	function get_duration () As Integer
cs	int get_duration ()
java	int get_duration ()
py	def get_duration ()

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.get_maxValue()**

Retourne la plus grande valeur observée durant cette séquence.

js	function get_maxValue ()
nodejs	function get_maxValue ()
php	function get_maxValue ()
cpp	double get_maxValue ()
m	-(double) max_value
pas	function get_maxValue (): double
vb	function get_maxValue () As Double
cs	double get_maxValue ()
java	double get_maxValue ()
py	def get_maxValue ()

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.get_minValue()**

Retourne la plus petite valeur observée durant cette séquence.

js	function get_minValue ()
nodejs	function get_minValue ()
php	function get_minValue ()
cpp	double get_minValue ()
m	-(double) minValue
pas	function get_minValue (): double
vb	function get_minValue () As Double
cs	double get_minValue ()
java	double get_minValue ()
py	def get_minValue ()

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.get_rowCount()**

Retourne le nombre d'enregistrement contenus dans la séquence.

js	function get_rowCount ()
nodejs	function get_rowCount ()
php	function get_rowCount ()
cpp	int get_rowCount ()
m	-(int) rowCount
pas	function get_rowCount (): LongInt
vb	function get_rowCount () As Integer
cs	int get_rowCount ()
java	int get_rowCount ()
py	def get_rowCount ()

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.get_runIndex()**

Retourne le numéro de Run de la séquence de données.

js	function get_runIndex ()
nodejs	function get_runIndex ()
php	function get_runIndex ()
cpp	int get_runIndex ()
m	-(int) runIndex
pas	function get_runIndex (): LongInt
vb	function get_runIndex () As Integer
cs	int get_runIndex ()
java	int get_runIndex ()
py	def get_runIndex ()

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.get_startTime()**

Retourne le temps de départ relatif de la séquence (en secondes).

js	function get_startTime ()
nodejs	function get_startTime ()
php	function get_startTime ()
cpp	int get_startTime ()
m	-(int) startTime
pas	function get_startTime (): LongInt
vb	function get_startTime () As Integer
cs	int get_startTime ()
java	int get_startTime ()
py	def get_startTime ()

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.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).

js	function get_startTimeUTC ()
nodejs	function get_startTimeUTC ()
php	function get_startTimeUTC ()
cpp	s64 get_startTimeUTC ()
m	-(s64) startTimeUTC
pas	function get_startTimeUTC (): int64
vb	function get_startTimeUTC () As Long
cs	long get_startTimeUTC ()
java	long get_startTimeUTC ()
py	def get_startTimeUTC ()

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.
digitalio→get_functionId()	Retourne l'identifiant matériel du port d'E/S digital, sans référence au module.
digitalio→get_hardwareId()	Retourne l'identifiant matériel unique du port d'E/S digital au format <code>SERIAL . FUNCTIONID</code> .
digitalio→get_logicalName()	Retourne le nom logique du port d'E/S digital.
digitalio→get_module()	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
digitalio→get_module_async(callback, context)	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
digitalio→get_outputVoltage()	Retourne la source de tension utilisée pour piloter les bits en sortie.
digitalio→get_portDirection()	Retourne la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.
digitalio→get_portOpenDrain()	Retourne le type d'interface électrique de chaque bit du port (bitmap).
digitalio→get_portPolarity()	Retourne la polarité des bits du port (bitmap).
digitalio→get_portSize()	Retourne le nombre de bits implémentés dans le port d'E/S.
digitalio→get_portState()	Retourne l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.
digitalio→get_userData()	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
digitalio→isOnline()	Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.
digitalio→isOnline_async(callback, context)	Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.
digitalio→load(msValidity)	Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.
digitalio→load_async(msValidity, callback, context)	Met en cache les valeurs courantes du port d'E/S digital, avec une durée de validité spécifiée.
digitalio→nextDigitalIO()	Continue l'énumération des ports d'E/S digitaux commencée à l'aide de <code>yFirstDigitalIO()</code> .
digitalio→pulse(bitno, ms_duration)	Déclenche une impulsion de durée spécifiée sur un bit choisi.
digitalio→registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
digitalio→set_bitDirection(bitno, bitdirection)	Change la direction d'un seul bit du port d'E/S.
digitalio→set_bitOpenDrain(bitno, opendrain)	Change le type d'interface électrique d'un seul bit du port d'E/S.
digitalio→set_bitPolarity(bitno, bitpolarity)	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()**YDigitalIO****yFindDigitalIO()****YDigitalIO.FindDigitalIO()**

Permet de retrouver un port d'E/S digital d'après un identifiant donné.

js	function yFindDigitalIO (func)
nodejs	function FindDigitalIO (func)
php	function yFindDigitalIO (\$func)
cpp	YDigitalIO* yFindDigitalIO (const string& func)
m	YDigitalIO* yFindDigitalIO (NSString* func)
pas	function yFindDigitalIO (func : string): TYDigitalIO
vb	function yFindDigitalIO (ByVal func As String) As YDigitalIO
cs	YDigitalIO FindDigitalIO (string func)
java	YDigitalIO FindDigitalIO (String func)
py	def FindDigitalIO (func)

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()**YDigitalIO****yFirstDigitalIO()****YDigitalIO.FirstDigitalIO()**

Commence l'énumération des ports d'E/S digitaux accessibles par la librairie.

js	function yFirstDigitalIO ()
nodejs	function FirstDigitalIO ()
php	function yFirstDigitalIO ()
c++	YDigitalIO* yFirstDigitalIO ()
m	YDigitalIO* yFirstDigitalIO ()
pas	function yFirstDigitalIO (): TYDigitalIO
vb	function yFirstDigitalIO () As YDigitalIO
cs	YDigitalIO FirstDigitalIO ()
java	YDigitalIO FirstDigitalIO ()
py	def FirstDigitalIO ()

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.

digitalio→delayedPulse()

YDigitalIO

digitalio.delayedPulse()

Préprogramme une impulsion de durée spécifiée sur un bit choisi.

js	function delayedPulse (bitno , ms_delay , ms_duration)
nodejs	function delayedPulse (bitno , ms_delay , ms_duration)
php	function delayedPulse (\$bitno , \$ms_delay , \$ms_duration)
c++	int delayedPulse (int bitno , int ms_delay , int ms_duration)
m	-(int) delayedPulse : (int) bitno : (int) ms_delay : (int) ms_duration
pas	function delayedPulse (bitno : LongInt, ms_delay : LongInt, ms_duration : LongInt): LongInt
vb	function delayedPulse () As Integer
cs	int delayedPulse (int bitno , int ms_delay , int ms_duration)
java	int delayedPulse (int bitno , int ms_delay , int ms_duration)
py	def delayedPulse (bitno , ms_delay , ms_duration)
cmd	YDigitalIO target delayedPulse bitno ms_delay 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_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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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()****YDigitalIO****digitalio**→**advertisedValue()****digitalio.get_advertisedValue()**

Retourne la valeur courante du port d'E/S digital (pas plus de 6 caractères).

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

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.get_bitDirection()**

Retourne la direction d'un seul bit du port d'E/S.

js	function get_bitDirection (bitno)
nodejs	function get_bitDirection (bitno)
php	function get_bitDirection (\$bitno)
cpp	int get_bitDirection (int bitno)
m	-(int) bitDirection : (int) bitno
pas	function get_bitDirection (bitno : LongInt): LongInt
vb	function get_bitDirection () As Integer
cs	int get_bitDirection (int bitno)
java	int get_bitDirection (int bitno)
py	def get_bitDirection (bitno)
cmd	YDigitalIO target get_bitDirection bitno

(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.get_bitOpenDrain()

Retourne la direction d'un seul bit du port d'E/S.

js	function get_bitOpenDrain (bitno)
nodejs	function get_bitOpenDrain (bitno)
php	function get_bitOpenDrain (\$bitno)
cpp	int get_bitOpenDrain (int bitno)
m	-(int) bitOpenDrain : (int) bitno
pas	function get_bitOpenDrain (bitno : LongInt): LongInt
vb	function get_bitOpenDrain () As Integer
cs	int get_bitOpenDrain (int bitno)
java	int get_bitOpenDrain (int bitno)
py	def get_bitOpenDrain (bitno)
cmd	YDigitalIO target get_bitOpenDrain bitno

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.get_bitPolarity()**

Retourne la polarité d'un seul bit du port d'E/S.

js	function get_bitPolarity (bitno)
nodejs	function get_bitPolarity (bitno)
php	function get_bitPolarity (\$bitno)
cpp	int get_bitPolarity (int bitno)
m	-(int) bitPolarity : (int) bitno
pas	function get_bitPolarity (bitno : LongInt): LongInt
vb	function get_bitPolarity () As Integer
cs	int get_bitPolarity (int bitno)
java	int get_bitPolarity (int bitno)
py	def get_bitPolarity (bitno)
cmd	YDigitalIO target get_bitPolarity bitno

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()****YDigitalIO****digitalio**→**bitState()****digitalio.get_bitState()**

Retourne l'état d'un seul bit du port d'E/S.

js	function get_bitState (bitno)
nodejs	function get_bitState (bitno)
php	function get_bitState (\$bitno)
c++	int get_bitState (int bitno)
m	-(int) bitState : (int) bitno
pas	function get_bitState (bitno : LongInt): LongInt
vb	function get_bitState () As Integer
cs	int get_bitState (int bitno)
java	int get_bitState (int bitno)
py	def get_bitState (bitno)
cmd	YDigitalIO target get_bitState bitno

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

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()****digitalio.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port d'E/S digital.

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

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.get_friendlyName()**

Retourne un identifiant global du port d'E/S digital au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du port d'E/S digital, sans référence au module.

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

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()****digitalio**→**hardwareId()****digitalio.get_hardwareId()**

Retourne l'identifiant matériel unique du port d'E/S digital au format `SERIAL.FUNCTIONID`.

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

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.get_logicalName()**

Retourne le nom logique du port d'E/S digital.

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

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()****YDigitalIO****digitalio**→**module()****digitalio.get_module()**

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

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

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 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.

digitalio→**get_outputVoltage()****YDigitalIO****digitalio**→**outputVoltage()****digitalio.get_outputVoltage()**

Retourne la source de tension utilisée pour piloter les bits en sortie.

js	function get_outputVoltage ()
nodejs	function get_outputVoltage ()
php	function get_outputVoltage ()
cpp	Y_OUTPUTVOLTAGE_enum get_outputVoltage ()
m	-(Y_OUTPUTVOLTAGE_enum) outputVoltage
pas	function get_outputVoltage (): Integer
vb	function get_outputVoltage () As Integer
cs	int get_outputVoltage ()
java	int get_outputVoltage ()
py	def get_outputVoltage ()
cmd	YDigitalIO target get_outputVoltage

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.get_portDirection()**

Retourne la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.

js	function get_portDirection ()
nodejs	function get_portDirection ()
php	function get_portDirection ()
c++	int get_portDirection ()
m	-(int) portDirection
pas	function get_portDirection (): LongInt
vb	function get_portDirection () As Integer
cs	int get_portDirection ()
java	int get_portDirection ()
py	def get_portDirection ()
cmd	YDigitalIO target get_portDirection

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.get_portOpenDrain()**

Retourne le type d'interface électrique de chaque bit du port (bitmap).

js	function get_portOpenDrain ()
nodejs	function get_portOpenDrain ()
php	function get_portOpenDrain ()
cpp	int get_portOpenDrain ()
m	-(int) portOpenDrain
pas	function get_portOpenDrain (): LongInt
vb	function get_portOpenDrain () As Integer
cs	int get_portOpenDrain ()
java	int get_portOpenDrain ()
py	def get_portOpenDrain ()
cmd	YDigitalIO target get_portOpenDrain

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.get_portPolarity()**

Retourne la polarité des bits du port (bitmap).

js	function get_portPolarity ()
nodejs	function get_portPolarity ()
php	function get_portPolarity ()
cpp	int get_portPolarity ()
m	-(int) portPolarity
pas	function get_portPolarity (): LongInt
vb	function get_portPolarity () As Integer
cs	int get_portPolarity ()
java	int get_portPolarity ()
py	def get_portPolarity ()
cmd	YDigitalIO target get_portPolarity

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.get_portSize()**

Retourne le nombre de bits implémentés dans le port d'E/S.

js	function get_portSize ()
nodejs	function get_portSize ()
php	function get_portSize ()
cpp	int get_portSize ()
m	-(int) portSize
pas	function get_portSize (): LongInt
vb	function get_portSize () As Integer
cs	int get_portSize ()
java	int get_portSize ()
py	def get_portSize ()
cmd	YDigitalIO target get_portSize

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.get_portState()**

Retourne l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.

js	function get_portState ()
nodejs	function get_portState ()
php	function get_portState ()
cpp	int get_portState ()
m	-(int) portState
pas	function get_portState (): LongInt
vb	function get_portState () As Integer
cs	int get_portState ()
java	int get_portState ()
py	def get_portState ()
cmd	YDigitalIO target get_portState

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.get_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.

digitalio→**isOnline()****digitalio.isOnline()****YDigitalIO**

Vérifie si le module hébergeant le port d'E/S digital est joignable, sans déclencher d'erreur.

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

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 (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.

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()****YDigitalIO****digitalio.nextDigitalIO()**

Continue l'énumération des ports d'E/S digitaux commencée à l'aide de `yFirstDigitalIO()`.

js	function nextDigitalIO ()
nodejs	function nextDigitalIO ()
php	function nextDigitalIO ()
cpp	YDigitalIO * nextDigitalIO ()
m	-(YDigitalIO*) nextDigitalIO
pas	function nextDigitalIO (): TYDigitalIO
vb	function nextDigitalIO () As YDigitalIO
cs	YDigitalIO nextDigitalIO ()
java	YDigitalIO nextDigitalIO ()
py	def nextDigitalIO ()

Retourne :

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

digitalio→pulse()`digitalio.pulse()`**YDigitalIO**

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()**YDigitalIO****digitalio.registerValueCallback()**

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 (YDigitalIOValueCallback callback)
m	-(int) registerValueCallback : (YDigitalIOValueCallback) callback
pas	function registerValueCallback (callback : TYDigitalIOValueCallback): 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.

digitalio→**set_bitDirection()****digitalio**→**setBitDirection()****digitalio.set_bitDirection()**

Change la direction d'un seul bit du port d'E/S.

js	function set_bitDirection (bitno , bitdirection)
nodejs	function set_bitDirection (bitno , bitdirection)
php	function set_bitDirection (\$bitno , \$bitdirection)
cpp	int set_bitDirection (int bitno , int bitdirection)
m	-(int) setBitDirection : (int) bitno : (int) bitdirection
pas	function set_bitDirection (bitno : LongInt, bitdirection : LongInt): LongInt
vb	function set_bitDirection () As Integer
cs	int set_bitDirection (int bitno , int bitdirection)
java	int set_bitDirection (int bitno , int bitdirection)
py	def set_bitDirection (bitno , bitdirection)
cmd	YDigitalIO target set_bitDirection bitno bitdirection

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()****YDigitalIO****digitalio**→**setBitOpenDrain()****digitalio.set_bitOpenDrain()**

Change le type d'interface électrique d'un seul bit du port d'E/S.

js	function set_bitOpenDrain (bitno , opendrain)
nodejs	function set_bitOpenDrain (bitno , opendrain)
php	function set_bitOpenDrain (\$bitno , \$opendrain)
cpp	int set_bitOpenDrain (int bitno , int opendrain)
m	-(int) setBitOpenDrain : (int) bitno : (int) opendrain
pas	function set_bitOpenDrain (bitno : LongInt, opendrain : LongInt): LongInt
vb	function set_bitOpenDrain () As Integer
cs	int set_bitOpenDrain (int bitno , int opendrain)
java	int set_bitOpenDrain (int bitno , int opendrain)
py	def set_bitOpenDrain (bitno , opendrain)
cmd	YDigitalIO target set_bitOpenDrain bitno opendrain

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()****digitalio**→**setBitPolarity()****digitalio.set_bitPolarity()**

Change la polarité d'un seul bit du port d'E/S.

js	function set_bitPolarity (bitno , bitpolarity)
nodejs	function set_bitPolarity (bitno , bitpolarity)
php	function set_bitPolarity (\$bitno , \$bitpolarity)
cpp	int set_bitPolarity (int bitno , int bitpolarity)
m	-(int) setBitPolarity : (int) bitno : (int) bitpolarity
pas	function set_bitPolarity (bitno : LongInt, bitpolarity : LongInt): LongInt
vb	function set_bitPolarity () As Integer
cs	int set_bitPolarity (int bitno , int bitpolarity)
java	int set_bitPolarity (int bitno , int bitpolarity)
py	def set_bitPolarity (bitno , bitpolarity)
cmd	YDigitalIO target set_bitPolarity bitno bitpolarity

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.set_bitState()**

Change l'état d'un seul bit du port d'E/S.

js	function set_bitState (bitno , bitstate)
nodejs	function set_bitState (bitno , bitstate)
php	function set_bitState (\$bitno , \$bitstate)
cpp	int set_bitState (int bitno , int bitstate)
m	-(int) setBitState : (int) bitno : (int) bitstate
pas	function set_bitState (bitno : LongInt, bitstate : LongInt): LongInt
vb	function set_bitState () As Integer
cs	int set_bitState (int bitno , int bitstate)
java	int set_bitState (int bitno , int bitstate)
py	def set_bitState (bitno , bitstate)
cmd	YDigitalIO target set_bitState bitno bitstate

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.set_logicalName()**

Modifie le nom logique du port d'E/S digital.

js	function set_logicalName (newval)
nodejs	function set_logicalName (newval)
php	function set_logicalName (\$newval)
c++	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	YDigitalIO 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 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()**YDigitalIO****digitalio→setOutputVoltage()****digitalio.set_outputVoltage()**

Modifie la source de tension utilisée pour piloter les bits en sortie.

js	function set_outputVoltage (newval)
nodejs	function set_outputVoltage (newval)
php	function set_outputVoltage (\$newval)
cpp	int set_outputVoltage (Y_OUTPUTVOLTAGE_enum newval)
m	-(int) setOutputVoltage : (Y_OUTPUTVOLTAGE_enum) newval
pas	function set_outputVoltage (newval : Integer): integer
vb	function set_outputVoltage (ByVal newval As Integer) As Integer
cs	int set_outputVoltage (int newval)
java	int set_outputVoltage (int newval)
py	def set_outputVoltage (newval)
cmd	YDigitalIO target set_outputVoltage newval

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.set_portDirection()**

Modifie la direction des bits du port (bitmap): 0 représente un bit en entrée, 1 représente un bit en sortie.

js	function set_portDirection (newval)
nodejs	function set_portDirection (newval)
php	function set_portDirection (\$newval)
cpp	int set_portDirection (int newval)
m	-(int) setPortDirection : (int) newval
pas	function set_portDirection (newval : LongInt): integer
vb	function set_portDirection (ByVal newval As Integer) As Integer
cs	int set_portDirection (int newval)
java	int set_portDirection (int newval)
py	def set_portDirection (newval)
cmd	YDigitalIO target set_portDirection newval

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.set_portOpenDrain()**

Modifie le type d'interface électrique de chaque bit du port (bitmap).

js	function set_portOpenDrain (newval)
nodejs	function set_portOpenDrain (newval)
php	function set_portOpenDrain (\$newval)
cpp	int set_portOpenDrain (int newval)
m	-(int) setPortOpenDrain : (int) newval
pas	function set_portOpenDrain (newval : LongInt): integer
vb	function set_portOpenDrain (ByVal newval As Integer) As Integer
cs	int set_portOpenDrain (int newval)
java	int set_portOpenDrain (int newval)
py	def set_portOpenDrain (newval)
cmd	YDigitalIO target set_portOpenDrain newval

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()****digitalio**→**setPortPolarity()****digitalio.set_portPolarity()**

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 set_portPolarity (newval)
nodejs	function set_portPolarity (newval)
php	function set_portPolarity (\$newval)
cpp	int set_portPolarity (int newval)
m	-(int) setPortPolarity : (int) newval
pas	function set_portPolarity (newval : LongInt): integer
vb	function set_portPolarity (ByVal newval As Integer) As Integer
cs	int set_portPolarity (int newval)
java	int set_portPolarity (int newval)
py	def set_portPolarity (newval)
cmd	YDigitalIO target set_portPolarity newval

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.set_portState()**

Modifie l'état du port d'E/S digital: le bit 0 représente l'input 0 et ainsi de suite.

js	function set_portState (newval)
nodejs	function set_portState (newval)
php	function set_portState (\$newval)
cpp	int set_portState (int newval)
m	-(int) setPortState : (int) newval
pas	function set_portState (newval : LongInt): integer
vb	function set_portState (ByVal newval As Integer) As Integer
cs	int set_portState (int newval)
java	int set_portState (int newval)
py	def set_portState (newval)
cmd	YDigitalIO target set_portState newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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()**YDigitalIO****digitalio.toggle_bitState()**

Inverse l'état d'un seul bit du port d'E/S.

js	function toggle_bitState (bitno)
nodejs	function toggle_bitState (bitno)
php	function toggle_bitState (\$bitno)
cpp	int toggle_bitState (int bitno)
m	-(int) toggle_bitState : (int) bitno
pas	function toggle_bitState (bitno : LongInt): LongInt
vb	function toggle_bitState () As Integer
cs	int toggle_bitState (int bitno)
java	int toggle_bitState (int bitno)
py	def toggle_bitState (bitno)
cmd	YDigitalIO target toggle_bitState bitno

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()**YDisplay****yFindDisplay()****YDisplay.FindDisplay()**

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

js	function yFindDisplay (func)
nodejs	function FindDisplay (func)
php	function yFindDisplay (\$func)
c++	YDisplay* yFindDisplay (string func)
m	+(YDisplay*) yFindDisplay : (NSString*) func
pas	function yFindDisplay (func : string): TYDisplay
vb	function yFindDisplay (ByVal func As String) As YDisplay
cs	YDisplay FindDisplay (string func)
java	YDisplay FindDisplay (String func)
py	def FindDisplay (func)

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()**YDisplay****yFirstDisplay()****YDisplay.FirstDisplay()**

Commence l'énumération des écran accessibles par la librairie.

js	function yFirstDisplay ()
nodejs	function FirstDisplay ()
php	function yFirstDisplay ()
cpp	YDisplay* yFirstDisplay ()
m	YDisplay* yFirstDisplay ()
pas	function yFirstDisplay (): TYDisplay
vb	function yFirstDisplay () As YDisplay
cs	YDisplay FirstDisplay ()
java	YDisplay FirstDisplay ()
py	def FirstDisplay ()

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 copyLayerContent (srcLayerId , dstLayerId)
nodejs	function copyLayerContent (srcLayerId , dstLayerId)
php	function copyLayerContent (\$ srcLayerId , \$ dstLayerId)
c++	int copyLayerContent (int srcLayerId , int dstLayerId)
m	-(int) copyLayerContent : (int) srcLayerId : (int) dstLayerId
pas	function copyLayerContent (srcLayerId : LongInt, dstLayerId : LongInt): LongInt
vb	function copyLayerContent () As Integer
cs	int copyLayerContent (int srcLayerId , int dstLayerId)
java	int copyLayerContent (int srcLayerId , int dstLayerId)
py	def copyLayerContent (srcLayerId , dstLayerId)
cmd	YDisplay target copyLayerContent srcLayerId dstLayerId

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

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 fade (brightness , duration)
nodejs	function fade (brightness , duration)
php	function fade (\$brightness , \$duration)
cpp	int fade (int brightness , int duration)
m	-(int) fade : (int) brightness : (int) duration
pas	function fade (brightness : LongInt, duration : LongInt): LongInt
vb	function fade () As Integer
cs	int fade (int brightness , int duration)
java	int fade (int brightness , int duration)
py	def fade (brightness , duration)
cmd	YDisplay target fade brightness duration

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.get_advertisedValue()**

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

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

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.get_brightness()**

Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).

js	function get_brightness ()
nodejs	function get_brightness ()
php	function get_brightness ()
cpp	int get_brightness ()
m	-(int) brightness
pas	function get_brightness (): LongInt
vb	function get_brightness () As Integer
cs	int get_brightness ()
java	int get_brightness ()
py	def get_brightness ()
cmd	YDisplay target get_brightness

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.get_displayHeight()**

Retourne la hauteur de l'écran, en pixels.

js	function get_displayHeight ()
nodejs	function get_displayHeight ()
php	function get_displayHeight ()
cpp	int get_displayHeight ()
m	-(int) displayHeight
pas	function get_displayHeight (): LongInt
vb	function get_displayHeight () As Integer
cs	int get_displayHeight ()
java	int get_displayHeight ()
py	def get_displayHeight ()
cmd	YDisplay target get_displayHeight

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.get_displayLayer()**

Retourne un objet YDisplayLayer utilisable pour dessiner sur la couche d'affichage correspondante.

js	function get_displayLayer (layerId)
nodejs	function get_displayLayer (layerId)
php	function get_displayLayer (\$layerId)
c++	YDisplayLayer* get_displayLayer (unsigned layerId)
m	-(YDisplayLayer*) displayLayer : (unsigned) layerId
vb	function get_displayLayer () As YDisplayLayer
cs	YDisplayLayer get_displayLayer (int layerId)
java	synchronized YDisplayLayer get_displayLayer (int layerId)
py	def get_displayLayer (layerId)

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.get_displayType()**

Retourne le type de l'écran: monochrome, niveaux de gris ou couleur.

js	function get_displayType ()
nodejs	function get_displayType ()
php	function get_displayType ()
cpp	Y_DISPLAYTYPE_enum get_displayType ()
m	-(Y_DISPLAYTYPE_enum) displayType
pas	function get_displayType (): Integer
vb	function get_displayType () As Integer
cs	int get_displayType ()
java	int get_displayType ()
py	def get_displayType ()
cmd	YDisplay target get_displayType

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.get_displayWidth()**

Retourne la largeur de l'écran, en pixels.

js	function get_displayWidth ()
nodejs	function get_displayWidth ()
php	function get_displayWidth ()
cpp	int get_displayWidth ()
m	-(int) displayWidth
pas	function get_displayWidth (): LongInt
vb	function get_displayWidth () As Integer
cs	int get_displayWidth ()
java	int get_displayWidth ()
py	def get_displayWidth ()
cmd	YDisplay target get_displayWidth

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()****YDisplay****display**→**enabled()****display.get_enabled()**

Retourne vrai si le l'ecran est alimenté, faux sinon.

js	function get_enabled ()
nodejs	function get_enabled ()
php	function get_enabled ()
cpp	Y_ENABLED_enum get_enabled ()
m	-(Y_ENABLED_enum) enabled
pas	function get_enabled (): Integer
vb	function get_enabled () As Integer
cs	int get_enabled ()
java	int get_enabled ()
py	def get_enabled ()
cmd	YDisplay target get_enabled

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()****display**→**errorMessage()****display.get_errorMessage()**

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

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

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()****YDisplay****display**→**errorType()****display.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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.get_functionDescriptor()

YDisplay

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

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

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

Retourne :

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

display→**get_layerCount()****YDisplay****display**→**layerCount()****display.get_layerCount()**

Retourne le nombre des couches affichables disponibles.

js	function get_layerCount ()
nodejs	function get_layerCount ()
php	function get_layerCount ()
cpp	int get_layerCount ()
m	-(int) layerCount
pas	function get_layerCount (): LongInt
vb	function get_layerCount () As Integer
cs	int get_layerCount ()
java	int get_layerCount ()
py	def get_layerCount ()
cmd	YDisplay target get_layerCount

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.get_layerHeight()**

Retourne la hauteur des couches affichables, en pixels.

js	function get_layerHeight ()
nodejs	function get_layerHeight ()
php	function get_layerHeight ()
cpp	int get_layerHeight ()
m	-(int) layerHeight
pas	function get_layerHeight (): LongInt
vb	function get_layerHeight () As Integer
cs	int get_layerHeight ()
java	int get_layerHeight ()
py	def get_layerHeight ()
cmd	YDisplay target get_layerHeight

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.get_layerWidth()**

Retourne la largeur des couches affichables, en pixels.

js	function get_layerWidth ()
nodejs	function get_layerWidth ()
php	function get_layerWidth ()
cpp	int get_layerWidth ()
m	-(int) layerWidth
pas	function get_layerWidth (): LongInt
vb	function get_layerWidth () As Integer
cs	int get_layerWidth ()
java	int get_layerWidth ()
py	def get_layerWidth ()
cmd	YDisplay target get_layerWidth

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.get_logicalName()**

Retourne le nom logique de l'ecran.

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

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.get_module()**

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

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

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 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.

display→**get_orientation()****YDisplay****display**→**orientation()****display.get_orientation()**

Retourne l'orientation sélectionnée pour l'écran.

js	function get_orientation ()
nodejs	function get_orientation ()
php	function get_orientation ()
cpp	Y_ORIENTATION_enum get_orientation ()
m	-(Y_ORIENTATION_enum) orientation
pas	function get_orientation (): Integer
vb	function get_orientation () As Integer
cs	int get_orientation ()
java	int get_orientation ()
py	def get_orientation ()
cmd	YDisplay target get_orientation

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.get_startupSeq()**

Retourne le nom de la séquence à jouer à la mise sous tension de l'écran.

js	function get_startupSeq()
nodejs	function get_startupSeq()
php	function get_startupSeq()
cpp	string get_startupSeq()
m	-(NSString*) startupSeq
pas	function get_startupSeq() : string
vb	function get_startupSeq() As String
cs	string get_startupSeq()
java	String get_startupSeq()
py	def get_startupSeq()
cmd	YDisplay target get_startupSeq

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

display→**isOnline()****display.isOnline()****YDisplay**

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

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

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 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.

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 newSequence ()
nodejs	function newSequence ()
php	function newSequence ()
cpp	int newSequence ()
m	-(int) newSequence
pas	function newSequence (): LongInt
vb	function newSequence () As Integer
cs	int newSequence ()
java	int newSequence ()
py	def newSequence ()
cmd	YDisplay target newSequence

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 nextDisplay ()
nodejs	function nextDisplay ()
php	function nextDisplay ()
cpp	YDisplay * nextDisplay ()
m	-(YDisplay*) nextDisplay
pas	function nextDisplay (): TYDisplay
vb	function nextDisplay () As YDisplay
cs	YDisplay nextDisplay ()
java	YDisplay nextDisplay ()
py	def nextDisplay ()

Retourne :

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

display→pauseSequence()**YDisplay****display.pauseSequence()**

Attend pour la durée spécifiée (en millisecondes) avant de jouer les commandes suivantes de la séquence active.

js	function pauseSequence (delay_ms)
nodejs	function pauseSequence (delay_ms)
php	function pauseSequence (\$delay_ms)
cpp	int pauseSequence (int delay_ms)
m	-(int) pauseSequence : (int) delay_ms
pas	function pauseSequence (delay_ms : LongInt): LongInt
vb	function pauseSequence () As Integer
cs	int pauseSequence (int delay_ms)
java	int pauseSequence (int delay_ms)
py	def pauseSequence (delay_ms)
cmd	YDisplay target pauseSequence delay_ms

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 playSequence (sequenceName)
nodejs	function playSequence (sequenceName)
php	function playSequence (\$sequenceName)
cpp	int playSequence (string sequenceName)
m	-(int) playSequence : (NSString*) sequenceName
pas	function playSequence (sequenceName : string): LongInt
vb	function playSequence () As Integer
cs	int playSequence (string sequenceName)
java	int playSequence (String sequenceName)
py	def playSequence (sequenceName)
cmd	YDisplay target playSequence sequenceName

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()**YDisplay****display.registerValueCallback()**

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 (YDisplayValueCallback callback)
m	-(int) registerValueCallback : (YDisplayValueCallback) callback
pas	function registerValueCallback (callback : TYDisplayValueCallback): 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.

display→resetAll()`display.resetAll()`**YDisplay**

Efface le contenu de l'écran et remet toutes les couches à leur état initial.

js	function resetAll ()
nodejs	function resetAll ()
php	function resetAll ()
cpp	int resetAll ()
m	-(int) resetAll
pas	function resetAll (): LongInt
vb	function resetAll () As Integer
cs	int resetAll ()
java	int resetAll ()
py	def resetAll ()
cmd	YDisplay target resetAll

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 saveSequence (sequenceName)
nodejs	function saveSequence (sequenceName)
php	function saveSequence (\$sequenceName)
c++	int saveSequence (string sequenceName)
m	-(int) saveSequence : (NSString*) sequenceName
pas	function saveSequence (sequenceName : string): LongInt
vb	function saveSequence () As Integer
cs	int saveSequence (string sequenceName)
java	int saveSequence (String sequenceName)
py	def saveSequence (sequenceName)
cmd	YDisplay target saveSequence sequenceName

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.set_brightness()**

Modifie la luminosité de l'écran.

js	function set_brightness (newval)
nodejs	function set_brightness (newval)
php	function set_brightness (\$newval)
cpp	int set_brightness (int newval)
m	-(int) setBrightness : (int) newval
pas	function set_brightness (newval : LongInt): integer
vb	function set_brightness (ByVal newval As Integer) As Integer
cs	int set_brightness (int newval)
java	int set_brightness (int newval)
py	def set_brightness (newval)
cmd	YDisplay target set_brightness newval

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.set_enabled()**

Modifie l'état d'activité de l'écran.

js	function set_enabled (newval)
nodejs	function set_enabled (newval)
php	function set_enabled (\$newval)
cpp	int set_enabled (Y_ENABLED_enum newval)
m	-(int) setEnabled : (Y_ENABLED_enum) newval
pas	function set_enabled (newval : Integer): integer
vb	function set_enabled (ByVal newval As Integer) As Integer
cs	int set_enabled (int newval)
java	int set_enabled (int newval)
py	def set_enabled (newval)
cmd	YDisplay target set_enabled newval

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.set_logicalName()**

Modifie le nom logique de l'ecran.

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	YDisplay 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 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()**
display→**setOrientation()**
display.set_orientation()

YDisplay

Modifie l'orientation de l'écran.

js	function set_orientation (newval)
nodejs	function set_orientation (newval)
php	function set_orientation (\$newval)
cpp	int set_orientation (Y_ORIENTATION_enum newval)
m	-(int) setOrientation : (Y_ORIENTATION_enum) newval
pas	function set_orientation (newval : Integer): integer
vb	function set_orientation (ByVal newval As Integer) As Integer
cs	int set_orientation (int newval)
java	int set_orientation (int newval)
py	def set_orientation (newval)
cmd	YDisplay target set_orientation newval

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.set_startupSeq()**

Modifie le nom de la séquence à jouer à la mise sous tension de l'écran.

js	function set_startupSeq (newval)
nodejs	function set_startupSeq (newval)
php	function set_startupSeq (\$newval)
cpp	int set_startupSeq (const string& newval)
m	-(int) setStartupSeq : (NSString*) newval
pas	function set_startupSeq (newval : string): integer
vb	function set_startupSeq (ByVal newval As String) As Integer
cs	int set_startupSeq (string newval)
java	int set_startupSeq (String newval)
py	def set_startupSeq (newval)
cmd	YDisplay target set_startupSeq newval

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.set_userdata()**

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

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 stopSequence ()
nodejs	function stopSequence ()
php	function stopSequence ()
cpp	int stopSequence ()
m	-(int) stopSequence
pas	function stopSequence (): LongInt
vb	function stopSequence () As Integer
cs	int stopSequence ()
java	int stopSequence ()
py	def stopSequence ()
cmd	YDisplay target stopSequence

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.

Permute le contenu de deux couches d'affichage.

js	function swapLayerContent (layerIdA , layerIdB)
node.js	function swapLayerContent (layerIdA , layerIdB)
php	function swapLayerContent (\$layerIdA , \$layerIdB)
cpp	int swapLayerContent (int layerIdA , int layerIdB)
m	-(int) swapLayerContent : (int) layerIdA : (int) layerIdB
pas	function swapLayerContent (layerIdA : LongInt, layerIdB : LongInt): LongInt
vb	function swapLayerContent () As Integer
cs	int swapLayerContent (int layerIdA , int layerIdB)
java	int swapLayerContent (int layerIdA , int layerIdB)
py	def swapLayerContent (layerIdA , layerIdB)
cmd	YDisplay target swapLayerContent layerIdA layerIdB

La couleur et la transparence de tous les pixels des deux couches sont permutées. Cette méthode modifie le contenu affiché, mais n'a aucun effet sur les propriétés de l'objet `layer` lui-même. En particulier, la visibilité des deux couches reste inchangée. Cela permet d'implémenter très efficacement un affichage par double-buffering, en utilisant une couche cachée et une couche visible. Notez que la couche zéro n'a pas de transparence (elle est toujours opaque).

Paramètres :

layerIdA l'identifiant de la première couche (un chiffre parmi 0..layerCount-1)
layerIdB l'identifiant de la deuxième couche (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→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.

<code>js</code>	<code>function upload(pathname, content)</code>
<code>nodejs</code>	<code>function upload(pathname, content)</code>
<code>php</code>	<code>function upload(\$pathname, \$content)</code>
<code>c++</code>	<code>int upload(string pathname, string content)</code>
<code>m</code>	<code>-(int) upload : (NSString*) pathname : (NSData*) content</code>
<code>pas</code>	<code>function upload(pathname: string, content: TByteArray): LongInt</code>
<code>vb</code>	<code>procedure upload()</code>
<code>cs</code>	<code>int upload(string pathname)</code>
<code>java</code>	<code>int upload(String pathname)</code>
<code>py</code>	<code>def upload(pathname, content)</code>
<code>cmd</code>	<code>YDisplay target upload pathname content</code>

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 clear ()
nodejs	function clear ()
php	function clear ()
cpp	int clear ()
m	-(int) clear
pas	function clear (): LongInt
vb	function clear () As Integer
cs	int clear ()
java	int clear ()
py	def clear ()
cmd	YDisplay target [-layer layerId] clear

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()**YDisplayLayer****displaylayer.clearConsole()**

Efface le contenu de la zone de console, et repositionne le curseur de la console en haut à gauche de la zone.

js	function clearConsole ()
nodejs	function clearConsole ()
php	function clearConsole ()
cpp	int clearConsole ()
m	-(int) clearConsole
pas	function clearConsole (): LongInt
vb	function clearConsole () As Integer
cs	int clearConsole ()
java	int clearConsole ()
py	def clearConsole ()
cmd	YDisplay target [-layer layerId] clearConsole

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()**YDisplayLayer****displaylayer.consoleOut()**

Affiche un message dans la zone de console, et déplace le curseur de la console à la fin du texte.

js	function consoleOut (text)
nodejs	function consoleOut (text)
php	function consoleOut (\$text)
c++	int consoleOut (string text)
m	-(int) consoleOut : (NSString*) text
pas	function consoleOut (text : string): LongInt
vb	function consoleOut () As Integer
cs	int consoleOut (string text)
java	int consoleOut (String text)
py	def consoleOut (text)
cmd	YDisplay target [-layer layerId] consoleOut text

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.

js	<code>function drawBar(x1, y1, x2, y2)</code>
nodejs	<code>function drawBar(x1, y1, x2, y2)</code>
php	<code>function drawBar(\$x1, \$y1, \$x2, \$y2)</code>
cpp	<code>int drawBar(int x1, int y1, int x2, int y2)</code>
m	<code>-(int) drawBar : (int) x1 : (int) y1 : (int) x2 : (int) y2</code>
pas	<code>function drawBar(x1: LongInt, y1: LongInt, x2: LongInt, y2: LongInt): LongInt</code>
vb	<code>function drawBar() As Integer</code>
cs	<code>int drawBar(int x1, int y1, int x2, int y2)</code>
java	<code>int drawBar(int x1, int y1, int x2, int y2)</code>
py	<code>def drawBar(x1, y1, x2, y2)</code>
cmd	<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()**YDisplayLayer****displaylayer.drawBitmap()**

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.

displaylayer→drawDisc()**YDisplayLayer****displaylayer.drawDisc()**

Dessine un disque plein à une position spécifiée.

js	function drawDisc (x , y , r)
nodejs	function drawDisc (x , y , r)
php	function drawDisc (\$x , \$y , \$r)
cpp	int drawDisc (int x , int y , int r)
m	-(int) drawDisc : (int) x : (int) y : (int) r
pas	function drawDisc (x : LongInt, y : LongInt, r : LongInt): LongInt
vb	function drawDisc () As Integer
cs	int drawDisc (int x , int y , int r)
java	int drawDisc (int x , int y , int r)
py	def drawDisc (x , y , r)
cmd	YDisplay target [-layer layerId] drawDisc x y r

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→drawImage()

YDisplayLayer

```
displaylayer.drawImage()
```

Dessine une image GIF à la position spécifiée de la couche.

js	function drawImage (x , y , imageName)
node.js	function drawImage (x , y , imageName)
php	function drawImage (\$x , \$y , \$imageName)
c++	int drawImage (int x , int y , string imageName)
m	-(int) drawImage : (int) x : (int) y : (NSString*) imageName
pas	function drawImage (x : LongInt, y : LongInt, imageName : string): LongInt
vb	function drawImage () As Integer
cs	int drawImage (int x , int y , string imageName)
java	int drawImage (int x , int y , String imageName)
py	def drawImage (x , y , imageName)
cmd	YDisplay target [-layer layerId] drawImage x y imageName

L'image GIF doit avoir été préalablement préchargée dans la mémoire du module. Si vous rencontrez des problèmes à l'utilisation d'une image bitmap, consultez les logs du module pour voir si vous n'y trouvez pas un message à propos d'un fichier d'image manquant ou d'un format de fichier invalide.

Paramètres :

x	la distance en pixels depuis la gauche de la couche jusqu'au bord gauche de l'image
y	la distance en pixels depuis le haut de la couche jusqu'au bord supérieur de l'image
imagename	le nom du fichier GIF 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→drawPixel()**YDisplayLayer****displaylayer.drawPixel()**

Dessine un pixel unique à une position spécifiée.

js	function drawPixel (x , y)
nodejs	function drawPixel (x , y)
php	function drawPixel (\$x , \$y)
cpp	int drawPixel (int x , int y)
m	-(int) drawPixel : (int) x : (int) y
pas	function drawPixel (x : LongInt, y : LongInt): LongInt
vb	function drawPixel () As Integer
cs	int drawPixel (int x , int y)
java	int drawPixel (int x , int y)
py	def drawPixel (x , y)
cmd	YDisplay target [-layer layerId] drawPixel x y

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()**YDisplayLayer****displaylayer.drawRect()**

Dessine un rectangle vide à une position spécifiée.

```

js      function drawRect( x1, y1, x2, y2)
nodejs  function drawRect( x1, y1, x2, y2)
php      function drawRect( $x1, $y1, $x2, $y2)
cpp      int drawRect( int x1, int y1, int x2, int y2)
m        -(int) drawRect : (int) x1
                        : (int) y1
                        : (int) x2
                        : (int) y2
pas      function drawRect( x1: LongInt,
                        y1: LongInt,
                        x2: LongInt,
                        y2: LongInt): LongInt
vb      function drawRect( ) As Integer
cs      int drawRect( int x1, int y1, int x2, int y2)
java     int drawRect( int x1, int y1, int x2, int y2)
py       def drawRect( x1, y1, x2, y2)
cmd      YDisplay target [-layer layerId] drawRect x1 y1 x2 y2

```

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()**YDisplayLayer****displaylayer.drawText ()**

Affiche un texte à la position spécifiée de la couche.

js	function drawText (x , y , anchor , text)
nodejs	function drawText (x , y , anchor , text)
php	function drawText (\$x , \$y , \$anchor , \$text)
cpp	int drawText (int x , int y , Y_ALIGN anchor , string text)
m	-(int) drawText : (int) x : (int) y : (Y_ALIGN) anchor : (NSString*) text
pas	function drawText (x : LongInt, y : LongInt, anchor : TYALIGN, text : string): LongInt
vb	function drawText () As Integer
cs	int drawText (int x , int y , ALIGN anchor , string text)
java	int drawText (int x , int y , ALIGN anchor , String text)
py	def drawText (x , y , anchor , text)
cmd	YDisplay target [-layer layerId] drawText x y anchor text

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()****YDisplayLayer****displaylayer**→**display()****displaylayer.get_display()**

Retourne l'YDisplay parent.

js	function get_display ()
nodejs	function get_display ()
php	function get_display ()
cpp	YDisplay* get_display ()
m	-(YDisplay*) display
pas	function get_display (): TYDisplay
vb	function get_display () As YDisplay
cs	YDisplay get_display ()
java	YDisplay get_display ()
py	def get_display ()

Retourne l'objet YDisplay parent du YDisplayLayer courant.

Retourne :

un objet YDisplay

displaylayer→get_displayHeight()**YDisplayLayer****displaylayer→displayHeight()****displaylayer.get_displayHeight()**

Retourne la hauteur de l'écran, en pixels.

js	function get_displayHeight ()
nodejs	function get_displayHeight ()
php	function get_displayHeight ()
cpp	int get_displayHeight ()
m	-(int) displayHeight
pas	function get_displayHeight (): LongInt
vb	function get_displayHeight () As Integer
cs	int get_displayHeight ()
java	int get_displayHeight ()
py	def get_displayHeight ()
cmd	YDisplay target [-layer layerId] get_displayHeight

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()**YDisplayLayer****displaylayer→displayWidth()****displaylayer.get_displayWidth()**

Retourne la largeur de l'écran, en pixels.

js	function get_displayWidth ()
nodejs	function get_displayWidth ()
php	function get_displayWidth ()
cpp	int get_displayWidth ()
m	-(int) displayWidth
pas	function get_displayWidth (): LongInt
vb	function get_displayWidth () As Integer
cs	int get_displayWidth ()
java	int get_displayWidth ()
py	def get_displayWidth ()
cmd	YDisplay target [-layer layerId] get_displayWidth

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.get_layerHeight()**

Retourne la hauteur des couches affichables, en pixels.

js	function get_layerHeight ()
nodejs	function get_layerHeight ()
php	function get_layerHeight ()
cpp	int get_layerHeight ()
m	-(int) layerHeight
pas	function get_layerHeight (): LongInt
vb	function get_layerHeight () As Integer
cs	int get_layerHeight ()
java	int get_layerHeight ()
py	def get_layerHeight ()
cmd	YDisplay target [-layer layerId] get_layerHeight

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.get_layerWidth()**

Retourne la largeur des couches affichables, en pixels.

js	function get_layerWidth ()
nodejs	function get_layerWidth ()
php	function get_layerWidth ()
cpp	int get_layerWidth ()
m	-(int) layerWidth
pas	function get_layerWidth (): LongInt
vb	function get_layerWidth () As Integer
cs	int get_layerWidth ()
java	int get_layerWidth ()
py	def get_layerWidth ()
cmd	YDisplay target [-layer layerId] get_layerWidth

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 hide ()
nodejs	function hide ()
php	function hide ()
cpp	int hide ()
m	-(int) hide
pas	function hide (): LongInt
vb	function hide () As Integer
cs	int hide ()
java	int hide ()
py	def hide ()
cmd	YDisplay target [-layer layerId] hide

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.

Dessine une ligne depuis le point de dessin courant jusqu'à la position spécifiée.

js	function lineTo(x, y)
node.js	function lineTo(x, y)
php	function lineTo(\$x, \$y)
c++	int lineTo(int x, int y)
m	-(int) lineTo : (int) x : (int) y
pas	function lineTo(x: LongInt, y: LongInt): LongInt
vb	function lineTo() As Integer
cs	int lineTo(int x, int y)
java	int lineTo(int x, int y)
py	def lineTo(x, y)
cmd	YDisplay target [-layer layerId] lineTo x y

Le pixel final spécifié est inclus dans la ligne dessinée. Le point de dessin courant est déplacé à au point final de la ligne.

Paramètres :

- x** la distance en pixels depuis la gauche de la couche jusqu'au point final
- y** la distance en pixels depuis le haut de la couche jusqu'au point final

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.

<code>js</code>	<code>function moveTo(x, y)</code>
<code>nodejs</code>	<code>function moveTo(x, y)</code>
<code>php</code>	<code>function moveTo(\$x, \$y)</code>
<code>cpp</code>	<code>int moveTo(int x, int y)</code>
<code>m</code>	<code>-(int) moveTo : (int) x : (int) y</code>
<code>pas</code>	<code>function moveTo(x: LongInt, y: LongInt): LongInt</code>
<code>vb</code>	<code>function moveTo() As Integer</code>
<code>cs</code>	<code>int moveTo(int x, int y)</code>
<code>java</code>	<code>int moveTo(int x, int y)</code>
<code>py</code>	<code>def moveTo(x, y)</code>
<code>cmd</code>	<code>YDisplay target [-layer layerId] moveTo x y</code>

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 target [-layer layerId] reset</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()**YDisplayLayer****displaylayer.selectColorPen()**

Choisit la couleur du crayon à utiliser pour tous les appels suivants aux fonctions de dessin.

js	function selectColorPen (color)
nodejs	function selectColorPen (color)
php	function selectColorPen (\$color)
cpp	int selectColorPen (int color)
m	-(int) selectColorPen : (int) color
pas	function selectColorPen (color : LongInt): LongInt
vb	function selectColorPen () As Integer
cs	int selectColorPen (int color)
java	int selectColorPen (int color)
py	def selectColorPen (color)
cmd	YDisplay target [-layer layerId] selectColorPen color

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()**YDisplayLayer****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.

js	function selectEraser ()
nodejs	function selectEraser ()
php	function selectEraser ()
cpp	int selectEraser ()
m	-(int) selectEraser
pas	function selectEraser (): LongInt
vb	function selectEraser () As Integer
cs	int selectEraser ()
java	int selectEraser ()
py	def selectEraser ()
cmd	YDisplay target [-layer layerId] selectEraser

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()**YDisplayLayer****displaylayer.selectFont()**

Sélectionne la police de caractères à utiliser pour les fonctions d'affichage de texte suivantes.

js	function selectFont (fontname)
nodejs	function selectFont (fontname)
php	function selectFont (\$fontname)
cpp	int selectFont (string fontname)
m	-(int) selectFont : (NSString*) fontname
pas	function selectFont (fontname : string): LongInt
vb	function selectFont () As Integer
cs	int selectFont (string fontname)
java	int selectFont (String fontname)
py	def selectFont (fontname)
cmd	YDisplay target [-layer layerId] selectFont fontname

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()**YDisplayLayer****displaylayer.selectGrayPen()**

Choisit le niveau de gris à utiliser pour tous les appels suivants aux fonctions de dessin.

js	function selectGrayPen (graylevel)
nodejs	function selectGrayPen (graylevel)
php	function selectGrayPen (\$graylevel)
cpp	int selectGrayPen (int graylevel)
m	-(int) selectGrayPen : (int) graylevel
pas	function selectGrayPen (graylevel : LongInt): LongInt
vb	function selectGrayPen () As Integer
cs	int selectGrayPen (int graylevel)
java	int selectGrayPen (int graylevel)
py	def selectGrayPen (graylevel)
cmd	YDisplay target [-layer layerId] selectGrayPen graylevel

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()**YDisplayLayer****displaylayer.setAntialiasingMode()**

Active ou désactive l'anti-aliasing pour tracer les lignes et les cercles.

js	function setAntialiasingMode (mode)
nodejs	function setAntialiasingMode (mode)
php	function setAntialiasingMode (\$mode)
cpp	int setAntialiasingMode (bool mode)
m	-(int) setAntialiasingMode : (bool) mode
pas	function setAntialiasingMode (mode : boolean): LongInt
vb	function setAntialiasingMode () As Integer
cs	int setAntialiasingMode (bool mode)
java	int setAntialiasingMode (boolean mode)
py	def setAntialiasingMode (mode)
cmd	YDisplay target [-layer layerId] setAntialiasingMode mode

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()**YDisplayLayer****displaylayer.setConsoleBackground()**

Configure la couleur de fond utilisée par la fonction `clearConsole` et par le défilement automatique de la console.

js	function setConsoleBackground (bgcol)
nodejs	function setConsoleBackground (bgcol)
php	function setConsoleBackground (\$bgcol)
c++	int setConsoleBackground (int bgcol)
m	-(int) setConsoleBackground : (int) bgcol
pas	function setConsoleBackground (bgcol : LongInt): LongInt
vb	function setConsoleBackground () As Integer
cs	int setConsoleBackground (int bgcol)
java	int setConsoleBackground (int bgcol)
py	def setConsoleBackground (bgcol)
cmd	YDisplay target [-layer layerId] setConsoleBackground bgcol

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.

displaylayer→setConsoleMargins()**YDisplayLayer****displaylayer.setConsoleMargins()**

Configure les marges d'affichage pour la fonction `consoleOut`.

js	function setConsoleMargins (x1 , y1 , x2 , y2)
nodejs	function setConsoleMargins (x1 , y1 , x2 , y2)
php	function setConsoleMargins (\$x1 , \$y1 , \$x2 , \$y2)
c++	int setConsoleMargins (int x1 , int y1 , int x2 , int y2)
m	-(int) setConsoleMargins : (int) x1 : (int) y1 : (int) x2 : (int) y2
pas	function setConsoleMargins (x1 : LongInt, : LongInt, : LongInt, : LongInt): LongInt
vb	function setConsoleMargins () As Integer
cs	int setConsoleMargins (int x1 , int y1 , int x2 , int y2)
java	int setConsoleMargins (int x1 , int y1 , int x2 , int y2)
py	def setConsoleMargins (x1 , y1 , x2 , y2)
cmd	YDisplay target [-layer layerId] setConsoleMargins x1 y1 x2 y2

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()**YDisplayLayer****displaylayer.setConsoleWordWrap()**

Configure le mode de retour à la ligne utilisé par la fonction `consoleOut`.

js	function setConsoleWordWrap (wordwrap)
nodejs	function setConsoleWordWrap (wordwrap)
php	function setConsoleWordWrap (\$wordwrap)
cpp	int setConsoleWordWrap (bool wordwrap)
m	-(int) setConsoleWordWrap : (bool) wordwrap
pas	function setConsoleWordWrap (wordwrap : boolean): LongInt
vb	function setConsoleWordWrap () As Integer
cs	int setConsoleWordWrap (bool wordwrap)
java	int setConsoleWordWrap (boolean wordwrap)
py	def setConsoleWordWrap (wordwrap)
cmd	YDisplay target [-layer layerId] setConsoleWordWrap wordwrap

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→setLayerPosition()**YDisplayLayer****displaylayer.setLayerPosition()**

Déplace la position de la couche de dessin par rapport au coin supérieur gauche de l'écran.

js	function setLayerPosition (x , y , scrollTime)
nodejs	function setLayerPosition (x , y , scrollTime)
php	function setLayerPosition (\$x , \$y , \$scrollTime)
cpp	int setLayerPosition (int x , int y , int scrollTime)
m	-(int) setLayerPosition : (int) x : (int) y : (int) scrollTime
pas	function setLayerPosition (x : LongInt, : LongInt, : LongInt): LongInt
vb	function setLayerPosition () As Integer
cs	int setLayerPosition (int x , int y , int scrollTime)
java	int setLayerPosition (int x , int y , int scrollTime)
py	def setLayerPosition (x , y , scrollTime)
cmd	YDisplay target [-layer layerId] setLayerPosition x y scrollTime

Lorsqu'une durée de défilement est configurée, la position d'affichage de la couche est automatiquement mise à jour durant les millisecondes suivantes pour animer le déplacement.

Paramètres :

- x** la distance en pixels depuis la gauche de l'écran jusqu'à l'origine de la couche.
- y** la distance en pixels depuis le haut de l'écran jusqu'à l'origine de la couche.
- scrollTime** durée en millisecondes du déplacement, ou 0 si le déplacement doit être immédiat.

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 unhide ()
nodejs	function unhide ()
php	function unhide ()
cpp	int unhide ()
m	-(int) unhide
pas	function unhide (): LongInt
vb	function unhide () As Integer
cs	int unhide ()
java	int unhide ()
py	def unhide ()
cmd	YDisplay target [-layer layerId] unhide

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

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()**YDualPower****yFindDualPower()**`YDualPower.FindDualPower()`

Permet de retrouver un contrôle d'alimentation d'après un identifiant donné.

js	function yFindDualPower (func)
nodejs	function FindDualPower (func)
php	function yFindDualPower (\$func)
cpp	YDualPower* yFindDualPower (const string& func)
m	YDualPower* yFindDualPower (NSString* func)
pas	function yFindDualPower (func : string): TYDualPower
vb	function yFindDualPower (ByVal func As String) As YDualPower
cs	YDualPower FindDualPower (string func)
java	YDualPower FindDualPower (String func)
py	def FindDualPower (func)

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()**YDualPower****yFirstDualPower()**`YDualPower.FirstDualPower()`

Commence l'énumération des contrôles d'alimentation accessibles par la librairie.

js	function yFirstDualPower ()
nodejs	function FirstDualPower ()
php	function yFirstDualPower ()
cpp	YDualPower* yFirstDualPower ()
m	YDualPower* yFirstDualPower ()
pas	function yFirstDualPower (): TYDualPower
vb	function yFirstDualPower () As YDualPower
cs	YDualPower FirstDualPower ()
java	YDualPower FirstDualPower ()
py	def FirstDualPower ()

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`.

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

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()**YDualPower****dualpower→advertisedValue()****dualpower.get_advertisedValue()**

Retourne la valeur courante du contrôle d'alimentation (pas plus de 6 caractères).

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

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

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()**YDualPower****dualpower→errorType()****dualpower.get_errorType()**

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

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

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**.**get_extVoltage()**

Retourne la tension mesurée sur l'alimentation de puissance externe, en millivolts.

js	function get_extVoltage ()
nodejs	function get_extVoltage ()
php	function get_extVoltage ()
cpp	int get_extVoltage ()
m	-(int) extVoltage
pas	function get_extVoltage (): LongInt
vb	function get_extVoltage () As Integer
cs	int get_extVoltage ()
java	int get_extVoltage ()
py	def get_extVoltage ()
cmd	YDualPower target get_extVoltage

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.get_friendlyName()**

Retourne un identifiant global du contrôle d'alimentation au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du contrôle d'alimentation, sans référence au module.

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

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**.**get_hardwareId()**

Retourne l'identifiant matériel unique du contrôle d'alimentation au format `SERIAL.FUNCTIONID`.

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

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.get_logicalName()**

Retourne le nom logique du contrôle d'alimentation.

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

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()****YDualPower****dualpower**→**module()****dualpower.get_module()**

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

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

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 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.

dualpower→**get_powerControl()****YDualPower****dualpower**→**powerControl()****dualpower.get_powerControl()**

Retourne le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

js	function get_powerControl ()
nodejs	function get_powerControl ()
php	function get_powerControl ()
cpp	Y_POWERCONTROL_enum get_powerControl ()
m	-(Y_POWERCONTROL_enum) powerControl
pas	function get_powerControl (): Integer
vb	function get_powerControl () As Integer
cs	int get_powerControl ()
java	int get_powerControl ()
py	def get_powerControl ()
cmd	YDualPower target get_powerControl

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.get_powerState()**

Retourne la source d'alimentation active pour les fonctions du module consommant beaucoup de courant.

js	function get_powerState ()
nodejs	function get_powerState ()
php	function get_powerState ()
cpp	Y_POWERSTATE_enum get_powerState ()
m	-(Y_POWERSTATE_enum) powerState
pas	function get_powerState (): Integer
vb	function get_powerState () As Integer
cs	int get_powerState ()
java	int get_powerState ()
py	def get_powerState ()
cmd	YDualPower target get_powerState

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 get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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

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 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.

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()**YDualPower****dualpower.nextDualPower()**

Continue l'énumération des contrôles d'alimentation commencée à l'aide de `yFirstDualPower()`.

js	function nextDualPower ()
nodejs	function nextDualPower ()
php	function nextDualPower ()
cpp	YDualPower * nextDualPower ()
m	-(YDualPower*) nextDualPower
pas	function nextDualPower (): TYDualPower
vb	function nextDualPower () As YDualPower
cs	YDualPower nextDualPower ()
java	YDualPower nextDualPower ()
py	def nextDualPower ()

Retourne :

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

dualpower→registerValueCallback()**YDualPower****dualpower.registerValueCallback()**

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)
c++	int registerValueCallback (YDualPowerValueCallback callback)
m	-(int) registerValueCallback : (YDualPowerValueCallback) callback
pas	function registerValueCallback (callback : TYDualPowerValueCallback): 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.

dualpower→**set_logicalName()****YDualPower****dualpower**→**setLogicalName()****dualpower.set_logicalName()**

Modifie le nom logique du contrôle d'alimentation.

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	YDualPower 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 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.set_powerControl()**

Modifie le mode d'alimentation choisi pour les fonctions du module consommant beaucoup de courant.

js	function set_powerControl (newval)
nodejs	function set_powerControl (newval)
php	function set_powerControl (\$newval)
cpp	int set_powerControl (Y_POWERCONTROL_enum newval)
m	-(int) setPowerControl : (Y_POWERCONTROL_enum) newval
pas	function set_powerControl (newval : Integer): integer
vb	function set_powerControl (ByVal newval As Integer) As Integer
cs	int set_powerControl (int newval)
java	int set_powerControl (int newval)
py	def set_powerControl (newval)
cmd	YDualPower target set_powerControl newval

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.set_userdata()**

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

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

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()**YFiles****yFindFiles()** **YFiles.FindFiles()**

Permet de retrouver un système de fichier d'après un identifiant donné.

js	function yFindFiles (func)
nodejs	function FindFiles (func)
php	function yFindFiles (\$func)
cpp	YFiles* yFindFiles (string func)
m	+(YFiles*) yFindFiles : (NSString*) func
pas	function yFindFiles (func : string): TYFiles
vb	function yFindFiles (ByVal func As String) As YFiles
cs	YFiles FindFiles (string func)
java	YFiles FindFiles (String func)
py	def FindFiles (func)

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()**YFiles****yFirstFiles()****YFiles.FirstFiles()**

Commence l'énumération des système de fichier accessibles par la librairie.

js	function yFirstFiles ()
nodejs	function FirstFiles ()
php	function yFirstFiles ()
cpp	YFiles* yFirstFiles ()
m	YFiles* yFirstFiles ()
pas	function yFirstFiles (): TYFiles
vb	function yFirstFiles () As YFiles
cs	YFiles FirstFiles ()
java	YFiles FirstFiles ()
py	def FirstFiles ()

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

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()

YFiles

Télécharge le fichier choisi du filesystem et retourne son contenu.

js	function download (pathname)
nodejs	function download (pathname)
php	function download (\$pathname)
cpp	string download (string pathname)
m	-(NSData*) download : (NSString*) pathname
pas	function download (pathname : string): TByteArray
vb	function download () As Byte
py	def download (pathname)
cmd	YFiles target download pathname

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()**

Rétabli le système de fichier dans on état original, défragmenté.

js	function format_fs ()
nodejs	function format_fs ()
php	function format_fs ()
cpp	int format_fs ()
m	-(int) format_fs
pas	function format_fs (): LongInt
vb	function format_fs () As Integer
cs	int format_fs ()
java	int format_fs ()
py	def format_fs ()
cmd	YFiles target format_fs

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.get_advertisedValue()**

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

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

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

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()****files.get_errorType()**

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

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

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.get_filesCount()**

Retourne le nombre de fichiers présents dans le système de fichier.

js	function get_filesCount ()
nodejs	function get_filesCount ()
php	function get_filesCount ()
cpp	int get_filesCount ()
m	-(int) filesCount
pas	function get_filesCount (): LongInt
vb	function get_filesCount () As Integer
cs	int get_filesCount ()
java	int get_filesCount ()
py	def get_filesCount ()
cmd	YFiles target get_filesCount

Retourne :

un entier représentant le nombre de fichiers présents dans le système de fichier

En cas d'erreur, déclenche une exception ou retourne Y_FILESCOUNT_INVALID.

files→**get_freeSpace()****YFiles****files**→**freeSpace()****files.get_freeSpace()**

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

js	function get_freeSpace ()
nodejs	function get_freeSpace ()
php	function get_freeSpace ()
cpp	int get_freeSpace ()
m	-(int) freeSpace
pas	function get_freeSpace (): LongInt
vb	function get_freeSpace () As Integer
cs	int get_freeSpace ()
java	int get_freeSpace ()
py	def get_freeSpace ()
cmd	YFiles target get_freeSpace

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.get_friendlyName()**

Retourne un identifiant global du système de fichier au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
c++	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du système de fichier, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

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

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.get_list()**

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

js	function get_list (pattern)
nodejs	function get_list (pattern)
php	function get_list (\$pattern)
cpp	vector<YFileRecord> get_list (string pattern)
m	-(NSMutableArray*) list : (NSString*) pattern
pas	function get_list (pattern : string): TYFileRecordArray
vb	function get_list () As List
cs	List<YFileRecord> get_list (string pattern)
java	ArrayList<YFileRecord> get_list (String pattern)
py	def get_list (pattern)
cmd	YFiles target get_list pattern

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.get_logicalName()**

Retourne le nom logique du système de fichier.

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

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.get_module()**

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

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

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 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.

files→**get_userData()****files**→**userData()****files.get_userData()**

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

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

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.

files→**isOnline()****files.isOnline()****YFiles**

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

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

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 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.

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 nextFiles ()
nodejs	function nextFiles ()
php	function nextFiles ()
cpp	YFiles * nextFiles ()
m	-(YFiles*) nextFiles
pas	function nextFiles (): TYFiles
vb	function nextFiles () As YFiles
cs	YFiles nextFiles ()
java	YFiles nextFiles ()
py	def nextFiles ()

Retourne :

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

files→registerValueCallback()**files.registerValueCallback()**

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 (YFilesValueCallback callback)
m	-(int) registerValueCallback : (YFilesValueCallback) callback
pas	function registerValueCallback (callback : TYFilesValueCallback): 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.

files→remove()`files.remove()`**YFiles**

Efface un fichier, spécifié par son path complet, du système de fichier.

js	function remove (pathname)
nodejs	function remove (pathname)
php	function remove (\$pathname)
cpp	int remove (string pathname)
m	-(int) remove : (NSString*) pathname
pas	function remove (pathname : string): LongInt
vb	function remove () As Integer
cs	int remove (string pathname)
java	int remove (String pathname)
py	def remove (pathname)
cmd	YFiles target remove pathname

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.set_logicalName()**

Modifie le nom logique du système de fichier.

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	YFiles 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 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.set_userdata()**

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

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.

<code>js</code>	<code>function upload(pathname, content)</code>
<code>nodejs</code>	<code>function upload(pathname, content)</code>
<code>php</code>	<code>function upload(\$pathname, \$content)</code>
<code>c++</code>	<code>int upload(string pathname, string content)</code>
<code>m</code>	<code>-(int) upload : (NSString*) pathname : (NSData*) content</code>
<code>pas</code>	<code>function upload(pathname: string, content: TByteArray): LongInt</code>
<code>vb</code>	<code>procedure upload()</code>
<code>cs</code>	<code>int upload(string pathname)</code>
<code>java</code>	<code>int upload(String pathname)</code>
<code>py</code>	<code>def upload(pathname, content)</code>
<code>cmd</code>	<code>YFiles target upload pathname content</code>

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()**YGenericSensor****yFindGenericSensor()****YGenericSensor.FindGenericSensor()**

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

js	function yFindGenericSensor (func)
nodejs	function FindGenericSensor (func)
php	function yFindGenericSensor (\$func)
cpp	YGenericSensor* yFindGenericSensor (const string& func)
m	YGenericSensor* yFindGenericSensor (NSString* func)
pas	function yFindGenericSensor (func : string): TYGenericSensor
vb	function yFindGenericSensor (ByVal func As String) As YGenericSensor
cs	YGenericSensor FindGenericSensor (string func)
java	YGenericSensor FindGenericSensor (String func)
py	def FindGenericSensor (func)

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()

YGenericSensor

yFirstGenericSensor()

YGenericSensor.FirstGenericSensor()

Commence l'énumération des capteurs génériques accessibles par la librairie.

js	function yFirstGenericSensor ()
nodejs	function FirstGenericSensor ()
php	function yFirstGenericSensor ()
cpp	YGenericSensor* yFirstGenericSensor ()
m	YGenericSensor* yFirstGenericSensor ()
pas	function yFirstGenericSensor (): TYGenericSensor
vb	function yFirstGenericSensor () As YGenericSensor
cs	YGenericSensor FirstGenericSensor ()
java	YGenericSensor FirstGenericSensor ()
py	def FirstGenericSensor ()

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()**YGenericSensor****genericsensor.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 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	YGenericSensor 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.

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()**YGenericSensor****genericsensor.describe()**

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

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

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 générique (ex:
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

genericsensor→get_advertisedValue()**YGenericSensor****genericsensor→advertisedValue()****genericsensor.get_advertisedValue()**

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

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

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.get_currentRawValue()**

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

js	function get_currentRawValue()
nodejs	function get_currentRawValue()
php	function get_currentRawValue()
cpp	double get_currentRawValue()
m	-(double) currentRawValue
pas	function get_currentRawValue() : double
vb	function get_currentRawValue() As Double
cs	double get_currentRawValue()
java	double get_currentRawValue()
py	def get_currentRawValue()
cmd	YGenericSensor target get_currentRawValue

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.get_currentValue()**

Retourne la valeur mesurée actuelle.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YGenericSensor target get_currentValue

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.get_errorMessage()**

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

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

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()**YGenericSensor****genericsensor→errorType()****genericsensor.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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.get_functionDescriptor()**

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

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

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.get_functionId()

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

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

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()**YGenericSensor****genericsensor→hardwareId()****genericsensor.get_hardwareId()**

Retourne l'identifiant matériel unique du capteur générique au format SERIAL.FUNCTIONID.

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

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()**
genericsensor→**highestValue()**
genericsensor.get_highestValue()

YGenericSensor

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

js	function get_highestValue()
nodejs	function get_highestValue()
php	function get_highestValue()
cpp	double get_highestValue()
m	-(double) highestValue
pas	function get_highestValue() : double
vb	function get_highestValue() As Double
cs	double get_highestValue()
java	double get_highestValue()
py	def get_highestValue()
cmd	YGenericSensor target get_highestValue

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.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.

js	function get_logFrequency ()
nodejs	function get_logFrequency ()
php	function get_logFrequency ()
c++	string get_logFrequency ()
m	-(NSString*) logFrequency
pas	function get_logFrequency (): string
vb	function get_logFrequency () As String
cs	string get_logFrequency ()
java	String get_logFrequency ()
py	def get_logFrequency ()
cmd	YGenericSensor target get_logFrequency

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.get_logicalName()**

Retourne le nom logique du capteur générique.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YGenericSensor target get_lowestValue

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.get_module()**

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

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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.

genericsensor→**get_recordedData()****YGenericSensor****genericsensor**→**recordedData()****genericsensor.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
c++	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YGenericSensor target get_recordedData startTime endTime

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.get_reportFrequency()**

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

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

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()****YGenericSensor****genericsensor**→**resolution()****genericsensor.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YGenericSensor target get_resolution

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.get_signalRange()**

Retourne la plage de signal électrique utilisée par le capteur.

js	function get_signalRange ()
nodejs	function get_signalRange ()
php	function get_signalRange ()
cpp	string get_signalRange ()
m	-(NSString*) signalRange
pas	function get_signalRange (): string
vb	function get_signalRange () As String
cs	string get_signalRange ()
java	String get_signalRange ()
py	def get_signalRange ()
cmd	YGenericSensor target get_signalRange

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.get_signalUnit()**

Retourne l'unité du signal électrique utilisée par le capteur.

js	function get_signalUnit ()
nodejs	function get_signalUnit ()
php	function get_signalUnit ()
cpp	string get_signalUnit ()
m	-(NSString*) signalUnit
pas	function get_signalUnit (): string
vb	function get_signalUnit () As String
cs	string get_signalUnit ()
java	String get_signalUnit ()
py	def get_signalUnit ()
cmd	YGenericSensor target get_signalUnit

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()**YGenericSensor****genericsensor→signalValue()****genericsensor.get_signalValue()**

Retourne la valeur mesurée du signal électrique utilisée par le capteur.

js	function get_signalValue ()
nodejs	function get_signalValue ()
php	function get_signalValue ()
cpp	double get_signalValue ()
m	-(double) signalValue
pas	function get_signalValue (): double
vb	function get_signalValue () As Double
cs	double get_signalValue ()
java	double get_signalValue ()
py	def get_signalValue ()
cmd	YGenericSensor target get_signalValue

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.get_unit()`

Retourne l'unité dans laquelle la mesure est exprimée.

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

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.get_userdata()**

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

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

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.

genericsensor→get_valueRange()**YGenericSensor****genericsensor→valueRange()****genericsensor.get_valueRange()**

Retourne la plage de valeurs physiques mesurés par le capteur.

js	function get_valueRange ()
nodejs	function get_valueRange ()
php	function get_valueRange ()
cpp	string get_valueRange ()
m	-(NSString*) valueRange
pas	function get_valueRange (): string
vb	function get_valueRange () As String
cs	string get_valueRange ()
java	String get_valueRange ()
py	def get_valueRange ()
cmd	YGenericSensor target get_valueRange

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()**YGenericSensor****genericsensor.isOnline()**

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

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

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 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.

genericsensor→loadCalibrationPoints()**YGenericSensor****genericsensor.loadCalibrationPoints()**

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()**YGenericSensor****genericsensor.nextGenericSensor()**

Continue l'énumération des capteurs génériques commencée à l'aide de `yFirstGenericSensor()`.

js	function nextGenericSensor ()
nodejs	function nextGenericSensor ()
php	function nextGenericSensor ()
cpp	YGenericSensor * nextGenericSensor ()
m	-(YGenericSensor*) nextGenericSensor
pas	function nextGenericSensor (): TYGenericSensor
vb	function nextGenericSensor () As YGenericSensor
cs	YGenericSensor nextGenericSensor ()
java	YGenericSensor nextGenericSensor ()
py	def nextGenericSensor ()

Retourne :

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

genericsensor→registerTimedReportCallback()**YGenericSensor**
**genericsensor.registerTimedReportCallback(
)**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
c++	int registerTimedReportCallback (YGenericSensorTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YGenericSensorTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYGenericSensorTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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'object fonction dont la valeur a changé, et un objet YMeasure décrivant la nouvelle valeur publiée.

genericsensor→registerValueCallback()**YGenericSensor****genericsensor.registerValueCallback()**

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 (YGenericSensorValueCallback callback)
m	-(int) registerValueCallback : (YGenericSensorValueCallback) callback
pas	function registerValueCallback (callback : TYGenericSensorValueCallback): 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.

genericsensor→set_highestValue()**YGenericSensor****genericsensor→setHighestValue()****genericsensor.set_highestValue()**

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	YGenericSensor 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.

genericsensor→**set_logFrequency()**
genericsensor→**setLogFrequency()**
genericsensor.set_logFrequency()

YGenericSensor

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	YGenericSensor 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.

genericsensor→set_logicalName()**YGenericSensor****genericsensor→setLogicalName()****genericsensor.set_logicalName()**

Modifie le nom logique du capteur générique.

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	YGenericSensor 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 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()**
genericsensor→**setLowestValue()**
genericsensor.set_lowestValue()

YGenericSensor

Modifie la mémoire de valeur minimale observée.

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

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()**YGenericSensor****genericsensor→setReportFrequency()****genericsensor.set_reportFrequency()**

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	YGenericSensor 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.

genericsensor→**set_resolution()****YGenericSensor****genericsensor**→**setResolution()****genericsensor.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_signalRange()**

Modifie la plage de signal électrique utilisée par le capteur.

js	function set_signalRange (newval)
nodejs	function set_signalRange (newval)
php	function set_signalRange (\$newval)
cpp	int set_signalRange (const string& newval)
m	-(int) setSignalRange : (NSString*) newval
pas	function set_signalRange (newval : string): integer
vb	function set_signalRange (ByVal newval As String) As Integer
cs	int set_signalRange (string newval)
java	int set_signalRange (String newval)
py	def set_signalRange (newval)
cmd	YGenericSensor target set_signalRange newval

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.set_unit()**

Change l'unité dans laquelle la valeur mesurée est exprimée.

js	function set_unit (newval)
nodejs	function set_unit (newval)
php	function set_unit (\$newval)
cpp	int set_unit (const string& newval)
m	-(int) setUnit : (NSString*) newval
pas	function set_unit (newval : string): integer
vb	function set_unit (ByVal newval As String) As Integer
cs	int set_unit (string newval)
java	int set_unit (String newval)
py	def set_unit (newval)
cmd	YGenericSensor target set_unit newval

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.set_userdata()**

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

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

genericsensor→set_valueRange()**YGenericSensor****genericsensor→setValueRange()****genericsensor.set_valueRange()**

Modifie la plage de valeurs physiques mesurés par le capteur.

js	function set_valueRange (newval)
nodejs	function set_valueRange (newval)
php	function set_valueRange (\$newval)
cpp	int set_valueRange (const string& newval)
m	-(int) setValueRange : (NSString*) newval
pas	function set_valueRange (newval : string): integer
vb	function set_valueRange (ByVal newval As String) As Integer
cs	int set_valueRange (string newval)
java	int set_valueRange (String newval)
py	def set_valueRange (newval)
cmd	YGenericSensor target set_valueRange newval

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()**YGyro****yFindGyro()****YGyro.FindGyro()**

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

js	function yFindGyro (func)
nodejs	function FindGyro (func)
php	function yFindGyro (\$func)
cpp	YGyro* yFindGyro (string func)
m	+(YGyro*) yFindGyro : (NSString*) func
pas	function yFindGyro (func : string): TYGyro
vb	function yFindGyro (ByVal func As String) As YGyro
cs	YGyro FindGyro (string func)
java	YGyro FindGyro (String func)
py	def FindGyro (func)

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()**YGyro****yFirstGyro()****YGyro.FirstGyro()**

Commence l'énumération des gyroscopes accessibles par la librairie.

js	function yFirstGyro ()
nodejs	function FirstGyro ()
php	function yFirstGyro ()
cpp	YGyro* yFirstGyro ()
m	YGyro* yFirstGyro ()
pas	function yFirstGyro (): TYGyro
vb	function yFirstGyro () As YGyro
cs	YGyro FirstGyro ()
java	YGyro FirstGyro ()
py	def FirstGyro ()

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()**YGyro****gyro.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 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.

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

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 gyroscope (ex:
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

gyro→**get_advertisedValue()**
gyro→**advertisedValue()**
gyro.get_advertisedValue()

YGyro

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

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

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.get_currentRawValue()**

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

js	function get_currentRawValue()
nodejs	function get_currentRawValue()
php	function get_currentRawValue()
cpp	double get_currentRawValue()
m	-(double) currentRawValue
pas	function get_currentRawValue() : double
vb	function get_currentRawValue() As Double
cs	double get_currentRawValue()
java	double get_currentRawValue()
py	def get_currentRawValue()
cmd	YGyro target get_currentRawValue

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()****YGyro****gyro**→**currentValue()****gyro.get_currentValue()**

Retourne la valeur actuelle de la vitesse angulaire.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YGyro target get_currentValue

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.get_errorMessage()**

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

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

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()**`gyro.get_errorType()`

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

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

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.get_friendlyName()`

Retourne un identifiant global du gyroscope au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
c++	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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()**YGyro****gyro→functionDescriptor()****gyro.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du gyroscope, sans référence au module.

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

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()****YGyro****gyro**→**hardwareId()****gyro.get_hardwareId()**

Retourne l'identifiant matériel unique du gyroscope au format `SERIAL.FUNCTIONID`.

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

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.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.

js	function get_heading ()
nodejs	function get_heading ()
php	function get_heading ()
cpp	double get_heading ()
m	-(double) heading
pas	function get_heading (): double
vb	function get_heading () As Double
cs	double get_heading ()
java	double get_heading ()
py	def get_heading ()

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.get_highestValue()**

Retourne la valeur maximale observée pour la vitesse angulaire depuis le démarrage du module.

js	function get_highestValue()
nodejs	function get_highestValue()
php	function get_highestValue()
cpp	double get_highestValue()
m	-(double) highestValue
pas	function get_highestValue() : double
vb	function get_highestValue() As Double
cs	double get_highestValue()
java	double get_highestValue()
py	def get_highestValue()
cmd	YGyro target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du gyroscope.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour la vitesse angulaire depuis le démarrage du module.

js	function get_lowestValue()
nodejs	function get_lowestValue()
php	function get_lowestValue()
cpp	double get_lowestValue()
m	-(double) lowestValue
pas	function get_lowestValue() : double
vb	function get_lowestValue() As Double
cs	double get_lowestValue()
java	double get_lowestValue()
py	def get_lowestValue()
cmd	YGyro target get_lowestValue

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()****YGyro****gyro**→**module()**`gyro.get_module()`

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

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

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.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.

js	function get_pitch ()
nodejs	function get_pitch ()
php	function get_pitch ()
cpp	double get_pitch ()
m	-(double) pitch
pas	function get_pitch (): double
vb	function get_pitch () As Double
cs	double get_pitch ()
java	double get_pitch ()
py	def get_pitch ()

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.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.

js	function get_quaternionW ()
nodejs	function get_quaternionW ()
php	function get_quaternionW ()
cpp	double get_quaternionW ()
m	-(double) quaternionW
pas	function get_quaternionW (): double
vb	function get_quaternionW () As Double
cs	double get_quaternionW ()
java	double get_quaternionW ()
py	def get_quaternionW ()

Retourne :

un nombre à virgule correspondant à la composante w du quaternion.

gyro→**get_quaternionX()****YGyro****gyro**→**quaternionX()****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.

js	function get_quaternionX ()
nodejs	function get_quaternionX ()
php	function get_quaternionX ()
cpp	double get_quaternionX ()
m	-(double) quaternionX
pas	function get_quaternionX (): double
vb	function get_quaternionX () As Double
cs	double get_quaternionX ()
java	double get_quaternionX ()
py	def get_quaternionX ()

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.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.

js	function get_quaternionY ()
nodejs	function get_quaternionY ()
php	function get_quaternionY ()
cpp	double get_quaternionY ()
m	-(double) quaternionY
pas	function get_quaternionY (): double
vb	function get_quaternionY () As Double
cs	double get_quaternionY ()
java	double get_quaternionY ()
py	def get_quaternionY ()

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.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.

js	function get_quaternionZ ()
nodejs	function get_quaternionZ ()
php	function get_quaternionZ ()
cpp	double get_quaternionZ ()
m	-(double) quaternionZ
pas	function get_quaternionZ (): double
vb	function get_quaternionZ () As Double
cs	double get_quaternionZ ()
java	double get_quaternionZ ()
py	def get_quaternionZ ()

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
c++	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YGyro target get_recordedData startTime endTime

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.get_reportFrequency()**

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

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

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()****YGyro****gyro**→**resolution()**`gyro.get_resolution()`

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YGyro target get_resolution

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.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.

js	function get_roll ()
nodejs	function get_roll ()
php	function get_roll ()
cpp	double get_roll ()
m	-(double) roll
pas	function get_roll (): double
vb	function get_roll () As Double
cs	double get_roll ()
java	double get_roll ()
py	def get_roll ()

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.get_unit()`

Retourne l'unité dans laquelle la vitesse angulaire est exprimée.

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

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.get_userData()`

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

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

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.get_xValue()**

Retourne la vitesse angulaire autour de l'axe X du module, sous forme de nombre à virgule.

js	function get_xValue ()
nodejs	function get_xValue ()
php	function get_xValue ()
cpp	double get_xValue ()
m	-(double) xValue
pas	function get_xValue (): double
vb	function get_xValue () As Double
cs	double get_xValue ()
java	double get_xValue ()
py	def get_xValue ()

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()**YGyro****gyro→yValue()**`gyro.get_yValue()`

Retourne la vitesse angulaire autour de l'axe Y du module, sous forme de nombre à virgule.

js	function get_yValue ()
nodejs	function get_yValue ()
php	function get_yValue ()
cpp	double get_yValue ()
m	-(double) yValue
pas	function get_yValue (): double
vb	function get_yValue () As Double
cs	double get_yValue ()
java	double get_yValue ()
py	def get_yValue ()

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.get_zValue()**

Retourne la vitesse angulaire autour de l'axe Z du module, sous forme de nombre à virgule.

js	function get_zValue ()
nodejs	function get_zValue ()
php	function get_zValue ()
cpp	double get_zValue ()
m	-(double) zValue
pas	function get_zValue (): double
vb	function get_zValue () As Double
cs	double get_zValue ()
java	double get_zValue ()
py	def get_zValue ()

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

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 (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.

gyro→loadCalibrationPoints()**YGyro****gyro.loadCalibrationPoints()**

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 nextGyro ()
nodejs	function nextGyro ()
php	function nextGyro ()
cpp	YGyro * nextGyro ()
m	-(YGyro*) nextGyro
pas	function nextGyro (): TYGyro
vb	function nextGyro () As YGyro
cs	YGyro nextGyro ()
java	YGyro nextGyro ()
py	def nextGyro ()

Retourne :

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

gyro→registerAnglesCallback()**YGyro****gyro.registerAnglesCallback()**

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

js	function registerAnglesCallback (callback)
nodejs	function registerAnglesCallback (callback)
php	function registerAnglesCallback (\$callback)
cpp	int registerAnglesCallback (YAnglesCallback callback)
m	-(int) registerAnglesCallback : (YAnglesCallback) callback
pas	function registerAnglesCallback (callback : TYAnglesCallback): LongInt
vb	function registerAnglesCallback () As Integer
cs	int registerAnglesCallback (YAnglesCallback callback)
java	int registerAnglesCallback (YAnglesCallback callback)
py	def registerAnglesCallback (callback)

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()**YGyro****gyro.registerQuaternionCallback()**

Enregistre une fonction de callback qui sera appelée à chaque changement de l'estimation de l'orientation du module.

js	function registerQuaternionCallback (callback)
nodejs	function registerQuaternionCallback (callback)
php	function registerQuaternionCallback (\$callback)
c++	int registerQuaternionCallback (YQuatCallback callback)
m	-(int) registerQuaternionCallback : (YQuatCallback) callback
pas	function registerQuaternionCallback (callback : TYQuatCallback): LongInt
vb	function registerQuaternionCallback () As Integer
cs	int registerQuaternionCallback (YQuatCallback callback)
java	int registerQuaternionCallback (YQuatCallback callback)
py	def registerQuaternionCallback (callback)

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()**YGyro****gyro.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
c++	int registerTimedReportCallback (YGyroTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YGyroTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYGyroTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YGyro****gyro.registerValueCallback()**

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 (YGyroValueCallback callback)
m	-(int) registerValueCallback : (YGyroValueCallback) callback
pas	function registerValueCallback (callback : TYGyroValueCallback): 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.

gyro→set_highestValue()**YGyro****gyro→setHighestValue()****gyro.set_highestValue()**

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	YGyro 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.

gyro→**set_logFrequency()****gyro**→**setLogFrequency()****gyro.set_logFrequency()**

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	YGyro 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.

gyro→set_logicalName()**YGyro****gyro→setLogicalName()**`gyro.set_logicalName()`

Modifie le nom logique du gyroscope.

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	YGyro 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

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

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()**YGyro****gyro→setReportFrequency()****gyro.set_reportFrequency()**

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	YGyro 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.

gyro→set_resolution()**YGyro****gyro→setResolution()**`gyro.set_resolution()`

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()`

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

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

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.
hubport→get_module()	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
hubport→get_module_async(callback, context)	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
hubport→get_portState()	Retourne l'état actuel du port de Yocto-hub.
hubport→get_userData()	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
hubport→isOnline()	Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.
hubport→isOnline_async(callback, context)	Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.
hubport→load(msValidity)	Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.
hubport→load_async(msValidity, callback, context)	Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.
hubport→nextHubPort()	Continue l'énumération des port de Yocto-hub commencée à l'aide de <code>yFirstHubPort()</code> .
hubport→registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
hubport→set_enabled(newval)	Modifie le mode d'activation du port du Yocto-hub.
hubport→set_logicalName(newval)	Modifie le nom logique du port de Yocto-hub.
hubport→set_userData(data)	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> .
hubport→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.

YHubPort.FindHubPort()**YHubPort****yFindHubPort()**`YHubPort.FindHubPort()`

Permet de retrouver un port de Yocto-hub d'après un identifiant donné.

<code>js</code>	<code>function yFindHubPort(func)</code>
<code>nodejs</code>	<code>function FindHubPort(func)</code>
<code>php</code>	<code>function yFindHubPort(\$func)</code>
<code>cpp</code>	<code>YHubPort* yFindHubPort(const string& func)</code>
<code>m</code>	<code>YHubPort* yFindHubPort(NSString* func)</code>
<code>pas</code>	<code>function yFindHubPort(func: string): TYHubPort</code>
<code>vb</code>	<code>function yFindHubPort(ByVal func As String) As YHubPort</code>
<code>cs</code>	<code>YHubPort FindHubPort(string func)</code>
<code>java</code>	<code>YHubPort FindHubPort(String func)</code>
<code>py</code>	<code>def FindHubPort(func)</code>

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()**YHubPort****yFirstHubPort()**`YHubPort.FirstHubPort()`

Commence l'énumération des port de Yocto-hub accessibles par la librairie.

js	function yFirstHubPort ()
nodejs	function FirstHubPort ()
php	function yFirstHubPort ()
cpp	YHubPort* yFirstHubPort ()
m	YHubPort* yFirstHubPort ()
pas	function yFirstHubPort (): TYHubPort
vb	function yFirstHubPort () As YHubPort
cs	YHubPort FirstHubPort ()
java	YHubPort FirstHubPort ()
py	def FirstHubPort ()

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

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()****YHubPort****hubport**→**advertisedValue()****hubport.get_advertisedValue()**

Retourne la valeur courante du port de Yocto-hub (pas plus de 6 caractères).

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

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.get_baudRate()**

Retourne la vitesse de transfert utilisée par le port de Yocto-hub, en kbps.

js	function get_baudRate ()
nodejs	function get_baudRate ()
php	function get_baudRate ()
cpp	int get_baudRate ()
m	-(int) baudRate
pas	function get_baudRate (): LongInt
vb	function get_baudRate () As Integer
cs	int get_baudRate ()
java	int get_baudRate ()
py	def get_baudRate ()
cmd	YHubPort target get_baudRate

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.get_enabled()**

Retourne vrai si le port du Yocto-hub est alimenté, faux sinon.

js	function get_enabled ()
nodejs	function get_enabled ()
php	function get_enabled ()
cpp	Y_ENABLED_enum get_enabled ()
m	-(Y_ENABLED_enum) enabled
pas	function get_enabled (): Integer
vb	function get_enabled () As Integer
cs	int get_enabled ()
java	int get_enabled ()
py	def get_enabled ()
cmd	YHubPort target get_enabled

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**.**get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

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

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()****hubport.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de Yocto-hub.

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

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.get_friendlyName()**

Retourne un identifiant global du port de Yocto-hub au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du port de Yocto-hub, sans référence au module.

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

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 get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_logicalName()**

Retourne le nom logique du port de Yocto-hub.

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

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.get_module()`

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

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

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 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.

hubport→**get_portState()****YHubPort****hubport**→**portState()****hubport.get_portState()**

Retourne l'état actuel du port de Yocto-hub.

js	function get_portState ()
nodejs	function get_portState ()
php	function get_portState ()
cpp	Y_PORTSTATE_enum get_portState ()
m	-(Y_PORTSTATE_enum) portState
pas	function get_portState (): Integer
vb	function get_portState () As Integer
cs	int get_portState ()
java	int get_portState ()
py	def get_portState ()
cmd	YHubPort target get_portState

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 get_userData ()
nodejs	function get_userData ()
php	function get_userData ()
cpp	void * get_userData ()
m	-(void*) userData
pas	function get_userData (): TObject
vb	function get_userData () As Object
cs	object get_userData ()
java	Object get_userData ()
py	def get_userData ()

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.

hubport→**isOnline()****hubport.isOnline()****YHubPort**

Vérifie si le module hébergeant le port de Yocto-hub est joignable, sans déclencher d'erreur.

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

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.

hubport→**load()**`hubport.load()`**YHubPort**

Met en cache les valeurs courantes du port de Yocto-hub, avec une durée de validité spécifiée.

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

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 nextHubPort ()
nodejs	function nextHubPort ()
php	function nextHubPort ()
cpp	YHubPort * nextHubPort ()
m	-(YHubPort*) nextHubPort
pas	function nextHubPort (): TYHubPort
vb	function nextHubPort () As YHubPort
cs	YHubPort nextHubPort ()
java	YHubPort nextHubPort ()
py	def nextHubPort ()

Retourne :

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

hubport→registerValueCallback()**YHubPort****hubport.registerValueCallback()**

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)
c++	int registerValueCallback (YHubPortValueCallback callback)
m	-(int) registerValueCallback : (YHubPortValueCallback) callback
pas	function registerValueCallback (callback : TYHubPortValueCallback): 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.

hubport→**set_enabled()****YHubPort****hubport**→**setEnabled()****hubport.setEnabled()**

Modifie le mode d'activation du port du Yocto-hub.

js	function set_enabled (newval)
nodejs	function set_enabled (newval)
php	function set_enabled (\$newval)
cpp	int set_enabled (Y_ENABLED_enum newval)
m	-(int) setEnabled : (Y_ENABLED_enum) newval
pas	function set_enabled (newval : Integer): integer
vb	function set_enabled (ByVal newval As Integer) As Integer
cs	int set_enabled (int newval)
java	int set_enabled (int newval)
py	def set_enabled (newval)
cmd	YHubPort target set_enabled newval

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()**YHubPort****hubport→setLogicalName()****hubport.set_logicalName()**

Modifie le nom logique du port de Yocto-hub.

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	YHubPort 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 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.set_userdata()**

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

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()**YHumidity****yFindHumidity()****YHumidity.FindHumidity()**

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

js	function yFindHumidity (func)
nodejs	function FindHumidity (func)
php	function yFindHumidity (\$func)
cpp	YHumidity* yFindHumidity (const string& func)
m	YHumidity* yFindHumidity (NSString* func)
pas	function yFindHumidity (func : string): TYHumidity
vb	function yFindHumidity (ByVal func As String) As YHumidity
cs	YHumidity FindHumidity (string func)
java	YHumidity FindHumidity (String func)
py	def FindHumidity (func)

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()**YHumidity****yFirstHumidity()****YHumidity.FirstHumidity()**

Commence l'énumération des capteurs d'humidité accessibles par la librairie.

js	function yFirstHumidity ()
nodejs	function FirstHumidity ()
php	function yFirstHumidity ()
cpp	YHumidity* yFirstHumidity ()
m	YHumidity* yFirstHumidity ()
pas	function yFirstHumidity (): TYHumidity
vb	function yFirstHumidity () As YHumidity
cs	YHumidity FirstHumidity ()
java	YHumidity FirstHumidity ()
py	def FirstHumidity ()

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()**YHumidity****humidity.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 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.

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

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()**YHumidity****humidity→advertisedValue()****humidity.get_advertisedValue()**

Retourne la valeur courante du capteur d'humidité (pas plus de 6 caractères).

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

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()****YHumidity****humidity**→**currentRawValue()****humidity.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YHumidity target get_currentRawValue

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.get_currentValue()**

Retourne la mesure actuelle de l'humidité.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YHumidity target get_currentValue

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.get_errorMessage()**

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

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

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()**YHumidity****humidity→errorType()humidity.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du capteur d'humidité au format `NOM_MODULE.NOM_FONCTION`.

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

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()**YHumidity****humidity→functionDescriptor()****humidity.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du capteur d'humidité, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du capteur d'humidité au format `SERIAL.FUNCTIONID`.

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

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**.**get_highestValue()**

Retourne la valeur maximale observée pour l'humidité.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YHumidity target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du capteur d'humidité.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour l'humidité.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YHumidity target get_lowestValue

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.get_module()**

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

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

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 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.

humidity→get_recordedData()**YHumidity****humidity→recordedData()****humidity.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YHumidity target get_recordedData startTime endTime

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()**YHumidity****humidity→reportFrequency()****humidity.get_reportFrequency()**

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

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
c++	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YHumidity target get_reportFrequency

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YHumidity target get_resolution

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.get_unit()**

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

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

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

humidity→isOnline()`humidity.isOnline()`**YHumidity**

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

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

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

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 (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.

humidity→loadCalibrationPoints()**YHumidity****humidity.loadCalibrationPoints()**

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 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()****YHumidity****humidity.nextHumidity()**

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()**YHumidity****humidity.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
c++	int registerTimedReportCallback (YHumidityTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YHumidityTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYHumidityTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YHumidity****humidity.registerValueCallback()**

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 (YHumidityValueCallback callback)
m	-(int) registerValueCallback : (YHumidityValueCallback) callback
pas	function registerValueCallback (callback : TYHumidityValueCallback): 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.

humidity→set_highestValue()**YHumidity****humidity→setHighestValue()****humidity.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour l'humidité.

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	YHumidity target set_highestValue newval

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()**YHumidity****humidity→setLogFrequency()****humidity.set_logFrequency()**

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	YHumidity 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.

humidity→set_logicalName()**YHumidity****humidity→setLogicalName()****humidity.set_logicalName()**

Modifie le nom logique du capteur d'humidité.

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	YHumidity 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 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()****YHumidity****humidity**→**setLowestValue()****humidity.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour l'humidité.

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

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()**YHumidity****humidity→setReportFrequency()****humidity.set_reportFrequency()**

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	YHumidity 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.

humidity→**set_resolution()****humidity**→**setResolution()****humidity.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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

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()**YLed****yFindLed()****YLed.FindLed()**

Permet de retrouver une led d'après un identifiant donné.

js	function yFindLed (func)
nodejs	function FindLed (func)
php	function yFindLed (\$func)
cpp	YLed* yFindLed (const string& func)
m	YLed* yFindLed (NSString* func)
pas	function yFindLed (func : string): TYLed
vb	function yFindLed (ByVal func As String) As YLed
cs	YLed FindLed (string func)
java	YLed FindLed (String func)
py	def FindLed (func)

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()**YLed****yFirstLed()****YLed.FirstLed()**

Commence l'énumération des leds accessibles par la librairie.

js	function yFirstLed ()
nodejs	function FirstLed ()
php	function yFirstLed ()
cpp	YLed* yFirstLed ()
m	YLed* yFirstLed ()
pas	function yFirstLed (): TYLed
vb	function yFirstLed () As YLed
cs	YLed FirstLed ()
java	YLed FirstLed ()
py	def FirstLed ()

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

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 (ex: `Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

led→**get_advertisedValue()****YLed****led**→**advertisedValue()****led.get_advertisedValue()**

Retourne la valeur courante de la led (pas plus de 6 caractères).

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

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.get_blinking()**

Retourne le mode de signalisation de la led.

js	function get_blinking ()
nodejs	function get_blinking ()
php	function get_blinking ()
cpp	Y_BLINKING_enum get_blinking ()
m	-(Y_BLINKING_enum) blinking
pas	function get_blinking (): Integer
vb	function get_blinking () As Integer
cs	int get_blinking ()
java	int get_blinking ()
py	def get_blinking ()
cmd	YLed target get_blinking

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()****YLed****led**→**errorMessage()****led.errorMessage()**

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

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

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()****led.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global de la led au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

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

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.get_logicalName()**

Retourne le nom logique de la led.

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

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.get_luminosity()**

Retourne l'intensité de la led en pour cent.

js	function get_luminosity ()
nodejs	function get_luminosity ()
php	function get_luminosity ()
cpp	int get_luminosity ()
m	-(int) luminosity
pas	function get_luminosity (): LongInt
vb	function get_luminosity () As Integer
cs	int get_luminosity ()
java	int get_luminosity ()
py	def get_luminosity ()
cmd	YLed target get_luminosity

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.get_module()**

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

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

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 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.

led→**get_power()****YLed****led**→**power()**`led.get_power()`

Retourne l'état courant de la led.

js	function get_power ()
nodejs	function get_power ()
php	function get_power ()
cpp	Y_POWER_enum get_power ()
m	-(Y_POWER_enum) power
pas	function get_power (): Integer
vb	function get_power () As Integer
cs	int get_power ()
java	int get_power ()
py	def get_power ()
cmd	YLed target get_power

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

led→**isOnline()****led.isOnline()****YLed**

Vérifie si le module hébergeant la led est joignable, sans déclencher d'erreur.

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

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 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.

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 nextLed ()
nodejs	function nextLed ()
php	function nextLed ()
cpp	YLed * nextLed ()
m	-(YLed*) nextLed
pas	function nextLed (): TYLed
vb	function nextLed () As YLed
cs	YLed nextLed ()
java	YLed nextLed ()
py	def nextLed ()

Retourne :

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

led→registerValueCallback()**YLed****led.registerValueCallback()**

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)
c++	int registerValueCallback (YLedValueCallback callback)
m	-(int) registerValueCallback : (YLedValueCallback) callback
pas	function registerValueCallback (callback : TYLedValueCallback): 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.

led→set_blinking()

YLed

led→setBlinking()`led.set_blinking()`

Modifie le mode de signalisation de la led.

js	function set_blinking (newval)
nodejs	function set_blinking (newval)
php	function set_blinking (\$newval)
cpp	int set_blinking (Y_BLINKING_enum newval)
m	-(int) setBlinking : (Y_BLINKING_enum) newval
pas	function set_blinking (newval : Integer): integer
vb	function set_blinking (ByVal newval As Integer) As Integer
cs	int set_blinking (int newval)
java	int set_blinking (int newval)
py	def set_blinking (newval)
cmd	YLed target set_blinking newval

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.set_logicalName()**

Modifie le nom logique de la led.

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	YLed 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 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.set_luminosity()**

Modifie l'intensité lumineuse de la led (en pour cent).

js	function set_luminosity (newval)
nodejs	function set_luminosity (newval)
php	function set_luminosity (\$newval)
cpp	int set_luminosity (int newval)
m	-(int) setLuminosity : (int) newval
pas	function set_luminosity (newval : LongInt): integer
vb	function set_luminosity (ByVal newval As Integer) As Integer
cs	int set_luminosity (int newval)
java	int set_luminosity (int newval)
py	def set_luminosity (newval)
cmd	YLed target set_luminosity newval

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.set_power()**

Modifie l'état courant de la led.

js	function set_power (newval)
nodejs	function set_power (newval)
php	function set_power (\$newval)
cpp	int set_power (Y_POWER_enum newval)
m	-(int) setPower : (Y_POWER_enum) newval
pas	function set_power (newval : Integer): integer
vb	function set_power (ByVal newval As Integer) As Integer
cs	int set_power (int newval)
java	int set_power (int newval)
py	def set_power (newval)
cmd	YLed target set_power newval

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.set_userdata()**

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

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

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()**YLightSensor****yFindLightSensor()****YLightSensor.FindLightSensor()**

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

js	function yFindLightSensor (func)
nodejs	function FindLightSensor (func)
php	function yFindLightSensor (\$func)
cpp	YLightSensor* yFindLightSensor (const string& func)
m	YLightSensor* yFindLightSensor (NSString* func)
pas	function yFindLightSensor (func : string): TYLightSensor
vb	function yFindLightSensor (ByVal func As String) As YLightSensor
cs	YLightSensor FindLightSensor (string func)
java	YLightSensor FindLightSensor (String func)
py	def FindLightSensor (func)

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()**YLightSensor****yFirstLightSensor()****YLightSensor.FirstLightSensor()**

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

js	function yFirstLightSensor ()
nodejs	function FirstLightSensor ()
php	function yFirstLightSensor ()
cpp	YLightSensor* yFirstLightSensor ()
m	YLightSensor* yFirstLightSensor ()
pas	function yFirstLightSensor (): TYLightSensor
vb	function yFirstLightSensor () As YLightSensor
cs	YLightSensor FirstLightSensor ()
java	YLightSensor FirstLightSensor ()
py	def FirstLightSensor ()

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).

js	function calibrate (calibratedVal)
nodejs	function calibrate (calibratedVal)
php	function calibrate (\$calibratedVal)
cpp	int calibrate (double calibratedVal)
m	-(int) calibrate : (double) calibratedVal
pas	function calibrate (calibratedVal : double): integer
vb	function calibrate (ByVal calibratedVal As Double) As Integer
cs	int calibrate (double calibratedVal)
java	int calibrate (double calibratedVal)
py	def calibrate (calibratedVal)
cmd	YLightSensor target calibrate calibratedVal

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()**YLightSensor****lightsensor.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 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.

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

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 lumière (ex:
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

lightsensor→**get_advertisedValue()****YLightSensor****lightsensor**→**advertisedValue()****lightsensor.get_advertisedValue()**

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

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

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()**YLightSensor****lightsensor→currentRawValue()****lightsensor.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YLightSensor target get_currentRawValue

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.get_currentValue()**

Retourne la mesure actuelle de la lumière ambiante.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YLightSensor target get_currentValue

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.get_errorMessage()**

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

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

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()****lightsensor.get_errorType()**

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

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

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()**YLightSensor****lightsensor→friendlyName()****lightsensor.get_friendlyName()**

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

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

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()****YLightSensor****lightsensor**→**functionDescriptor()****lightsensor.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

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

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()**YLightSensor****lightsensor→highestValue()****lightsensor.get_highestValue()**

Retourne la valeur maximale observée pour la lumière ambiante.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YLightSensor target get_highestValue

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()**YLightSensor****lightsensor→logFrequency()****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.

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

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.get_logicalName()**

Retourne le nom logique du capteur de lumière.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour la lumière ambiante.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YLightSensor target get_lowestValue

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.get_module()**

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

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YLightSensor target get_recordedData startTime endTime

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()**YLightSensor****lightsensor→reportFrequency()****lightsensor.get_reportFrequency()**

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

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

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YLightSensor target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle la lumière ambiante est exprimée.

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

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 get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
c++	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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

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 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.

lightsensor→loadCalibrationPoints()**YLightSensor****lightsensor.loadCalibrationPoints()**

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-qu'el à la fonction de callback

Retourne :

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

lightsensor→nextLightSensor()**YLightSensor****lightsensor.nextLightSensor()**

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()**YLightSensor****lightsensor.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YLightSensorTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YLightSensorTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYLightSensorTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YLightSensor****lightsensor.registerValueCallback()**

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 (YLightSensorValueCallback callback)
m	-(int) registerValueCallback : (YLightSensorValueCallback) callback
pas	function registerValueCallback (callback : TYLightSensorValueCallback): 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.

lightsensor→**set_highestValue()****YLightSensor****lightsensor**→**setHighestValue()****lightsensor.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour la lumière ambiante.

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	YLightSensor target set_highestValue newval

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.set_logFrequency()**

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	YLightSensor 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.

lightsensor→set_logicalName()**YLightSensor****lightsensor→setLogicalName()****lightsensor.set_logicalName()**

Modifie le nom logique du capteur de lumière.

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	YLightSensor 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 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()****YLightSensor****lightsensor**→**setLowestValue()****lightsensor.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour la lumière ambiante.

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

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.set_reportFrequency()**

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	YLightSensor 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.

lightsensor→set_resolution()**YLightSensor****lightsensor→setResolution()****lightsensor.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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()**YMagnetometer****yFindMagnetometer()****YMagnetometer.FindMagnetometer()**

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

js	function yFindMagnetometer (func)
nodejs	function FindMagnetometer (func)
php	function yFindMagnetometer (\$func)
cpp	YMagnetometer* yFindMagnetometer (const string& func)
m	YMagnetometer* yFindMagnetometer (NSString* func)
pas	function yFindMagnetometer (func : string): TYMagnetometer
vb	function yFindMagnetometer (ByVal func As String) As YMagnetometer
cs	YMagnetometer FindMagnetometer (string func)
java	YMagnetometer FindMagnetometer (String func)
py	def FindMagnetometer (func)

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()

YMagnetometer

yFirstMagnetometer()

YMagnetometer.FirstMagnetometer()

Commence l'énumération des magnétomètres accessibles par la librairie.

js	function yFirstMagnetometer ()
nodejs	function FirstMagnetometer ()
php	function yFirstMagnetometer ()
cpp	YMagnetometer* yFirstMagnetometer ()
m	YMagnetometer* yFirstMagnetometer ()
pas	function yFirstMagnetometer (): TYMagnetometer
vb	function yFirstMagnetometer () As YMagnetometer
cs	YMagnetometer FirstMagnetometer ()
java	YMagnetometer FirstMagnetometer ()
py	def FirstMagnetometer ()

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 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	YMagnetometer 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.

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()**YMagnetometer****magnetometer.describe()**

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

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

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 magnétomètre (ex:
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

magnetometer→get_advertisedValue()**YMagnetometer****magnetometer→advertisedValue()****magnetometer.get_advertisedValue()**

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

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

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.get_currentRawValue()**

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

js	function get_currentRawValue()
nodejs	function get_currentRawValue()
php	function get_currentRawValue()
cpp	double get_currentRawValue()
m	-(double) currentRawValue
pas	function get_currentRawValue() : double
vb	function get_currentRawValue() As Double
cs	double get_currentRawValue()
java	double get_currentRawValue()
py	def get_currentRawValue()
cmd	YMagnetometer target get_currentRawValue

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()**YMagnetometer****magnetometer→currentValue()****magnetometer.get_currentValue()**

Retourne la valeur actuelle du champ magnétique.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YMagnetometer target get_currentValue

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.get_errorMessage()**

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

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

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()**YMagnetometer****magnetometer→errorType()****magnetometer.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du magnétomètre au format `NOM_MODULE.NOM_FONCTION`.

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

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()**YMagnetometer****magnetometer→functionDescriptor()****magnetometer.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du magnétomètre, sans référence au module.

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

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()**YMagnetometer****magnetometer→hardwareId()****magnetometer.get_hardwareId()**

Retourne l'identifiant matériel unique du magnétomètre au format SERIAL.FUNCTIONID.

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

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.get_highestValue()**

Retourne la valeur maximale observée pour le champ magnétique depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YMagnetometer target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du magnétomètre.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour le champ magnétique depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YMagnetometer target get_lowestValue

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.get_module()**

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

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

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 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-qu'el 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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
c++	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YMagnetometer target get_recordedData startTime endTime

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()**YMagnetometer****magnetometer→reportFrequency()****magnetometer.get_reportFrequency()**

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

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

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YMagnetometer target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle le champ magnétique est exprimée.

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

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

magnetometer→get_xValue()**YMagnetometer****magnetometer→xValue()****magnetometer.get_xValue()**

Retourne la composante X du champ magnétique, sous forme de nombre à virgule.

js	function get_xValue ()
nodejs	function get_xValue ()
php	function get_xValue ()
cpp	double get_xValue ()
m	-(double) xValue
pas	function get_xValue (): double
vb	function get_xValue () As Double
cs	double get_xValue ()
java	double get_xValue ()
py	def get_xValue ()
cmd	YMagnetometer target get_xValue

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.get_yValue()**

Retourne la composante Y du champ magnétique, sous forme de nombre à virgule.

js	function get_yValue ()
nodejs	function get_yValue ()
php	function get_yValue ()
cpp	double get_yValue ()
m	-(double) yValue
pas	function get_yValue (): double
vb	function get_yValue () As Double
cs	double get_yValue ()
java	double get_yValue ()
py	def get_yValue ()
cmd	YMagnetometer target get_yValue

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.get_zValue()**

Retourne la composante Z du champ magnétique, sous forme de nombre à virgule.

js	function get_zValue ()
nodejs	function get_zValue ()
php	function get_zValue ()
cpp	double get_zValue ()
m	-(double) zValue
pas	function get_zValue (): double
vb	function get_zValue () As Double
cs	double get_zValue ()
java	double get_zValue ()
py	def get_zValue ()
cmd	YMagnetometer target get_zValue

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()**YMagnetometer****magnetometer.isOnline()**

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

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

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 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.

magnetometer→loadCalibrationPoints()**YMagnetometer****magnetometer.loadCalibrationPoints()**

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	YMagnetometer 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.

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()****YMagnetometer****magnetometer.nextMagnetometer()**

Continue l'énumération des magnétomètres commencée à l'aide de `yFirstMagnetometer()`.

js	function nextMagnetometer ()
nodejs	function nextMagnetometer ()
php	function nextMagnetometer ()
cpp	YMagnetometer * nextMagnetometer ()
m	-(YMagnetometer*) nextMagnetometer
pas	function nextMagnetometer (): TYMagnetometer
vb	function nextMagnetometer () As YMagnetometer
cs	YMagnetometer nextMagnetometer ()
java	YMagnetometer nextMagnetometer ()
py	def nextMagnetometer ()

Retourne :

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

magnetometer→registerTimedReportCallback()**YMagnetometer****magnetometer.registerTimedReportCallback()**

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

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

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 registerValueCallback (callback)
nodejs	function registerValueCallback (callback)
php	function registerValueCallback (\$callback)
c++	int registerValueCallback (YMagnetometerValueCallback callback)
m	-(int) registerValueCallback : (YMagnetometerValueCallback) callback
pas	function registerValueCallback (callback : TYMagnetometerValueCallback): 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'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

magnetometer→**set_highestValue()****YMagnetometer****magnetometer**→**setHighestValue()****magnetometer.set_highestValue()**

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.set_logFrequency()**

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	YMagnetometer 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.

magnetometer→**set_logicalName()****YMagnetometer****magnetometer**→**setLogicalName()****magnetometer.set_logicalName()**

Modifie le nom logique du magnétomètre.

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	YMagnetometer 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

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

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.set_reportFrequency()**

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	YMagnetometer 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.

magnetometer→set_resolution()**YMagnetometer****magnetometer→setResolution()****magnetometer.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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.get_averageValue()**

Retourne la valeur moyenne observée durant l'intervalle de temps couvert par la mesure.

js	function get_averageValue ()
nodejs	function get_averageValue ()
php	function get_averageValue ()
cpp	double get_averageValue ()
m	-(double) averageValue
pas	function get_averageValue (): double
vb	function get_averageValue () As Double
cs	double get_averageValue ()
java	double get_averageValue ()
py	def get_averageValue ()

Retourne :

un nombre décimal correspondant à la valeur moyenne observée.

measure→**get_endTimeUTC()**

YMeasure

measure→**endTimeUTC()**

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).

js	function get_endTimeUTC ()
nodejs	function get_endTimeUTC ()
php	function get_endTimeUTC ()
cpp	double get_endTimeUTC ()
m	-(double) endTimeUTC
pas	function get_endTimeUTC (): double
vb	function get_endTimeUTC () As Double
cs	double get_endTimeUTC ()
java	double get_endTimeUTC ()
py	def get_endTimeUTC ()

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.getMaxValue()`

Retourne la plus grande valeur observée durant l'intervalle de temps couvert par la mesure.

js	function get_maxValue ()
nodejs	function get_maxValue ()
php	function get_maxValue ()
cpp	double get_maxValue ()
m	-(double) maxValue
pas	function get_maxValue (): double
vb	function get_maxValue () As Double
cs	double get_maxValue ()
java	double get_maxValue ()
py	def get_maxValue ()

Retourne :

un nombre décimal correspondant à la plus grande valeur observée.

measure→**get_minValue()**

YMeasure

measure→**minValue()**`measure.get_minValue()`

Retourne la plus petite valeur observée durant l'intervalle de temps couvert par la mesure.

js	function get_minValue ()
nodejs	function get_minValue ()
php	function get_minValue ()
cpp	double get_minValue ()
m	-(double) minValue
pas	function get_minValue (): double
vb	function get_minValue () As Double
cs	double get_minValue ()
java	double get_minValue ()
py	def get_minValue ()

Retourne :

un nombre décimal correspondant à la plus petite valeur observée.

measure→**get_startTimeUTC()****YMeasure****measure**→**startTimeUTC()****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).

js	function get_startTimeUTC ()
nodejs	function get_startTimeUTC ()
php	function get_startTimeUTC ()
c++	double get_startTimeUTC ()
m	-(double) startTimeUTC
pas	function get_startTimeUTC (): double
vb	function get_startTimeUTC () As Double
cs	double get_startTimeUTC ()
java	double get_startTimeUTC ()
py	def get_startTimeUTC ()

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
yFindModule(func) Permet de retrouver un module d'après son numéro de série ou son nom logique.
yFirstModule() Commence l'énumération des modules accessibles par la librairie.
Méthodes des objets YModule
module→describe() Retourne un court texte décrivant le module.
module→download(pathname) Télécharge le fichier choisi du module et retourne son contenu.
module→functionCount() Retourne le nombre de fonctions (sans compter l'interface "module") existant sur le module.
module→functionId(functionIndex) Retourne l'identifiant matériel de la <i>nième</i> fonction du module.
module→functionName(functionIndex) Retourne le nom logique de la <i>nième</i> fonction du module.
module→functionValue(functionIndex) Retourne la valeur publiée par la <i>nième</i> fonction du module.
module→get_beacon() Retourne l'état de la balise de localisation.
module→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.
module→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'objet module.
module→get_firmwareRelease() Retourne la version du logiciel embarqué du module.
module→get_hardwareId() Retourne l'identifiant unique du module.
module→get_icon2d()

	Retourne l'icône du module.
module → get_lastLogs()	Retourne une chaîne de caractère contenant les derniers logs du module.
module → get_logicalName()	Retourne le nom logique du module.
module → get_luminosity()	Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).
module → get_persistentSettings()	Retourne l'état courant des réglages persistents du module.
module → get_productId()	Retourne l'identifiant USB du module, préprogrammé en usine.
module → get_productName()	Retourne le nom commercial du module, préprogrammé en usine.
module → get_productRelease()	Retourne le numéro de version matériel du module, préprogrammé en usine.
module → get_rebootCountdown()	Retourne le nombre de secondes restantes avant un redémarrage du module, ou zéro si aucun redémarrage n'a été agendé.
module → get_serialNumber()	Retourne le numéro de série du module, préprogrammé en usine.
module → get_upTime()	Retourne le nombre de millisecondes écoulées depuis la mise sous tension du module
module → get_usbBandwidth()	Retourne le nombre d'interface USB utilisé par le module.
module → get_usbCurrent()	Retourne le courant consommé par le module sur le bus USB, en milliampères.
module → get_userData()	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode <code>set_userData</code> .
module → isOnline()	Vérifie si le module est joignable, sans déclencher d'erreur.
module → isOnline_async(callback, context)	Vérifie si le module est joignable, sans déclencher d'erreur.
module → load(msValidity)	Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.
module → load_async(msValidity, callback, context)	Met en cache les valeurs courantes du module, avec une durée de validité spécifiée.
module → nextModule()	Continue l'énumération des modules commencée à l'aide de <code>yFirstModule()</code> .
module → reboot(secBeforeReboot)	Agende un simple redémarrage du module dans un nombre donné de secondes.
module → registerLogCallback(callback)	todo
module → revertFromFlash()	Recharge les réglages stockés dans le mémoire non volatile du module, comme à la mise sous tension du module.
module → saveToFlash()	

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()**YModule****yFindModule()**`YModule.FindModule()`

Permet de retrouver un module d'après son numéro de série ou son nom logique.

js	function yFindModule (func)
nodejs	function FindModule (func)
php	function yFindModule (\$func)
cpp	YModule* yFindModule (string func)
m	+(YModule*) yFindModule : (NSString*) func
pas	function yFindModule (func : string): TYModule
vb	function yFindModule (ByVal func As String) As YModule
cs	YModule FindModule (string func)
java	YModule FindModule (String func)
py	def FindModule (func)

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()**YModule****yFirstModule()**`YModule.FirstModule()`

Commence l'énumération des modules accessibles par la librairie.

js	function yFirstModule ()
nodejs	function FirstModule ()
php	function yFirstModule ()
cpp	YModule* yFirstModule ()
m	YModule* yFirstModule ()
pas	function yFirstModule (): TYModule
vb	function yFirstModule () As YModule
cs	YModule FirstModule ()
java	YModule FirstModule ()
py	def FirstModule ()

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

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()**YModule**

Télécharge le fichier choisi du module et retourne son contenu.

js	function download (pathname)
nodejs	function download (pathname)
php	function download (\$pathname)
cpp	string download (string pathname)
m	-(NSData*) download : (NSString*) pathname
pas	function download (pathname : string): TByteArray
vb	function download () As Byte
py	def download (pathname)
cmd	YModule target download pathname

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()****YModule**

Retourne le nombre de fonctions (sans compter l'interface "module") existant sur le module.

js	function functionCount ()
nodejs	function functionCount ()
php	function functionCount ()
cpp	int functionCount ()
m	-(int) functionCount
pas	function functionCount (): integer
vb	function functionCount () As Integer
cs	int functionCount ()
py	def functionCount ()

Retourne :

le nombre de fonctions sur le module

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

module→**functionId()****YModule**

Retourne l'identifiant matériel de la *n*ième fonction du module.

js	function functionId (functionIndex)
nodejs	function functionId (functionIndex)
php	function functionId (\$functionIndex)
cpp	string functionId (int functionIndex)
m	-(NSString*) functionId : (int) functionIndex
pas	function functionId (functionIndex : integer): string
vb	function functionId (ByVal functionIndex As Integer) As String
cs	string functionId (int functionIndex)
py	def functionId (functionIndex)

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()****YModule**

Retourne le nom logique de la *nième* fonction du module.

js	function functionName (functionIndex)
nodejs	function functionName (functionIndex)
php	function functionName (\$functionIndex)
cpp	string functionName (int functionIndex)
m	-(NSString*) functionName : (int) functionIndex
pas	function functionName (functionIndex : integer): string
vb	function functionName (ByVal functionIndex As Integer) As String
cs	string functionName (int functionIndex)
py	def functionName (functionIndex)

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()****YModule**

Retourne la valeur publiée par la *nième* fonction du module.

js	function functionValue (functionIndex)
nodejs	function functionValue (functionIndex)
php	function functionValue (\$functionIndex)
cpp	string functionValue (int functionIndex)
m	-(NSString*) functionValue : (int) functionIndex
pas	function functionValue (functionIndex : integer): string
vb	function functionValue (ByVal functionIndex As Integer) As String
cs	string functionValue (int functionIndex)
py	def functionValue (functionIndex)

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.get_beacon()**

Retourne l'état de la balise de localisation.

js	function get_beacon ()
nodejs	function get_beacon ()
php	function get_beacon ()
cpp	Y_BEACON_enum get_beacon ()
m	-(Y_BEACON_enum) beacon
pas	function get_beacon (): Integer
vb	function get_beacon () As Integer
cs	int get_beacon ()
java	int get_beacon ()
py	def get_beacon ()
cmd	YModule target get_beacon

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

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()****YModule****module**→**errorType()**`module.get_errorType()`

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

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

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()****YModule****module**→**firmwareRelease()****module.get_firmwareRelease()**

Retourne la version du logiciel embarqué du module.

js	function get_firmwareRelease ()
nodejs	function get_firmwareRelease ()
php	function get_firmwareRelease ()
cpp	string get_firmwareRelease ()
m	-(NSString*) firmwareRelease
pas	function get_firmwareRelease (): string
vb	function get_firmwareRelease () As String
cs	string get_firmwareRelease ()
java	String get_firmwareRelease ()
py	def get_firmwareRelease ()
cmd	YModule target get_firmwareRelease

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

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()**

Retourne l'icône du module.

js	function get_icon2d ()
nodejs	function get_icon2d ()
php	function get_icon2d ()
cpp	string get_icon2d ()
m	-(NSData*) icon2d
pas	function get_icon2d (): TByteArray
vb	function get_icon2d () As Byte
py	def get_icon2d ()
cmd	YModule target get_icon2d

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.get_lastLogs()`

Retourne une chaine de caractère contenant les derniers logs du module.

js	function get_lastLogs()
nodejs	function get_lastLogs()
php	function get_lastLogs()
cpp	string get_lastLogs()
m	-(NSString*) lastLogs
pas	function get_lastLogs() : string
vb	function get_lastLogs() As String
cs	string get_lastLogs()
java	String get_lastLogs()
py	def get_lastLogs()
cmd	YModule target get_lastLogs

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.get_logicalName()**

Retourne le nom logique du module.

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

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.get_luminosity()`

Retourne la luminosité des leds informatives du module (valeur entre 0 et 100).

js	function get_luminosity ()
nodejs	function get_luminosity ()
php	function get_luminosity ()
cpp	int get_luminosity ()
m	-(int) luminosity
pas	function get_luminosity (): LongInt
vb	function get_luminosity () As Integer
cs	int get_luminosity ()
java	int get_luminosity ()
py	def get_luminosity ()
cmd	YModule target get_luminosity

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()****YModule****module**→**persistentSettings()****module.get_persistentSettings()**

Retourne l'état courant des réglages persistents du module.

js	function get_persistentSettings ()
nodejs	function get_persistentSettings ()
php	function get_persistentSettings ()
cpp	Y_PERSISTENTSETTINGS_enum get_persistentSettings ()
m	-(Y_PERSISTENTSETTINGS_enum) persistentSettings
pas	function get_persistentSettings (): Integer
vb	function get_persistentSettings () As Integer
cs	int get_persistentSettings ()
java	int get_persistentSettings ()
py	def get_persistentSettings ()
cmd	YModule target get_persistentSettings

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.get_productId()`

Retourne l'identifiant USB du module, préprogrammé en usine.

js	function get_productId ()
nodejs	function get_productId ()
php	function get_productId ()
cpp	int get_productId ()
m	-(int) productId
pas	function get_productId (): LongInt
vb	function get_productId () As Integer
cs	int get_productId ()
java	int get_productId ()
py	def get_productId ()
cmd	YModule target get_productId

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.get_productName()**

Retourne le nom commercial du module, préprogrammé en usine.

js	function get_productName ()
nodejs	function get_productName ()
php	function get_productName ()
cpp	string get_productName ()
m	-(NSString*) productName
pas	function get_productName (): string
vb	function get_productName () As String
cs	string get_productName ()
java	String get_productName ()
py	def get_productName ()
cmd	YModule target get_productName

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.get_productRelease()**

Retourne le numéro de version matériel du module, préprogrammé en usine.

js	function get_productRelease()
nodejs	function get_productRelease()
php	function get_productRelease()
cpp	int get_productRelease()
m	-(int) productRelease
pas	function get_productRelease() : LongInt
vb	function get_productRelease() As Integer
cs	int get_productRelease()
java	int get_productRelease()
py	def get_productRelease()
cmd	YModule target get_productRelease

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()****YModule****module**→**rebootCountdown()****module.get_rebootCountdown()**

Retourne le nombre de secondes restantes avant un redémarrage du module, ou zéro si aucun redémarrage n'a été agendé.

js	function get_rebootCountdown ()
nodejs	function get_rebootCountdown ()
php	function get_rebootCountdown ()
cpp	int get_rebootCountdown ()
m	-(int) rebootCountdown
pas	function get_rebootCountdown (): LongInt
vb	function get_rebootCountdown () As Integer
cs	int get_rebootCountdown ()
java	int get_rebootCountdown ()
py	def get_rebootCountdown ()
cmd	YModule target get_rebootCountdown

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.get_serialNumber()**

Retourne le numéro de série du module, préprogrammé en usine.

js	function get_serialNumber ()
nodejs	function get_serialNumber ()
php	function get_serialNumber ()
cpp	string get_serialNumber ()
m	-(NSString*) serialNumber
pas	function get_serialNumber (): string
vb	function get_serialNumber () As String
cs	string get_serialNumber ()
java	String get_serialNumber ()
py	def get_serialNumber ()
cmd	YModule target get_serialNumber

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.get_upTime()`

Retourne le nombre de millisecondes écoulées depuis la mise sous tension du module

js	function get_upTime ()
nodejs	function get_upTime ()
php	function get_upTime ()
cpp	s64 get_upTime ()
m	-(s64) upTime
pas	function get_upTime (): int64
vb	function get_upTime () As Long
cs	long get_upTime ()
java	long get_upTime ()
py	def get_upTime ()
cmd	YModule target get_upTime

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.get_usbBandwidth()**

Retourne le nombre d'interface USB utilisé par le module.

js	function get_usbBandwidth ()
nodejs	function get_usbBandwidth ()
php	function get_usbBandwidth ()
cpp	Y_USBBANDWIDTH_enum get_usbBandwidth ()
m	-(Y_USBBANDWIDTH_enum) usbBandwidth
pas	function get_usbBandwidth (): Integer
vb	function get_usbBandwidth () As Integer
cs	int get_usbBandwidth ()
java	int get_usbBandwidth ()
py	def get_usbBandwidth ()
cmd	YModule target get_usbBandwidth

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.get_usbCurrent()**

Retourne le courant consommé par le module sur le bus USB, en milliampères.

js	function get_usbCurrent ()
nodejs	function get_usbCurrent ()
php	function get_usbCurrent ()
cpp	int get_usbCurrent ()
m	-(int) usbCurrent
pas	function get_usbCurrent (): LongInt
vb	function get_usbCurrent () As Integer
cs	int get_usbCurrent ()
java	int get_usbCurrent ()
py	def get_usbCurrent ()
cmd	YModule target get_usbCurrent

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 get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
c++	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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

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 load (msValidity)
nodejs	function load (msValidity)
php	function load (\$msValidity)
c++	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.

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 nextModule ()
nodejs	function nextModule ()
php	function nextModule ()
cpp	YModule * nextModule ()
m	-(YModule*) nextModule
pas	function nextModule (): TYModule
vb	function nextModule () As YModule
cs	YModule nextModule ()
java	YModule nextModule ()
py	def nextModule ()

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 reboot (secBeforeReboot)
nodejs	function reboot (secBeforeReboot)
php	function reboot (\$secBeforeReboot)
cpp	int reboot (int secBeforeReboot)
m	-(int) reboot : (int) secBeforeReboot
pas	function reboot (secBeforeReboot : LongInt): LongInt
vb	function reboot () As Integer
cs	int reboot (int secBeforeReboot)
java	int reboot (int secBeforeReboot)
py	def reboot (secBeforeReboot)
cmd	YModule target reboot secBeforeReboot

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()**YModule**

todo

cpp	void registerLogCallback (YModuleLogCallback callback)
m	-(void) registerLogCallback : (YModuleLogCallback) callback
vb	function registerLogCallback (ByVal callback As YModuleLogCallback) As Integer
cs	int registerLogCallback (LogCallback callback)
py	def registerLogCallback (callback)

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()**YModule****module.revertFromFlash()**

Recharge les réglages stockés dans le mémoire non volatile du module, comme à la mise sous tension du module.

js	function revertFromFlash ()
nodejs	function revertFromFlash ()
php	function revertFromFlash ()
cpp	int revertFromFlash ()
m	-(int) revertFromFlash
pas	function revertFromFlash (): LongInt
vb	function revertFromFlash () As Integer
cs	int revertFromFlash ()
java	int revertFromFlash ()
py	def revertFromFlash ()
cmd	YModule target revertFromFlash

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 saveToFlash ()
nodejs	function saveToFlash ()
php	function saveToFlash ()
cpp	int saveToFlash ()
m	-(int) saveToFlash
pas	function saveToFlash (): LongInt
vb	function saveToFlash () As Integer
cs	int saveToFlash ()
java	int saveToFlash ()
py	def saveToFlash ()
cmd	YModule target saveToFlash

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.set_beacon()**

Allume ou éteint la balise de localisation du module.

js	function set_beacon (newval)
nodejs	function set_beacon (newval)
php	function set_beacon (\$newval)
cpp	int set_beacon (Y_BEACON_enum newval)
m	-(int) setBeacon : (Y_BEACON_enum) newval
pas	function set_beacon (newval : Integer): integer
vb	function set_beacon (ByVal newval As Integer) As Integer
cs	int set_beacon (int newval)
java	int set_beacon (int newval)
py	def set_beacon (newval)
cmd	YModule target set_beacon newval

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.set_logicalName()**

Change le nom logique du module.

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	YModule 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**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.set_luminosity()**

Modifie la luminosité des leds informatives du module.

js	function set_luminosity (newval)
nodejs	function set_luminosity (newval)
php	function set_luminosity (\$newval)
cpp	int set_luminosity (int newval)
m	-(int) setLuminosity : (int) newval
pas	function set_luminosity (newval : LongInt): integer
vb	function set_luminosity (ByVal newval As Integer) As Integer
cs	int set_luminosity (int newval)
java	int set_luminosity (int newval)
py	def set_luminosity (newval)
cmd	YModule target set_luminosity newval

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()****YModule****module**→**setUsbBandwidth()****module.set_usbBandwidth()**

Modifie le nombre d'interface USB utilisé par le module.

js	function set_usbBandwidth (newval)
nodejs	function set_usbBandwidth (newval)
php	function set_usbBandwidth (\$newval)
cpp	int set_usbBandwidth (Y_USBBANDWIDTH_enum newval)
m	-(int) setUsbBandwidth : (Y_USBBANDWIDTH_enum) newval
pas	function set_usbBandwidth (newval : Integer): integer
vb	function set_usbBandwidth (ByVal newval As Integer) As Integer
cs	int set_usbBandwidth (int newval)
java	int set_usbBandwidth (int newval)
py	def set_usbBandwidth (newval)
cmd	YModule target set_usbBandwidth newval

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.set_userdata()**

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

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()**YModule****module.triggerFirmwareUpdate()**

Agende un redémarrage du module en mode spécial de reprogrammation du logiciel embarqué.

js	function triggerFirmwareUpdate (secBeforeReboot)
nodejs	function triggerFirmwareUpdate (secBeforeReboot)
php	function triggerFirmwareUpdate (\$secBeforeReboot)
cpp	int triggerFirmwareUpdate (int secBeforeReboot)
m	-(int) triggerFirmwareUpdate : (int) secBeforeReboot
pas	function triggerFirmwareUpdate (secBeforeReboot : LongInt): LongInt
vb	function triggerFirmwareUpdate () As Integer
cs	int triggerFirmwareUpdate (int secBeforeReboot)
java	int triggerFirmwareUpdate (int secBeforeReboot)
py	def triggerFirmwareUpdate (secBeforeReboot)
cmd	YModule target triggerFirmwareUpdate secBeforeReboot

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()**YNetwork****yFindNetwork()** `YNetwork.FindNetwork()`

Permet de retrouver une interface réseau d'après un identifiant donné.

js	function yFindNetwork (func)
nodejs	function FindNetwork (func)
php	function yFindNetwork (\$func)
cpp	YNetwork* yFindNetwork (const string& func)
m	YNetwork* yFindNetwork (NSString* func)
pas	function yFindNetwork (func : string): TYNetwork
vb	function yFindNetwork (ByVal func As String) As YNetwork
cs	YNetwork FindNetwork (string func)
java	YNetwork FindNetwork (String func)
py	def FindNetwork (func)

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()**YNetwork****yFirstNetwork()**`YNetwork.FirstNetwork()`

Commence l'énumération des interfaces réseau accessibles par la librairie.

js	function yFirstNetwork ()
nodejs	function FirstNetwork ()
php	function yFirstNetwork ()
cpp	YNetwork* yFirstNetwork ()
m	YNetwork* yFirstNetwork ()
pas	function yFirstNetwork (): TYNetwork
vb	function yFirstNetwork () As YNetwork
cs	YNetwork FirstNetwork ()
java	YNetwork FirstNetwork ()
py	def FirstNetwork ()

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()**

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

js	function callbackLogin (username , password)
nodejs	function callbackLogin (username , password)
php	function callbackLogin (\$username , \$password)
c++	int callbackLogin (string username , string password)
m	-(int) callbackLogin : (NSString*) username : (NSString*) password
pas	function callbackLogin (username : string, password : string): integer
vb	function callbackLogin (ByVal username As String, ByVal password As String) As Integer
cs	int callbackLogin (string username , string password)
java	int callbackLogin (String username , String password)
py	def callbackLogin (username , password)
cmd	YNetwork target callbackLogin username password

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

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()****YNetwork****network**→**adminPassword()****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.

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()**YNetwork****network→advertisedValue()****network.get_advertisedValue()**

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

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

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()**YNetwork****network→callbackCredentials()****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.

js	function get_callbackCredentials ()
nodejs	function get_callbackCredentials ()
php	function get_callbackCredentials ()
cpp	string get_callbackCredentials ()
m	-(NSString*) callbackCredentials
pas	function get_callbackCredentials (): string
vb	function get_callbackCredentials () As String
cs	string get_callbackCredentials ()
java	String get_callbackCredentials ()
py	def get_callbackCredentials ()
cmd	YNetwork target get_callbackCredentials

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.get_callbackEncoding()**

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function get_callbackEncoding ()
nodejs	function get_callbackEncoding ()
php	function get_callbackEncoding ()
cpp	Y_CALLBACKENCODING_enum get_callbackEncoding ()
m	-(Y_CALLBACKENCODING_enum) callbackEncoding
pas	function get_callbackEncoding (): Integer
vb	function get_callbackEncoding () As Integer
cs	int get_callbackEncoding ()
java	int get_callbackEncoding ()
py	def get_callbackEncoding ()
cmd	YNetwork target get_callbackEncoding

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.get_callbackMaxDelay()**

Retourne l'attente maximale entre deux notifications par callback, en secondes.

js	function get_callbackMaxDelay ()
nodejs	function get_callbackMaxDelay ()
php	function get_callbackMaxDelay ()
cpp	int get_callbackMaxDelay ()
m	-(int) callbackMaxDelay
pas	function get_callbackMaxDelay (): LongInt
vb	function get_callbackMaxDelay () As Integer
cs	int get_callbackMaxDelay ()
java	int get_callbackMaxDelay ()
py	def get_callbackMaxDelay ()
cmd	YNetwork target get_callbackMaxDelay

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.get_callbackMethod()**

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

js	function get_callbackMethod ()
nodejs	function get_callbackMethod ()
php	function get_callbackMethod ()
cpp	Y_CALLBACKMETHOD_enum get_callbackMethod ()
m	-(Y_CALLBACKMETHOD_enum) callbackMethod
pas	function get_callbackMethod (): Integer
vb	function get_callbackMethod () As Integer
cs	int get_callbackMethod ()
java	int get_callbackMethod ()
py	def get_callbackMethod ()
cmd	YNetwork target get_callbackMethod

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.get_callbackMinDelay()**

Retourne l'attente minimale entre deux notifications par callback, en secondes.

js	function get_callbackMinDelay ()
nodejs	function get_callbackMinDelay ()
php	function get_callbackMinDelay ()
cpp	int get_callbackMinDelay ()
m	-(int) callbackMinDelay
pas	function get_callbackMinDelay (): LongInt
vb	function get_callbackMinDelay () As Integer
cs	int get_callbackMinDelay ()
java	int get_callbackMinDelay ()
py	def get_callbackMinDelay ()
cmd	YNetwork target get_callbackMinDelay

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.get_callbackUrl()**

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

js	function get_callbackUrl ()
nodejs	function get_callbackUrl ()
php	function get_callbackUrl ()
cpp	string get_callbackUrl ()
m	-(NSString*) callbackUrl
pas	function get_callbackUrl (): string
vb	function get_callbackUrl () As String
cs	string get_callbackUrl ()
java	String get_callbackUrl ()
py	def get_callbackUrl ()
cmd	YNetwork target get_callbackUrl

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.get_discoverable()**

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

js	function get_discoverable ()
nodejs	function get_discoverable ()
php	function get_discoverable ()
cpp	Y_DISCOVERABLE_enum get_discoverable ()
m	-(Y_DISCOVERABLE_enum) discoverable
pas	function get_discoverable (): Integer
vb	function get_discoverable () As Integer
cs	int get_discoverable ()
java	int get_discoverable ()
py	def get_discoverable ()
cmd	YNetwork target get_discoverable

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.get_errorMessage()**

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

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

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()****network.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_ipAddress()`

Retourne l'adresse IP utilisée par le module Yoctopuce.

js	function get_ipAddress ()
nodejs	function get_ipAddress ()
php	function get_ipAddress ()
cpp	string get_ipAddress ()
m	-(NSString*) ipAddress
pas	function get_ipAddress (): string
vb	function get_ipAddress () As String
cs	string get_ipAddress ()
java	String get_ipAddress ()
py	def get_ipAddress ()
cmd	YNetwork target get_ipAddress

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()****network**→**logicalName()****network.get_logicalName()**

Retourne le nom logique de l'interface réseau.

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

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.get_macAddress()**

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

js	function get_macAddress ()
nodejs	function get_macAddress ()
php	function get_macAddress ()
cpp	string get_macAddress ()
m	-(NSString*) macAddress
pas	function get_macAddress (): string
vb	function get_macAddress () As String
cs	string get_macAddress ()
java	String get_macAddress ()
py	def get_macAddress ()
cmd	YNetwork target get_macAddress

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.get_module()`

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

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

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 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.

network→**get_poeCurrent()****YNetwork****network**→**poeCurrent()****network.get_poeCurrent()**

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

js	function get_poeCurrent ()
nodejs	function get_poeCurrent ()
php	function get_poeCurrent ()
cpp	int get_poeCurrent ()
m	-(int) poeCurrent
pas	function get_poeCurrent (): LongInt
vb	function get_poeCurrent () As Integer
cs	int get_poeCurrent ()
java	int get_poeCurrent ()
py	def get_poeCurrent ()
cmd	YNetwork target get_poeCurrent

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.get_primaryDNS()**

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function get_primaryDNS ()
nodejs	function get_primaryDNS ()
php	function get_primaryDNS ()
cpp	string get_primaryDNS ()
m	-(NSString*) primaryDNS
pas	function get_primaryDNS (): string
vb	function get_primaryDNS () As String
cs	string get_primaryDNS ()
java	String get_primaryDNS ()
py	def get_primaryDNS ()
cmd	YNetwork target get_primaryDNS

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.get_readiness()**

Retourne l'état de fonctionnement atteint par l'interface réseau.

js	function get_readiness ()
nodejs	function get_readiness ()
php	function get_readiness ()
cpp	Y_READINESS_enum get_readiness ()
m	-(Y_READINESS_enum) readiness
pas	function get_readiness (): Integer
vb	function get_readiness () As Integer
cs	int get_readiness ()
java	int get_readiness ()
py	def get_readiness ()
cmd	YNetwork target get_readiness

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.get_router()`

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

js	function get_router ()
nodejs	function get_router ()
php	function get_router ()
cpp	string get_router ()
m	-(NSString*) router
pas	function get_router (): string
vb	function get_router () As String
cs	string get_router ()
java	String get_router ()
py	def get_router ()
cmd	YNetwork target get_router

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()****network**→**secondaryDNS()****network.get_secondaryDNS()**

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

js	function get_secondaryDNS ()
nodejs	function get_secondaryDNS ()
php	function get_secondaryDNS ()
cpp	string get_secondaryDNS ()
m	-(NSString*) secondaryDNS
pas	function get_secondaryDNS (): string
vb	function get_secondaryDNS () As String
cs	string get_secondaryDNS ()
java	String get_secondaryDNS ()
py	def get_secondaryDNS ()
cmd	YNetwork target get_secondaryDNS

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.get_subnetMask()**

Retourne le masque de sous-réseau utilisé par le module.

js	function get_subnetMask ()
nodejs	function get_subnetMask ()
php	function get_subnetMask ()
cpp	string get_subnetMask ()
m	-(NSString*) subnetMask
pas	function get_subnetMask (): string
vb	function get_subnetMask () As String
cs	string get_subnetMask ()
java	String get_subnetMask ()
py	def get_subnetMask ()
cmd	YNetwork target get_subnetMask

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.get_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 get_userPassword ()
nodejs	function get_userPassword ()
php	function get_userPassword ()
cpp	string get_userPassword ()
m	-(NSString*) userPassword
pas	function get_userPassword (): string
vb	function get_userPassword () As String
cs	string get_userPassword ()
java	String get_userPassword ()
py	def get_userPassword ()
cmd	YNetwork target get_userPassword

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()****YNetwork****network**→**wwwWatchdogDelay()****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.

js	function get_wwwWatchdogDelay ()
nodejs	function get_wwwWatchdogDelay ()
php	function get_wwwWatchdogDelay ()
c++	int get_wwwWatchdogDelay ()
m	-(int) wwwWatchdogDelay
pas	function get_wwwWatchdogDelay (): LongInt
vb	function get_wwwWatchdogDelay () As Integer
cs	int get_wwwWatchdogDelay ()
java	int get_wwwWatchdogDelay ()
py	def get_wwwWatchdogDelay ()
cmd	YNetwork target get_wwwWatchdogDelay

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

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 (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.

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-quel 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 nextNetwork ()
nodejs	function nextNetwork ()
php	function nextNetwork ()
cpp	YNetwork * nextNetwork ()
m	-(YNetwork*) nextNetwork
pas	function nextNetwork (): TYNetwork
vb	function nextNetwork () As YNetwork
cs	YNetwork nextNetwork ()
java	YNetwork nextNetwork ()
py	def nextNetwork ()

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 ping (host)
nodejs	function ping (host)
php	function ping (\$host)
c++	string ping (string host)
m	-(NSString*) ping : (NSString*) host
pas	function ping (host : string): string
vb	function ping () As String
cs	string ping (string host)
java	String ping (String host)
py	def ping (host)
cmd	YNetwork target ping host

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()**YNetwork****network.registerValueCallback()**

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)
c++	int registerValueCallback (YNetworkValueCallback callback)
m	-(int) registerValueCallback : (YNetworkValueCallback) callback
pas	function registerValueCallback (callback : TYNetworkValueCallback): 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.

network→set_adminPassword()**YNetwork****network→setAdminPassword()****network.set_adminPassword()**

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

js	function set_adminPassword (newval)
nodejs	function set_adminPassword (newval)
php	function set_adminPassword (\$newval)
cpp	int set_adminPassword (const string& newval)
m	-(int) setAdminPassword : (NSString*) newval
pas	function set_adminPassword (newval : string): integer
vb	function set_adminPassword (ByVal newval As String) As Integer
cs	int set_adminPassword (string newval)
java	int set_adminPassword (String newval)
py	def set_adminPassword (newval)
cmd	YNetwork target set_adminPassword newval

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()**YNetwork****network→setCallbackCredentials()****network.set_callbackCredentials()**

Modifie le laisser-passer pour se connecter à l'adresse de callback.

js	function set_callbackCredentials (newval)
nodejs	function set_callbackCredentials (newval)
php	function set_callbackCredentials (\$newval)
cpp	int set_callbackCredentials (const string& newval)
m	-(int) setCallbackCredentials : (NSString*) newval
pas	function set_callbackCredentials (newval : string): integer
vb	function set_callbackCredentials (ByVal newval As String) As Integer
cs	int set_callbackCredentials (string newval)
java	int set_callbackCredentials (String newval)
py	def set_callbackCredentials (newval)
cmd	YNetwork target set_callbackCredentials newval

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.set_callbackEncoding()**

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function set_callbackEncoding (newval)
nodejs	function set_callbackEncoding (newval)
php	function set_callbackEncoding (\$newval)
cpp	int set_callbackEncoding (Y_CALLBACKENCODING_enum newval)
m	-(int) setCallbackEncoding : (Y_CALLBACKENCODING_enum) newval
pas	function set_callbackEncoding (newval : Integer): integer
vb	function set_callbackEncoding (ByVal newval As Integer) As Integer
cs	int set_callbackEncoding (int newval)
java	int set_callbackEncoding (int newval)
py	def set_callbackEncoding (newval)
cmd	YNetwork target set_callbackEncoding newval

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()**YNetwork****network→setCallbackMaxDelay()****network.set_callbackMaxDelay()**

Modifie l'attente maximale entre deux notifications par callback, en secondes.

js	function set_callbackMaxDelay (newval)
nodejs	function set_callbackMaxDelay (newval)
php	function set_callbackMaxDelay (\$newval)
cpp	int set_callbackMaxDelay (int newval)
m	-(int) setCallbackMaxDelay : (int) newval
pas	function set_callbackMaxDelay (newval : LongInt): integer
vb	function set_callbackMaxDelay (ByVal newval As Integer) As Integer
cs	int set_callbackMaxDelay (int newval)
java	int set_callbackMaxDelay (int newval)
py	def set_callbackMaxDelay (newval)
cmd	YNetwork target set_callbackMaxDelay newval

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.set_callbackMethod()**

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

js	function set_callbackMethod (newval)
nodejs	function set_callbackMethod (newval)
php	function set_callbackMethod (\$newval)
cpp	int set_callbackMethod (Y_CALLBACKMETHOD_enum newval)
m	-(int) setCallbackMethod : (Y_CALLBACKMETHOD_enum) newval
pas	function set_callbackMethod (newval : Integer): integer
vb	function set_callbackMethod (ByVal newval As Integer) As Integer
cs	int set_callbackMethod (int newval)
java	int set_callbackMethod (int newval)
py	def set_callbackMethod (newval)
cmd	YNetwork target set_callbackMethod newval

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()**YNetwork****network→setCallbackMinDelay()****network.set_callbackMinDelay()**

Modifie l'attente minimale entre deux notifications par callback, en secondes.

js	function set_callbackMinDelay (newval)
nodejs	function set_callbackMinDelay (newval)
php	function set_callbackMinDelay (\$newval)
cpp	int set_callbackMinDelay (int newval)
m	-(int) setCallbackMinDelay : (int) newval
pas	function set_callbackMinDelay (newval : LongInt): integer
vb	function set_callbackMinDelay (ByVal newval As Integer) As Integer
cs	int set_callbackMinDelay (int newval)
java	int set_callbackMinDelay (int newval)
py	def set_callbackMinDelay (newval)
cmd	YNetwork target set_callbackMinDelay newval

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()****network**→**setCallbackUrl()****network.set_callbackUrl()**

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

js	function set_callbackUrl (newval)
nodejs	function set_callbackUrl (newval)
php	function set_callbackUrl (\$newval)
cpp	int set_callbackUrl (const string& newval)
m	-(int) setCallbackUrl : (NSString*) newval
pas	function set_callbackUrl (newval : string): integer
vb	function set_callbackUrl (ByVal newval As String) As Integer
cs	int set_callbackUrl (string newval)
java	int set_callbackUrl (String newval)
py	def set_callbackUrl (newval)
cmd	YNetwork target set_callbackUrl newval

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.set_discoverable()**

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour).

js	function set_discoverable (newval)
nodejs	function set_discoverable (newval)
php	function set_discoverable (\$newval)
cpp	int set_discoverable (Y_DISCOVERABLE_enum newval)
m	-(int) setDiscoverable : (Y_DISCOVERABLE_enum) newval
pas	function set_discoverable (newval : Integer): integer
vb	function set_discoverable (ByVal newval As Integer) As Integer
cs	int set_discoverable (int newval)
java	int set_discoverable (int newval)
py	def set_discoverable (newval)
cmd	YNetwork target set_discoverable newval

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.set_logicalName()**

Modifie le nom logique de l'interface réseau.

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	YNetwork 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 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.set_primaryDNS()**

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function set_primaryDNS (newval)
nodejs	function set_primaryDNS (newval)
php	function set_primaryDNS (\$newval)
cpp	int set_primaryDNS (const string& newval)
m	-(int) setPrimaryDNS : (NSString*) newval
pas	function set_primaryDNS (newval : string): integer
vb	function set_primaryDNS (ByVal newval As String) As Integer
cs	int set_primaryDNS (string newval)
java	int set_primaryDNS (String newval)
py	def set_primaryDNS (newval)
cmd	YNetwork target set_primaryDNS newval

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.set_secondaryDNS ()**

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

js	function set_secondaryDNS (newval)
nodejs	function set_secondaryDNS (newval)
php	function set_secondaryDNS (\$newval)
c++	int set_secondaryDNS (const string& newval)
m	-(int) setSecondaryDNS : (NSString*) newval
pas	function set_secondaryDNS (newval : string): integer
vb	function set_secondaryDNS (ByVal newval As String) As Integer
cs	int set_secondaryDNS (string newval)
java	int set_secondaryDNS (String newval)
py	def set_secondaryDNS (newval)
cmd	YNetwork target set_secondaryDNS newval

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.set_userdata()**

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

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.set_userPassword()**

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

js	function set_userPassword (newval)
nodejs	function set_userPassword (newval)
php	function set_userPassword (\$newval)
cpp	int set_userPassword (const string& newval)
m	-(int) setUserPassword : (NSString*) newval
pas	function set_userPassword (newval : string): integer
vb	function set_userPassword (ByVal newval As String) As Integer
cs	int set_userPassword (string newval)
java	int set_userPassword (String newval)
py	def set_userPassword (newval)
cmd	YNetwork target set_userPassword newval

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.set_wwwWatchdogDelay()**

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 set_wwwWatchdogDelay (newval)
nodejs	function set_wwwWatchdogDelay (newval)
php	function set_wwwWatchdogDelay (\$newval)
c++	int set_wwwWatchdogDelay (int newval)
m	-(int) setWwwWatchdogDelay : (int) newval
pas	function set_wwwWatchdogDelay (newval : LongInt): integer
vb	function set_wwwWatchdogDelay (ByVal newval As Integer) As Integer
cs	int set_wwwWatchdogDelay (int newval)
java	int set_wwwWatchdogDelay (int newval)
py	def set_wwwWatchdogDelay (newval)
cmd	YNetwork target set_wwwWatchdogDelay newval

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><script type='text/javascript' src='yocto_oscontrol.js'></script></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()**YOsControl****yFindOsControl()**`YOsControl.FindOsControl()`

Permet de retrouver un contrôle d'OS d'après un identifiant donné.

js	function yFindOsControl (func)
nodejs	function FindOsControl (func)
php	function yFindOsControl (\$func)
cpp	YOsControl* yFindOsControl (const string& func)
m	YOsControl* yFindOsControl (NSString* func)
pas	function yFindOsControl (func : string): TYOsControl
vb	function yFindOsControl (ByVal func As String) As YOsControl
cs	YOsControl FindOsControl (string func)
java	YOsControl FindOsControl (String func)
py	def FindOsControl (func)

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()**YOsControl****yFirstOsControl()**`YOsControl.FirstOsControl()`

Commence l'énumération des contrôle d'OS accessibles par la librairie.

<code>js</code>	<code>function yFirstOsControl()</code>
<code>nodejs</code>	<code>function FirstOsControl()</code>
<code>php</code>	<code>function yFirstOsControl()</code>
<code>cpp</code>	<code>YOsControl* yFirstOsControl()</code>
<code>m</code>	<code>YOsControl* yFirstOsControl()</code>
<code>pas</code>	<code>function yFirstOsControl(): TYOsControl</code>
<code>vb</code>	<code>function yFirstOsControl() As YOsControl</code>
<code>cs</code>	<code>YOsControl FirstOsControl()</code>
<code>java</code>	<code>YOsControl FirstOsControl()</code>
<code>py</code>	<code>def FirstOsControl()</code>

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

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()**YOsControl****oscontrol→advertisedValue()****oscontrol.get_advertisedValue()**

Retourne la valeur courante du contrôle d'OS (pas plus de 6 caractères).

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

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.get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du contrôle d'OS.

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

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()****oscontrol.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du contrôle d'OS au format `NOM_MODULE.NOM_FONCTION`.

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

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()****YOsControl****oscontrol**→**functionDescriptor()****oscontrol.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du contrôle d'OS, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du contrôle d'OS au format SERIAL.FUNCTIONID.

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

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.get_logicalName()**

Retourne le nom logique du contrôle d'OS.

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

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.get_module()`

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

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

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 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.

oscontrol→get_shutdownCountdown()**YOsControl****oscontrol→shutdownCountdown()****oscontrol.get_shutdownCountdown()**

Retourne le nombre de secondes restantes avant un arrêt de l'OS, ou zéro si aucun arrêt n'a été agendé.

js	function get_shutdownCountdown ()
nodejs	function get_shutdownCountdown ()
php	function get_shutdownCountdown ()
cpp	int get_shutdownCountdown ()
m	-(int) shutdownCountdown
pas	function get_shutdownCountdown (): LongInt
vb	function get_shutdownCountdown () As Integer
cs	int get_shutdownCountdown ()
java	int get_shutdownCountdown ()
py	def get_shutdownCountdown ()
cmd	YOsControl target get_shutdownCountdown

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 get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

oscontrol→**isOnline()**`oscontrol.isOnline()`**YOsControl**

Vérifie si le module hébergeant le contrôle d'OS est joignable, sans déclencher d'erreur.

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

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 load (msValidity)
nodejs	function load (msValidity)
php	function load (\$msValidity)
c++	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.

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()****YOsControl****oscontrol.nextOsControl()**

Continue l'énumération des contrôle d'OS commencée à l'aide de `yFirstOsControl()`.

js	function nextOsControl ()
nodejs	function nextOsControl ()
php	function nextOsControl ()
c++	YOsControl * nextOsControl ()
m	-(YOsControl*) nextOsControl
pas	function nextOsControl (): TYOsControl
vb	function nextOsControl () As YOsControl
cs	YOsControl nextOsControl ()
java	YOsControl nextOsControl ()
py	def nextOsControl ()

Retourne :

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

oscontrol→registerValueCallback()**YOsControl****oscontrol.registerValueCallback()**

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 (YOsControlValueCallback callback)
m	-(int) registerValueCallback : (YOsControlValueCallback) callback
pas	function registerValueCallback (callback : TYOsControlValueCallback): 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.

oscontrol→**set_logicalName()****YOsControl****oscontrol**→**setLogicalName()****oscontrol.set_logicalName()**

Modifie le nom logique du contrôle d'OS.

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	YOsControl 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 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.set_userdata()**

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

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

oscontrol→**shutdown()**`oscontrol.shutdown()`**YOsControl**

Agende un arrêt de l'OS dans un nombre donné de secondes.

js	function shutdown (secBeforeShutDown)
nodejs	function shutdown (secBeforeShutDown)
php	function shutdown (\$secBeforeShutDown)
c++	int shutdown (int secBeforeShutDown)
m	-(int) shutdown : (int) secBeforeShutDown
pas	function shutdown (secBeforeShutDown : LongInt): LongInt
vb	function shutdown () As Integer
cs	int shutdown (int secBeforeShutDown)
java	int shutdown (int secBeforeShutDown)
py	def shutdown (secBeforeShutDown)
cmd	YOsControl target shutdown secBeforeShutDown

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)`

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()**YPower****yFindPower()**`YPower.FindPower()`

Permet de retrouver un capteur de puissance électrique d'après un identifiant donné.

<code>js</code>	<code>function yFindPower(func)</code>
<code>nodejs</code>	<code>function FindPower(func)</code>
<code>php</code>	<code>function yFindPower(\$func)</code>
<code>cpp</code>	<code>YPower* yFindPower(const string& func)</code>
<code>m</code>	<code>YPower* yFindPower(NSString* func)</code>
<code>pas</code>	<code>function yFindPower(func: string): TYPower</code>
<code>vb</code>	<code>function yFindPower(ByVal func As String) As YPower</code>
<code>cs</code>	<code>YPower FindPower(string func)</code>
<code>java</code>	<code>YPower FindPower(String func)</code>
<code>py</code>	<code>def FindPower(func)</code>

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 électrique soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YPower.isOnline()` pour tester si le capteur de puissance électrique 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 électrique sans ambiguïté

Retourne :

un objet de classe `YPower` qui permet ensuite de contrôler le capteur de puissance électrique.

YPower.FirstPower()**YPower****yFirstPower()**`YPower.FirstPower()`

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

js	function yFirstPower ()
nodejs	function FirstPower ()
php	function yFirstPower ()
cpp	YPower* yFirstPower ()
m	YPower* yFirstPower ()
pas	function yFirstPower (): TYPower
vb	function yFirstPower () As YPower
cs	YPower FirstPower ()
java	YPower FirstPower ()
py	def FirstPower ()

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()**YPower****power.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 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.

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

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.get_advertisedValue()**

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

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

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.get_cosPhi()`

Retourne le facteur de puissance (rapport entre la puissance réelle consommée, en W, et la puissance apparente fournie, en VA).

js	function get_cosPhi ()
nodejs	function get_cosPhi ()
php	function get_cosPhi ()
cpp	double get_cosPhi ()
m	-(double) cosPhi
pas	function get_cosPhi (): double
vb	function get_cosPhi () As Double
cs	double get_cosPhi ()
java	double get_cosPhi ()
py	def get_cosPhi ()
cmd	YPower target get_cosPhi

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.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YPower target get_currentRawValue

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.get_currentValue()**

Retourne la valeur instantanée de la puissance électrique.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YPower target get_currentValue

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.get_errorMessage()**

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

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

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()****power.get_errorType()**

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

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

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.get_friendlyName()`

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

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

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.get_functionDescriptor()**

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

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

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.get_functionId()`

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

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

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.get_hardwareId()**

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

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

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.get_highestValue()`

Retourne la valeur maximale observée pour la puissance électrique.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YPower target get_highestValue

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.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.

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

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.get_logicalName()`

Retourne le nom logique du capteur de puissance electrique.

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

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.get_lowestValue()`

Retourne la valeur minimale observée pour la puissance électrique.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YPower target get_lowestValue

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.get_meter()`

Retourne la valeur actuelle du compteur d'energie, calculée par le wattmètre en intégrant la consommation instantanée.

js	function get_meter ()
nodejs	function get_meter ()
php	function get_meter ()
cpp	double get_meter ()
m	-(double) meter
pas	function get_meter (): double
vb	function get_meter () As Double
cs	double get_meter ()
java	double get_meter ()
py	def get_meter ()
cmd	YPower target get_meter

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.get_meterTimer()**

Retourne le temps écoulé depuis la dernière initialisation du compteur d'énergie, en secondes

js	function get_meterTimer ()
nodejs	function get_meterTimer ()
php	function get_meterTimer ()
cpp	int get_meterTimer ()
m	-(int) meterTimer
pas	function get_meterTimer (): LongInt
vb	function get_meterTimer () As Integer
cs	int get_meterTimer ()
java	int get_meterTimer ()
py	def get_meterTimer ()
cmd	YPower target get_meterTimer

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.get_module()`

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

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

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YPower target get_recordedData startTime endTime

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.

power→**get_reportFrequency()****YPower****power**→**reportFrequency()****power.get_reportFrequency()**

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

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

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.get_resolution()`

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YPower target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle la puissance électrique est exprimée.

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

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.get_userData()`

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

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

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

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 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.

power→loadCalibrationPoints()**YPower****power.loadCalibrationPoints()**

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()`.

js	function nextPower ()
nodejs	function nextPower ()
php	function nextPower ()
cpp	YPower * nextPower ()
m	-(YPower*) nextPower
pas	function nextPower (): TYPower
vb	function nextPower () As YPower
cs	YPower nextPower ()
java	YPower nextPower ()
py	def nextPower ()

Retourne :

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

power→registerTimedReportCallback()**YPower****power.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
c++	int registerTimedReportCallback (YPowerTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YPowerTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYPowerTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YPower****power.registerValueCallback()**

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 (YPowerValueCallback callback)
m	-(int) registerValueCallback : (YPowerValueCallback) callback
pas	function registerValueCallback (callback : TYPowerValueCallback): 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.

power→**reset()****power.reset ()****YPower**

Réinitialise le compteur d'énergie.

js	function reset ()
nodejs	function reset ()
php	function reset ()
cpp	int reset ()
m	-(int) reset
pas	function reset (): LongInt
vb	function reset () As Integer
cs	int reset ()
java	int reset ()
py	def reset ()
cmd	YPower target 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.

power→**set_highestValue()****YPower****power**→**setHighestValue()****power.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour la puissance électrique.

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	YPower target set_highestValue newval

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()****power**→**setLogFrequency()****power.set_logFrequency()**

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	YPower 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.

power→**set_logicalName()****YPower****power**→**setLogicalName()****power.set_logicalName()**

Modifie le nom logique du capteur de puissance electrique.

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	YPower 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour la puissance électrique.

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

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()****YPower****power**→**setReportFrequency()****power.set_reportFrequency()**

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	YPower 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.

power→**set_resolution()****YPower****power**→**setResolution()**`power.set_resolution()`

Modifie la résolution des valeurs mesurées.

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

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.set_userdata()**

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

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

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><script type='text/javascript' src='yocto_pressure.js'></script></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()

	Retourne l'identifiant matériel unique du capteur de pression au format <code>SERIAL.FUNCTIONID</code> .
pressure→get_highestValue()	Retourne la valeur maximale observée pour la pression.
pressure→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.
pressure→get_logicalName()	Retourne le nom logique du capteur de pression.
pressure→get_lowestValue()	Retourne la valeur minimale observée pour la pression.
pressure→get_module()	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
pressure→get_module_async(callback, context)	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
pressure→get_recordedData(startTime, endTime)	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.
pressure→get_reportFrequency()	Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.
pressure→get_resolution()	Retourne la résolution des valeurs mesurées.
pressure→get_unit()	Retourne l'unité dans laquelle la pression est exprimée.
pressure→get_userData()	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
pressure→isOnline()	Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.
pressure→isOnline_async(callback, context)	Vérifie si le module hébergeant le capteur de pression est joignable, sans déclencher d'erreur.
pressure→load(msValidity)	Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.
pressure→loadCalibrationPoints(rawValues, refValues)	Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode <code>calibrateFromPoints</code> .
pressure→load_async(msValidity, callback, context)	Met en cache les valeurs courantes du capteur de pression, avec une durée de validité spécifiée.
pressure→nextPressure()	Continue l'énumération des capteurs de pression commencée à l'aide de <code>yFirstPressure()</code> .
pressure→registerTimedReportCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque notification périodique.
pressure→registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
pressure→set_highestValue(newval)	Modifie la mémoire de valeur maximale observée pour la pression.
pressure→set_logFrequency(newval)	

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()**YPressure****yFindPressure()****YPressure.FindPressure()**

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

js	function yFindPressure (func)
nodejs	function FindPressure (func)
php	function yFindPressure (\$func)
cpp	YPressure* yFindPressure (const string& func)
m	YPressure* yFindPressure (NSString* func)
pas	function yFindPressure (func : string): TYPpressure
vb	function yFindPressure (ByVal func As String) As YPressure
cs	YPressure FindPressure (string func)
java	YPressure FindPressure (String func)
py	def FindPressure (func)

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()**YPressure****yFirstPressure()****YPressure.FirstPressure()**

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

js	function yFirstPressure ()
nodejs	function FirstPressure ()
php	function yFirstPressure ()
cpp	YPressure* yFirstPressure ()
m	YPressure* yFirstPressure ()
pas	function yFirstPressure (): TYPressure
vb	function yFirstPressure () As YPressure
cs	YPressure FirstPressure ()
java	YPressure FirstPressure ()
py	def FirstPressure ()

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()**YPressure****pressure.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 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	YPressure 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.

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

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()****YPressure****pressure**→**advertisedValue()****pressure.get_advertisedValue()**

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

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

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()****YPressure****pressure**→**currentRawValue()****pressure.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YPressure target get_currentRawValue

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.get_currentValue()**

Retourne la mesure actuelle de la pression.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YPressure target get_currentValue

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.get_errorMessage()**

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

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

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()****pressure.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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()****YPressure****pressure**→**functionDescriptor()****pressure.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()**

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

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

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.get_highestValue()**

Retourne la valeur maximale observée pour la pression.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YPressure target get_highestValue

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.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.

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

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.get_logicalName()**

Retourne le nom logique du capteur de pression.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour la pression.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YPressure target get_lowestValue

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.get_module()`

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

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

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YPressure target get_recordedData startTime endTime

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.

pressure→**get_reportFrequency()****YPressure****pressure**→**reportFrequency()****pressure.get_reportFrequency()**

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

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

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YPressure target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle la pression est exprimée.

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

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.get_userData()**

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

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

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

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 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.

pressure→loadCalibrationPoints()**YPressure****pressure.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(&\$rawValues, &\$refValues)</code>
<code>cpp</code>	<code>int loadCalibrationPoints(vector<double>& rawValues, vector<double>& 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<double> rawValues, List<double> refValues)</code>
<code>java</code>	<code>int loadCalibrationPoints(ArrayList<Double> rawValues, ArrayList<Double> refValues)</code>
<code>py</code>	<code>def loadCalibrationPoints(rawValues, refValues)</code>
<code>cmd</code>	<code>YPressure 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.

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()****YPressure****pressure.nextPressure()**

Continue l'énumération des capteurs de pression commencée à l'aide de `yFirstPressure()`.

js	function nextPressure ()
nodejs	function nextPressure ()
php	function nextPressure ()
cpp	YPressure * nextPressure ()
m	-(YPressure*) nextPressure
pas	function nextPressure (): TYPpressure
vb	function nextPressure () As YPressure
cs	YPressure nextPressure ()
java	YPressure nextPressure ()
py	def nextPressure ()

Retourne :

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

pressure→registerTimedReportCallback()**YPressure****pressure.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YPressureTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YPressureTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYPressureTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YPressure****pressure.registerValueCallback()**

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 (YPressureValueCallback callback)
m	-(int) registerValueCallback : (YPressureValueCallback) callback
pas	function registerValueCallback (callback : TYPresureValueCallback): 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.

pressure→**set_highestValue()****YPressure****pressure**→**setHighestValue()****pressure.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour la pression.

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	YPressure target set_highestValue newval

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.set_logFrequency()

YPressure

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	YPressure 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.

pressure→**set_logicalName()****YPressure****pressure**→**setLogicalName()****pressure.set_logicalName()**

Modifie le nom logique du capteur de pression.

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	YPressure 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 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()****YPressure****pressure**→**setLowestValue()****pressure.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour la pression.

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

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.set_reportFrequency()

YPressure

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	YPressure 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.

pressure→**set_resolution()**
pressure→**setResolution()**
pressure.set_resolution()

YPressure

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()**

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

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

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()**YPwmOutput****yFindPwmOutput()** `YPwmOutput.FindPwmOutput()`

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

<code>js</code>	<code>function yFindPwmOutput(func)</code>
<code>nodejs</code>	<code>function FindPwmOutput(func)</code>
<code>php</code>	<code>function yFindPwmOutput(\$func)</code>
<code>cpp</code>	<code>YPwmOutput* yFindPwmOutput(const string& func)</code>
<code>m</code>	<code>YPwmOutput* yFindPwmOutput(NSString* func)</code>
<code>pas</code>	<code>function yFindPwmOutput(func: string): TYPwmOutput</code>
<code>vb</code>	<code>function yFindPwmOutput(ByVal func As String) As YPwmOutput</code>
<code>cs</code>	<code>YPwmOutput FindPwmOutput(string func)</code>
<code>java</code>	<code>YPwmOutput FindPwmOutput(String func)</code>
<code>py</code>	<code>def FindPwmOutput(func)</code>

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()**YPwmOutput****yFirstPwmOutput()****YPwmOutput.FirstPwmOutput()**

Commence l'énumération des PWM accessibles par la librairie.

js	function yFirstPwmOutput ()
nodejs	function FirstPwmOutput ()
php	function yFirstPwmOutput ()
cpp	YPwmOutput* yFirstPwmOutput ()
m	YPwmOutput* yFirstPwmOutput ()
pas	function yFirstPwmOutput (): TYPwmOutput
vb	function yFirstPwmOutput () As YPwmOutput
cs	YPwmOutput FirstPwmOutput ()
java	YPwmOutput FirstPwmOutput ()
py	def FirstPwmOutput ()

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

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()**YPwmOutput****pwmoutput.dutyCycleMove()**

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

js	function dutyCycleMove (target , ms_duration)
nodejs	function dutyCycleMove (target , ms_duration)
php	function dutyCycleMove (\$target , \$ms_duration)
cpp	int dutyCycleMove (double target , int ms_duration)
m	-(int) dutyCycleMove : (double) target : (int) ms_duration
pas	function dutyCycleMove (target : double, ms_duration : LongInt): LongInt
vb	function dutyCycleMove () As Integer
cs	int dutyCycleMove (double target , int ms_duration)
java	int dutyCycleMove (double target , int ms_duration)
py	def dutyCycleMove (target , ms_duration)
cmd	YPwmOutput target dutyCycleMove target ms_duration

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.get_advertisedValue()**

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

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

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.get_dutyCycle()**

Retourne le duty cycle du \$FUNCTION\$ sous la forme d'un nombre a virgule entre 0 et 1.

js	function get_dutyCycle ()
nodejs	function get_dutyCycle ()
php	function get_dutyCycle ()
cpp	double get_dutyCycle ()
m	-(double) dutyCycle
pas	function get_dutyCycle (): double
vb	function get_dutyCycle () As Double
cs	double get_dutyCycle ()
java	double get_dutyCycle ()
py	def get_dutyCycle ()
cmd	YPwmOutput target get_dutyCycle

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.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.

js	function get_dutyCycleAtPowerOn ()
nodejs	function get_dutyCycleAtPowerOn ()
php	function get_dutyCycleAtPowerOn ()
cpp	double get_dutyCycleAtPowerOn ()
m	-(double) dutyCycleAtPowerOn
pas	function get_dutyCycleAtPowerOn (): double
vb	function get_dutyCycleAtPowerOn () As Double
cs	double get_dutyCycleAtPowerOn ()
java	double get_dutyCycleAtPowerOn ()
py	def get_dutyCycleAtPowerOn ()
cmd	YPwmOutput target get_dutyCycleAtPowerOn

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.get_enabled()`

Retourne l'état de fonctionnement du \$FUNCTION\$.

js	function get_enabled ()
nodejs	function get_enabled ()
php	function get_enabled ()
cpp	Y_ENABLED_enum get_enabled ()
m	-(Y_ENABLED_enum) enabled
pas	function get_enabled (): Integer
vb	function get_enabled () As Integer
cs	int get_enabled ()
java	int get_enabled ()
py	def get_enabled ()
cmd	YPwmOutput target get_enabled

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.get_enabledAtPowerOn()

YPwmOutput

Retourne l'état de fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

js	function get_enabledAtPowerOn ()
nodejs	function get_enabledAtPowerOn ()
php	function get_enabledAtPowerOn ()
cpp	Y_ENABLEDATPOWERON_enum get_enabledAtPowerOn ()
m	-(Y_ENABLEDATPOWERON_enum) enabledAtPowerOn
pas	function get_enabledAtPowerOn (): Integer
vb	function get_enabledAtPowerOn () As Integer
cs	int get_enabledAtPowerOn ()
java	int get_enabledAtPowerOn ()
py	def get_enabledAtPowerOn ()
cmd	YPwmOutput target get_enabledAtPowerOn

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()**YPwmOutput****pwmoutput→errorMessage()****pwmoutput.get_errorMessage()**

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

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

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()**YPwmOutput****pwmoutput→errorType()****pwmoutput.get_errorType()**

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

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

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.get_frequency()**

Retourne la fréquence du \$FUNCTION\$ en Hz.

js	function get_frequency ()
nodejs	function get_frequency ()
php	function get_frequency ()
cpp	int get_frequency ()
m	-(int) frequency
pas	function get_frequency (): LongInt
vb	function get_frequency () As Integer
cs	int get_frequency ()
java	int get_frequency ()
py	def get_frequency ()
cmd	YPwmOutput target get_frequency

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()**YPwmOutput****pwmoutput→friendlyName()****pwmoutput.get_friendlyName()**

Retourne un identifiant global du PWM au format `NOM_MODULE.NOM_FONCTION`.

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

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()**YPwmOutput****pwmoutput→functionDescriptor()****pwmoutput.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du PWM, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du PWM au format SERIAL.FUNCTIONID.

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

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()
pwmoutput→logicalName()
pwmoutput.get_logicalName()

YPwmOutput

Retourne le nom logique du PWM.

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

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.get_module()`

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

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

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 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.

pwmoutput→get_period()**YPwmOutput****pwmoutput→period()**`pwmoutput.get_period()`

Retourne la période du \$FUNCTION\$ en nano secondes.

js	function get_period ()
nodejs	function get_period ()
php	function get_period ()
cpp	double get_period ()
m	-(double) period
pas	function get_period (): double
vb	function get_period () As Double
cs	double get_period ()
java	double get_period ()
py	def get_period ()
cmd	YPwmOutput target get_period

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()**YPwmOutput****pwmoutput→pulseDuration()****pwmoutput.get_pulseDuration()**

Retourne la longueur d'une impulsion du \$FUNCTION\$ en millisecondes.

js	function get_pulseDuration ()
nodejs	function get_pulseDuration ()
php	function get_pulseDuration ()
cpp	double get_pulseDuration ()
m	-(double) pulseDuration
pas	function get_pulseDuration (): double
vb	function get_pulseDuration () As Double
cs	double get_pulseDuration ()
java	double get_pulseDuration ()
py	def get_pulseDuration ()
cmd	YPwmOutput target get_pulseDuration

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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

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 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.

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()**YPwmOutput****pwmoutput.nextPwmOutput()**

Continue l'énumération des PWM commencée à l'aide de `yFirstPwmOutput()`.

js	function nextPwmOutput ()
nodejs	function nextPwmOutput ()
php	function nextPwmOutput ()
cpp	YPwmOutput * nextPwmOutput ()
m	-(YPwmOutput*) nextPwmOutput
pas	function nextPwmOutput (): TYPwmOutput
vb	function nextPwmOutput () As YPwmOutput
cs	YPwmOutput nextPwmOutput ()
java	YPwmOutput nextPwmOutput ()
py	def nextPwmOutput ()

Retourne :

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

pwmoutput→pulseDurationMove()**YPwmOutput****pwmoutput.pulseDurationMove()**

Déclenche une variation progressive de la longueur des impulsions vers une valeur donnée.

js	function pulseDurationMove (ms_target , ms_duration)
nodejs	function pulseDurationMove (ms_target , ms_duration)
php	function pulseDurationMove (\$ms_target , \$ms_duration)
cpp	int pulseDurationMove (double ms_target , int ms_duration)
m	-(int) pulseDurationMove : (double) ms_target : (int) ms_duration
pas	function pulseDurationMove (ms_target : double, ms_duration : LongInt): LongInt
vb	function pulseDurationMove () As Integer
cs	int pulseDurationMove (double ms_target , int ms_duration)
java	int pulseDurationMove (double ms_target , int ms_duration)
py	def pulseDurationMove (ms_target , ms_duration)
cmd	YPwmOutput target pulseDurationMove ms_target ms_duration

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()**YPwmOutput****pwmoutput.registerValueCallback()**

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 (YPwmOutputValueCallback callback)
m	-(int) registerValueCallback : (YPwmOutputValueCallback) callback
pas	function registerValueCallback (callback : TYPwmOutputValueCallback): 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.

pwmoutput→set_dutyCycle()**YPwmOutput****pwmoutput→setDutyCycle()****pwmoutput.set_dutyCycle()**

Configure le duty cycle du \$FUNCTION\$.

js	function set_dutyCycle (newval)
nodejs	function set_dutyCycle (newval)
php	function set_dutyCycle (\$newval)
cpp	int set_dutyCycle (double newval)
m	-(int) setDutyCycle : (double) newval
pas	function set_dutyCycle (newval : double): integer
vb	function set_dutyCycle (ByVal newval As Double) As Integer
cs	int set_dutyCycle (double newval)
java	int set_dutyCycle (double newval)
py	def set_dutyCycle (newval)
cmd	YPwmOutput target set_dutyCycle newval

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.set_dutyCycleAtPowerOn()**

Configure le duty cycle du \$FUNCTION\$ au démarrage du module.

js	function set_dutyCycleAtPowerOn (newval)
nodejs	function set_dutyCycleAtPowerOn (newval)
php	function set_dutyCycleAtPowerOn (\$newval)
cpp	int set_dutyCycleAtPowerOn (double newval)
m	-(int) setDutyCycleAtPowerOn : (double) newval
pas	function set_dutyCycleAtPowerOn (newval : double): integer
vb	function set_dutyCycleAtPowerOn (ByVal newval As Double) As Integer
cs	int set_dutyCycleAtPowerOn (double newval)
java	int set_dutyCycleAtPowerOn (double newval)
py	def set_dutyCycleAtPowerOn (newval)
cmd	YPwmOutput target set_dutyCycleAtPowerOn newval

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.set_enabled()**

Démarre ou arrête le \$FUNCTION\$.

js	function set_enabled (newval)
nodejs	function set_enabled (newval)
php	function set_enabled (\$newval)
cpp	int set_enabled (Y_ENABLED_enum newval)
m	-(int) setEnabled : (Y_ENABLED_enum) newval
pas	function set_enabled (newval : Integer): integer
vb	function set_enabled (ByVal newval As Integer) As Integer
cs	int set_enabled (int newval)
java	int set_enabled (int newval)
py	def set_enabled (newval)
cmd	YPwmOutput target set_enabled newval

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()
pwmoutput→setEnabledAtPowerOn()
pwmoutput.set_enabledAtPowerOn()

YPwmOutput

Configure l'état du fonctionnement du \$FUNCTION\$ à la mise sous tension du module.

js	function set_enabledAtPowerOn (newval)
nodejs	function set_enabledAtPowerOn (newval)
php	function set_enabledAtPowerOn (\$newval)
cpp	int set_enabledAtPowerOn (Y_ENABLEDATPOWERON_enum newval)
m	-(int) setEnabledAtPowerOn : (Y_ENABLEDATPOWERON_enum) newval
pas	function set_enabledAtPowerOn (newval : Integer): integer
vb	function set_enabledAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_enabledAtPowerOn (int newval)
java	int set_enabledAtPowerOn (int newval)
py	def set_enabledAtPowerOn (newval)
cmd	YPwmOutput target set_enabledAtPowerOn newval

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.set_frequency()**

Configure la fréquence du \$FUNCTION\$.

js	function set_frequency (newval)
nodejs	function set_frequency (newval)
php	function set_frequency (\$newval)
cpp	int set_frequency (int newval)
m	-(int) setFrequency : (int) newval
pas	function set_frequency (newval : LongInt): integer
vb	function set_frequency (ByVal newval As Integer) As Integer
cs	int set_frequency (int newval)
java	int set_frequency (int newval)
py	def set_frequency (newval)
cmd	YPwmOutput target set_frequency newval

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()
pwmoutput→setLogicalName()
pwmoutput.set_logicalName()

YPwmOutput

Modifie le nom logique du PWM.

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	YPwmOutput 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 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.set_period()`

Configure la période du \$FUNCTION\$.

js	function set_period (newval)
nodejs	function set_period (newval)
php	function set_period (\$newval)
cpp	int set_period (double newval)
m	-(int) setPeriod : (double) newval
pas	function set_period (newval : double): integer
vb	function set_period (ByVal newval As Double) As Integer
cs	int set_period (double newval)
java	int set_period (double newval)
py	def set_period (newval)
cmd	YPwmOutput target set_period newval

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()**YPwmOutput****pwmoutput→setPulseDuration()****pwmoutput.set_pulseDuration()**

Configure la longueur d'une impulsion du \$FUNCTION\$.

js	function set_pulseDuration (newval)
nodejs	function set_pulseDuration (newval)
php	function set_pulseDuration (\$newval)
cpp	int set_pulseDuration (double newval)
m	-(int) setPulseDuration : (double) newval
pas	function set_pulseDuration (newval : double): integer
vb	function set_pulseDuration (ByVal newval As Double) As Integer
cs	int set_pulseDuration (double newval)
java	int set_pulseDuration (double newval)
py	def set_pulseDuration (newval)
cmd	YPwmOutput target set_pulseDuration newval

Attention la longueur d'une impulsion ne peut pas être plus grande que la période, dans le cas contraire, la longueur sera automatiquement tronquée 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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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() YPwmPowerSource.FindPwmPowerSource()

YPwmPowerSource

Permet de retrouver une source de tension d'après un identifiant donné.

js	function yFindPwmPowerSource (func)
nodejs	function FindPwmPowerSource (func)
php	function yFindPwmPowerSource (\$func)
cpp	YPwmPowerSource* yFindPwmPowerSource (const string& func)
m	YPwmPowerSource* yFindPwmPowerSource (NSString* func)
pas	function yFindPwmPowerSource (func : string): TYPwmPowerSource
vb	function yFindPwmPowerSource (ByVal func As String) As YPwmPowerSource
cs	YPwmPowerSource FindPwmPowerSource (string func)
java	YPwmPowerSource FindPwmPowerSource (String func)
py	def FindPwmPowerSource (func)

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() YPwmPowerSource.FirstPwmPowerSource()

YPwmPowerSource

Commence l'énumération des Source de tension accessibles par la librairie.

js	function yFirstPwmPowerSource ()
nodejs	function FirstPwmPowerSource ()
php	function yFirstPwmPowerSource ()
cpp	YPwmPowerSource* yFirstPwmPowerSource ()
m	YPwmPowerSource* yFirstPwmPowerSource ()
pas	function yFirstPwmPowerSource (): TYPwmPowerSource
vb	function yFirstPwmPowerSource () As YPwmPowerSource
cs	YPwmPowerSource FirstPwmPowerSource ()
java	YPwmPowerSource FirstPwmPowerSource ()
py	def FirstPwmPowerSource ()

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.

pwpowersource→describe()**YPwmPowerSource****pwpowersource.describe()**

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

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

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 la source de tension (ex:
`Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1`)

pwmpowersource→get_advertisedValue()

YPwmPowerSource

pwmpowersource→advertisedValue()

pwmpowersource.get_advertisedValue()

Retourne la valeur courante de la source de tension (pas plus de 6 caractères).

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

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()****YPwmPowerSource****pwmpowersource**→**errorMessage()****pwmpowersource.get_errorMessage()**

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

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

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()**YPwmPowerSource****pwmpowersource→errorType()****pwmpowersource.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global de la source de tension au format `NOM_MODULE.NOM_FONCTION`.

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

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()**YPwmPowerSource****pwmpowersource→functionDescriptor()****pwmpowersource.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel de la source de tension, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique de la source de tension au format SERIAL.FUNCTIONID.

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

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.get_logicalName()**

Retourne le nom logique de la source de tension.

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

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()**YPwmPowerSource****pwmpowersource→module()****pwmpowersource.get_module()**

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

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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.get_powerMode()**

Retourne la source de tension utilisé par tous les PWM du même module.

js	function get_powerMode ()
nodejs	function get_powerMode ()
php	function get_powerMode ()
cpp	Y_POWERMODE_enum get_powerMode ()
m	-(Y_POWERMODE_enum) powerMode
pas	function get_powerMode (): Integer
vb	function get_powerMode () As Integer
cs	int get_powerMode ()
java	int get_powerMode ()
py	def get_powerMode ()

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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()**YPwmPowerSource****pwmpowersource.isOnline()**

Vérifie si le module hébergeant la source de tension est joignable, sans déclencher d'erreur.

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

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 (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.

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-qu'el à la fonction de callback

Retourne :

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

pwpmpowersource→nextPwmPowerSource()**YPwmPowerSource****pwpmpowersource.nextPwmPowerSource()**

Continue l'énumération des Source de tension commencée à l'aide de `yFirstPwmPowerSource()`.

js	function nextPwmPowerSource ()
nodejs	function nextPwmPowerSource ()
php	function nextPwmPowerSource ()
cpp	YPwmPowerSource * nextPwmPowerSource ()
m	-(YPwmPowerSource*) nextPwmPowerSource
pas	function nextPwmPowerSource (): TYPwmPowerSource
vb	function nextPwmPowerSource () As YPwmPowerSource
cs	YPwmPowerSource nextPwmPowerSource ()
java	YPwmPowerSource nextPwmPowerSource ()
py	def nextPwmPowerSource ()

Retourne :

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

pwmpowersource→registerValueCallback()**YPwmPowerSource****pwmpowersource.registerValueCallback()**

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 (YPwmPowerSourceValueCallback callback)
m	-(int) registerValueCallback : (YPwmPowerSourceValueCallback) callback
pas	function registerValueCallback (callback : TYPwmPowerSourceValueCallback): 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.

pwmpowersource→set_logicalName()**YPwmPowerSource****pwmpowersource→setLogicalName()****pwmpowersource.set_logicalName()**

Modifie le nom logique de la source de tension.

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	YPwmPowerSource 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 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.set_powerMode()**

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 set_powerMode (newval)
nodejs	function set_powerMode (newval)
php	function set_powerMode (\$newval)
cpp	int set_powerMode (Y_POWERMODE_enum newval)
m	-(int) setPowerMode : (Y_POWERMODE_enum) newval
pas	function set_powerMode (newval : Integer): integer
vb	function set_powerMode (ByVal newval As Integer) As Integer
cs	int set_powerMode (int newval)
java	int set_powerMode (int newval)
py	def set_powerMode (newval)
cmd	YPwmPowerSource target set_powerMode newval

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.set_userdata()**

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

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

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><script type='text/javascript' src='yocto_gyro.js'></script></code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>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()**YQt****yFindQt()**`YQt.FindQt()`

Permet de retrouver un élément de quaternion d'après un identifiant donné.

js	function yFindQt (func)
nodejs	function FindQt (func)
php	function yFindQt (\$func)
cpp	YQt* yFindQt (string func)
m	+(YQt*) yFindQt : (NSString*) func
pas	function yFindQt (func : string): TYQt
vb	function yFindQt (ByVal func As String) As YQt
cs	YQt FindQt (string func)
java	YQt FindQt (String func)
py	def FindQt (func)

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()**YQt****yFirstQt()**`YQt.FirstQt()`

Commence l'énumération des éléments de quaternion accessibles par la librairie.

<code>js</code>	<code>function yFirstQt()</code>
<code>nodejs</code>	<code>function FirstQt()</code>
<code>php</code>	<code>function yFirstQt()</code>
<code>cpp</code>	<code>YQt* yFirstQt()</code>
<code>m</code>	<code>YQt* yFirstQt()</code>
<code>pas</code>	<code>function yFirstQt(): TYQt</code>
<code>vb</code>	<code>function yFirstQt() As YQt</code>
<code>cs</code>	<code>YQt FirstQt()</code>
<code>java</code>	<code>YQt FirstQt()</code>
<code>py</code>	<code>def FirstQt()</code>

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()**

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.

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

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.get_advertisedValue()**

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

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

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.get_currentRawValue()`

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YSensor target get_currentRawValue

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.get_currentValue()**

Retourne la valeur actuelle de la coordonnée.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YSensor target get_currentValue

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.get_errorMessage()`

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

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

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()**`qt.get_errorType()`

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

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

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.get_friendlyName()`

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

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

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()**qt→functionDescriptor()****qt.get_functionDescriptor()**

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

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

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.get_functionId()`

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

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

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.get_hardwareId()`

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

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

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.get_highestValue()`

Retourne la valeur maximale observée pour la coordonnée depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YSensor target get_highestValue

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.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.

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

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.get_logicalName()`

Retourne le nom logique de l'élément de quaternion.

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

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.get_lowestValue()`

Retourne la valeur minimale observée pour la coordonnée depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YSensor target get_lowestValue

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.get_module()`

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

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

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 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-qu'el 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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YSensor target get_recordedData startTime endTime

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.get_reportFrequency()`

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

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

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.get_resolution()`

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YSensor target get_resolution

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.get_unit()`

Retourne l'unité dans laquelle la coordonnée est exprimée.

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

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.get_userdata()`

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): TObject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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

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-qu'el 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 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.

qt→loadCalibrationPoints()**YQt****qt.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(&\$rawValues, &\$refValues)</code>
cpp	<code>int loadCalibrationPoints(vector<double>& rawValues, vector<double>& 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<double> rawValues, List<double> refValues)</code>
java	<code>int loadCalibrationPoints(ArrayList<Double> rawValues, ArrayList<Double> refValues)</code>
py	<code>def loadCalibrationPoints(rawValues, refValues)</code>
cmd	<code>YSensor 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.

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()**YQt****qt.registerTimedReportCallback()**

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

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YQtTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YQtTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYQtTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YQt****qt.registerValueCallback()**

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 (YQtValueCallback callback)
m	-(int) registerValueCallback : (YQtValueCallback) callback
pas	function registerValueCallback (callback : TYQtValueCallback): 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.

qt→set_highestValue()**YQt****qt→setHighestValue()**`qt.set_highestValue()`

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	YSensor 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.

qt→set_logFrequency()**YQt****qt→setLogFrequency()**`qt.set_logFrequency()`

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	YSensor 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.

qt→set_logicalName()**YQt****qt→setLogicalName()**`qt.set_logicalName()`

Modifie le nom logique de l'élément de quaternion.

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	YSensor 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 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()**YQt****qt→setLowestValue()**`qt.set_lowestValue()`

Modifie la mémoire de valeur minimale observée.

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

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.set_reportFrequency()**

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	YSensor 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.

qt→set_resolution()**YQt****qt→setResolution()**`qt.set_resolution()`

Modifie la résolution des valeurs physique mesurées.

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

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.set_userdata()`

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

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

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
yFindRealTimeClock(func) Permet de retrouver une horloge d'après un identifiant donné.
yFirstRealTimeClock() Commence l'énumération des horloge accessibles par la librairie.
Méthodes des objets YRealTimeClock
realtimeclock→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance de l'horloge au format TYPE (NAME) = SERIAL . FUNCTIONID.
realtimeclock→get_advertisedValue() Retourne la valeur courante de l'horloge (pas plus de 6 caractères).
realtimeclock→get_dateTime() Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss"
realtimeclock→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.
realtimeclock→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge.
realtimeclock→get_friendlyName() Retourne un identifiant global de l'horloge au format NOM_MODULE . NOM_FONCTION.
realtimeclock→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
realtimeclock→get_functionId() Retourne l'identifiant matériel de l'horloge, sans référence au module.
realtimeclock→get_hardwareId() Retourne l'identifiant matériel unique de l'horloge au format SERIAL . FUNCTIONID.
realtimeclock→get_logicalName() Retourne le nom logique de l'horloge.
realtimeclock→get_module()

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()**YRealTimeClock****yFindRealTimeClock()****YRealTimeClock.FindRealTimeClock()**

Permet de retrouver une horloge d'après un identifiant donné.

js	function yFindRealTimeClock (func)
nodejs	function FindRealTimeClock (func)
php	function yFindRealTimeClock (\$func)
cpp	YRealTimeClock* yFindRealTimeClock (const string& func)
m	YRealTimeClock* yFindRealTimeClock (NSString* func)
pas	function yFindRealTimeClock (func : string): TYRealTimeClock
vb	function yFindRealTimeClock (ByVal func As String) As YRealTimeClock
cs	YRealTimeClock FindRealTimeClock (string func)
java	YRealTimeClock FindRealTimeClock (String func)
py	def FindRealTimeClock (func)

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()**YRealTimeClock****yFirstRealTimeClock()****YRealTimeClock.FirstRealTimeClock()**

Commence l'énumération des horloge accessibles par la librairie.

js	function yFirstRealTimeClock ()
nodejs	function FirstRealTimeClock ()
php	function yFirstRealTimeClock ()
cpp	YRealTimeClock* yFirstRealTimeClock ()
m	YRealTimeClock* yFirstRealTimeClock ()
pas	function yFirstRealTimeClock (): TYRealTimeClock
vb	function yFirstRealTimeClock () As YRealTimeClock
cs	YRealTimeClock FirstRealTimeClock ()
java	YRealTimeClock FirstRealTimeClock ()
py	def FirstRealTimeClock ()

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()**YRealTimeClock****realtimeclock.describe()**

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

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

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'horloge (ex:
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

realtimeclock→get_advertisedValue()**YRealTimeClock****realtimeclock→advertisedValue()****realtimeclock.get_advertisedValue()**

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

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

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.get_dateTime()**

Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss"

js	function get_dateTime ()
nodejs	function get_dateTime ()
php	function get_dateTime ()
cpp	string get_dateTime ()
m	-(NSString*) dateTime
pas	function get_dateTime (): string
vb	function get_dateTime () As String
cs	string get_dateTime ()
java	String get_dateTime ()
py	def get_dateTime ()

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()**YRealTimeClock****realtimeclock→errorMessage()****realtimeclock.get_errorMessage()**

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

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

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()**YRealTimeClock****realtimeclock→errorType()****realtimeclock.get_errorType()**

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

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

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.get_friendlyName()**

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

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

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

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

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.get_hardwareId()

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

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

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.get_logicalName()

YRealTimeClock

Retourne le nom logique de l'horloge.

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

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.get_module()**

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

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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.

realtimeclock→get_timeSet()**YRealTimeClock****realtimeclock→timeSet()****realtimeclock.get_timeSet()**

Retourne vrai si l'horloge à été mise à l'heure, sinon faux.

js	function get_timeSet ()
nodejs	function get_timeSet ()
php	function get_timeSet ()
cpp	Y_TIMESET_enum get_timeSet ()
m	-(Y_TIMESET_enum) timeSet
pas	function get_timeSet (): Integer
vb	function get_timeSet () As Integer
cs	int get_timeSet ()
java	int get_timeSet ()
py	def get_timeSet ()
cmd	YRealTimeClock target get_timeSet

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.get_unixTime()**

Retourne l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

js	function get_unixTime ()
nodejs	function get_unixTime ()
php	function get_unixTime ()
cpp	s64 get_unixTime ()
m	-(s64) unixTime
pas	function get_unixTime (): int64
vb	function get_unixTime () As Long
cs	long get_unixTime ()
java	long get_unixTime ()
py	def get_unixTime ()
cmd	YRealTimeClock target get_unixTime

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.get_userdata()**

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

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.get_utcOffset()**

Retourne le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

js	function get_utcOffset ()
nodejs	function get_utcOffset ()
php	function get_utcOffset ()
cpp	int get_utcOffset ()
m	-(int) utcOffset
pas	function get_utcOffset (): LongInt
vb	function get_utcOffset () As Integer
cs	int get_utcOffset ()
java	int get_utcOffset ()
py	def get_utcOffset ()
cmd	YRealTimeClock target get_utcOffset

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()**YRealTimeClock****realtimeclock.isOnline()**

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

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

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-quel 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.

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

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()**YRealTimeClock****realtimeclock.nextRealTimeClock()**

Continue l'énumération des horloge commencée à l'aide de `yFirstRealTimeClock()`.

js	function nextRealTimeClock ()
nodejs	function nextRealTimeClock ()
php	function nextRealTimeClock ()
cpp	YRealTimeClock * nextRealTimeClock ()
m	-(YRealTimeClock*) nextRealTimeClock
pas	function nextRealTimeClock (): TYRealTimeClock
vb	function nextRealTimeClock () As YRealTimeClock
cs	YRealTimeClock nextRealTimeClock ()
java	YRealTimeClock nextRealTimeClock ()
py	def nextRealTimeClock ()

Retourne :

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

realtimeclock→registerValueCallback()**YRealTimeClock****realtimeclock.registerValueCallback()**

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 (YRealTimeClockValueCallback callback)
m	-(int) registerValueCallback : (YRealTimeClockValueCallback) callback
pas	function registerValueCallback (callback : TYRealTimeClockValueCallback): 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.

realtimeclock→set_logicalName()
realtimeclock→setLogicalName()
realtimeclock.set_logicalName()

YRealTimeClock

Modifie le nom logique de l'horloge.

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	YRealTimeClock 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 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.set_unixTime()

YRealTimeClock

Modifie l'heure courante.

js	function set_unixTime (newval)
nodejs	function set_unixTime (newval)
php	function set_unixTime (\$newval)
cpp	int set_unixTime (s64 newval)
m	-(int) setUnixTime : (s64) newval
pas	function set_unixTime (newval : int64): integer
vb	function set_unixTime (ByVal newval As Long) As Integer
cs	int set_unixTime (long newval)
java	int set_unixTime (long newval)
py	def set_unixTime (newval)
cmd	YRealTimeClock target set_unixTime newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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.set_utcOffset()**

Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

js	function set_utcOffset (newval)
nodejs	function set_utcOffset (newval)
php	function set_utcOffset (\$newval)
cpp	int set_utcOffset (int newval)
m	-(int) setUtcOffset : (int) newval
pas	function set_utcOffset (newval : LongInt): integer
vb	function set_utcOffset (ByVal newval As Integer) As Integer
cs	int set_utcOffset (int newval)
java	int set_utcOffset (int newval)
py	def set_utcOffset (newval)
cmd	YRealTimeClock target set_utcOffset newval

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><script type='text/javascript' src='yocto_refframe.js'></script></code>
nodejs	<code>var yoctolib = require('yoctolib'); var YRefFrame = yoctolib.YRefFrame;</code>
php	<code>require_once('yocto_refframe.php');</code>
cpp	<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()

3. Reference

	Retourne le cap de référence utilisé par le compas.
refframe → get_errorMessage()	Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.
refframe → get_errorType()	Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du référentiel.
refframe → get_friendlyName()	Retourne un identifiant global du référentiel au format <code>NOM_MODULE . NOM_FONCTION</code> .
refframe → get_functionDescriptor()	Retourne un identifiant unique de type <code>YFUN_DESCR</code> correspondant à la fonction.
refframe → get_functionId()	Retourne l'identifiant matériel du référentiel, sans référence au module.
refframe → get_hardwareId()	Retourne l'identifiant matériel unique du référentiel au format <code>SERIAL . FUNCTIONID</code> .
refframe → get_logicalName()	Retourne le nom logique du référentiel.
refframe → get_module()	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
refframe → get_module_async(callback, context)	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
refframe → get_mountOrientation()	Retourne l'orientation à l'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.
refframe → get_mountPosition()	Retourne la position d'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.
refframe → get_userData()	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
refframe → isOnline()	Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.
refframe → isOnline_async(callback, context)	Vérifie si le module hébergeant le référentiel est joignable, sans déclencher d'erreur.
refframe → load(msValidity)	Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.
refframe → load_async(msValidity, callback, context)	Met en cache les valeurs courantes du référentiel, avec une durée de validité spécifiée.
refframe → more3DCalibration()	Continue le processus de calibration tridimensionnelle des capteurs initié avec la méthode <code>start3DCalibration</code> .
refframe → nextRefFrame()	Continue l'énumération des référentiels commencée à l'aide de <code>yFirstRefFrame()</code> .
refframe → registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
refframe → save3DCalibration()	Applique les paramètres de calibration tridimensionnelle précédemment calculés.
refframe → set_bearing(newval)	

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()**YRefFrame****yFindRefFrame()****YRefFrame.FindRefFrame()**

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

js	function yFindRefFrame (func)
nodejs	function FindRefFrame (func)
php	function yFindRefFrame (\$func)
c++	YRefFrame* yFindRefFrame (const string& func)
m	YRefFrame* yFindRefFrame (NSString* func)
pas	function yFindRefFrame (func : string): TYRefFrame
vb	function yFindRefFrame (ByVal func As String) As YRefFrame
cs	YRefFrame FindRefFrame (string func)
java	YRefFrame FindRefFrame (String func)
py	def FindRefFrame (func)

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()**YRefFrame****yFirstRefFrame()****YRefFrame.FirstRefFrame()**

Commence l'énumération des référentiels accessibles par la librairie.

js	function yFirstRefFrame ()
nodejs	function FirstRefFrame ()
php	function yFirstRefFrame ()
cpp	YRefFrame* yFirstRefFrame ()
m	YRefFrame* yFirstRefFrame ()
pas	function yFirstRefFrame (): TYRefFrame
vb	function yFirstRefFrame () As YRefFrame
cs	YRefFrame FirstRefFrame ()
java	YRefFrame FirstRefFrame ()
py	def FirstRefFrame ()

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()****YRefFrame****refframe.cancel3DCalibration()**

Annule la calibration tridimensionnelle en cours, et rétabli les réglages normaux.

js	function cancel3DCalibration ()
nodejs	function cancel3DCalibration ()
php	function cancel3DCalibration ()
cpp	int cancel3DCalibration ()
m	-(int) cancel3DCalibration
pas	function cancel3DCalibration (): LongInt
vb	function cancel3DCalibration () As Integer
cs	int cancel3DCalibration ()
java	int cancel3DCalibration ()
py	def cancel3DCalibration ()
cmd	YRefFrame target cancel3DCalibration

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

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()****YRefFrame****refframe**→**3DCalibrationHint()****refframe.get_3DCalibrationHint()**

Retourne les instructions à suivre pour procéder à la calibration tridimensionnelle initiée avec la méthode `start3DCalibration`.

js	function get_3DCalibrationHint ()
nodejs	function get_3DCalibrationHint ()
php	function get_3DCalibrationHint ()
cpp	string get_3DCalibrationHint ()
m	-(NSString*) 3DCalibrationHint
pas	function get_3DCalibrationHint (): string
vb	function get_3DCalibrationHint () As String
cs	string get_3DCalibrationHint ()
java	String get_3DCalibrationHint ()
py	def get_3DCalibrationHint ()
cmd	YRefFrame target get_3DCalibrationHint

Retourne :

une chaîne de caractères.

refframe→get_3DCalibrationLogMsg()**YRefFrame****refframe→3DCalibrationLogMsg()****refframe.get_3DCalibrationLogMsg()**

Retourne le dernier message de log produit par le processus de calibration.

js	function get_3DCalibrationLogMsg ()
nodejs	function get_3DCalibrationLogMsg ()
php	function get_3DCalibrationLogMsg ()
cpp	string get_3DCalibrationLogMsg ()
m	-(NSString*) 3DCalibrationLogMsg
pas	function get_3DCalibrationLogMsg (): string
vb	function get_3DCalibrationLogMsg () As String
cs	string get_3DCalibrationLogMsg ()
java	String get_3DCalibrationLogMsg ()
py	def get_3DCalibrationLogMsg ()
cmd	YRefFrame target get_3DCalibrationLogMsg

Si aucun nouveau message n'est disponible, retourne une chaîne vide.

Retourne :

une chaîne de caractères.

refframe→get_3DCalibrationProgress()

YRefFrame

refframe→3DCalibrationProgress()

refframe.get_3DCalibrationProgress()

Retourne l'avancement global du processus de calibration tridimensionnelle initié avec la méthode `start3DCalibration`.

js	function get_3DCalibrationProgress ()
nodejs	function get_3DCalibrationProgress ()
php	function get_3DCalibrationProgress ()
cpp	int get_3DCalibrationProgress ()
m	-(int) 3DCalibrationProgress
pas	function get_3DCalibrationProgress (): LongInt
vb	function get_3DCalibrationProgress () As Integer
cs	int get_3DCalibrationProgress ()
java	int get_3DCalibrationProgress ()
py	def get_3DCalibrationProgress ()
cmd	YRefFrame target get_3DCalibrationProgress

Retourne :

une nombre entier entre 0 (pas commencé) et 100 (terminé).

refframe→get_3DCalibrationStage()**YRefFrame****refframe→3DCalibrationStage()****refframe.get_3DCalibrationStage()**

Retourne l'index de l'étape courante de la calibration initiée avec la méthode `start3DCalibration`.

js	function get_3DCalibrationStage ()
nodejs	function get_3DCalibrationStage ()
php	function get_3DCalibrationStage ()
cpp	int get_3DCalibrationStage ()
m	-(int) 3DCalibrationStage
pas	function get_3DCalibrationStage (): LongInt
vb	function get_3DCalibrationStage () As Integer
cs	int get_3DCalibrationStage ()
java	int get_3DCalibrationStage ()
py	def get_3DCalibrationStage ()
cmd	YRefFrame target get_3DCalibrationStage

Retourne :

une nombre entier, croissant au fur et à mesure de la complétion des étapes.

refframe→get_3DCalibrationStageProgress()**YRefFrame****refframe→3DCalibrationStageProgress()****refframe.get_3DCalibrationStageProgress()**

Retourne l'avancement de l'étape courante de la calibration initiée avec la méthode `start3DCalibration`.

js	function get_3DCalibrationStageProgress ()
nodejs	function get_3DCalibrationStageProgress ()
php	function get_3DCalibrationStageProgress ()
cpp	int get_3DCalibrationStageProgress ()
m	-(int) 3DCalibrationStageProgress
pas	function get_3DCalibrationStageProgress (): LongInt
vb	function get_3DCalibrationStageProgress () As Integer
cs	int get_3DCalibrationStageProgress ()
java	int get_3DCalibrationStageProgress ()
py	def get_3DCalibrationStageProgress ()
cmd	YRefFrame target get_3DCalibrationStageProgress

Retourne :

une nombre entier entre 0 (pas commencé) et 100 (terminé).

refframe→get_advertisedValue()**YRefFrame****refframe→advertisedValue()****refframe.get_advertisedValue()**

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

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

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.get_bearing()**

Retourne le cap de référence utilisé par le compas.

js	function get_bearing ()
nodejs	function get_bearing ()
php	function get_bearing ()
cpp	double get_bearing ()
m	-(double) bearing
pas	function get_bearing (): double
vb	function get_bearing () As Double
cs	double get_bearing ()
java	double get_bearing ()
py	def get_bearing ()
cmd	YRefFrame target get_bearing

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.get_errorMessage()**

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

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

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()****refframe.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du référentiel au format `NOM_MODULE.NOM_FONCTION`.

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

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()**YRefFrame****refframe→functionDescriptor()****refframe.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du référentiel, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du référentiel au format SERIAL . FUNCTIONID.

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

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.get_logicalName()**

Retourne le nom logique du référentiel.

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

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.get_module()**

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

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

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 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.

refframe→get_mountOrientation()**YRefFrame****refframe→mountOrientation()****refframe.get_mountOrientation()**

Retourne l'orientation à l'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.

js	function get_mountOrientation ()
nodejs	function get_mountOrientation ()
php	function get_mountOrientation ()
cpp	Y_MOUNTORIENTATION get_mountOrientation ()
m	-(Y_MOUNTORIENTATION) mountOrientation
pas	function get_mountOrientation (): TY_MOUNTORIENTATION
vb	function get_mountOrientation () As Y_MOUNTORIENTATION
cs	MOUNTORIENTATION get_mountOrientation ()
java	MOUNTORIENTATION get_mountOrientation ()
py	def get_mountOrientation ()
cmd	YRefFrame target get_mountOrientation

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.get_mountPosition()**

Retourne la position d'installation du module, telle que configurée afin de définir le référentiel de la boussole et des inclinomètres.

js	function get_mountPosition ()
nodejs	function get_mountPosition ()
php	function get_mountPosition ()
cpp	Y_MOUNTPOSITION get_mountPosition ()
m	-(Y_MOUNTPOSITION) mountPosition
pas	function get_mountPosition (): TYMOUNTPOSITION
vb	function get_mountPosition () As Y_MOUNTPOSITION
cs	MOUNTPOSITION get_mountPosition ()
java	MOUNTPOSITION get_mountPosition ()
py	def get_mountPosition ()
cmd	YRefFrame target get_mountPosition

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.get_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.

refframe→**isOnline()****refframe.isOnline()****YRefFrame**

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

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

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 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.

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-quel à la fonction de callback

Retourne :

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

refframe→more3DCalibration()**YRefFrame****refframe.more3DCalibration()**

Continue le processus de calibration tridimensionnelle des capteurs initié avec la méthode `start3DCalibration`.

js	function more3DCalibration ()
nodejs	function more3DCalibration ()
php	function more3DCalibration ()
cpp	int more3DCalibration ()
m	-(int) more3DCalibration
pas	function more3DCalibration (): LongInt
vb	function more3DCalibration () As Integer
cs	int more3DCalibration ()
java	int more3DCalibration ()
py	def more3DCalibration ()
cmd	YRefFrame target more3DCalibration

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()**YRefFrame****refframe.nextRefFrame()**

Continue l'énumération des référentiels commencée à l'aide de `yFirstRefFrame()`.

js	function nextRefFrame ()
nodejs	function nextRefFrame ()
php	function nextRefFrame ()
c++	YRefFrame * nextRefFrame ()
m	-(YRefFrame*) nextRefFrame
pas	function nextRefFrame (): TYRefFrame
vb	function nextRefFrame () As YRefFrame
cs	YRefFrame nextRefFrame ()
java	YRefFrame nextRefFrame ()
py	def nextRefFrame ()

Retourne :

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

refframe→registerValueCallback()**YRefFrame****refframe.registerValueCallback()**

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 (YRefFrameValueCallback callback)
m	-(int) registerValueCallback : (YRefFrameValueCallback) callback
pas	function registerValueCallback (callback : TYRefFrameValueCallback): 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.

refframe→save3DCalibration()**YRefFrame****refframe.save3DCalibration()**

Applique les paramètres de calibration tridimensionnelle précédemment calculés.

js	function save3DCalibration ()
nodejs	function save3DCalibration ()
php	function save3DCalibration ()
cpp	int save3DCalibration ()
m	-(int) save3DCalibration
pas	function save3DCalibration (): LongInt
vb	function save3DCalibration () As Integer
cs	int save3DCalibration ()
java	int save3DCalibration ()
py	def save3DCalibration ()
cmd	YRefFrame target save3DCalibration

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.set_bearing()**

Modifie le cap de référence utilisé par le compas.

js	function set_bearing (newval)
nodejs	function set_bearing (newval)
php	function set_bearing (\$newval)
cpp	int set_bearing (double newval)
m	-(int) setBearing : (double) newval
pas	function set_bearing (newval : double): integer
vb	function set_bearing (ByVal newval As Double) As Integer
cs	int set_bearing (double newval)
java	int set_bearing (double newval)
py	def set_bearing (newval)
cmd	YRefFrame target set_bearing newval

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.set_logicalName()**

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.set_mountPosition()**

Modifie le référentiel de la boussole et des inclinomètres.

js	function set_mountPosition (position , orientation)
nodejs	function set_mountPosition (position , orientation)
php	function set_mountPosition (\$position , \$orientation)
cpp	int set_mountPosition (Y_MOUNTPOSITION position , Y_MOUNTORIENTATION orientation)
m	-(int) setMountPosition : (Y_MOUNTPOSITION) position : (Y_MOUNTORIENTATION) orientation
pas	function set_mountPosition (position : TYMOUNTPOSITION, orientation : TYMOUNTORIENTATION): LongInt
vb	function set_mountPosition () As Integer
cs	int set_mountPosition (MOUNTPOSITION position , MOUNTORIENTATION orientation)
java	int set_mountPosition (MOUNTPOSITION position , MOUNTORIENTATION orientation)
py	def set_mountPosition (position , orientation)
cmd	YRefFrame target set_mountPosition position orientation

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.set_userdata()**

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

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

refframe→start3DCalibration()**YRefFrame****refframe.start3DCalibration()**

Initie le processus de calibration tridimensionnelle des capteurs.

js	function start3DCalibration ()
nodejs	function start3DCalibration ()
php	function start3DCalibration ()
cpp	int start3DCalibration ()
m	-(int) start3DCalibration
pas	function start3DCalibration (): LongInt
vb	function start3DCalibration () As Integer
cs	int start3DCalibration ()
java	int start3DCalibration ()
py	def start3DCalibration ()
cmd	YRefFrame target start3DCalibration

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><script type='text/javascript' src='yocto_relay.js'></script></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()

YRelay

yFindRelay()YRelay.FindRelay()

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

js	function yFindRelay (func)
nodejs	function FindRelay (func)
php	function yFindRelay (\$func)
cpp	YRelay* yFindRelay (const string& func)
m	YRelay* yFindRelay (NSString* func)
pas	function yFindRelay (func : string): TYRelay
vb	function yFindRelay (ByVal func As String) As YRelay
cs	YRelay FindRelay (string func)
java	YRelay FindRelay (String func)
py	def FindRelay (func)

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()**YRelay****yFirstRelay()****YRelay.FirstRelay()**

Commence l'énumération des relais accessibles par la librairie.

js	function yFirstRelay ()
nodejs	function FirstRelay ()
php	function yFirstRelay ()
cpp	YRelay* yFirstRelay ()
m	YRelay* yFirstRelay ()
pas	function yFirstRelay (): TYRelay
vb	function yFirstRelay () As YRelay
cs	YRelay FirstRelay ()
java	YRelay FirstRelay ()
py	def FirstRelay ()

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 delayedPulse (ms_delay , ms_duration)
nodejs	function delayedPulse (ms_delay , ms_duration)
php	function delayedPulse (\$ms_delay , \$ms_duration)
cpp	int delayedPulse (int ms_delay , int ms_duration)
m	-(int) delayedPulse : (int) ms_delay : (int) ms_duration
pas	function delayedPulse (ms_delay : LongInt, ms_duration : LongInt): integer
vb	function delayedPulse (ByVal ms_delay As Integer, ByVal ms_duration As Integer) As Integer
cs	int delayedPulse (int ms_delay , int ms_duration)
java	int delayedPulse (int ms_delay , int ms_duration)
py	def delayedPulse (ms_delay , ms_duration)
cmd	YRelay target delayedPulse ms_delay ms_duration

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

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.get_advertisedValue()**

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

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

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.get_countdown()`

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`.

js	function get_countdown ()
nodejs	function get_countdown ()
php	function get_countdown ()
cpp	s64 get_countdown ()
m	-(s64) countdown
pas	function get_countdown (): int64
vb	function get_countdown () As Long
cs	long get_countdown ()
java	long get_countdown ()
py	def get_countdown ()
cmd	YRelay target get_countdown

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

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()****relay.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du relais au format `NOM_MODULE.NOM_FONCTION`.

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

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.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du relais, sans référence au module.

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

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()****YRelay****relay**→**hardwareId()****relay.get_hardwareId()**

Retourne l'identifiant matériel unique du relais au format `SERIAL.FUNCTIONID`.

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

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.get_logicalName()**

Retourne le nom logique du relais.

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

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.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.

js	function get_maxTimeOnStateA ()
nodejs	function get_maxTimeOnStateA ()
php	function get_maxTimeOnStateA ()
cpp	s64 get_maxTimeOnStateA ()
m	-(s64) maxTimeOnStateA
pas	function get_maxTimeOnStateA (): int64
vb	function get_maxTimeOnStateA () As Long
cs	long get_maxTimeOnStateA ()
java	long get_maxTimeOnStateA ()
py	def get_maxTimeOnStateA ()
cmd	YRelay target get_maxTimeOnStateA

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.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.

js	function get_maxTimeOnStateB ()
nodejs	function get_maxTimeOnStateB ()
php	function get_maxTimeOnStateB ()
cpp	s64 get_maxTimeOnStateB ()
m	-(s64) maxTimeOnStateB
pas	function get_maxTimeOnStateB (): int64
vb	function get_maxTimeOnStateB () As Long
cs	long get_maxTimeOnStateB ()
java	long get_maxTimeOnStateB ()
py	def get_maxTimeOnStateB ()
cmd	YRelay target get_maxTimeOnStateB

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()****YRelay****relay**→**module()****relay.get_module()**

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

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

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.get_output()**

Retourne l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

js	function get_output ()
nodejs	function get_output ()
php	function get_output ()
cpp	Y_OUTPUT_enum get_output ()
m	-(Y_OUTPUT_enum) output
pas	function get_output (): Integer
vb	function get_output () As Integer
cs	int get_output ()
java	int get_output ()
py	def get_output ()
cmd	YRelay target get_output

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.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.

js	function get_pulseTimer ()
nodejs	function get_pulseTimer ()
php	function get_pulseTimer ()
cpp	s64 get_pulseTimer ()
m	-(s64) pulseTimer
pas	function get_pulseTimer (): int64
vb	function get_pulseTimer () As Long
cs	long get_pulseTimer ()
java	long get_pulseTimer ()
py	def get_pulseTimer ()
cmd	YRelay target get_pulseTimer

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.get_state()**

Retourne l'état du relais (A pour la position de repos, B pour l'état actif).

js	function get_state ()
nodejs	function get_state ()
php	function get_state ()
cpp	Y_STATE_enum get_state ()
m	-(Y_STATE_enum) state
pas	function get_state (): Integer
vb	function get_state () As Integer
cs	int get_state ()
java	int get_state ()
py	def get_state ()
cmd	YRelay target get_state

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.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).

js	function get_stateAtPowerOn ()
nodejs	function get_stateAtPowerOn ()
php	function get_stateAtPowerOn ()
cpp	Y_STATEATPOWERON_enum get_stateAtPowerOn ()
m	-(Y_STATEATPOWERON_enum) stateAtPowerOn
pas	function get_stateAtPowerOn (): Integer
vb	function get_stateAtPowerOn () As Integer
cs	int get_stateAtPowerOn ()
java	int get_stateAtPowerOn ()
py	def get_stateAtPowerOn ()
cmd	YRelay target get_stateAtPowerOn

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

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.

relay→**isOnline()****relay.isOnline()****YRelay**

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

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

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 load (msValidity)
nodejs	function load (msValidity)
php	function load (\$msValidity)
c++	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.

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 nextRelay ()
nodejs	function nextRelay ()
php	function nextRelay ()
cpp	YRelay * nextRelay ()
m	-(YRelay*) nextRelay
pas	function nextRelay (): TYRelay
vb	function nextRelay () As YRelay
cs	YRelay nextRelay ()
java	YRelay nextRelay ()
py	def nextRelay ()

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()****YRelay****relay.registerValueCallback()**

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.set_logicalName()**

Modifie le nom logique du relais.

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	YRelay 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 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.set_maxTimeOnStateA()**

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 set_maxTimeOnStateA( newval)
nodejs function set_maxTimeOnStateA( newval)
php    function set_maxTimeOnStateA( $newval)
cpp    int set_maxTimeOnStateA( s64 newval)
m      -(int) setMaxTimeOnStateA : (s64) newval
pas    function set_maxTimeOnStateA( newval: int64): integer
vb      function set_maxTimeOnStateA( ByVal newval As Long) As Integer
cs      int set_maxTimeOnStateA( long newval)
java    int set_maxTimeOnStateA( long newval)
py      def set_maxTimeOnStateA( newval)
cmd     YRelay target set_maxTimeOnStateA newval

```

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()****YRelay****relay**→**setMaxTimeOnStateB()****relay.set_maxTimeOnStateB()**

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 set_maxTimeOnStateB (newval)
nodejs	function set_maxTimeOnStateB (newval)
php	function set_maxTimeOnStateB (\$newval)
cpp	int set_maxTimeOnStateB (s64 newval)
m	-(int) setMaxTimeOnStateB : (s64) newval
pas	function set_maxTimeOnStateB (newval : int64): integer
vb	function set_maxTimeOnStateB (ByVal newval As Long) As Integer
cs	int set_maxTimeOnStateB (long newval)
java	int set_maxTimeOnStateB (long newval)
py	def set_maxTimeOnStateB (newval)
cmd	YRelay target set_maxTimeOnStateB newval

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.set_output()**

Modifie l'état de la sortie du relais, lorsqu'il est utilisé comme un simple interrupteur.

js	function set_output (newval)
nodejs	function set_output (newval)
php	function set_output (\$newval)
cpp	int set_output (Y_OUTPUT_enum newval)
m	-(int) setOutput : (Y_OUTPUT_enum) newval
pas	function set_output (newval : Integer): integer
vb	function set_output (ByVal newval As Integer) As Integer
cs	int set_output (int newval)
java	int set_output (int newval)
py	def set_output (newval)
cmd	YRelay target set_output newval

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.set_state()**

Modifie l'état du relais (A pour la position de repos, B pour l'état actif).

js	function set_state (newval)
nodejs	function set_state (newval)
php	function set_state (\$newval)
cpp	int set_state (Y_STATE_enum newval)
m	-(int) setState : (Y_STATE_enum) newval
pas	function set_state (newval : Integer): integer
vb	function set_state (ByVal newval As Integer) As Integer
cs	int set_state (int newval)
java	int set_state (int newval)
py	def set_state (newval)
cmd	YRelay target set_state newval

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.set_stateAtPowerOn()**

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 set_stateAtPowerOn (newval)
nodejs	function set_stateAtPowerOn (newval)
php	function set_stateAtPowerOn (\$newval)
cpp	int set_stateAtPowerOn (Y_STATEATPOWERON_enum newval)
m	-(int) setStateAtPowerOn : (Y_STATEATPOWERON_enum) newval
pas	function set_stateAtPowerOn (newval : Integer): integer
vb	function set_stateAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_stateAtPowerOn (int newval)
java	int set_stateAtPowerOn (int newval)
py	def set_stateAtPowerOn (newval)
cmd	YRelay target set_stateAtPowerOn newval

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.set_userdata()**

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

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

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()**YSensor****yFindSensor()****YSensor.FindSensor()**

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

js	function yFindSensor (func)
nodejs	function FindSensor (func)
php	function yFindSensor (\$func)
cpp	YSensor* yFindSensor (string func)
m	+(YSensor*) yFindSensor : (NSString*) func
pas	function yFindSensor (func : string): TYSensor
vb	function yFindSensor (ByVal func As String) As YSensor
cs	YSensor FindSensor (string func)
java	YSensor FindSensor (String func)
py	def FindSensor (func)

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()**YSensor****yFirstSensor()****YSensor.FirstSensor()**

Commence l'énumération des senseurs accessibles par la librairie.

js	function yFirstSensor ()
nodejs	function FirstSensor ()
php	function yFirstSensor ()
cpp	YSensor* yFirstSensor ()
m	YSensor* yFirstSensor ()
pas	function yFirstSensor (): TYSensor
vb	function yFirstSensor () As YSensor
cs	YSensor FirstSensor ()
java	YSensor FirstSensor ()
py	def FirstSensor ()

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()**YSensor****sensor.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 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.

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

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.get_advertisedValue()**

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

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

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()****YSensor****sensor**→**currentRawValue()****sensor.get_currentRawValue()**

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

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YSensor target get_currentRawValue

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.get_currentValue()**

Retourne la valeur actuelle de la mesure.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YSensor target get_currentValue

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 senseur.

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

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 senseur.

sensor→**get_errorType()****YSensor****sensor**→**errorType()****sensor.get_errorType()**

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

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

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.get_friendlyName()**

Retourne un identifiant global du senseur au format `NOM_MODULE.NOM_FONCTION`.

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

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()****YSensor****sensor**→**functionDescriptor()****sensor.get_functionDescriptor()**

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

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

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.get_functionId()**

Retourne l'identifiant matériel du senseur, sans référence au module.

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

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.get_hardwareId()**

Retourne l'identifiant matériel unique du senseur au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_highestValue()**

Retourne la valeur maximale observée pour la mesure depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YSensor target get_highestValue

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.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.

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

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()****YSensor****sensor**→**logicalName()****sensor.get_logicalName()**

Retourne le nom logique du senseur.

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

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.get_lowestValue()**

Retourne la valeur minimale observée pour la mesure depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YSensor target get_lowestValue

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.get_module()**

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

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

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.get_recordedData()**

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

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

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.

sensor→**get_reportFrequency()****YSensor****sensor**→**reportFrequency()****sensor.get_reportFrequency()**

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

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

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YSensor target get_resolution

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()****YSensor****sensor**→**unit()****sensor.get_unit()**

Retourne l'unité dans laquelle la mesure est exprimée.

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

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 get_userData ()
nodejs	function get_userData ()
php	function get_userData ()
cpp	void * get_userData ()
m	-(void*) userData
pas	function get_userData (): Tobject
vb	function get_userData () As Object
cs	object get_userData ()
java	Object get_userData ()
py	def get_userData ()

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.

sensor→**isOnline()****sensor.isOnline()****YSensor**

Vérifie si le module hébergeant le senseur est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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-quel 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 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.

sensor→loadCalibrationPoints()**YSensor****sensor.loadCalibrationPoints()**

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 nextSensor ()
nodejs	function nextSensor ()
php	function nextSensor ()
cpp	YSensor * nextSensor ()
m	-(YSensor*) nextSensor
pas	function nextSensor (): TYSensor
vb	function nextSensor () As YSensor
cs	YSensor nextSensor ()
java	YSensor nextSensor ()
py	def nextSensor ()

Retourne :

un pointeur sur un objet `YSensor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

sensor→registerTimedReportCallback()**YSensor****sensor.registerTimedReportCallback()**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YSensorTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YSensorTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYSensorTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YSensor****sensor.registerValueCallback()**

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 (YSensorValueCallback callback)
m	-(int) registerValueCallback : (YSensorValueCallback) callback
pas	function registerValueCallback (callback : TYSensorValueCallback): 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.

sensor→**set_highestValue()****YSensor****sensor**→**setHighestValue()****sensor.set_highestValue()**

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	YSensor 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.

sensor→**set_logFrequency()**
sensor→**setLogFrequency()**
sensor.set_logFrequency()

YSensor

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	YSensor 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.

sensor→**set_logicalName()****YSensor****sensor**→**setLogicalName()****sensor.set_logicalName()**

Modifie le nom logique du senseur.

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	YSensor 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

js	function set_lowestValue (newval)
nodejs	function set_lowestValue (newval)
php	function set_lowestValue (\$newval)
cpp	int set_lowestValue (double newval)
m	-(int) setLowestValue : (double) newval
pas	function set_lowestValue (newval : double): integer
vb	function set_lowestValue (ByVal newval As Double) As Integer
cs	int set_lowestValue (double newval)
java	int set_lowestValue (double newval)
py	def set_lowestValue (newval)
cmd	YSensor target set_lowestValue newval

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()****YSensor****sensor**→**setReportFrequency()****sensor.set_reportFrequency()**

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	YSensor 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.

sensor→**set_resolution()****YSensor****sensor**→**setResolution()****sensor.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

js	function set_resolution (newval)
nodejs	function set_resolution (newval)
php	function set_resolution (\$newval)
cpp	int set_resolution (double newval)
m	-(int) setResolution : (double) newval
pas	function set_resolution (newval : double): integer
vb	function set_resolution (ByVal newval As Double) As Integer
cs	int set_resolution (double newval)
java	int set_resolution (double newval)
py	def set_resolution (newval)
cmd	YSensor target set_resolution newval

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.set_userData()**

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 set_userData (data)
nodejs	function set_userData (data)
php	function set_userData (\$data)
cpp	void set_userData (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userData (data : Tobject)
vb	procedure set_userData (ByVal data As Object)
cs	void set_userData (object data)
java	void set_userData (Object data)
py	def set_userData (data)

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><script type='text/javascript' src='yocto_servo.js'></script></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()**YServo****yFindServo()****YServo.FindServo()**

Permet de retrouver un servo d'après un identifiant donné.

js	function yFindServo (func)
nodejs	function FindServo (func)
php	function yFindServo (\$func)
c++	YServo* yFindServo (const string& func)
m	YServo* yFindServo (NSString* func)
pas	function yFindServo (func : string): TYServo
vb	function yFindServo (ByVal func As String) As YServo
cs	YServo FindServo (string func)
java	YServo FindServo (String func)
py	def FindServo (func)

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()**YServo****yFirstServo()****YServo.FirstServo()**

Commence l'énumération des servo accessibles par la librairie.

js	function yFirstServo ()
nodejs	function FirstServo ()
php	function yFirstServo ()
cpp	YServo* yFirstServo ()
m	YServo* yFirstServo ()
pas	function yFirstServo (): TYServo
vb	function yFirstServo () As YYServo
cs	YServo FirstServo ()
java	YServo FirstServo ()
py	def FirstServo ()

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()**

Retourne la valeur courante du servo (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YServo target get_advertisedValue

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.get_enabled()`

Retourne l'état de fonctionnement du \$FUNCTION\$.

js	function get_enabled ()
nodejs	function get_enabled ()
php	function get_enabled ()
cpp	Y_ENABLED_enum get_enabled ()
m	-(Y_ENABLED_enum) enabled
pas	function get_enabled (): Integer
vb	function get_enabled () As Integer
cs	int get_enabled ()
java	int get_enabled ()
py	def get_enabled ()
cmd	YServo target get_enabled

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.get_enabledAtPowerOn()**

Retourne l'état du générateur de signal de commande du servo au démarrage du module.

js	function get_enabledAtPowerOn ()
nodejs	function get_enabledAtPowerOn ()
php	function get_enabledAtPowerOn ()
cpp	Y_ENABLEDATPOWERON_enum get_enabledAtPowerOn ()
m	-(Y_ENABLEDATPOWERON_enum) enabledAtPowerOn
pas	function get_enabledAtPowerOn (): Integer
vb	function get_enabledAtPowerOn () As Integer
cs	int get_enabledAtPowerOn ()
java	int get_enabledAtPowerOn ()
py	def get_enabledAtPowerOn ()
cmd	YServo target get_enabledAtPowerOn

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 get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****servo.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du servo.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()`

Retourne un identifiant global du servo au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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()****YServo****servo**→**functionId()****servo.get_functionId()**

Retourne l'identifiant matériel du servo, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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()****YServo****servo**→**hardwareId()****servo.get_hardwareId()**

Retourne l'identifiant matériel unique du servo au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_logicalName()**

Retourne le nom logique du servo.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YServo target get_logicalName

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.get_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
c++	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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.

servo→**get_neutral()****YServo****servo**→**neutral()****servo.get_neutral()**

Retourne la durée en microsecondes de l'impulsion correspondant au neutre du servo.

js	function get_neutral ()
nodejs	function get_neutral ()
php	function get_neutral ()
cpp	int get_neutral ()
m	-(int) neutral
pas	function get_neutral (): LongInt
vb	function get_neutral () As Integer
cs	int get_neutral ()
java	int get_neutral ()
py	def get_neutral ()
cmd	YServo target get_neutral

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.get_position()`

Retourne la position courante du servo.

js	function get_position ()
nodejs	function get_position ()
php	function get_position ()
cpp	int get_position ()
m	-(int) position
pas	function get_position (): LongInt
vb	function get_position () As Integer
cs	int get_position ()
java	int get_position ()
py	def get_position ()
cmd	YServo target get_position

Retourne :

un entier représentant la position courante du servo

En cas d'erreur, déclenche une exception ou retourne Y_POSITION_INVALID.

servo→**get_positionAtPowerOn()****YServo****servo**→**positionAtPowerOn()****servo.get_positionAtPowerOn()**

Retourne la position du servo au démarrage du module.

js	function get_positionAtPowerOn ()
nodejs	function get_positionAtPowerOn ()
php	function get_positionAtPowerOn ()
cpp	int get_positionAtPowerOn ()
m	-(int) positionAtPowerOn
pas	function get_positionAtPowerOn (): LongInt
vb	function get_positionAtPowerOn () As Integer
cs	int get_positionAtPowerOn ()
java	int get_positionAtPowerOn ()
py	def get_positionAtPowerOn ()
cmd	YServo target get_positionAtPowerOn

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()****YServo****servo**→**range()****servo.get_range()**

Retourne la plage d'utilisation du servo.

js	function get_range ()
nodejs	function get_range ()
php	function get_range ()
cpp	int get_range ()
m	-(int) range
pas	function get_range (): LongInt
vb	function get_range () As Integer
cs	int get_range ()
java	int get_range ()
py	def get_range ()
cmd	YServo target get_range

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.get_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function get_userData ()
nodejs	function get_userData ()
php	function get_userData ()
cpp	void * get_userData ()
m	-(void*) userData
pas	function get_userData (): Tobject
vb	function get_userData () As Object
cs	object get_userData ()
java	Object get_userData ()
py	def get_userData ()

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.

servo→**isOnline()****servo.isOnline()****YServo**

Vérifie si le module hébergeant le servo est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 (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.

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 move (target , ms_duration)
nodejs	function move (target , ms_duration)
php	function move (\$target , \$ms_duration)
cpp	int move (int target , int ms_duration)
m	-(int) move : (int) target : (int) ms_duration
pas	function move (target : LongInt, ms_duration : LongInt): integer
vb	function move (ByVal target As Integer, ByVal ms_duration As Integer) As Integer
cs	int move (int target , int ms_duration)
java	int move (int target , int ms_duration)
py	def move (target , ms_duration)
cmd	YServo target move target ms_duration

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 nextServo ()
nodejs	function nextServo ()
php	function nextServo ()
cpp	YServo * nextServo ()
m	-(YServo*) nextServo
pas	function nextServo (): TYServo
vb	function nextServo () As YServo
cs	YServo nextServo ()
java	YServo nextServo ()
py	def nextServo ()

Retourne :

un pointeur sur un objet `YServo` accessible en ligne, ou `null` lorsque l'énumération est terminée.

servo→registerValueCallback()**YServo****servo.registerValueCallback()**

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 (YServoValueCallback callback)
m	-(int) registerValueCallback : (YServoValueCallback) callback
pas	function registerValueCallback (callback : TYServoValueCallback): 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'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 set_enabled (newval)
nodejs	function set_enabled (newval)
php	function set_enabled (\$newval)
cpp	int set_enabled (Y_ENABLED_enum newval)
m	-(int) setEnabled : (Y_ENABLED_enum) newval
pas	function set_enabled (newval : Integer): integer
vb	function set_enabled (ByVal newval As Integer) As Integer
cs	int set_enabled (int newval)
java	int set_enabled (int newval)
py	def set_enabled (newval)
cmd	YServo target set_enabled newval

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()**YServo****servo→setEnabledAtPowerOn()****servo.set_enabledAtPowerOn()**

Configure l'état du générateur de signal de commande du servo au démarrage du module.

js	function set_enabledAtPowerOn (newval)
nodejs	function set_enabledAtPowerOn (newval)
php	function set_enabledAtPowerOn (\$newval)
cpp	int set_enabledAtPowerOn (Y_ENABLEDATPOWERON_enum newval)
m	-(int) setEnabledAtPowerOn : (Y_ENABLEDATPOWERON_enum) newval
pas	function set_enabledAtPowerOn (newval : Integer): integer
vb	function set_enabledAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_enabledAtPowerOn (int newval)
java	int set_enabledAtPowerOn (int newval)
py	def set_enabledAtPowerOn (newval)
cmd	YServo target set_enabledAtPowerOn newval

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()****servo**→**setLogicalName()****servo.set_logicalName()**

Modifie le nom logique du servo.

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	YServo 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 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.set_neutral()**

Modifie la durée de l'impulsion correspondant à la position neutre du servo.

js	function set_neutral (newval)
nodejs	function set_neutral (newval)
php	function set_neutral (\$newval)
cpp	int set_neutral (int newval)
m	-(int) setNeutral : (int) newval
pas	function set_neutral (newval : LongInt): integer
vb	function set_neutral (ByVal newval As Integer) As Integer
cs	int set_neutral (int newval)
java	int set_neutral (int newval)
py	def set_neutral (newval)
cmd	YServo target set_neutral newval

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()****YServo****servo**→**setPosition()****servo.set_position()**

Modifie immédiatement la consigne de position du servo.

js	function set_position (newval)
nodejs	function set_position (newval)
php	function set_position (\$newval)
cpp	int set_position (int newval)
m	-(int) setPosition : (int) newval
pas	function set_position (newval : LongInt): integer
vb	function set_position (ByVal newval As Integer) As Integer
cs	int set_position (int newval)
java	int set_position (int newval)
py	def set_position (newval)
cmd	YServo target set_position newval

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()**YServo****servo→setPositionAtPowerOn()****servo.set_positionAtPowerOn()**

Configure la position du servo au démarrage du module.

js	function set_positionAtPowerOn (newval)
nodejs	function set_positionAtPowerOn (newval)
php	function set_positionAtPowerOn (\$newval)
cpp	int set_positionAtPowerOn (int newval)
m	-(int) setPositionAtPowerOn : (int) newval
pas	function set_positionAtPowerOn (newval : LongInt): integer
vb	function set_positionAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_positionAtPowerOn (int newval)
java	int set_positionAtPowerOn (int newval)
py	def set_positionAtPowerOn (newval)
cmd	YServo target set_positionAtPowerOn newval

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.set_range()**

Modifie la plage d'utilisation du servo, en pourcents.

js	function set_range (newval)
nodejs	function set_range (newval)
php	function set_range (\$newval)
cpp	int set_range (int newval)
m	-(int) setRange : (int) newval
pas	function set_range (newval : LongInt): integer
vb	function set_range (ByVal newval As Integer) As Integer
cs	int set_range (int newval)
java	int set_range (int newval)
py	def set_range (newval)
cmd	YServo target set_range newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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	<script type='text/javascript' src='yocto_temperature.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YTemperature = yoctolib.YTemperature;
php	require_once('yocto_temperature.php');
c++	#include "yocto_temperature.h"
m	#import "yocto_temperature.h"
pas	uses yocto_temperature;
vb	yocto_temperature.vb
cs	yocto_temperature.cs
java	import com.yoctopuce.YoctoAPI.YTemperature;
py	from yocto_temperature import *

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.
temperature → get_highestValue()	Retourne la valeur maximale observée pour la température depuis le démarrage du module.
temperature → 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.
temperature → get_logicalName()	Retourne le nom logique du capteur de température.
temperature → get_lowestValue()	Retourne la valeur minimale observée pour la température depuis le démarrage du module.
temperature → get_module()	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
temperature → get_module_async(callback, context)	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
temperature → 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.
temperature → get_reportFrequency()	Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.
temperature → get_resolution()	Retourne la résolution des valeurs mesurées.
temperature → get_sensorType()	Retourne le type de capteur de température utilisé par le module
temperature → get_unit()	Retourne l'unité dans laquelle la température est exprimée.
temperature → get_userData()	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.
temperature → isOnline()	Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.
temperature → isOnline_async(callback, context)	Vérifie si le module hébergeant le capteur de température est joignable, sans déclencher d'erreur.
temperature → load(msValidity)	Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.
temperature → loadCalibrationPoints(rawValues, refValues)	Récupère les points de correction de mesure précédemment enregistrés à l'aide de la méthode calibrateFromPoints.
temperature → load_async(msValidity, callback, context)	Met en cache les valeurs courantes du capteur de température, avec une durée de validité spécifiée.
temperature → nextTemperature()	Continue l'énumération des capteurs de température commencée à l'aide de yFirstTemperature ().
temperature → registerTimedReportCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque notification périodique.
temperature → registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
temperature → set_highestValue(newval)	

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()**YTemperature****yFindTemperature()****YTemperature.FindTemperature()**

Permet de retrouver un capteur de température d'après un identifiant donné.

js	function yFindTemperature (func)
nodejs	function FindTemperature (func)
php	function yFindTemperature (\$func)
cpp	YTemperature* yFindTemperature (const string& func)
m	YTemperature* yFindTemperature (NSString* func)
pas	function yFindTemperature (func : string): TYTemperature
vb	function yFindTemperature (ByVal func As String) As YTemperature
cs	YTemperature FindTemperature (string func)
java	YTemperature FindTemperature (String func)
py	def FindTemperature (func)

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()**YTemperature****yFirstTemperature()****YTemperature.FirstTemperature()**

Commence l'énumération des capteurs de température accessibles par la librairie.

js	function yFirstTemperature ()
nodejs	function FirstTemperature ()
php	function yFirstTemperature ()
cpp	YTemperature* yFirstTemperature ()
m	YTemperature* yFirstTemperature ()
pas	function yFirstTemperature (): TYTemperature
vb	function yFirstTemperature () As YTemperature
cs	YTemperature FirstTemperature ()
java	YTemperature FirstTemperature ()
py	def FirstTemperature ()

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()**YTemperature****temperature.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 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.

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()

Retourne la valeur courante du capteur de température (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YTemperature target get_advertisedValue

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()****YTemperature****temperature**→**currentRawValue()****temperature.get_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YTemperature target get_currentRawValue

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()****YTemperature****temperature**→**currentValue()****temperature.get_currentValue()**

Retourne la valeur actuelle de la température.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YTemperature target get_currentValue

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()****YTemperature****temperature**→**errorMessage()****temperature.errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()

temperature.get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de température.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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()****YTemperature****temperature**→**friendlyName()****temperature.get_friendlyName()**

Retourne un identifiant global du capteur de température au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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()** **YTemperature**
temperature→**functionDescriptor()**
temperature.get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()**

Retourne l'identifiant matériel du capteur de température, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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.get_hardwareId()

Retourne l'identifiant matériel unique du capteur de température au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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()****YTemperature****temperature**→**highestValue()****temperature.get_highestValue()**

Retourne la valeur maximale observée pour la température depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YTemperature target get_highestValue

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()****YTemperature****temperature**→**logFrequency()****temperature.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.

js	function get_logFrequency ()
nodejs	function get_logFrequency ()
php	function get_logFrequency ()
cpp	string get_logFrequency ()
m	-(NSString*) logFrequency
pas	function get_logFrequency (): string
vb	function get_logFrequency () As String
cs	string get_logFrequency ()
java	String get_logFrequency ()
py	def get_logFrequency ()
cmd	YTemperature target get_logFrequency

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()****YTemperature****temperature**→**logicalName()****temperature.get_logicalName()**

Retourne le nom logique du capteur de température.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YTemperature target get_logicalName

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.get_lowestValue()**

Retourne la valeur minimale observée pour la température depuis le démarrage du module.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YTemperature target get_lowestValue

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.get_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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()**
temperature→**module_async()**

YTemperature

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()****YTemperature****temperature**→**recordedData()****temperature.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YTemperature target get_recordedData startTime endTime

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.

temperature→**get_reportFrequency()****YTemperature****temperature**→**reportFrequency()****temperature.get_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
cpp	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YTemperature target get_reportFrequency

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YTemperature target get_resolution

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.get_sensorType()**

Retourne le type de capteur de température utilisé par le module

js	function get_sensorType ()
nodejs	function get_sensorType ()
php	function get_sensorType ()
cpp	Y_SENSORTYPE_enum get_sensorType ()
m	-(Y_SENSORTYPE_enum) sensorType
pas	function get_sensorType (): Integer
vb	function get_sensorType () As Integer
cs	int get_sensorType ()
java	int get_sensorType ()
py	def get_sensorType ()
cmd	YTemperature target get_sensorType

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.get_unit()`

Retourne l'unité dans laquelle la température est exprimée.

js	function get_unit ()
nodejs	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()
cmd	YTemperature target get_unit

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.get_userdata()

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) <code>userData</code>
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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 isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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	<code>function load(msValidity)</code>
nodejs	<code>function load(msValidity)</code>
php	<code>function load(\$msValidity)</code>
cpp	<code>YRETCODE load(int msValidity)</code>
m	<code>-(YRETCODE) load : (int) msValidity</code>
pas	<code>function load(msValidity: integer): YRETCODE</code>
vb	<code>function load(ByVal msValidity As Integer) As YRETCODE</code>
cs	<code>YRETCODE load(int msValidity)</code>
java	<code>int load(long msValidity)</code>
py	<code>def load(msValidity)</code>

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()**YTemperature****temperature.loadCalibrationPoints()**

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	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()**

YTemperature

temperature.nextTemperature()

Continue l'énumération des capteurs de température commencée à l'aide de `yFirstTemperature()`.

<code>js</code>	<code>function nextTemperature()</code>
<code>nodejs</code>	<code>function nextTemperature()</code>
<code>php</code>	<code>function nextTemperature()</code>
<code>cpp</code>	<code>YTemperature * nextTemperature()</code>
<code>m</code>	<code>-(YTemperature*) nextTemperature</code>
<code>pas</code>	<code>function nextTemperature(): TYTemperature</code>
<code>vb</code>	<code>function nextTemperature() As YTemperature</code>
<code>cs</code>	<code>YTemperature nextTemperature()</code>
<code>java</code>	<code>YTemperature nextTemperature()</code>
<code>py</code>	<code>def nextTemperature()</code>

Retourne :

un pointeur sur un objet `YTemperature` accessible en ligne, ou `null` lorsque l'énumération est terminée.

temperature→registerTimedReportCallback()**YTemperature****temperature.registerTimedReportCallback()**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YTemperatureTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YTemperatureTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYTemperatureTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YTemperature****temperature.registerValueCallback()**

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 (YTemperatureValueCallback callback)
m	-(int) registerValueCallback : (YTemperatureValueCallback) callback
pas	function registerValueCallback (callback : TYTemperatureValueCallback): 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.

temperature→**set_highestValue()****YTemperature****temperature**→**setHighestValue()****temperature.set_highestValue()**

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	YTemperature 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.

temperature→**set_logFrequency()****YTemperature****temperature**→**setLogFrequency()****temperature.set_logFrequency()**

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()****YTemperature****temperature**→**setLogicalName()****temperature.set_logicalName()**

Modifie le nom logique du capteur de température.

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	YTemperature 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

js	function set_lowestValue (newval)
nodejs	function set_lowestValue (newval)
php	function set_lowestValue (\$newval)
cpp	int set_lowestValue (double newval)
m	-(int) setLowestValue : (double) newval
pas	function set_lowestValue (newval : double): integer
vb	function set_lowestValue (ByVal newval As Double) As Integer
cs	int set_lowestValue (double newval)
java	int set_lowestValue (double newval)
py	def set_lowestValue (newval)
cmd	YTemperature target set_lowestValue newval

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.set_reportFrequency()**

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	YTemperature 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.

temperature→**set_resolution()****YTemperature****temperature**→**setResolution()****temperature.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

js	function set_resolution (newval)
nodejs	function set_resolution (newval)
php	function set_resolution (\$newval)
cpp	int set_resolution (double newval)
m	-(int) setResolution : (double) newval
pas	function set_resolution (newval : double): integer
vb	function set_resolution (ByVal newval As Double) As Integer
cs	int set_resolution (double newval)
java	int set_resolution (double newval)
py	def set_resolution (newval)
cmd	YTemperature target set_resolution newval

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.set_sensorType()**

Change le type de senseur utilisé par le module.

js	function set_sensorType (newval)
nodejs	function set_sensorType (newval)
php	function set_sensorType (\$newval)
cpp	int set_sensorType (Y_SENSORTYPE_enum newval)
m	-(int) setSensorType : (Y_SENSORTYPE_enum) newval
pas	function set_sensorType (newval : Integer): integer
vb	function set_sensorType (ByVal newval As Integer) As Integer
cs	int set_sensorType (int newval)
java	int set_sensorType (int newval)
py	def set_sensorType (newval)
cmd	YTemperature target set_sensorType newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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

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
yFindTilt(func) Permet de retrouver un inclinomètre d'après un identifiant donné.
yFirstTilt() Commence l'énumération des inclinomètres accessibles par la librairie.
Méthodes des objets YTilt
tilt→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.
tilt→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance de l'inclinomètre au format TYPE (NAME) = SERIAL . FUNCTIONID.
tilt→get_advertisedValue() Retourne la valeur courante de l'inclinomètre (pas plus de 6 caractères).
tilt→get_currentRawValue() Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).
tilt→get_currentValue() Retourne la valeur actuelle de l'inclinaison.
tilt→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.
tilt→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.
tilt→get_friendlyName() Retourne un identifiant global de l'inclinomètre au format NOM_MODULE . NOM_FONCTION.
tilt→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
tilt→get_functionId() Retourne l'identifiant matériel de l'inclinomètre, sans référence au module.
tilt→get_hardwareId() 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()****YTilt.FindTilt()**

Permet de retrouver un inclinomètre d'après un identifiant donné.

js	function yFindTilt (func)
nodejs	function FindTilt (func)
php	function yFindTilt (\$func)
c++	YTilt* yFindTilt (const string& func)
m	YTilt* yFindTilt (NSString* func)
pas	function yFindTilt (func : string): YTilt
vb	function yFindTilt (ByVal func As String) As YTilt
cs	YTilt FindTilt (string func)
java	YTilt FindTilt (String func)
py	def FindTilt (func)

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 où 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()**YTilt****yFirstTilt()****YTilt.FirstTilt()**

Commence l'énumération des inclinomètres accessibles par la librairie.

js	function yFirstTilt ()
nodejs	function FirstTilt ()
php	function yFirstTilt ()
cpp	YTilt* yFirstTilt ()
m	YTilt* yFirstTilt ()
pas	function yFirstTilt (): TYTilt
vb	function yFirstTilt () As YTilt
cs	YTilt FirstTilt ()
java	YTilt FirstTilt ()
py	def FirstTilt ()

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()**YTilt****tilt.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 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.

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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'inclinomètre (ex:
 Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

tilt→get_advertisedValue()**YTilt****tilt→advertisedValue()****tilt.get_advertisedValue()**

Retourne la valeur courante de l'inclinomètre (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YTilt target get_advertisedValue

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()****tilt**→**currentRawValue()****tilt.get_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YTilt target get_currentRawValue

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.get_currentValue()**

Retourne la valeur actuelle de l'inclinaison.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YTilt target get_currentValue

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 get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****tilt.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'inclinomètre.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()`

Retourne un identifiant global de l'inclinomètre au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()`

Retourne l'identifiant matériel de l'inclinomètre, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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.get_hardwareId()**

Retourne l'identifiant matériel unique de l'inclinomètre au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
c++	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_highestValue()**

Retourne la valeur maximale observée pour l'inclinaison depuis le démarrage du module.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YTilt target get_highestValue

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.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.

js	function get_logFrequency ()
nodejs	function get_logFrequency ()
php	function get_logFrequency ()
cpp	string get_logFrequency ()
m	-(NSString*) logFrequency
pas	function get_logFrequency (): string
vb	function get_logFrequency () As String
cs	string get_logFrequency ()
java	String get_logFrequency ()
py	def get_logFrequency ()
cmd	YTilt target get_logFrequency

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()****YTilt****tilt**→**logicalName()**`tilt.get_logicalName()`

Retourne le nom logique de l'inclinomètre.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YTilt target get_logicalName

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.get_lowestValue()**

Retourne la valeur minimale observée pour l'inclinaison depuis le démarrage du module.

js	function get_lowestValue()
nodejs	function get_lowestValue()
php	function get_lowestValue()
cpp	double get_lowestValue()
m	-(double) lowestValue
pas	function get_lowestValue() : double
vb	function get_lowestValue() As Double
cs	double get_lowestValue()
java	double get_lowestValue()
py	def get_lowestValue()
cmd	YTilt target get_lowestValue

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.get_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
c++	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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-qu'el 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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YTilt target get_recordedData startTime endTime

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.get_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
cpp	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YTilt target get_reportFrequency

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YTilt target get_resolution

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.get_unit()**

Retourne l'unité dans laquelle l'inclinaison est exprimée.

js	function get_unit ()
nodejs	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()
cmd	YTilt target get_unit

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`.

<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.

tilt→isOnline()`tilt.isOnline()`**YTilt**

Vérifie si le module hébergeant l'inclinomètre est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 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.

tilt→loadCalibrationPoints()**tilt.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(&\$rawValues, &\$refValues)</code>
cpp	<code>int loadCalibrationPoints(vector<double>& rawValues, vector<double>& 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<double> rawValues, List<double> refValues)</code>
java	<code>int loadCalibrationPoints(ArrayList<Double> rawValues, ArrayList<Double> refValues)</code>
py	<code>def loadCalibrationPoints(rawValues, refValues)</code>
cmd	<code>YTilt 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.

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 nextTilt ()
nodejs	function nextTilt ()
php	function nextTilt ()
cpp	YTilt * nextTilt ()
m	-(YTilt*) nextTilt
pas	function nextTilt (): TYTilt
vb	function nextTilt () As YTilt
cs	YTilt nextTilt ()
java	YTilt nextTilt ()
py	def nextTilt ()

Retourne :

un pointeur sur un objet `YTilt` accessible en ligne, ou `null` lorsque l'énumération est terminée.

tilt→registerTimedReportCallback()**YTilt****tilt.registerTimedReportCallback()**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YTiltTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YTiltTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYTiltTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YTilt****tilt.registerValueCallback()**

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.set_highestValue()**

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	YTilt 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.

tilt→set_logFrequency()**tilt→setLogFrequency()**`tilt.set_logFrequency()`

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	YTilt 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.

tilt→**set_logicalName()****YTilt****tilt**→**setLogicalName()****tilt.set_logicalName()**

Modifie le nom logique de l'inclinomètre.

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	YTilt 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée.

js	function set_lowestValue (newval)
nodejs	function set_lowestValue (newval)
php	function set_lowestValue (\$newval)
cpp	int set_lowestValue (double newval)
m	-(int) setLowestValue : (double) newval
pas	function set_lowestValue (newval : double): integer
vb	function set_lowestValue (ByVal newval As Double) As Integer
cs	int set_lowestValue (double newval)
java	int set_lowestValue (double newval)
py	def set_lowestValue (newval)
cmd	YTilt target set_lowestValue newval

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.set_reportFrequency()**

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	YTilt 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.

tilt→**set_resolution()****tilt**→**setResolution()****tilt.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

js	function set_resolution (newval)
nodejs	function set_resolution (newval)
php	function set_resolution (\$newval)
cpp	int set_resolution (double newval)
m	-(int) setResolution : (double) newval
pas	function set_resolution (newval : double): integer
vb	function set_resolution (ByVal newval As Double) As Integer
cs	int set_resolution (double newval)
java	int set_resolution (double newval)
py	def set_resolution (newval)
cmd	YTilt target set_resolution newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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	<script type='text/javascript' src='yocto_voc.js'></script>
nodejs	var yoctolib = require('yoctolib'); var YVoc = yoctolib.YVoc;
php	require_once('yocto_voc.php');
c++	#include "yocto_voc.h"
m	#import "yocto_voc.h"
pas	uses yocto_voc;
vb	yocto_voc.vb
cs	yocto_voc.cs
java	import com.yoctopuce.YoctoAPI.YVoc;
py	from yocto_voc import *

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()

YVoc

yFindVoc()YVoc.FindVoc()

Permet de retrouver un capteur de Composés Organiques Volatils d'après un identifiant donné.

js	function yFindVoc (func)
nodejs	function FindVoc (func)
php	function yFindVoc (\$func)
cpp	YVoc* yFindVoc (const string& func)
m	YVoc* yFindVoc (NSString* func)
pas	function yFindVoc (func : string): TYVoc
vb	function yFindVoc (ByVal func As String) As YVoc
cs	YVoc FindVoc (string func)
java	YVoc FindVoc (String func)
py	def FindVoc (func)

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()**YVoc****yFirstVoc()****YVoc.FirstVoc()**

Commence l'énumération des capteurs de Composés Organiques Volatils accessibles par la librairie.

js	function yFirstVoc ()
nodejs	function FirstVoc ()
php	function yFirstVoc ()
cpp	YVoc* yFirstVoc ()
m	YVoc* yFirstVoc ()
pas	function yFirstVoc (): TYVoc
vb	function yFirstVoc () As YVoc
cs	YVoc FirstVoc ()
java	YVoc FirstVoc ()
py	def FirstVoc ()

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()**YVoc****voc.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 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.

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()**

Retourne la valeur courante du capteur de Composés Organiques Volatils (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YVoc target get_advertisedValue

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.get_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function get_currentRawValue ()
nodejs	function get_currentRawValue ()
php	function get_currentRawValue ()
cpp	double get_currentRawValue ()
m	-(double) currentRawValue
pas	function get_currentRawValue (): double
vb	function get_currentRawValue () As Double
cs	double get_currentRawValue ()
java	double get_currentRawValue ()
py	def get_currentRawValue ()
cmd	YVoc target get_currentRawValue

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.get_currentValue()**

Retourne la mesure actuelle du taux de VOC estimé.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YVoc target get_currentValue

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.get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****voc.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de Composés Organiques Volatils.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global du capteur de Composés Organiques Volatils au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName()
nodejs	function get_friendlyName()
php	function get_friendlyName()
cpp	string get_friendlyName()
m	-(NSString*) friendlyName
cs	string get_friendlyName()
java	String get_friendlyName()
py	def get_friendlyName()

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.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor()
nodejs	function get_functionDescriptor()
php	function get_functionDescriptor()
cpp	YFUN_DESCR get_functionDescriptor()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor() : YFUN_DESCR
vb	function get_functionDescriptor() As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor()
java	String get_functionDescriptor()
py	def get_functionDescriptor()

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.get_functionId()**

Retourne l'identifiant matériel du capteur de Composés Organiques Volatils, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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 get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_highestValue()**

Retourne la valeur maximale observée pour le taux de VOC estimé.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YVoc target get_highestValue

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.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.

js	function get_logFrequency ()
nodejs	function get_logFrequency ()
php	function get_logFrequency ()
cpp	string get_logFrequency ()
m	-(NSString*) logFrequency
pas	function get_logFrequency (): string
vb	function get_logFrequency () As String
cs	string get_logFrequency ()
java	String get_logFrequency ()
py	def get_logFrequency ()
cmd	YVoc target get_logFrequency

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.get_logicalName()**

Retourne le nom logique du capteur de Composés Organiques Volatils.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YVoc target get_logicalName

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.get_lowestValue()**

Retourne la valeur minimale observée pour le taux de VOC estimé.

js	function get_lowestValue()
nodejs	function get_lowestValue()
php	function get_lowestValue()
cpp	double get_lowestValue()
m	-(double) lowestValue
pas	function get_lowestValue() : double
vb	function get_lowestValue() As Double
cs	double get_lowestValue()
java	double get_lowestValue()
py	def get_lowestValue()
cmd	YVoc target get_lowestValue

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.get_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	<code>YModule *</code> get_module ()
m	-(<code>YModule*</code>) module
pas	function get_module (): <code>TYModule</code>
vb	function get_module () As <code>YModule</code>
cs	<code>YModule</code> get_module ()
java	<code>YModule</code> get_module ()
py	def get_module ()

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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
cpp	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YVoc target get_recordedData startTime endTime

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.get_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
cpp	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YVoc target get_reportFrequency

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YVoc target get_resolution

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()****YVoc****voc**→**unit()****voc.get_unit()**

Retourne l'unité dans laquelle le taux de VOC estimé est exprimée.

js	function get_unit ()
nodejs	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()
cmd	YVoc target get_unit

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.get_userdata()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
c++	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.

voc→**isOnline()****voc.isOnline()****YVoc**

Vérifie si le module hébergeant le capteur de Composés Organiques Volatils est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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-quel 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 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.

voc→loadCalibrationPoints()**YVoc****voc.loadCalibrationPoints()**

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 nextVoc ()
nodejs	function nextVoc ()
php	function nextVoc ()
cpp	YVoc * nextVoc ()
m	-(YVoc*) nextVoc
pas	function nextVoc (): TYVoc
vb	function nextVoc () As YVoc
cs	YVoc nextVoc ()
java	YVoc nextVoc ()
py	def nextVoc ()

Retourne :

un pointeur sur un objet `YVoc` accessible en ligne, ou `null` lorsque l'énumération est terminée.

voc→registerTimedReportCallback()**YVoc****voc.registerTimedReportCallback()**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YVocTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YVocTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYVocTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YVoc****voc.registerValueCallback()**

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)
c++	int registerValueCallback (YVocValueCallback callback)
m	-(int) registerValueCallback : (YVocValueCallback) callback
pas	function registerValueCallback (callback : TYVocValueCallback): 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'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

voc→**set_highestValue()****YVoc****voc**→**setHighestValue()****voc.set_highestValue()**

Modifie la mémoire de valeur maximale observée pour le taux de VOC estimé.

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	YVoc target set_highestValue newval

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.set_logFrequency()`

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	YVoc 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.

voc→**set_logicalName()****YVoc****voc**→**setLogicalName()****voc.set_logicalName()**

Modifie le nom logique du capteur de Composés Organiques Volatils.

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	YVoc 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour le taux de VOC estimé.

js	function set_lowestValue (newval)
nodejs	function set_lowestValue (newval)
php	function set_lowestValue (\$newval)
cpp	int set_lowestValue (double newval)
m	-(int) setLowestValue : (double) newval
pas	function set_lowestValue (newval : double): integer
vb	function set_lowestValue (ByVal newval As Double) As Integer
cs	int set_lowestValue (double newval)
java	int set_lowestValue (double newval)
py	def set_lowestValue (newval)
cmd	YVoc target set_lowestValue newval

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.set_reportFrequency()**

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	YVoc 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.

voc→**set_resolution()****YVoc****voc**→**setResolution()****voc.set_resolution()**

Modifie la résolution des valeurs physique mesurées.

js	function set_resolution (newval)
nodejs	function set_resolution (newval)
php	function set_resolution (\$newval)
cpp	int set_resolution (double newval)
m	-(int) setResolution : (double) newval
pas	function set_resolution (newval : double): integer
vb	function set_resolution (ByVal newval As Double) As Integer
cs	int set_resolution (double newval)
java	int set_resolution (double newval)
py	def set_resolution (newval)
cmd	YVoc target set_resolution newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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

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	<code><script type='text/javascript' src='yocto_voltage.js'></script></code>
nodejs	<code>var yoctolib = require('yoctolib');</code> <code>var YVoltage = yoctolib.YVoltage;</code>
php	<code>require_once('yocto_voltage.php');</code>
c++	<code>#include "yocto_voltage.h"</code>
m	<code>#import "yocto_voltage.h"</code>
pas	<code>uses yocto_voltage;</code>
vb	<code>yocto_voltage.vb</code>
cs	<code>yocto_voltage.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YVoltage;</code>
py	<code>from yocto_voltage import *</code>

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()**YVoltage****yFindVoltage()** `YVoltage.FindVoltage()`

Permet de retrouver un capteur de tension d'après un identifiant donné.

<code>js</code>	<code>function yFindVoltage(func)</code>
<code>nodejs</code>	<code>function FindVoltage(func)</code>
<code>php</code>	<code>function yFindVoltage(\$func)</code>
<code>cpp</code>	<code>YVoltage* yFindVoltage(const string& func)</code>
<code>m</code>	<code>YVoltage* yFindVoltage(NSString* func)</code>
<code>pas</code>	<code>function yFindVoltage(func: string): TYVoltage</code>
<code>vb</code>	<code>function yFindVoltage(ByVal func As String) As YVoltage</code>
<code>cs</code>	<code>YVoltage FindVoltage(string func)</code>
<code>java</code>	<code>YVoltage FindVoltage(String func)</code>
<code>py</code>	<code>def FindVoltage(func)</code>

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()**YVoltage****yFirstVoltage()****YVoltage.FirstVoltage()**

Commence l'énumération des capteurs de tension accessibles par la librairie.

js	function yFirstVoltage ()
nodejs	function FirstVoltage ()
php	function yFirstVoltage ()
cpp	YVoltage* yFirstVoltage ()
m	YVoltage* yFirstVoltage ()
pas	function yFirstVoltage (): TYVoltage
vb	function yFirstVoltage () As YVoltage
cs	YVoltage FirstVoltage ()
java	YVoltage FirstVoltage ()
py	def FirstVoltage ()

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()**YVoltage****voltage.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 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.

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()**

Retourne la valeur courante du capteur de tension (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YVoltage target get_advertisedValue

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()****YVoltage****voltage**→**currentRawValue()****voltage.get_currentRawValue()**

Retourne la valeur brute retournée par le capteur (sans arrondi ni calibration).

js	function get_currentRawValue()
nodejs	function get_currentRawValue()
php	function get_currentRawValue()
cpp	double get_currentRawValue()
m	-(double) currentRawValue
pas	function get_currentRawValue() : double
vb	function get_currentRawValue() As Double
cs	double get_currentRawValue()
java	double get_currentRawValue()
py	def get_currentRawValue()
cmd	YVoltage target get_currentRawValue

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()**
voltage→**currentValue()**
voltage.get_currentValue()

YVoltage

Retourne la valeur instantanée de la tension.

js	function get_currentValue ()
nodejs	function get_currentValue ()
php	function get_currentValue ()
cpp	double get_currentValue ()
m	-(double) currentValue
pas	function get_currentValue (): double
vb	function get_currentValue () As Double
cs	double get_currentValue ()
java	double get_currentValue ()
py	def get_currentValue ()
cmd	YVoltage target get_currentValue

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.get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****YVoltage****voltage**→**errorType()****voltage.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du capteur de tension.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global du capteur de tension au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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()****YVoltage****voltage**→**functionDescriptor()****voltage.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()**

Retourne l'identifiant matériel du capteur de tension, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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 get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_highestValue()**

Retourne la valeur maximale observée pour la tension.

js	function get_highestValue ()
nodejs	function get_highestValue ()
php	function get_highestValue ()
cpp	double get_highestValue ()
m	-(double) highestValue
pas	function get_highestValue (): double
vb	function get_highestValue () As Double
cs	double get_highestValue ()
java	double get_highestValue ()
py	def get_highestValue ()
cmd	YVoltage target get_highestValue

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**.**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.

js	function get_logFrequency ()
nodejs	function get_logFrequency ()
php	function get_logFrequency ()
cpp	string get_logFrequency ()
m	-(NSString*) logFrequency
pas	function get_logFrequency (): string
vb	function get_logFrequency () As String
cs	string get_logFrequency ()
java	String get_logFrequency ()
py	def get_logFrequency ()
cmd	YVoltage target get_logFrequency

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.get_logicalName()**

Retourne le nom logique du capteur de tension.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YVoltage target get_logicalName

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.get_lowestValue()**

Retourne la valeur minimale observée pour la tension.

js	function get_lowestValue ()
nodejs	function get_lowestValue ()
php	function get_lowestValue ()
cpp	double get_lowestValue ()
m	-(double) lowestValue
pas	function get_lowestValue (): double
vb	function get_lowestValue () As Double
cs	double get_lowestValue ()
java	double get_lowestValue ()
py	def get_lowestValue ()
cmd	YVoltage target get_lowestValue

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.get_module()`

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
c++	<code>YModule *</code> get_module ()
m	<code>-(YModule*)</code> module
pas	function get_module (): <code>TYModule</code>
vb	function get_module () As <code>YModule</code>
cs	<code>YModule</code> get_module ()
java	<code>YModule</code> get_module ()
py	def get_module ()

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()****YVoltage****voltage**→**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-qu'el 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.get_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 get_recordedData (startTime , endTime)
nodejs	function get_recordedData (startTime , endTime)
php	function get_recordedData (\$startTime , \$endTime)
c++	YDataSet get_recordedData (s64 startTime , s64 endTime)
m	-(YDataSet*) recordedData : (s64) startTime : (s64) endTime
pas	function get_recordedData (startTime : int64, endTime : int64): TYDataSet
vb	function get_recordedData () As YDataSet
cs	YDataSet get_recordedData (long startTime , long endTime)
java	YDataSet get_recordedData (long startTime , long endTime)
py	def get_recordedData (startTime , endTime)
cmd	YVoltage target get_recordedData startTime endTime

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()****YVoltage****voltage**→**reportFrequency()****voltage.get_reportFrequency()**

Retourne la fréquence de notification périodique des valeurs mesurées, ou "OFF" si les notifications périodiques sont désactivées pour cette fonction.

js	function get_reportFrequency ()
nodejs	function get_reportFrequency ()
php	function get_reportFrequency ()
c++	string get_reportFrequency ()
m	-(NSString*) reportFrequency
pas	function get_reportFrequency (): string
vb	function get_reportFrequency () As String
cs	string get_reportFrequency ()
java	String get_reportFrequency ()
py	def get_reportFrequency ()
cmd	YVoltage target get_reportFrequency

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.get_resolution()**

Retourne la résolution des valeurs mesurées.

js	function get_resolution ()
nodejs	function get_resolution ()
php	function get_resolution ()
cpp	double get_resolution ()
m	-(double) resolution
pas	function get_resolution (): double
vb	function get_resolution () As Double
cs	double get_resolution ()
java	double get_resolution ()
py	def get_resolution ()
cmd	YVoltage target get_resolution

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.get_unit()**

Retourne l'unité dans laquelle la tension est exprimée.

js	function get_unit ()
nodejs	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()
cmd	YVoltage target get_unit

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.get_userdata()`

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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 isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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()**YVoltage****voltage.loadCalibrationPoints()**

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 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 nextVoltage ()
nodejs	function nextVoltage ()
php	function nextVoltage ()
cpp	YVoltage * nextVoltage ()
m	-(YVoltage*) nextVoltage
pas	function nextVoltage (): TYVoltage
vb	function nextVoltage () As YVoltage
cs	YVoltage nextVoltage ()
java	YVoltage nextVoltage ()
py	def nextVoltage ()

Retourne :

un pointeur sur un objet `YVoltage` accessible en ligne, ou `null` lorsque l'énumération est terminée.

voltage→registerTimedReportCallback()**YVoltage****voltage.registerTimedReportCallback()**

Enregistre la fonction de callback qui est appelée à chaque notification périodique.

js	function registerTimedReportCallback (callback)
nodejs	function registerTimedReportCallback (callback)
php	function registerTimedReportCallback (\$callback)
cpp	int registerTimedReportCallback (YVoltageTimedReportCallback callback)
m	-(int) registerTimedReportCallback : (YVoltageTimedReportCallback) callback
pas	function registerTimedReportCallback (callback : TYVoltageTimedReportCallback): LongInt
vb	function registerTimedReportCallback () As Integer
cs	int registerTimedReportCallback (TimedReportCallback callback)
java	int registerTimedReportCallback (TimedReportCallback callback)
py	def registerTimedReportCallback (callback)

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()**YVoltage****voltage.registerValueCallback()**

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 (YVoltageValueCallback callback)
m	-(int) registerValueCallback : (YVoltageValueCallback) callback
pas	function registerValueCallback (callback : TYVoltageValueCallback): 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.

voltage→**set_highestValue()**
voltage→**setHighestValue()**
voltage.set_highestValue()

YVoltage

Modifie la mémoire de valeur maximale observée pour la tension.

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	YVoltage target set_highestValue newval

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.set_logFrequency()**

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)
c++	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	YVoltage 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.

voltage→**set_logicalName()****YVoltage****voltage**→**setLogicalName()****voltage.set_logicalName()**

Modifie le nom logique du capteur de tension.

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	YVoltage 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 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.set_lowestValue()**

Modifie la mémoire de valeur minimale observée pour la tension.

js	function set_lowestValue (newval)
nodejs	function set_lowestValue (newval)
php	function set_lowestValue (\$newval)
cpp	int set_lowestValue (double newval)
m	-(int) setLowestValue : (double) newval
pas	function set_lowestValue (newval : double): integer
vb	function set_lowestValue (ByVal newval As Double) As Integer
cs	int set_lowestValue (double newval)
java	int set_lowestValue (double newval)
py	def set_lowestValue (newval)
cmd	YVoltage target set_lowestValue newval

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()****YVoltage****voltage**→**setReportFrequency()****voltage.set_reportFrequency()**

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	YVoltage 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.

voltage→**set_resolution()****YVoltage****voltage**→**setResolution()****voltage.set_resolution()**

Modifie la résolution des valeurs mesurées.

js	function set_resolution (newval)
nodejs	function set_resolution (newval)
php	function set_resolution (\$newval)
c++	int set_resolution (double newval)
m	-(int) setResolution : (double) newval
pas	function set_resolution (newval : double): integer
vb	function set_resolution (ByVal newval As Double) As Integer
cs	int set_resolution (double newval)
java	int set_resolution (double newval)
py	def set_resolution (newval)
cmd	YVoltage target set_resolution newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
c++	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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');
c++	#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
yFindVSource(func) Permet de retrouver une source de tension d'après un identifiant donné.
yFirstVSource() Commence l'énumération des sources de tension accessibles par la librairie.
Méthodes des objets YVSource
vsource→describe() Retourne un court texte décrivant la fonction au format TYPE (NAME) = SERIAL . FUNCTIONID.
vsource→get_advertisedValue() Retourne la valeur courante de la source de tension (pas plus de 6 caractères).
vsource→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.
vsource→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.
vsource→get_extPowerFailure() Rend TRUE si le voltage de l'alimentation externe est trop bas.
vsource→get_failure() Indique si le module est en condition d'erreur.
vsource→get_friendlyName() Retourne un identifiant global de la fonction au format NOM_MODULE . NOM_FONCTION.
vsource→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
vsource→get_functionId() Retourne l'identifiant matériel de la fonction, sans référence au module.
vsource→get_hardwareId() Retourne l'identifiant matériel unique de la fonction au format SERIAL . FUNCTIONID.
vsource→get_logicalName() Retourne le nom logique de la source de tension.
vsource→get_module() Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
vsource→get_module_async(callback, context)

3. Reference

	Retourne l'objet <code>YModule</code> correspondant au module Yoctopuce qui héberge la fonction.
<code>vsource→get_overCurrent()</code>	Rend TRUE si l'appareil connecté à la sortie du module consomme trop de courant.
<code>vsource→get_overHeat()</code>	Rend TRUE si le module est en surchauffe.
<code>vsource→get_overLoad()</code>	Rend TRUE si le module n'est pas capable de tenir la tension de sortie demandée.
<code>vsource→get_regulationFailure()</code>	Rend TRUE si le voltage de sortie de trop élevé par report à la tension demandée demandée.
<code>vsource→get_unit()</code>	Retourne l'unité dans laquelle la tension est exprimée.
<code>vsource→get_userData()</code>	Retourne le contenu de l'attribut <code>userData</code> , précédemment stocké à l'aide de la méthode <code>set_userData</code> .
<code>vsource→get_voltage()</code>	Retourne la valeur de la commande de tension de sortie en mV
<code>vsource→isOnline()</code>	Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.
<code>vsource→isOnline_async(callback, context)</code>	Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.
<code>vsource→load(msValidity)</code>	Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.
<code>vsource→load_async(msValidity, callback, context)</code>	Met en cache les valeurs courantes de la fonction, avec une durée de validité spécifiée.
<code>vsource→nextVSource()</code>	Continue l'énumération des sources de tension commencée à l'aide de <code>yFirstVSource()</code> .
<code>vsource→pulse(voltage, ms_duration)</code>	Active la sortie à une tension donnée, et pour durée spécifiée, puis revient ensuite spontanément à zéro volt.
<code>vsource→registerValueCallback(callback)</code>	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
<code>vsource→set_logicalName(newval)</code>	Modifie le nom logique de la source de tension.
<code>vsource→set_userData(data)</code>	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> .
<code>vsource→set_voltage(newval)</code>	Règle la tension de sortie du module (en milliVolts).
<code>vsource→voltageMove(target, ms_duration)</code>	Déclenche une variation constante de la sortie vers une valeur donnée.
<code>vsource→wait_async(callback, context)</code>	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()****yVSource.FindVSource()**

Permet de retrouver une source de tension d'après un identifiant donné.

js	function yFindVSource (func)
php	function yFindVSource (\$func)
cpp	YVSource* yFindVSource (const string& func)
m	YVSource* yFindVSource (NSString* func)
pas	function yFindVSource (func : string): TYVSource
vb	function yFindVSource (ByVal func As String) As YVSource
cs	YVSource FindVSource (string func)
java	YVSource FindVSource (String func)
py	def FindVSource (func)

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()****YVSource.FirstVSource()**

Commence l'énumération des sources de tension accessibles par la librairie.

js	function yFirstVSource ()
php	function yFirstVSource ()
cpp	YVSource* yFirstVSource ()
m	YVSource* yFirstVSource ()
pas	function yFirstVSource (): TYVSource
vb	function yFirstVSource () As YVSource
cs	YVSource FirstVSource ()
java	YVSource FirstVSource ()
py	def FirstVSource ()

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 describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()

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.get_advertisedValue()**

Retourne la valeur courante de la source de tension (pas plus de 6 caractères).

js	function get_advertisedValue()
php	function get_advertisedValue()
cpp	string get_advertisedValue()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue() : string
vb	function get_advertisedValue() As String
cs	string get_advertisedValue()
java	String get_advertisedValue()
py	def get_advertisedValue()
cmd	YVSource target get_advertisedValue

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 get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****vsSource.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de la fonction.

js	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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

vsource→**get_extPowerFailure()****YVSource****vsource**→**extPowerFailure()****vsource.get_extPowerFailure()**

Rend TRUE si le voltage de l'alimentation externe est trop bas.

js	function get_extPowerFailure ()
php	function get_extPowerFailure ()
cpp	Y_EXTPOWERFAILURE_enum get_extPowerFailure ()
m	-(Y_EXTPOWERFAILURE_enum) extPowerFailure
pas	function get_extPowerFailure (): Integer
vb	function get_extPowerFailure () As Integer
cs	int get_extPowerFailure ()
java	int get_extPowerFailure ()
py	def get_extPowerFailure ()
cmd	YVSource target get_extPowerFailure

Retourne :

soit Y_EXTPOWERFAILURE_FALSE, soit Y_EXTPOWERFAILURE_TRUE

En cas d'erreur, déclenche une exception ou retourne Y_EXTPOWERFAILURE_INVALID.

vsourc→**get_failure()****YVSource****vsourc**→**failure()****vsourc.get_failure()**

Indique si le module est en condition d'erreur.

js	function get_failure ()
php	function get_failure ()
cpp	Y_FAILURE_enum get_failure ()
m	-(Y_FAILURE_enum) failure
pas	function get_failure (): Integer
vb	function get_failure () As Integer
cs	int get_failure ()
java	int get_failure ()
py	def get_failure ()
cmd	YVSource target get_failure

Il possible de savoir de quelle erreur il s'agit en testant get_overheat, get_overcurrent etc... Lorsqu'un condition d'erreur est rencontrée, la tension de sortie est mise à zéro est ne peut pas être changée tant la fonction reset() n'aura pas appelée.

Retourne :

soit Y_FAILURE_FALSE, soit Y_FAILURE_TRUE

En cas d'erreur, déclenche une exception ou retourne Y_FAILURE_INVALID.

vsources→**get_friendlyName()****YVSource****vsources**→**friendlyName()****vsources.get_friendlyName()**

Retourne un identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName()
php	function get_friendlyName()
cpp	virtual string get_friendlyName()
m	-(NSString*) friendlyName
cs	override string get_friendlyName()
java	String get_friendlyName()

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.get_vsSourceDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor()
php	function get_functionDescriptor()
cpp	YFUN_DESCR get_functionDescriptor()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor() : YFUN_DESCR
vb	function get_functionDescriptor() As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor()
java	String get_functionDescriptor()
py	def get_functionDescriptor()

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.get_vsourceId()`

Retourne l'identifiant matériel de la fonction, sans référence au module.

js	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()

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`.

vsSource→**get_hardwareId()****YVSource****vsSource**→**hardwareId()****vsSource.hardwareId()**

Retourne l'identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()

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()**
vsource→**logicalName()**
vsource.get_logicalName()

YVSource

Retourne le nom logique de la source de tension.

js	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YVSource target get_logicalName

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.get_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
php	function get_module ()
cpp	<code>YModule *</code> get_module ()
m	<code>-(YModule*)</code> module
pas	function get_module (): <code>TYModule</code>
vb	function get_module () As <code>YModule</code>
cs	<code>YModule</code> get_module ()
java	<code>YModule</code> get_module ()
py	def get_module ()

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.

vsource→**get_overCurrent()****vsource**→**overCurrent()****vsource.get_overCurrent()**

Rend TRUE si l'appareil connecté à la sortie du module consomme trop de courant.

js	function get_overCurrent ()
php	function get_overCurrent ()
cpp	Y_OVERCURRENT_enum get_overCurrent ()
m	-(Y_OVERCURRENT_enum) overCurrent
pas	function get_overCurrent (): Integer
vb	function get_overCurrent () As Integer
cs	int get_overCurrent ()
java	int get_overCurrent ()
py	def get_overCurrent ()
cmd	YVSource target get_overCurrent

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.get_overHeat()**

Rend TRUE si le module est en surchauffe.

js	function get_overHeat()
php	function get_overHeat()
cpp	Y_OVERHEAT_enum get_overHeat()
m	-(Y_OVERHEAT_enum) overHeat
pas	function get_overHeat() : Integer
vb	function get_overHeat() As Integer
cs	int get_overHeat()
java	int get_overHeat()
py	def get_overHeat()
cmd	YVSource target get_overHeat

Retourne :

soit Y_OVERHEAT_FALSE, soit Y_OVERHEAT_TRUE

En cas d'erreur, déclenche une exception ou retourne Y_OVERHEAT_INVALID.

vsources→**get_overLoad()****YVSource****vsources**→**overLoad()****vsources.get_overLoad()**

Rend TRUE si le module n'est pas capable de tenir la tension de sortie demandée.

js	function get_overLoad ()
php	function get_overLoad ()
cpp	Y_OVERLOAD_enum get_overLoad ()
m	-(Y_OVERLOAD_enum) overLoad
pas	function get_overLoad (): Integer
vb	function get_overLoad () As Integer
cs	int get_overLoad ()
java	int get_overLoad ()
py	def get_overLoad ()
cmd	YVSource target get_overLoad

Retourne :

soit Y_OVERLOAD_FALSE, soit Y_OVERLOAD_TRUE

En cas d'erreur, déclenche une exception ou retourne Y_OVERLOAD_INVALID.

vsource→**get_regulationFailure()****YVSource****vsource**→**regulationFailure()****vsource.get_regulationFailure()**

Rend TRUE si le voltage de sortie de trop élevé par report à la tension demandée demandée.

js	function get_regulationFailure ()
php	function get_regulationFailure ()
cpp	Y_REGULATIONFAILURE_enum get_regulationFailure ()
m	-(Y_REGULATIONFAILURE_enum) regulationFailure
pas	function get_regulationFailure (): Integer
vb	function get_regulationFailure () As Integer
cs	int get_regulationFailure ()
java	int get_regulationFailure ()
py	def get_regulationFailure ()
cmd	YVSource target get_regulationFailure

Retourne :

soit Y_REGULATIONFAILURE_FALSE, soit Y_REGULATIONFAILURE_TRUE

En cas d'erreur, déclenche une exception ou retourne Y_REGULATIONFAILURE_INVALID.

vsource→**get_unit()****YVSource****vsource**→**unit()**`vsource.get_unit()`

Retourne l'unité dans laquelle la tension est exprimée.

js	function get_unit ()
php	function get_unit ()
cpp	string get_unit ()
m	-(NSString*) unit
pas	function get_unit (): string
vb	function get_unit () As String
cs	string get_unit ()
java	String get_unit ()
py	def get_unit ()
cmd	YVSource target get_unit

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.get_userData()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.

js	function get_userData ()
php	function get_userData ()
cpp	void * get_userData ()
m	-(void*) userData
pas	function get_userData (): Tobject
vb	function get_userData () As Object
cs	object get_userData ()
java	Object get_userData ()
py	def get_userData ()

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.get_voltage()**

Retourne la valeur de la commande de tension de sortie en mV

js	function get_voltage ()
php	function get_voltage ()
cpp	int get_voltage ()
m	-(int) voltage
pas	function get_voltage (): LongInt
vb	function get_voltage () As Integer
cs	int get_voltage ()
java	int get_voltage ()
py	def get_voltage ()

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 isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 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.

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 nextVSource ()
php	function nextVSource ()
cpp	YVSource * nextVSource ()
m	-(YVSource*) nextVSource
pas	function nextVSource (): TYVSource
vb	function nextVSource () As YVSource
cs	YVSource nextVSource ()
java	YVSource nextVSource ()
py	def nextVSource ()

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 pulse (voltage , ms_duration)
php	function pulse (\$voltage , \$ms_duration)
cpp	int pulse (int voltage , int ms_duration)
m	-(int) pulse : (int) voltage : (int) ms_duration
pas	function pulse (voltage : integer, ms_duration : integer): integer
vb	function pulse (ByVal voltage As Integer, ByVal ms_duration As Integer) As Integer
cs	int pulse (int voltage , int ms_duration)
java	int pulse (int voltage , int ms_duration)
py	def pulse (voltage , ms_duration)
cmd	YVSource target pulse voltage ms_duration

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.

vsources.registerValueCallback()**YVSource****vsources.registerValueCallback()**

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function registerValueCallback (callback)
php	function registerValueCallback (\$callback)
cpp	void registerValueCallback (YDisplayUpdateCallback callback)
pas	procedure registerValueCallback (callback : TGenericUpdateCallback)
vb	procedure registerValueCallback (ByVal callback As GenericUpdateCallback)
cs	void registerValueCallback (UpdateCallback callback)
java	void registerValueCallback (UpdateCallback callback)
py	def registerValueCallback (callback)
m	-(void) registerValueCallback : (YFunctionUpdateCallback) 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'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

vsourceset_logicalName()**YVSource****vsourcesetLogicalName()****vsourceset_logicalName()**

Modifie le nom logique de la source de tension.

js	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	YVSource 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 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.set_userdata()**

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 set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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()**`vsourceset_voltage()`

Règle la tension de sortie du module (en milliVolts).

js	function set_voltage (newval)
php	function set_voltage (\$newval)
cpp	int set_voltage (int newval)
m	-(int) setVoltage : (int) newval
pas	function set_voltage (newval : LongInt): integer
vb	function set_voltage (ByVal newval As Integer) As Integer
cs	int set_voltage (int newval)
java	int set_voltage (int newval)
py	def set_voltage (newval)
cmd	YVSource target set_voltage newval

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 voltageMove (target , ms_duration)
php	function voltageMove (\$target , \$ms_duration)
cpp	int voltageMove (int target , int ms_duration)
m	-(int) voltageMove : (int) target : (int) ms_duration
pas	function voltageMove (target : integer, ms_duration : integer): integer
vb	function voltageMove (ByVal target As Integer, ByVal ms_duration As Integer) As Integer
cs	int voltageMove (int target , int ms_duration)
java	int voltageMove (int target , int ms_duration)
py	def voltageMove (target , ms_duration)
cmd	YVSource target voltageMove target ms_duration

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.

vssource→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()

YWakeUpMonitor

YWakeUpMonitor.FindWakeUpMonitor()

Permet de retrouver un moniteur d'après un identifiant donné.

js	function yFindWakeUpMonitor (func)
nodejs	function FindWakeUpMonitor (func)
php	function yFindWakeUpMonitor (\$func)
c++	YWakeUpMonitor* yFindWakeUpMonitor (const string& func)
m	YWakeUpMonitor* yFindWakeUpMonitor (NSString* func)
pas	function yFindWakeUpMonitor (func : string): TYWakeUpMonitor
vb	function yFindWakeUpMonitor (ByVal func As String) As YWakeUpMonitor
cs	YWakeUpMonitor FindWakeUpMonitor (string func)
java	YWakeUpMonitor FindWakeUpMonitor (String func)
py	def FindWakeUpMonitor (func)

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()**YWakeUpMonitor****yFirstWakeUpMonitor()****YWakeupMonitor.FirstWakeUpMonitor()**

Commence l'énumération des Moniteurs accessibles par la librairie.

js	function yFirstWakeUpMonitor ()
nodejs	function FirstWakeUpMonitor ()
php	function yFirstWakeUpMonitor ()
cpp	YWakeupMonitor* yFirstWakeUpMonitor ()
m	YWakeupMonitor* yFirstWakeUpMonitor ()
pas	function yFirstWakeUpMonitor (): TYWakeUpMonitor
vb	function yFirstWakeUpMonitor () As YWakeUpMonitor
cs	YWakeupMonitor FirstWakeUpMonitor ()
java	YWakeupMonitor FirstWakeUpMonitor ()
py	def FirstWakeUpMonitor ()

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()**YWakeUpMonitor****wakeupmonitor.describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur au format
 TYPE(NAME)=SERIAL.FUNCTIONID.

js	function describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()**

Retourne la valeur courante du moniteur (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YWakeUpMonitor target get_advertisedValue

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.get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()**YWakeUpMonitor****wakeupmonitor→errorType()****wakeupmonitor.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global du moniteur au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName()
nodejs	function get_friendlyName()
php	function get_friendlyName()
cpp	string get_friendlyName()
m	-(NSString*) friendlyName
cs	string get_friendlyName()
java	String get_friendlyName()
py	def get_friendlyName()

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.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()

Retourne l'identifiant matériel du moniteur, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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**.**get_hardwareId()**

Retourne l'identifiant matériel unique du moniteur au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_logicalName()**

Retourne le nom logique du moniteur.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YWakeUpMonitor target get_logicalName

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.get_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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.get_nextWakeUp()**

Retourne la prochaine date/heure de réveil agendée (format UNIX)

js	function get_nextWakeUp ()
nodejs	function get_nextWakeUp ()
php	function get_nextWakeUp ()
cpp	s64 get_nextWakeUp ()
m	-(s64) nextWakeUp
pas	function get_nextWakeUp (): int64
vb	function get_nextWakeUp () As Long
cs	long get_nextWakeUp ()
java	long get_nextWakeUp ()
py	def get_nextWakeUp ()

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.get_powerDuration()

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

js	function get_powerDuration ()
nodejs	function get_powerDuration ()
php	function get_powerDuration ()
cpp	int get_powerDuration ()
m	-(int) powerDuration
pas	function get_powerDuration (): LongInt
vb	function get_powerDuration () As Integer
cs	int get_powerDuration ()
java	int get_powerDuration ()
py	def get_powerDuration ()
cmd	YWakeUpMonitor target get_powerDuration

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.get_sleepCountdown()**

Retourne le temps avant le prochain sommeil.

js	function get_sleepCountdown ()
nodejs	function get_sleepCountdown ()
php	function get_sleepCountdown ()
cpp	int get_sleepCountdown ()
m	-(int) sleepCountdown
pas	function get_sleepCountdown (): LongInt
vb	function get_sleepCountdown () As Integer
cs	int get_sleepCountdown ()
java	int get_sleepCountdown ()
py	def get_sleepCountdown ()
cmd	YWakeUpMonitor target get_sleepCountdown

Retourne :

un entier représentant le temps avant le prochain sommeil

En cas d'erreur, déclenche une exception ou retourne Y_SLEEPDOWNDOWN_INVALID.

wakeupmonitor→**get_userdata()****YWakeUpMonitor****wakeupmonitor**→**userData()****wakeupmonitor.get_userdata()**

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) <code>userData</code>
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.get_wakeUpReason()**

Renvoie la raison du dernier réveil.

js	function get_wakeUpReason ()
nodejs	function get_wakeUpReason ()
php	function get_wakeUpReason ()
cpp	Y_WAKEUPREASON_enum get_wakeUpReason ()
m	-(Y_WAKEUPREASON_enum) wakeUpReason
pas	function get_wakeUpReason (): Integer
vb	function get_wakeUpReason () As Integer
cs	int get_wakeUpReason ()
java	int get_wakeUpReason ()
py	def get_wakeUpReason ()
cmd	YWakeUpMonitor target get_wakeUpReason

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.get_wakeUpState()**

Revoie l'état actuel du moniteur

js	function get_wakeUpState()
nodejs	function get_wakeUpState()
php	function get_wakeUpState()
cpp	Y_WAKEUPSTATE_enum get_wakeUpState()
m	-(Y_WAKEUPSTATE_enum) wakeUpState
pas	function get_wakeUpState() : Integer
vb	function get_wakeUpState() As Integer
cs	int get_wakeUpState()
java	int get_wakeUpState()
py	def get_wakeUpState()

Retourne :

soit Y_WAKEUPSTATE_SLEEPING, soit Y_WAKEUPSTATE_AWAKE

En cas d'erreur, déclenche une exception ou retourne Y_WAKEUPSTATE_INVALID.

wakeupmonitor→isOnline()**YWakeUpMonitor****wakeupmonitor.isOnline()**

Vérifie si le module hébergeant le moniteur est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 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.

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-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→**nextWakeUpMonitor()****YWakeUpMonitor****wakeupmonitor.nextWakeUpMonitor()**

Continue l'énumération des Moniteurs commencée à l'aide de `yFirstWakeUpMonitor()`.

js	function nextWakeUpMonitor ()
nodejs	function nextWakeUpMonitor ()
php	function nextWakeUpMonitor ()
cpp	YWakeupMonitor * nextWakeUpMonitor ()
m	-(YWakeupMonitor*) nextWakeUpMonitor
pas	function nextWakeUpMonitor (): TYWakeUpMonitor
vb	function nextWakeUpMonitor () As YWakeUpMonitor
cs	YWakeupMonitor nextWakeUpMonitor ()
java	YWakeupMonitor nextWakeUpMonitor ()
py	def nextWakeUpMonitor ()

Retourne :

un pointeur sur un objet `YWakeupMonitor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupmonitor→registerValueCallback()**YWakeUpMonitor****wakeupmonitor.registerValueCallback()**

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 (YWakeUpMonitorValueCallback callback)
m	-(int) registerValueCallback : (YWakeUpMonitorValueCallback) callback
pas	function registerValueCallback (callback : TYWakeUpMonitorValueCallback): 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.

wakeupmonitor→resetSleepCountDown()

wakeupmonitor.resetSleepCountDown()

YWakeUpMonitor

Réinitialise le compteur de mise en sommeil.

js	function resetSleepCountDown ()
nodejs	function resetSleepCountDown ()
php	function resetSleepCountDown ()
cpp	int resetSleepCountDown ()
m	-(int) resetSleepCountDown
pas	function resetSleepCountDown (): LongInt
vb	function resetSleepCountDown () As Integer
cs	int resetSleepCountDown ()
java	int resetSleepCountDown ()
py	def resetSleepCountDown ()
cmd	YWakeUpMonitor target resetSleepCountDown

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.set_logicalName()**

Modifie le nom logique du moniteur.

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	YWakeUpMonitor 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 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.set_nextWakeUp()**

Modifie les jours de la semaine où un réveil doit avoir lieu.

js	function set_nextWakeUp (newval)
nodejs	function set_nextWakeUp (newval)
php	function set_nextWakeUp (\$newval)
cpp	int set_nextWakeUp (s64 newval)
m	-(int) setNextWakeUp : (s64) newval
pas	function set_nextWakeUp (newval : int64): integer
vb	function set_nextWakeUp (ByVal newval As Long) As Integer
cs	int set_nextWakeUp (long newval)
java	int set_nextWakeUp (long newval)
py	def set_nextWakeUp (newval)
cmd	YWakeUpMonitor target set_nextWakeUp newval

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.set_powerDuration()**

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()****YWakeUpMonitor****wakeupmonitor**→**setSleepCountdown()****wakeupmonitor.set_sleepCountdown()**

Modifie le temps avant le prochain sommeil .

js	function set_sleepCountdown (newval)
nodejs	function set_sleepCountdown (newval)
php	function set_sleepCountdown (\$newval)
cpp	int set_sleepCountdown (int newval)
m	-(int) setSleepCountdown : (int) newval
pas	function set_sleepCountdown (newval : LongInt): integer
vb	function set_sleepCountdown (ByVal newval As Integer) As Integer
cs	int set_sleepCountdown (int newval)
java	int set_sleepCountdown (int newval)
py	def set_sleepCountdown (newval)
cmd	YWakeUpMonitor target set_sleepCountdown newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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

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 sleep (secBeforeSleep)
nodejs	function sleep (secBeforeSleep)
php	function sleep (\$secBeforeSleep)
cpp	int sleep (int secBeforeSleep)
m	-(int) sleep : (int) secBeforeSleep
pas	function sleep (secBeforeSleep : LongInt): LongInt
vb	function sleep () As Integer
cs	int sleep (int secBeforeSleep)
java	int sleep (int secBeforeSleep)
py	def sleep (secBeforeSleep)
cmd	YWakeUpMonitor target sleep secBeforeSleep

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()**YWakeUpMonitor****wakeupmonitor.sleepFor()**

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 sleepFor (secUntilWakeUp , secBeforeSleep)
nodejs	function sleepFor (secUntilWakeUp , secBeforeSleep)
php	function sleepFor (\$secUntilWakeUp , \$secBeforeSleep)
c++	int sleepFor (int secUntilWakeUp , int secBeforeSleep)
m	-(int) sleepFor : (int) secUntilWakeUp : (int) secBeforeSleep
pas	function sleepFor (secUntilWakeUp : LongInt, secBeforeSleep : LongInt): LongInt
vb	function sleepFor () As Integer
cs	int sleepFor (int secUntilWakeUp , int secBeforeSleep)
java	int sleepFor (int secUntilWakeUp , int secBeforeSleep)
py	def sleepFor (secUntilWakeUp , secBeforeSleep)
cmd	YWakeUpMonitor target sleepFor secUntilWakeUp secBeforeSleep

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()**YWakeUpMonitor****wakeupmonitor.sleepUntil()**

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 sleepUntil (wakeUpTime , secBeforeSleep)
nodejs	function sleepUntil (wakeUpTime , secBeforeSleep)
php	function sleepUntil (\$wakeUpTime , \$secBeforeSleep)
cpp	int sleepUntil (int wakeUpTime , int secBeforeSleep)
m	-(int) sleepUntil : (int) wakeUpTime : (int) secBeforeSleep
pas	function sleepUntil (wakeUpTime : LongInt, secBeforeSleep : LongInt): LongInt
vb	function sleepUntil () As Integer
cs	int sleepUntil (int wakeUpTime , int secBeforeSleep)
java	int sleepUntil (int wakeUpTime , int secBeforeSleep)
py	def sleepUntil (wakeUpTime , secBeforeSleep)
cmd	YWakeUpMonitor target sleepUntil wakeUpTime secBeforeSleep

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 wakeUp()
nodejs	function wakeUp()
php	function wakeUp()
cpp	int wakeUp()
m	-(int) wakeUp
pas	function wakeUp() : LongInt
vb	function wakeUp() As Integer
cs	int wakeUp()
java	int wakeUp()
py	def wakeUp()
cmd	YWakeUpMonitor target wakeUp

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
yFindWakeUpSchedule(func) Permet de retrouver un réveil agendé d'après un identifiant donné.
yFirstWakeUpSchedule() Commence l'énumération des réveils agendés accessibles par la librairie.
Méthodes des objets YWakeUpSchedule
wakeupschedule→describe() Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE (NAME) = SERIAL . FUNCTIONID.
wakeupschedule→get_advertisedValue() Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).
wakeupschedule→get_errorMessage() Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
wakeupschedule→get_errorType() Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.
wakeupschedule→get_friendlyName() Retourne un identifiant global du réveil agendé au format NOM_MODULE . NOM_FONCTION.
wakeupschedule→get_functionDescriptor() Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.
wakeupschedule→get_functionId() Retourne l'identifiant matériel du réveil agendé, sans référence au module.
wakeupschedule→get_hardwareId() Retourne l'identifiant matériel unique du réveil agendé au format SERIAL . FUNCTIONID.
wakeupschedule→get_hours() Retourne les heures où le réveil est actif..
wakeupschedule→get_logicalName() Retourne le nom logique du réveil agendé.
wakeupschedule→get_minutes() Retourne toutes les minutes de chaque heure où le réveil est actif.
wakeupschedule→get_minutesA()

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()

YWakeupSchedule

YWakeupSchedule.FindWakeUpSchedule()

Permet de retrouver un réveil agendé d'après un identifiant donné.

js	function yFindWakeUpSchedule (func)
nodejs	function FindWakeUpSchedule (func)
php	function yFindWakeUpSchedule (\$func)
cpp	YWakeupSchedule* yFindWakeUpSchedule (const string& func)
m	YWakeupSchedule* yFindWakeUpSchedule (NSString* func)
pas	function yFindWakeUpSchedule (func : string): TYWakeUpSchedule
vb	function yFindWakeUpSchedule (ByVal func As String) As YWakeupSchedule
cs	YWakeupSchedule FindWakeUpSchedule (string func)
java	YWakeupSchedule FindWakeUpSchedule (String func)
py	def FindWakeUpSchedule (func)

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()

YWakeUpSchedule

YWakeUpSchedule.FirstWakeUpSchedule()

Commence l'énumération des réveils agendés accessibles par la librairie.

js	function yFirstWakeUpSchedule ()
nodejs	function FirstWakeUpSchedule ()
php	function yFirstWakeUpSchedule ()
cpp	YWakeUpSchedule* yFirstWakeUpSchedule ()
m	YWakeUpSchedule* yFirstWakeUpSchedule ()
pas	function yFirstWakeUpSchedule (): TYWakeUpSchedule
vb	function yFirstWakeUpSchedule () As YWakeUpSchedule
cs	YWakeUpSchedule FirstWakeUpSchedule ()
java	YWakeUpSchedule FirstWakeUpSchedule ()
py	def FirstWakeUpSchedule ()

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()**YWakeUpSchedule****wakeupschedule.describe()**

Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format
 TYPE(NAME)=SERIAL.FUNCTIONID.

js	function describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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.get_advertisedValue()**

Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YWakeUpSchedule target get_advertisedValue

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.get_errorMessage()

YWakeUpSchedule

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****wakeupschedule.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global du réveil agendé au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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()****YWakeUpSchedule****wakeupschedule**→**functionDescriptor()****wakeupschedule.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor()
nodejs	function get_functionDescriptor()
php	function get_functionDescriptor()
cpp	YFUN_DESCR get_functionDescriptor()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor() : YFUN_DESCR
vb	function get_functionDescriptor() As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor()
java	String get_functionDescriptor()
py	def get_functionDescriptor()

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.get_functionId()**

Retourne l'identifiant matériel du réveil agendé, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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.get_hardwareId()

Retourne l'identifiant matériel unique du réveil agendé au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_hours()**

Retourne les heures où le réveil est actif..

js	function get_hours ()
nodejs	function get_hours ()
php	function get_hours ()
cpp	int get_hours ()
m	-(int) hours
pas	function get_hours (): LongInt
vb	function get_hours () As Integer
cs	int get_hours ()
java	int get_hours ()
py	def get_hours ()
cmd	YWakeUpSchedule target get_hours

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.get_logicalName()**

Retourne le nom logique du réveil agendé.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YWakeUpSchedule target get_logicalName

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.get_minutes()**

Retourne toutes les minutes de chaque heure où le réveil est actif.

js	function get_minutes ()
nodejs	function get_minutes ()
php	function get_minutes ()
cpp	s64 get_minutes ()
m	-(s64) minutes
pas	function get_minutes (): int64
vb	function get_minutes () As Long
cs	long get_minutes ()
java	long get_minutes ()
py	def get_minutes ()
cmd	YWakeUpSchedule target get_minutes

wakeupschedule→**get_minutesA()****YWakeUpSchedule****wakeupschedule**→**minutesA()****wakeupschedule.get_minutesA()**

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

js	function get_minutesA ()
nodejs	function get_minutesA ()
php	function get_minutesA ()
cpp	int get_minutesA ()
m	-(int) minutesA
pas	function get_minutesA (): LongInt
vb	function get_minutesA () As Integer
cs	int get_minutesA ()
java	int get_minutesA ()
py	def get_minutesA ()
cmd	YWakeUpSchedule target get_minutesA

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.get_minutesB()**

Retourne les minutes de l'interval 30-59 de chaque heure où le réveil est actif.

js	function get_minutesB ()
nodejs	function get_minutesB ()
php	function get_minutesB ()
cpp	int get_minutesB ()
m	-(int) minutesB
pas	function get_minutesB (): LongInt
vb	function get_minutesB () As Integer
cs	int get_minutesB ()
java	int get_minutesB ()
py	def get_minutesB ()
cmd	YWakeUpSchedule target get_minutesB

Retourne :

un entier représentant les minutes de l'interval 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.get_module()**

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
cpp	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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.

wakeupschedule→**get_monthDays()**

YWakeUpSchedule

wakeupschedule→**monthDays()**

wakeupschedule.get_monthDays()

Retourne les jours du mois où le réveil est actif..

js	function get_monthDays ()
nodejs	function get_monthDays ()
php	function get_monthDays ()
cpp	int get_monthDays ()
m	-(int) monthDays
pas	function get_monthDays (): LongInt
vb	function get_monthDays () As Integer
cs	int get_monthDays ()
java	int get_monthDays ()
py	def get_monthDays ()
cmd	YWakeUpSchedule target get_monthDays

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()**YWakeUpSchedule****wakeupschedule→months()****wakeupschedule.get_months()**

Retourne les mois où le réveil est actif..

js	function get_months ()
nodejs	function get_months ()
php	function get_months ()
cpp	int get_months ()
m	-(int) months
pas	function get_months (): LongInt
vb	function get_months () As Integer
cs	int get_months ()
java	int get_months ()
py	def get_months ()
cmd	YWakeUpSchedule target get_months

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.get_nextOccurence()**

Retourne la date/heure de la prochaine occurrence de réveil

js	function get_nextOccurence ()
nodejs	function get_nextOccurence ()
php	function get_nextOccurence ()
cpp	s64 get_nextOccurence ()
m	-(s64) nextOccurence
pas	function get_nextOccurence (): int64
vb	function get_nextOccurence () As Long
cs	long get_nextOccurence ()
java	long get_nextOccurence ()
py	def get_nextOccurence ()

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.get_userdata()**

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userdata.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
c++	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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.get_weekDays()

Retourne les jours de la semaine où le réveil est actif..

js	function get_weekDays ()
nodejs	function get_weekDays ()
php	function get_weekDays ()
cpp	int get_weekDays ()
m	-(int) weekDays
pas	function get_weekDays (): LongInt
vb	function get_weekDays () As Integer
cs	int get_weekDays ()
java	int get_weekDays ()
py	def get_weekDays ()
cmd	YWakeUpSchedule target get_weekDays

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()**YWakeUpSchedule****wakeupschedule.isOnline()**

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

js	function isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 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.

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()**YWakeUpSchedule****wakeupschedule.nextWakeUpSchedule()**

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`.

<code>js</code>	<code>function nextWakeUpSchedule()</code>
<code>nodejs</code>	<code>function nextWakeUpSchedule()</code>
<code>php</code>	<code>function nextWakeUpSchedule()</code>
<code>cpp</code>	<code>YWakeupSchedule * nextWakeUpSchedule()</code>
<code>m</code>	<code>-(YWakeupSchedule*) nextWakeUpSchedule</code>
<code>pas</code>	<code>function nextWakeUpSchedule(): TYWakeUpSchedule</code>
<code>vb</code>	<code>function nextWakeUpSchedule() As YWakeUpSchedule</code>
<code>cs</code>	<code>YWakeupSchedule nextWakeUpSchedule()</code>
<code>java</code>	<code>YWakeupSchedule nextWakeUpSchedule()</code>
<code>py</code>	<code>def nextWakeUpSchedule()</code>

Retourne :

un pointeur sur un objet `YWakeupSchedule` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupschedule→registerValueCallback()**YWakeUpSchedule****wakeupschedule.registerValueCallback()**

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 (YWakeUpScheduleValueCallback callback)
m	-(int) registerValueCallback : (YWakeUpScheduleValueCallback) callback
pas	function registerValueCallback (callback : TYWakeUpScheduleValueCallback): 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.

wakeupschedule→set_hours()**YWakeUpSchedule****wakeupschedule→setHours()****wakeupschedule.set_hours ()**

Modifie les heures où un réveil doit avoir lieu

js	function set_hours (newval)
nodejs	function set_hours (newval)
php	function set_hours (\$newval)
cpp	int set_hours (int newval)
m	-(int) setHours : (int) newval
pas	function set_hours (newval : LongInt): integer
vb	function set_hours (ByVal newval As Integer) As Integer
cs	int set_hours (int newval)
java	int set_hours (int newval)
py	def set_hours (newval)
cmd	YWakeUpSchedule target set_hours newval

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.set_logicalName()**

Modifie le nom logique du réveil agendé.

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	YWakeUpSchedule 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 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.set_minutes()**

Modifie toutes les minutes où un réveil doit avoir lieu

js	function set_minutes (bitmap)
nodejs	function set_minutes (bitmap)
php	function set_minutes (\$bitmap)
cpp	int set_minutes (s64 bitmap)
m	-(int) setMinutes : (s64) bitmap
pas	function set_minutes (bitmap : int64): LongInt
vb	function set_minutes () As Integer
cs	int set_minutes (long bitmap)
java	int set_minutes (long bitmap)
py	def set_minutes (bitmap)
cmd	YWakeUpSchedule target set_minutes bitmap

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.set_minutesA()**

Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu

js	function set_minutesA (newval)
nodejs	function set_minutesA (newval)
php	function set_minutesA (\$newval)
cpp	int set_minutesA (int newval)
m	-(int) setMinutesA : (int) newval
pas	function set_minutesA (newval : LongInt): integer
vb	function set_minutesA (ByVal newval As Integer) As Integer
cs	int set_minutesA (int newval)
java	int set_minutesA (int newval)
py	def set_minutesA (newval)
cmd	YWakeUpSchedule target set_minutesA newval

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.set_minutesB()**

Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

js	function set_minutesB (newval)
nodejs	function set_minutesB (newval)
php	function set_minutesB (\$newval)
cpp	int set_minutesB (int newval)
m	-(int) setMinutesB : (int) newval
pas	function set_minutesB (newval : LongInt): integer
vb	function set_minutesB (ByVal newval As Integer) As Integer
cs	int set_minutesB (int newval)
java	int set_minutesB (int newval)
py	def set_minutesB (newval)
cmd	YWakeUpSchedule target set_minutesB newval

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.set_monthDays()**

Modifie les jours du mois où un réveil doit avoir lieu

js	function set_monthDays (newval)
nodejs	function set_monthDays (newval)
php	function set_monthDays (\$newval)
cpp	int set_monthDays (int newval)
m	-(int) setMonthDays : (int) newval
pas	function set_monthDays (newval : LongInt): integer
vb	function set_monthDays (ByVal newval As Integer) As Integer
cs	int set_monthDays (int newval)
java	int set_monthDays (int newval)
py	def set_monthDays (newval)
cmd	YWakeUpSchedule target set_monthDays newval

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.set_months()**

Modifie les mois où un réveil doit avoir lieu

js	function set_months (newval)
nodejs	function set_months (newval)
php	function set_months (\$newval)
cpp	int set_months (int newval)
m	-(int) setMonths : (int) newval
pas	function set_months (newval : LongInt): integer
vb	function set_months (ByVal newval As Integer) As Integer
cs	int set_months (int newval)
java	int set_months (int newval)
py	def set_months (newval)
cmd	YWakeUpSchedule target set_months newval

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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : TObject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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.set_weekDays ()**

Modifie les jours de la semaine où un réveil doit avoir lieu

js	function set_weekDays (newval)
nodejs	function set_weekDays (newval)
php	function set_weekDays (\$newval)
cpp	int set_weekDays (int newval)
m	-(int) setWeekDays : (int) newval
pas	function set_weekDays (newval : LongInt): integer
vb	function set_weekDays (ByVal newval As Integer) As Integer
cs	int set_weekDays (int newval)
java	int set_weekDays (int newval)
py	def set_weekDays (newval)
cmd	YWakeUpSchedule target set_weekDays newval

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()

YWatchdog

yFindWatchdog()YWatchdog.FindWatchdog()

Permet de retrouver un watchdog d'après un identifiant donné.

js	function yFindWatchdog (func)
nodejs	function FindWatchdog (func)
php	function yFindWatchdog (\$func)
cpp	YWatchdog* yFindWatchdog (const string& func)
m	YWatchdog* yFindWatchdog (NSString* func)
pas	function yFindWatchdog (func : string): TYWatchdog
vb	function yFindWatchdog (ByVal func As String) As YWatchdog
cs	YWatchdog FindWatchdog (string func)
java	YWatchdog FindWatchdog (String func)
py	def FindWatchdog (func)

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()**YWatchdog****yFirstWatchdog()****YWatchdog.FirstWatchdog()**

Commence l'énumération des watchdog accessibles par la librairie.

js	function yFirstWatchdog()
nodejs	function FirstWatchdog()
php	function yFirstWatchdog()
cpp	YWatchdog* yFirstWatchdog()
m	YWatchdog* yFirstWatchdog()
pas	function yFirstWatchdog() : TYWatchdog
vb	function yFirstWatchdog() As YWatchdog
cs	YWatchdog FirstWatchdog()
java	YWatchdog FirstWatchdog()
py	def FirstWatchdog()

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()**YWatchdog****watchdog.delayedPulse()**

Pré-programme une impulsion

js	function delayedPulse (ms_delay , ms_duration)
nodejs	function delayedPulse (ms_delay , ms_duration)
php	function delayedPulse (\$ms_delay , \$ms_duration)
cpp	int delayedPulse (int ms_delay , int ms_duration)
m	-(int) delayedPulse : (int) ms_delay : (int) ms_duration
pas	function delayedPulse (ms_delay : LongInt, ms_duration : LongInt): integer
vb	function delayedPulse (ByVal ms_delay As Integer, ByVal ms_duration As Integer) As Integer
cs	int delayedPulse (int ms_delay , int ms_duration)
java	int delayedPulse (int ms_delay , int ms_duration)
py	def delayedPulse (ms_delay , ms_duration)
cmd	YWatchdog target delayedPulse ms_delay ms_duration

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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()****YWatchdog****watchdog**→**advertisedValue()****watchdog.get_advertisedValue()**

Retourne la valeur courante du watchdog (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YWatchdog target get_advertisedValue

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.get_autoStart()`

Retourne l'état du watchdog à la mise sous tension du module.

js	function get_autoStart ()
nodejs	function get_autoStart ()
php	function get_autoStart ()
cpp	Y_AUTOSTART_enum get_autoStart ()
m	-(Y_AUTOSTART_enum) autoStart
pas	function get_autoStart (): Integer
vb	function get_autoStart () As Integer
cs	int get_autoStart ()
java	int get_autoStart ()
py	def get_autoStart ()
cmd	YWatchdog target get_autoStart

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.get_countdown()**

Retourne le nombre de millisecondes restantes avant le déclenchement d'une impulsion préprogrammée par un appel à `delayedPulse()`.

js	function get_countdown ()
nodejs	function get_countdown ()
php	function get_countdown ()
cpp	s64 get_countdown ()
m	-(s64) countdown
pas	function get_countdown (): int64
vb	function get_countdown () As Long
cs	long get_countdown ()
java	long get_countdown ()
py	def get_countdown ()
cmd	YWatchdog target get_countdown

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.get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

js	function get_errorMessage ()
nodejs	function get_errorMessage ()
php	function get_errorMessage ()
cpp	string get_errorMessage ()
m	-(NSString*) errorMessage
pas	function get_errorMessage (): string
vb	function get_errorMessage () As String
cs	string get_errorMessage ()
java	String get_errorMessage ()
py	def get_errorMessage ()

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()****watchdog.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du watchdog.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global du watchdog au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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()****YWatchdog****watchdog**→**functionDescriptor()****watchdog.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()**

Retourne l'identifiant matériel du watchdog, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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.**get_hardwareId()**

Retourne l'identifiant matériel unique du watchdog au format `SERIAL.FUNCTIONID`.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_logicalName()**

Retourne le nom logique du watchdog.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YWatchdog target get_logicalName

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()****YWatchdog****watchdog**→**maxTimeOnStateA()****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.

js	function get_maxTimeOnStateA ()
nodejs	function get_maxTimeOnStateA ()
php	function get_maxTimeOnStateA ()
cpp	s64 get_maxTimeOnStateA ()
m	-(s64) maxTimeOnStateA
pas	function get_maxTimeOnStateA (): int64
vb	function get_maxTimeOnStateA () As Long
cs	long get_maxTimeOnStateA ()
java	long get_maxTimeOnStateA ()
py	def get_maxTimeOnStateA ()
cmd	YWatchdog target get_maxTimeOnStateA

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.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.

js	function get_maxTimeOnStateB ()
nodejs	function get_maxTimeOnStateB ()
php	function get_maxTimeOnStateB ()
cpp	s64 get_maxTimeOnStateB ()
m	-(s64) maxTimeOnStateB
pas	function get_maxTimeOnStateB (): int64
vb	function get_maxTimeOnStateB () As Long
cs	long get_maxTimeOnStateB ()
java	long get_maxTimeOnStateB ()
py	def get_maxTimeOnStateB ()
cmd	YWatchdog target get_maxTimeOnStateB

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.get_module()**

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
c++	<code>YModule *</code> get_module ()
m	<code>-(YModule*)</code> module
pas	function get_module (): <code>TYModule</code>
vb	function get_module () As <code>YModule</code>
cs	<code>YModule</code> get_module ()
java	<code>YModule</code> get_module ()
py	def get_module ()

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 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-qu'el 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.get_output()**

Retourne l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

js	function get_output ()
nodejs	function get_output ()
php	function get_output ()
cpp	Y_OUTPUT_enum get_output ()
m	-(Y_OUTPUT_enum) output
pas	function get_output (): Integer
vb	function get_output () As Integer
cs	int get_output ()
java	int get_output ()
py	def get_output ()
cmd	YWatchdog target get_output

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.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.

js	function get_pulseTimer ()
nodejs	function get_pulseTimer ()
php	function get_pulseTimer ()
c++	s64 get_pulseTimer ()
m	-(s64) pulseTimer
pas	function get_pulseTimer (): int64
vb	function get_pulseTimer () As Long
cs	long get_pulseTimer ()
java	long get_pulseTimer ()
py	def get_pulseTimer ()
cmd	YWatchdog target get_pulseTimer

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.get_running()`

Retourne l'état du watchdog.

js	function get_running ()
nodejs	function get_running ()
php	function get_running ()
cpp	Y_RUNNING_enum get_running ()
m	-(Y_RUNNING_enum) running
pas	function get_running (): Integer
vb	function get_running () As Integer
cs	int get_running ()
java	int get_running ()
py	def get_running ()
cmd	YWatchdog target get_running

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.get_state()`

Retourne l'état du watchdog (A pour la position de repos, B pour l'état actif).

js	function get_state ()
nodejs	function get_state ()
php	function get_state ()
cpp	Y_STATE_enum get_state ()
m	-(Y_STATE_enum) state
pas	function get_state (): Integer
vb	function get_state () As Integer
cs	int get_state ()
java	int get_state ()
py	def get_state ()
cmd	YWatchdog target get_state

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.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).

js	function get_stateAtPowerOn ()
nodejs	function get_stateAtPowerOn ()
php	function get_stateAtPowerOn ()
cpp	Y_STATEATPOWERON_enum get_stateAtPowerOn ()
m	-(Y_STATEATPOWERON_enum) stateAtPowerOn
pas	function get_stateAtPowerOn (): Integer
vb	function get_stateAtPowerOn () As Integer
cs	int get_stateAtPowerOn ()
java	int get_stateAtPowerOn ()
py	def get_stateAtPowerOn ()
cmd	YWatchdog target get_stateAtPowerOn

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.get_triggerDelay()**

Retourne le délai d'attente avant qu'un reset ne soit automatiquement généré par le watchdog, en millisecondes.

js	function get_triggerDelay ()
nodejs	function get_triggerDelay ()
php	function get_triggerDelay ()
c++	s64 get_triggerDelay ()
m	-(s64) triggerDelay
pas	function get_triggerDelay (): int64
vb	function get_triggerDelay () As Long
cs	long get_triggerDelay ()
java	long get_triggerDelay ()
py	def get_triggerDelay ()
cmd	YWatchdog target get_triggerDelay

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.get_triggerDuration()**

Retourne la durée d'un reset généré par le watchdog, en millisecondes.

js	function get_triggerDuration ()
nodejs	function get_triggerDuration ()
php	function get_triggerDuration ()
cpp	s64 get_triggerDuration ()
m	-(s64) triggerDuration
pas	function get_triggerDuration (): int64
vb	function get_triggerDuration () As Long
cs	long get_triggerDuration ()
java	long get_triggerDuration ()
py	def get_triggerDuration ()
cmd	YWatchdog target get_triggerDuration

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.get_userdata()`

Retourne le contenu de l'attribut `userData`, précédemment stocké à l'aide de la méthode `set_userdata`.

js	function get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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 isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 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.

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()****YWatchdog****watchdog.nextWatchdog()**

Continue l'énumération des watchdog commencée à l'aide de `yFirstWatchdog()`.

js	function nextWatchdog ()
nodejs	function nextWatchdog ()
php	function nextWatchdog ()
c++	YWatchdog * nextWatchdog ()
m	-(YWatchdog*) nextWatchdog
pas	function nextWatchdog (): TYWatchdog
vb	function nextWatchdog () As YWatchdog
cs	YWatchdog nextWatchdog ()
java	YWatchdog nextWatchdog ()
py	def nextWatchdog ()

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).

<code>js</code>	<code>function pulse(ms_duration)</code>
<code>nodejs</code>	<code>function pulse(ms_duration)</code>
<code>php</code>	<code>function pulse(\$ms_duration)</code>
<code>cpp</code>	<code>int pulse(int ms_duration)</code>
<code>m</code>	<code>-(int) pulse : (int) ms_duration</code>
<code>pas</code>	<code>function pulse(ms_duration: LongInt): integer</code>
<code>vb</code>	<code>function pulse(ByVal ms_duration As Integer) As Integer</code>
<code>cs</code>	<code>int pulse(int ms_duration)</code>
<code>java</code>	<code>int pulse(int ms_duration)</code>
<code>py</code>	<code>def pulse(ms_duration)</code>
<code>cmd</code>	<code>YWatchdog target pulse ms_duration</code>

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()**YWatchdog****watchdog.registerValueCallback()**

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 (YWatchdogValueCallback callback)
m	-(int) registerValueCallback : (YWatchdogValueCallback) callback
pas	function registerValueCallback (callback : TYWatchdogValueCallback): 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.

watchdog→resetWatchdog()**YWatchdog****watchdog.resetWatchdog()**

Réinitialise le WatchDog.

js	function resetWatchdog ()
nodejs	function resetWatchdog ()
php	function resetWatchdog ()
cpp	int resetWatchdog ()
m	-(int) resetWatchdog
pas	function resetWatchdog (): integer
vb	function resetWatchdog () As Integer
cs	int resetWatchdog ()
java	int resetWatchdog ()
py	def resetWatchdog ()
cmd	YWatchdog target resetWatchdog

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()****watchdog**→**setAutoStart()****watchdog.set_autoStart()**

Modifie l'état du watching au démarrage du module.

js	function set_autoStart (newval)
nodejs	function set_autoStart (newval)
php	function set_autoStart (\$newval)
cpp	int set_autoStart (Y_AUTOSTART_enum newval)
m	-(int) setAutoStart : (Y_AUTOSTART_enum) newval
pas	function set_autoStart (newval : Integer): integer
vb	function set_autoStart (ByVal newval As Integer) As Integer
cs	int set_autoStart (int newval)
java	int set_autoStart (int newval)
py	def set_autoStart (newval)
cmd	YWatchdog target set_autoStart newval

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()**YWatchdog****watchdog→setLogicalName()****watchdog.set_logicalName()**

Modifie le nom logique du watchdog.

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	YWatchdog 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 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.set_maxTimeOnStateA()**

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 set_maxTimeOnStateA (newval)
nodejs	function set_maxTimeOnStateA (newval)
php	function set_maxTimeOnStateA (\$newval)
cpp	int set_maxTimeOnStateA (s64 newval)
m	-(int) setMaxTimeOnStateA : (s64) newval
pas	function set_maxTimeOnStateA (newval : int64): integer
vb	function set_maxTimeOnStateA (ByVal newval As Long) As Integer
cs	int set_maxTimeOnStateA (long newval)
java	int set_maxTimeOnStateA (long newval)
py	def set_maxTimeOnStateA (newval)
cmd	YWatchdog target set_maxTimeOnStateA newval

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.set_maxTimeOnStateB()**

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 set_maxTimeOnStateB (newval)
nodejs	function set_maxTimeOnStateB (newval)
php	function set_maxTimeOnStateB (\$newval)
cpp	int set_maxTimeOnStateB (s64 newval)
m	-(int) setMaxTimeOnStateB : (s64) newval
pas	function set_maxTimeOnStateB (newval : int64): integer
vb	function set_maxTimeOnStateB (ByVal newval As Long) As Integer
cs	int set_maxTimeOnStateB (long newval)
java	int set_maxTimeOnStateB (long newval)
py	def set_maxTimeOnStateB (newval)
cmd	YWatchdog target set_maxTimeOnStateB newval

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.set_output ()`

Modifie l'état de la sortie du watchdog, lorsqu'il est utilisé comme un simple interrupteur.

js	function set_output (newval)
nodejs	function set_output (newval)
php	function set_output (\$newval)
cpp	int set_output (Y_OUTPUT_enum newval)
m	-(int) setOutput : (Y_OUTPUT_enum) newval
pas	function set_output (newval : Integer): integer
vb	function set_output (ByVal newval As Integer) As Integer
cs	int set_output (int newval)
java	int set_output (int newval)
py	def set_output (newval)
cmd	YWatchdog target set_output newval

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.set_running()`

Modifie manuellement l'état de fonctionnement du watchdog.

js	function set_running (newval)
nodejs	function set_running (newval)
php	function set_running (\$newval)
cpp	int set_running (Y_RUNNING_enum newval)
m	-(int) setRunning : (Y_RUNNING_enum) newval
pas	function set_running (newval : Integer): integer
vb	function set_running (ByVal newval As Integer) As Integer
cs	int set_running (int newval)
java	int set_running (int newval)
py	def set_running (newval)
cmd	YWatchdog target set_running newval

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()**

YWatchdog

watchdog→**setState()**`watchdog.set_state()`

Modifie l'état du watchdog (A pour la position de repos, B pour l'état actif).

js	function set_state (newval)
nodejs	function set_state (newval)
php	function set_state (\$newval)
cpp	int set_state (Y_STATE_enum newval)
m	-(int) setState : (Y_STATE_enum) newval
pas	function set_state (newval : Integer): integer
vb	function set_state (ByVal newval As Integer) As Integer
cs	int set_state (int newval)
java	int set_state (int newval)
py	def set_state (newval)
cmd	YWatchdog target set_state newval

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.set_stateAtPowerOn()**

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 set_stateAtPowerOn (newval)
nodejs	function set_stateAtPowerOn (newval)
php	function set_stateAtPowerOn (\$newval)
cpp	int set_stateAtPowerOn (Y_STATEATPOWERON_enum newval)
m	-(int) setStateAtPowerOn : (Y_STATEATPOWERON_enum) newval
pas	function set_stateAtPowerOn (newval : Integer): integer
vb	function set_stateAtPowerOn (ByVal newval As Integer) As Integer
cs	int set_stateAtPowerOn (int newval)
java	int set_stateAtPowerOn (int newval)
py	def set_stateAtPowerOn (newval)
cmd	YWatchdog target set_stateAtPowerOn newval

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.set_triggerDelay()**

Modifie le délai d'attente avant qu'un reset ne soit généré par le watchdog, en millisecondes.

js	function set_triggerDelay (newval)
nodejs	function set_triggerDelay (newval)
php	function set_triggerDelay (\$newval)
cpp	int set_triggerDelay (s64 newval)
m	-(int) setTriggerDelay : (s64) newval
pas	function set_triggerDelay (newval : int64): integer
vb	function set_triggerDelay (ByVal newval As Long) As Integer
cs	int set_triggerDelay (long newval)
java	int set_triggerDelay (long newval)
py	def set_triggerDelay (newval)
cmd	YWatchdog target set_triggerDelay newval

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()**YWatchdog****watchdog→setTriggerDuration()****watchdog.set_triggerDuration()**

Modifie la durée des resets générés par le watchdog, en millisecondes.

js	function set_triggerDuration (newval)
nodejs	function set_triggerDuration (newval)
php	function set_triggerDuration (\$newval)
cpp	int set_triggerDuration (s64 newval)
m	-(int) setTriggerDuration : (s64) newval
pas	function set_triggerDuration (newval : int64): integer
vb	function set_triggerDuration (ByVal newval As Long) As Integer
cs	int set_triggerDuration (long newval)
java	int set_triggerDuration (long newval)
py	def set_triggerDuration (newval)
cmd	YWatchdog target set_triggerDuration newval

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()****watchdog**→**setUserData()****watchdog.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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()**YWireless****yFindWireless()****YWireless.FindWireless()**

Permet de retrouver une interface réseau sans fil d'après un identifiant donné.

js	function yFindWireless (func)
nodejs	function FindWireless (func)
php	function yFindWireless (\$func)
cpp	YWireless* yFindWireless (string func)
m	+(YWireless*) yFindWireless : (NSString*) func
pas	function yFindWireless (func : string): TYWireless
vb	function yFindWireless (ByVal func As String) As YWireless
cs	YWireless FindWireless (string func)
java	YWireless FindWireless (String func)
py	def FindWireless (func)

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()**YWireless****yFirstWireless()****ywireless.FirstWireless()**

Commence l'énumération des interfaces réseau sans fil accessibles par la librairie.

js	function yFirstWireless ()
nodejs	function FirstWireless ()
php	function yFirstWireless ()
cpp	YWireless* yFirstWireless ()
m	YWireless* yFirstWireless ()
pas	function yFirstWireless (): TYWireless
vb	function yFirstWireless () As YWireless
cs	YWireless FirstWireless ()
java	YWireless FirstWireless ()
py	def FirstWireless ()

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()**YWireless****wireless.adhocNetwork()**

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 adhocNetwork (ssid , securityKey)
nodejs	function adhocNetwork (ssid , securityKey)
php	function adhocNetwork (\$ssid , \$securityKey)
cpp	int adhocNetwork (string ssid , string securityKey)
m	-(int) adhocNetwork : (NSString*) ssid : (NSString*) securityKey
pas	function adhocNetwork (ssid : string, securityKey : string): integer
vb	function adhocNetwork (ByVal ssid As String, ByVal securityKey As String) As Integer
cs	int adhocNetwork (string ssid , string securityKey)
java	int adhocNetwork (String ssid , String securityKey)
py	def adhocNetwork (ssid , securityKey)
cmd	YWireless target adhocNetwork ssid securityKey

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 describe ()
nodejs	function describe ()
php	function describe ()
cpp	string describe ()
m	-(NSString*) describe
pas	function describe (): string
vb	function describe () As String
cs	string describe ()
java	String describe ()
py	def describe ()

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()****YWireless****wireless**→**advertisedValue()****wireless.get_advertisedValue()**

Retourne la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

js	function get_advertisedValue ()
nodejs	function get_advertisedValue ()
php	function get_advertisedValue ()
cpp	string get_advertisedValue ()
m	-(NSString*) advertisedValue
pas	function get_advertisedValue (): string
vb	function get_advertisedValue () As String
cs	string get_advertisedValue ()
java	String get_advertisedValue ()
py	def get_advertisedValue ()
cmd	YWireless target get_advertisedValue

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.get_channel()**

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

js	function get_channel ()
nodejs	function get_channel ()
php	function get_channel ()
cpp	int get_channel ()
m	-(int) channel
pas	function get_channel (): LongInt
vb	function get_channel () As Integer
cs	int get_channel ()
java	int get_channel ()
py	def get_channel ()
cmd	YWireless target get_channel

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.get_detectedWlans()**

Retourne une liste d'objets objet YFileRecord qui décrivent les réseaux sans fils détectés.

js	function get_detectedWlans ()
nodejs	function get_detectedWlans ()
php	function get_detectedWlans ()
cpp	vector<YWlanRecord> get_detectedWlans ()
m	-(NSMutableArray*) detectedWlans
pas	function get_detectedWlans (): TYWlanRecordArray
vb	function get_detectedWlans () As List
cs	List<YWlanRecord> get_detectedWlans ()
java	ArrayList<YWlanRecord> get_detectedWlans ()
py	def get_detectedWlans ()
cmd	YWireless target get_detectedWlans

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.get_errorMessage()**

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

js	function get_errorMessage()
nodejs	function get_errorMessage()
php	function get_errorMessage()
c++	string get_errorMessage()
m	-(NSString*) errorMessage
pas	function get_errorMessage() : string
vb	function get_errorMessage() As String
cs	string get_errorMessage()
java	String get_errorMessage()
py	def get_errorMessage()

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()****wireless.get_errorType()**

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau sans fil.

js	function get_errorType ()
nodejs	function get_errorType ()
php	function get_errorType ()
cpp	YRETCODE get_errorType ()
pas	function get_errorType (): YRETCODE
vb	function get_errorType () As YRETCODE
cs	YRETCODE get_errorType ()
java	int get_errorType ()
py	def get_errorType ()

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.get_friendlyName()**

Retourne un identifiant global de l'interface réseau sans fil au format `NOM_MODULE.NOM_FONCTION`.

js	function get_friendlyName ()
nodejs	function get_friendlyName ()
php	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	def get_friendlyName ()

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.get_functionDescriptor()**

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function get_functionDescriptor ()
nodejs	function get_functionDescriptor ()
php	function get_functionDescriptor ()
cpp	YFUN_DESCR get_functionDescriptor ()
m	-(YFUN_DESCR) functionDescriptor
pas	function get_functionDescriptor (): YFUN_DESCR
vb	function get_functionDescriptor () As YFUN_DESCR
cs	YFUN_DESCR get_functionDescriptor ()
java	String get_functionDescriptor ()
py	def get_functionDescriptor ()

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.get_functionId()**

Retourne l'identifiant matériel de l'interface réseau sans fil, sans référence au module.

js	function get_functionId ()
nodejs	function get_functionId ()
php	function get_functionId ()
cpp	string get_functionId ()
m	-(NSString*) functionId
vb	function get_functionId () As String
cs	string get_functionId ()
java	String get_functionId ()
py	def get_functionId ()

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.get_hardwareId()**

Retourne l'identifiant matériel unique de l'interface réseau sans fil au format SERIAL.FUNCTIONID.

js	function get_hardwareId ()
nodejs	function get_hardwareId ()
php	function get_hardwareId ()
cpp	string get_hardwareId ()
m	-(NSString*) hardwareId
vb	function get_hardwareId () As String
cs	string get_hardwareId ()
java	String get_hardwareId ()
py	def get_hardwareId ()

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.get_linkQuality()**

Retourne la qualité de la connection, exprimée en pourcents.

js	function get_linkQuality ()
nodejs	function get_linkQuality ()
php	function get_linkQuality ()
cpp	int get_linkQuality ()
m	-(int) linkQuality
pas	function get_linkQuality (): LongInt
vb	function get_linkQuality () As Integer
cs	int get_linkQuality ()
java	int get_linkQuality ()
py	def get_linkQuality ()
cmd	YWireless target get_linkQuality

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.get_logicalName()**

Retourne le nom logique de l'interface réseau sans fil.

js	function get_logicalName ()
nodejs	function get_logicalName ()
php	function get_logicalName ()
cpp	string get_logicalName ()
m	-(NSString*) logicalName
pas	function get_logicalName (): string
vb	function get_logicalName () As String
cs	string get_logicalName ()
java	String get_logicalName ()
py	def get_logicalName ()
cmd	YWireless target get_logicalName

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.get_message()`

Retourne le dernier message de diagnostic de l'interface au réseau sans fil.

js	function get_message ()
nodejs	function get_message ()
php	function get_message ()
cpp	string get_message ()
m	-(NSString*) message
pas	function get_message (): string
vb	function get_message () As String
cs	string get_message ()
java	String get_message ()
py	def get_message ()
cmd	YWireless target get_message

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.get_module()`

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function get_module ()
nodejs	function get_module ()
php	function get_module ()
c++	YModule * get_module ()
m	-(YModule*) module
pas	function get_module (): TYModule
vb	function get_module () As YModule
cs	YModule get_module ()
java	YModule get_module ()
py	def get_module ()

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 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-qu'el à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wireless→**get_security()****YWireless****wireless**→**security()****wireless.get_security()**

Retourne l'algorithme de sécurité utilisé par le réseau sans-fil sélectionné.

js	function get_security ()
nodejs	function get_security ()
php	function get_security ()
cpp	Y_SECURITY_enum get_security ()
m	-(Y_SECURITY_enum) security
pas	function get_security (): Integer
vb	function get_security () As Integer
cs	int get_security ()
java	int get_security ()
py	def get_security ()
cmd	YWireless target get_security

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.get_ssid()**

Retourne le nom (SSID) du réseau sans-fil sélectionné.

js	function get_ssid ()
nodejs	function get_ssid ()
php	function get_ssid ()
cpp	string get_ssid ()
m	-(NSString*) ssid
pas	function get_ssid (): string
vb	function get_ssid () As String
cs	string get_ssid ()
java	String get_ssid ()
py	def get_ssid ()
cmd	YWireless target get_ssid

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 get_userdata ()
nodejs	function get_userdata ()
php	function get_userdata ()
cpp	void * get_userdata ()
m	-(void*) userData
pas	function get_userdata (): Tobject
vb	function get_userdata () As Object
cs	object get_userdata ()
java	Object get_userdata ()
py	def get_userdata ()

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 isOnline ()
nodejs	function isOnline ()
php	function isOnline ()
cpp	bool isOnline ()
m	-(BOOL) isOnline
pas	function isOnline (): boolean
vb	function isOnline () As Boolean
cs	bool isOnline ()
java	boolean isOnline ()
py	def isOnline ()

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 joinNetwork (ssid , securityKey)
nodejs	function joinNetwork (ssid , securityKey)
php	function joinNetwork (\$ssid , \$securityKey)
cpp	int joinNetwork (string ssid , string securityKey)
m	-(int) joinNetwork : (NSString*) ssid : (NSString*) securityKey
pas	function joinNetwork (ssid : string, securityKey : string): integer
vb	function joinNetwork (ByVal ssid As String, ByVal securityKey As String) As Integer
cs	int joinNetwork (string ssid , string securityKey)
java	int joinNetwork (String ssid , String securityKey)
py	def joinNetwork (ssid , securityKey)
cmd	YWireless target joinNetwork ssid securityKey

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 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.

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()**

YWireless

wireless.nextWireless()

Continue l'énumération des interfaces réseau sans fil commencée à l'aide de `yFirstWireless()`.

<code>js</code>	<code>function nextWireless()</code>
<code>nodejs</code>	<code>function nextWireless()</code>
<code>php</code>	<code>function nextWireless()</code>
<code>cpp</code>	<code>YWireless * nextWireless()</code>
<code>m</code>	<code>-(YWireless*) nextWireless</code>
<code>pas</code>	<code>function nextWireless(): TYWireless</code>
<code>vb</code>	<code>function nextWireless() As YWireless</code>
<code>cs</code>	<code>YWireless nextWireless()</code>
<code>java</code>	<code>YWireless nextWireless()</code>
<code>py</code>	<code>def nextWireless()</code>

Retourne :

un pointeur sur un objet `YWireless` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wireless→registerValueCallback()**YWireless****wireless.registerValueCallback()**

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)
c++	int registerValueCallback (YWirelessValueCallback callback)
m	-(int) registerValueCallback : (YWirelessValueCallback) callback
pas	function registerValueCallback (callback : TYWirelessValueCallback): 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.

wireless→**set_logicalName()****YWireless****wireless**→**setLogicalName()****wireless.set_logicalName()**

Modifie le nom logique de l'interface réseau sans fil.

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	YWireless 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 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.set_userdata()**

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 set_userdata (data)
nodejs	function set_userdata (data)
php	function set_userdata (\$data)
cpp	void set_userdata (void* data)
m	-(void) setUserData : (void*) data
pas	procedure set_userdata (data : Tobject)
vb	procedure set_userdata (ByVal data As Object)
cs	void set_userdata (object data)
java	void set_userdata (Object data)
py	def set_userdata (data)

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 1563
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_reportFrequency, YCurrent	267
get_period, YPwmOutput	1103	get_reportFrequency, YGenericSensor	576
get_persistentSettings, YModule	877	get_reportFrequency, YGyro	632
get_pitch, YGyro	626	get_reportFrequency, YHumidity	710
get_poeCurrent, YNetwork	931	get_reportFrequency, YLightSensor	785
get_portDirection, YDigitalIO	382	get_reportFrequency, YMagnetometer	828
get_portOpenDrain, YDigitalIO	383	get_reportFrequency, YPower	1017
get_portPolarity, YDigitalIO	384	get_reportFrequency, YPressure	1061
get_portSize, YDigitalIO	385	get_reportFrequency, YQt	1173
get_portState, YDigitalIO	386	get_reportFrequency, YSensor	1327
get_portState, YHubPort	675	get_reportFrequency, YTemperature	1409
get_position, YServo	1366	get_reportFrequency, YTilt	1454
get_positionAtPowerOn, YServo	1367	get_reportFrequency, YVoc	1497
get_power, YLed	747	get_reportFrequency, YVoltage	1540
get_powerControl, YDualPower	507	get_resolution, YAccelerometer	60
get_powerDuration, YWakeUpMonitor	1613	get_resolution, YCarbonDioxide	148
get_powerMode, YPwmPowerSource	1138	get_resolution, YCompass	225
get_powerState, YDualPower	508	get_resolution, YCurrent	268
get_preview, YDataSet	338	get_resolution, YGenericSensor	577
get_primaryDNS, YNetwork	932	get_resolution, YGyro	633
get_productId, YModule	878	get_resolution, YHumidity	711
get_productName, YModule	879	get_resolution, YLightSensor	786
get_productRelease, YModule	880	get_resolution, YMagnetometer	829
get_progress, YDataSet	339	get_resolution, YPower	1018
get_pulseCounter, YAnButton	105	get_resolution, YPressure	1062
get_pulseDuration, YPwmOutput	1104	get_resolution, YQt	1174
get_pulseTimer, YAnButton	106	get_resolution, YSensor	1328
get_pulseTimer, YRelay	1285	get_resolution, YTemperature	1410
get_pulseTimer, YWatchdog	1698	get_resolution, YTilt	1455
get_quaternionW, YGyro	627	get_resolution, YVoc	1498
get_quaternionX, YGyro	628	get_resolution, YVoltage	1541
get_quaternionY, YGyro	629	get_rgbColor, YColorLed	183
get_quaternionZ, YGyro	630	get_rgbColorAtPowerOn, YColorLed	184
get_range, YServo	1368	get_roll, YGyro	634
get_rawValue, YAnButton	107	get_router, YNetwork	934
get_readiness, YNetwork	933	get_rowCount, YDataStream	356
get_rebootCountdown, YModule	881	get_runIndex, YDataStream	357
get_recordedData, YAccelerometer	58	get_running, YWatchdog	1699
get_recordedData, YCarbonDioxide	146	get_secondaryDNS, YNetwork	935
get_recordedData, YCompass	223	get_security, YWireless	1745
get_recordedData, YCurrent	266	get_sensitivity, YAnButton	108
get_recordedData, YGenericSensor	575	get_sensorType, YTemperature	1411
get_recordedData, YGyro	631	get_serialNumber, YModule	882
get_recordedData, YHumidity	709	get_shutdownCountdown, YOsControl	979
get_recordedData, YLightSensor	784	get_signalRange, YGenericSensor	578
get_recordedData, YMagnetometer	827	get_signalUnit, YGenericSensor	579
get_recordedData, YPower	1016	get_signalValue, YGenericSensor	580
get_recordedData, YPressure	1060	get_sleepCountdown, YWakeUpMonitor	1614
get_recordedData, YQt	1172	get_ssid, YWireless	1746
get_recordedData, YSensor	1326	get_startTime, YDataStream	358
get_recordedData, YTemperature	1408	get_startTimeUTC, YDataRun	328
get_recordedData, YTilt	1453	get_startTimeUTC, YDataSet	340
get_recordedData, YVoc	1496	get_startTimeUTC, YDataStream	359
get_recordedData, YVoltage	1539	get_startTimeUTC, YMeasure	856
get_recording, YDataLogger	307	get_startupSeq, YDisplay	436
get_regulationFailure, YVSource	1580	get_state, YRelay	1286
get_reportFrequency, YAccelerometer	59	get_state, YWatchdog	1700
get_reportFrequency, YCarbonDioxide	147	get_stateAtPowerOn, YRelay	1287
get_reportFrequency, YCompass	224	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

Java 3
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

Objets 460

P

pauseSequence, YDisplay 444
ping, YNetwork 945
playSequence, YDisplay 445
Port 659
Power 992
Préparation 3
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_resolution, YCarbonDioxide	164
set_lowestValue, YPressure	1076	set_resolution, YCompass	241
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_userdata, YPwmPowerSource 1148
- 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
- yCheckLogicalName 12
- YColorLed 169-199
- YCompass 203-243
- YCurrent 247-286
- YDataLogger 289-321
- YDataRun 323-331
- YDataSet 334-344
- YDataStream 347-359
- YDigitalIO 363-407
- yDisableExceptions 13
- YDisplay 411-458
- YDisplayLayer 461-490
- YDualPower 493-519
- yEnableExceptions 14
- yEnableUSBHost 15
- YFiles 522-552
- yFindAccelerometer 39
- yFindAnButton 85
- yFindCarbonDioxide 127
- yFindColorLed 169
- yFindCompass 203
- yFindCurrent 247
- yFindDataLogger 289
- yFindDigitalIO 363
- yFindDisplay 411
- yFindDualPower 493
- yFindFiles 522
- yFindGenericSensor 556
- yFindGyro 606
- yFindHubPort 660
- yFindHumidity 690
- yFindLed 732
- yFindLightSensor 764
- yFindMagnetometer 808
- yFindModule 860
- yFindNetwork 907
- yFindOsControl 966
- yFindPower 994
- yFindPressure 1041
- yFindPwmOutput 1084
- yFindPwmPowerSource 1125
- yFindQt 1153
- yFindRealTimeClock 1195
- yFindRefFrame 1227
- yFindRelay 1267
- yFindSensor 1307
- yFindServo 1350
- yFindTemperature 1389
- yFindTilt 1434
- yFindVoc 1477
- yFindVoltage 1520
- yFindVSource 1562
- yFindWakeUpMonitor 1599
- yFindWakeUpSchedule 1638
- yFindWatchdog 1679
- yFindWireless 1727
- yFirstAccelerometer 40
- yFirstAnButton 86
- yFirstCarbonDioxide 128
- yFirstColorLed 170
- yFirstCompass 204
- yFirstCurrent 248
- yFirstDataLogger 290
- yFirstDigitalIO 364
- yFirstDisplay 412
- yFirstDualPower 494
- yFirstFiles 523
- yFirstGenericSensor 557
- yFirstGyro 607
- yFirstHubPort 661
- yFirstHumidity 691
- yFirstLed 733
- yFirstLightSensor 765
- yFirstMagnetometer 809
- yFirstModule 861
- yFirstNetwork 908
- yFirstOsControl 967
- yFirstPower 995
- yFirstPressure 1042
- yFirstPwmOutput 1085
- yFirstPwmPowerSource 1126
- yFirstQt 1154
- yFirstRealTimeClock 1196
- yFirstRefFrame 1228
- yFirstRelay 1268
- yFirstSensor 1308
- yFirstServo 1351
- yFirstTemperature 1390
- yFirstTilt 1435
- yFirstVoc 1478
- yFirstVoltage 1521
- yFirstVSource 1563
- yFirstWakeUpMonitor 1600
- yFirstWakeUpSchedule 1639
- yFirstWatchdog 1680
- yFirstWireless 1728
- yFreeAPI 16
- YGenericSensor 556-602
- yGetAPIVersion 17
- yGetTickCount 18
- YGyro 606-657
- yHandleEvents 19
- YHubPort 660-686
- YHumidity 690-729
- yInitAPI 20
- YLed 732-760
- YLightSensor 764-804
- YMagnetometer 808-850
- YMeasure 852-856
- YModule 860-902
- YNetwork 907-963
- Yocto-Demo 3
- Yocto-hub 659
- YOsControl 966-990
- YPower 994-1037
- yPreregisterHub 21

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