

2021

YMLIB API Referenzen USB

Version: 1.0



YMLib API Referenz für USB Module (Windows)

Inhaltsverzeichnis

1	YMLIB API Referenzen USB.....	2
2	Klasse DeviceServer	2
3	Klasse Device	3
4	Klasse ReadData.....	3
5	Erklärung der Funktionen und Methoden.....	4

1 YMLIB API Referenzen USB

Dank **des HID-Verfahrens** benötigen die Yamutec® USB-Module **keine Treiberinstallation**. Die Yamutec® USB-Module werden nach dem anschließen von der Betriebssystem automatisch erkannt.

Die YMLib-API erlaubt Ihnen die individuelle Anpassung der Yamutec® USB-Module per VB6, VB.NET, C# oder LabView an Ihren Anwendungsfall.

Anhand der gelieferten Programmierbeispiele, können Sie die gesamte Programmierung entnehmen.

2 Klasse DeviceServer

DeviceServer ist die Hauptklasse in der YMLib.dll-Datei, um die Yamutec® USB-Module anzusprechen.

Methoden der Klasse DeviceServer

YMcom.StartAsync()

startet den Server. Es erkennt angeschlossene/getrennte Geräte und liest Daten von allen angeschlossenen Geräten.

YMcom.StopAsync()

Stoppt den Server vollständig.

YMcom.TIMEOUT_INTERVAL_AND_CHANNEL()

Schaltet die ausgewählten Ausgänge aus/ein, wenn sich das Gerät im Timeout-Status befindet.

YMcom.TIMEOUT_DISABLE()

Deaktiviert den Timeout-Status.

YMcom.HW_RESET()

Setzt das Modul zurück.

YMcom.GetDevice()

Ruft Informationen vom Modul

YMcom.CALL_ABOUT_BOX()

Ruft den YMLib.dll Version auf

YMcom.SEND_OUTPUT_STATUS_X()

Sendet ein Schaltzustand an das Modul

3 Klasse Device

Eigenschaften der Klasse Device

device.Adresse = Integer

Gibt die Adresse zurück

Eigenschaften der Klasse device.definition

device.Definition.ProductName = String

Gibt den Modellnamen zurück

device.Definition.SerialNumber = String

Gibt die Seriennummer zurück

device.Definition.ProductID = Integer

Gibt die Produkt ID zurück

device.Definition.VendorID = Integer

Gibt die Hersteller ID zurück

device.Definition.VendorName = String

Gibt den Herstellername zurück

4 Klasse ReadData

ReadData ist die Klasse, um gelesene Daten vom Gerät zu kapseln.

Eigenschaften der Klasse ReadData

Timeout_ausgeloest = Byte

Gibt zurück, wenn sich das Gerät derzeit im Timeout-Status befindet

Zt_Reset = Byte

Gibt den Timeout Intervall zurück

**RelaisCount1, RelaisCount2, RelaisCount3, RelaisCount4, RelaisCount5, RelaisCount6,
RelaisCount7, RelaisCount8 = Byte**

Gibt die Zustände der Ausgänge zurück

Isttimeraktiviert = Byte

Gibt zurück, ob Timeout aktiviert ist

5 Erklärung der Funktionen und Methoden

Initialisieren und konfigurieren des Servers

```
YMcom = New DeviceServer With
```

```
{
```

Registriert eine Funktion, die ausgeführt wird, wenn ein Gerät verbunden wird

```
.DeviceConnectedCallback = AddressOf DeviceConnectedCallback,
```

Registriert eine Funktion, die ausgeführt wird, wenn ein Gerät getrennt wird

```
.DeviceDisconnectedCallback = AddressOf DeviceDisconnectedCallback,
```

Registriert eine Funktion, die ausgeführt wird, wenn sich der Status eines Geräts ändert, verbunden oder getrennt wird

```
.DevicesChangedCallback = AddressOf DevicesChangedCallback,
```

```
.ReadAllDevicesContinuously = True,
```

```
.ReadAllDevicesInterval = 1,
```

Registriert eine Funktion, die ausgeführt wird, wenn Daten vom Gerät gelesen werden sollen

```
.ReadCallback = AddressOf ReadCallback
```

```
}
```

```
YMcom.StartAsync()
```

Wird aufgerufen, wenn sich ein Gerät verbindet

```
Private Function DeviceConnectedCallback(device As Device) As Task
```

```
Debug.WriteLine("****Device Connected: " & device.Adresse)
```

```
Return Task.FromResult(Of Integer)(0)
```

```
End Function
```

Wird aufgerufen, wenn ein Gerät die Verbindung trennt

```
Private Function DeviceDisconnectedCallback(device As Device) As Task
```

```
Debug.WriteLine("****Device Disconnected: " & device.Adresse)
```

```
Return Task.FromResult(Of Integer)(0)
```

```
End Function
```

Wird aufgerufen, wenn sich der Status eines Geräts ändert, verbunden oder getrennt

Parameter connectedDevices: Geräte, die sich kürzlich verbunden haben

Parameter disconnectedDevices: Geräte, die kürzlich getrennt wurden

Parameter currentConnectedDevices: alle Geräte die aktuell verbunden sind

```
Private Function DevicesChangedCallback(connectedDevices As List(Of Device),
```

```
disconnectedDevices As List(Of Device), currentConnectedDevices As List(Of
```

```
Device)) As Task
```

```
.
```

```
.
```

```
.
```

```
End Function
```

Schaltzustand an das Modul senden

Je nach welches Modul angesprochen werden soll, muss für das Senden, die richtige Methode ausgewählt werden.

Relais Module

USB Modul mit 4 Ausgänge

SEND_OUTPUT_STATUS_4(adresse, DataWert1)

USB Modul mit 8 Ausgänge

SEND_OUTPUT_STATUS_8(adresse, DataWert1)

USB Modul mit 16 Ausgänge

SEND_OUTPUT_STATUS_16(adresse, DataWert1, DataWert2)

USB Modul mit 32 Ausgänge

SEND_OUTPUT_STATUS_32(adresse, DataWert1, DataWert2, DataWert3, DataWert4)

USB Modul mit 64 Ausgänge

SEND_OUTPUT_STATUS_64(adresse, DataWert1, DataWert2, DataWert3, DataWert4, DataWert5, DataWert6, DataWert7, DataWert8)

Hybrid Module

USB Modul mit 8 Ausgänge und 8 Eingänge

SEND_HYBRID_STATUS_8(adresse, DataWert1)

USB Modul mit 16 Ausgänge und 16 Eingänge

SEND_HYBRID_STATUS_16(adresse, DataWert1, DataWert2)

USB Modul mit 32 Ausgänge und 32 Eingänge

SEND_HYBRID_STATUS_32(adresse, DataWert1, DataWert2, DataWert3, DataWert4)

Parameter adresse

Gibt das Modul an, welches angesprochen werden soll. Device_adresse lässt sich an dem DIP Schalter am Modul einstellen.

adresse = 0 → 1. Modul

adresse = 1 → 2. Modul

Bis zu 8 USB Module können an einem System angeschlossen werden.

Parameter DataWert

Die Ausgangszustände des Moduls werden ständig gelesen und in der Variable `Dim SendeByte1 As Byte` festgehalten. Zum Schalten der Ausgänge muss daher nur geschrieben werden was ein- bzw. ausgeschaltet werden soll.

z.B. Variablenwert `SendeByte1=77` bedeutet, dass die Ausgänge 1,3,4, und die 7 eingeschaltet sind. Möchte man jetzt nur den Ausgang 3 ausschalten, so ist aus dem Variablenwert 77 die 4 zu subtrahieren und der Wert 73 wird an das Modul gesendet.