

PTC04-LIN PRODUCT SPECIFIC FUNCTIONS

SOFTWARE LIBRARY

1 Contents

1	CONTENTS	2
2	INTRODUCTION	4
3	SOFTWARE STRUCTURE.....	4
3.1	Object Model	4
3.2	Object Structure and Hierarchy	4
3.3	Objects with Interfaces	5
4	PSFPTCLINAAMLXMANAGER OBJECT	6
4.1	Background.....	6
5	PSFPTCLINAAMLXDEVICE OBJECT.....	7
5.1	Background.....	7
5.2	Logging Property	8
5.3	PTC04 Property	9
5.4	Exchange Method	9
5.5	_Exchange Method	11
5.6	GetLinBusState Method.....	12
5.7	GetSetting Method.....	12
5.8	OpenProfile Method	13
5.9	SaveProfile Method	13
5.10	SaveProfileAs Method	14
5.11	SendWakeUpPulse Method	14
5.12	SetSetting Method.....	15
5.13	ValidateId Method	16
5.14	SetVbatConnected Method	16
5.15	SetVbatLevel Method	17
6	MELEXCM LOADER METHODS OF PSFPTCLINAAMLXDEVICE.....	17
6.1	How to upload a HEX file	18
6.2	How to download a HEX file.....	18
6.3	MlxcMUploadHexFile Method	18
6.4	MlxcMVerifyHexFile Method	19
6.5	MlxcMDownloadHexFile Method	20
6.6	MlxcMUpload Method.....	21
6.7	MlxcMDownload Method.....	21
6.8	ConfigureMlxcMLoader Method	22
6.9	ConfigureLinLoader Method	23
6.10	ConfigureFastLoader Method	24
6.11	MlxcMAbortLoader Method	24
6.12	MlxcMSaveHexFileInXram Method.....	25
6.13	MlxcMUploadFromXram Method	26
6.14	MlxcMVerifyWithXram Method	27
6.15	LoaderLog Property	27
6.16	MlxcMNvramUploadHexFile Method	28
6.17	MlxcMNvramDownloadHexFile Method	28
6.18	MlxcMNvramUpload Method.....	29
6.19	MlxcMNvramDownload Method.....	30
7	ENUMERATION CONSTANTS	31
7.1	SettingCodes enumeration	31
7.2	LoaderProtocolCodes enumeration	32

7.3	NvramOptionCodes enumeration	32
8	DISCLAIMER	33

2 Introduction

MLXPTCLIN PSF is MS Windows software library, which meets the requirements for a Product Specific Functions (PSF) module, defined in Melexis Programmable Toolbox (MPT) object model. The library implements in-process COM objects for interaction with MLXPTCLIN firmware. It is designed primarily to be used by MPT Framework application, but also can be loaded as a standalone in-process COM server by other applications that need to communicate with the above-mentioned Melexis hardware.

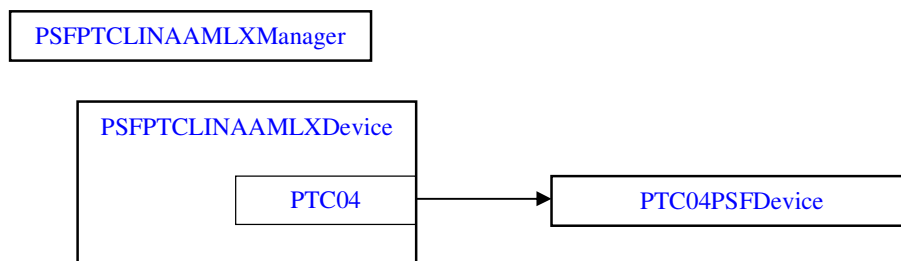
3 Software Structure

3.1 Object Model

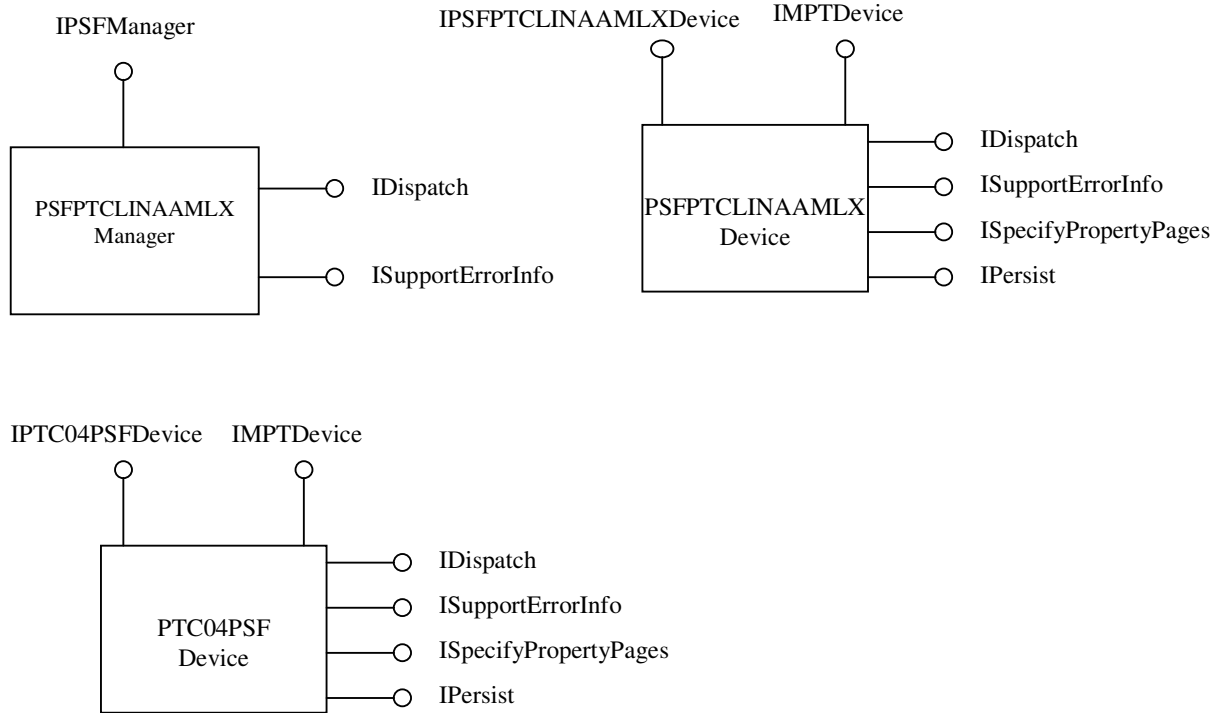
MPT object model specifies that a PSF module must expose two COM objects which implement certain COM interfaces. MLXPTCLIN PSF implements these two objects and two additional objects for advanced operations.

- **PSFPTCLINAAMLXManager object** – implements IPSFManager standard MPT interface. This is a standard PSFManager object. MPT Framework and other client applications create a temporary instance of that object, just for device scanning procedure. After that this instance is released. This is the first required object. Refer to MPT Developer Reference document for more information about PSFManager object and IPSFManager interface.
- **PSFPTCLINAAMLXDevice object** – implements IPSFPTCLINAAMLXDevice specific interface. However, this interface derives from IMPTDevice standard MPT interface and therefore PSFPTCLINAAMLXDevice also implements the functionality of MPTDevice standard MPT object. In addition to standard IMPTDevice methods, IPSFPTCLINAAMLXDevice interface exposes methods, which are specific to this library. They are described in this document. This is the second required COM object. Refer to MPT Developer Reference document for more information about MPTDevice object and IMPTDevice interface.

3.2 Object Structure and Hierarchy



3.3 Objects with Interfaces



4 PSFPTCLINAAMLXManager Object

4.1 Background

This object is created only once and is destroyed when the library is unmapped from process address space. Each subsequent request for this object returns the same instance.

PSFPTCLINAAMLXManager object implements standard MPT category **CATID_MLXMPTPSFSerialModule**, which is required for automatic device scanning. C++ standalone client applications can create an instance of this object by using the standard COM API CoCreateInstance with class ID **CLSID_PSFPTCLINAAMLXManager**, or ProgID “**MPT.PSFPTCLINAAMLXManager**”:

```
hRes = ::CoCreateInstance(CLSID_PSFPTCLINAAMLXManager, NULL,  
CLSCCTX_INPROC, IID_IPSFManager, (void**) &pPSFMan);
```

Visual Basic applications should call CreateObject function to instantiate PSFPTCLINAAMLXManager:

```
Set PSFMan = CreateObject(“MPT.PSFPTCLINAAMLXManager”)
```

The primary objective of this instantiation is to call ScanStandalone method. C++:

```
hRes = pPSFMan->ScanStandalone(dtSerial, varDevices, &pDevArray);
```

Or in Visual Basic:

```
Set DevArray = PSFMan.ScanStandalone(dtSerial)
```

ScanStandalone function returns collection of PSFPTCLINAAMLXDevice objects, one for each connected PTC04. The collection is empty if there are no connected devices.

5 PSFPTCLINAAMLXDevice Object

5.1 Background

This object implements standard MPT category **CATID_MLXMPTPSFSerialDevice** as well as library specific **CATID_MLXMPTLINPTC04Device** category. It also declares required specific category **CATID_MLXMPTLINPTC04UIModule** for identification of required user interface modules.

This object can be created directly with `CoCreateInstance/GetObject` or by calling the device scanning procedure `ScanStandalone` of **PSFPTCLINAAMLXManager** object. The following Visual Basic subroutine shows how to instantiate **PSFPTCLINAAMLXDevice** object by performing device scan on the system:

```
Sub CreateDevice()
    Dim PSFMan As PSFPTCLINAAMLXManager, DevicesCol As ObjectCollection, I As Long
    On Error GoTo IError

    Set PSFMan = CreateObject("MPT.PSFPTCLINAAMLXManager")
    Set DevicesCol = PSFMan.ScanStandalone(dtSerial)
    If DevicesCol.Count <= 0 Then
        MsgBox ("No PTC-04 programmers found!")
        Exit Sub
    End If

    ' Dev is a global variable of type PSFPTCLINAAMLXDevice
    ' Select first device from the collection
    Set Dev = DevicesCol(0)
    MsgBox (Dev.Name & " device found on " & Dev.Channel.Name)
    If DevicesCol.Count > 1 Then
        For I = 1 To DevicesCol.Count - 1
            ' We are responsible to call Destroy(True) on the device objects we do not need
            Call DevicesCol(I).Destroy(True)
        Next I
    End If
    Exit Sub

IError:
    MsgBox Err.Description
    Err.Clear
End Sub
```

Developers can also manually connect the device object to a serial channel object thus bypassing standard device scanning procedure. The following Visual Basic subroutine allows manual connection along with standard device scanning depending on input parameter `bAutomatic`:

```
Sub CreateDevice(bAutomatic As Boolean)
    Dim PSFMan As PSFPTCLINAAMLXManager, DevicesCol As ObjectCollection, I As Long
    Dim CommMan As CommManager, Chan As MPTChannel
    On Error GoTo IError

    If bAutomatic Then
        ' Automatic device scanning begins here
        Set PSFMan = CreateObject("MPT.PSFPTCLINAAMLXManager")
        Set DevicesCol = PSFMan.ScanStandalone(dtSerial)
        If DevicesCol.Count <= 0 Then
            MsgBox ("No PTC-04 programmers found!")
            Exit Sub
        End If

        If DevicesCol.Count > 1 Then
            For I = 1 To DevicesCol.Count - 1
                'We are responsible to call Destroy(True) on device objects we do not need
                Call DevicesCol(I).Destroy(True)
            Next I
        End If
        Set MyDev = DevicesCol(0)
    Else
        IError:
        MsgBox Err.Description
        Err.Clear
    End Sub
```

```
' Manual connection begins here
Set CommMan = CreateObject("MPT.CommManager")
Set MyDev = CreateObject("MPT.PSFPTCLINAAMLXDevice")
I = ActiveWorkbook.Names("SerialPort").RefersToRange.Value2
Set Chan = CommMan.Channels.CreateChannel(CVar(I), ctSerial)
MyDev.Channel = Chan
' Check if a PTC04 programmer is connected to this channel
Call MyDev.CheckSetup(False)
End If
MsgBox (MyDev.Name & " programmer found on " & MyDev.Channel.Name)
Exit Sub
```

```
IError:
MsgBox Err.Description
Err.Clear
End Sub
```

PSFPTCLINAAMLXDevice object implements IMPTDevice standard MPT interface. Please refer to MPT Developer reference document for description of the properties and methods of this interface.

In addition PSFPTCLINAAMLXDevice object implements IPSFPTCLINAAMLXDevice library specific interface, which derives from IMPTDevice. The following is a description of its properties and methods.

5.2 **Logging Property**

Specifies whether logging information is generated while working with the library.

Syntax

Visual Basic:
Property Logging as Boolean

C++:
HRESULT get_Logging([out,retval] VARIANT_BOOL* pValue);
HRESULT set_Logging([in] VARIANT_BOOL Value);

Parameters

pValue
An address of **VARIANT_BOOL** variable that receives current value of the property. **VARIANT_TRUE** means that logging is active, **VARIANT_FALSE** means inactive.

Value
A **VARIANT_BOOL** specifying new value for the property. **VARIANT_TRUE** activates the logging, **VARIANT_FALSE** deactivates it.

Return value

Visual Basic:
True if logging is active, **False** otherwise.

C++:
The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pValue contains valid value.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.3 **PTC04 Property**

This property holds a reference to PTC04PSFDevice co-object.

Syntax

Visual Basic:

Property PTC04 as Object

C++:

```
HRESULT get_PTC04([out][retval] LPDISPATCH* pVal);
HRESULT set_PTC04([in] LPDISPATCH Value);
```

Parameters

Value

An **IDispatch*** specifying new PTC04 device object. Nothing happens if the object is the same instance as the existing one. Otherwise PSFPTCLINAAMLXDevice object releases its current PTC04 device object and connects to the new one. This also includes replacing of the communication Channel object with the one from the new PTC04PSFDevice object.

pVal

Address of **IDispatch*** pointer variable that receives the interface pointer to the PTC04 device object. If the invocation succeeds, the caller is responsible for calling **IUnknown::Release()** on the pointer when it is no longer needed.

Return value

Visual Basic:

A reference to the PTC04PSFDevice co-object.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pVal contains valid pointer.
Any other error code	The operation failed. *pVal contains NULL.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.4 **Exchange Method**

Sends a frame to and receives a response from the LIN bus.

Syntax

Visual Basic:

Function Exchange(Buffer, bParameter As Byte, isCRC2 As Boolean, pbtDiag As Byte, ResponseTimeout As Long)

C++:

```
HRESULT Exchange([in] VARIANT Buffer, [in] unsigned char bParameter, [in]
VARIANT_BOOL isCRC2, [out] unsigned char* pbtDiag, [in, defaultvalue(0)] long
ResponseTimeout, [out, retval] VARIANT * pvResult );
```

Parameters

Buffer

A **VARIANT** value containing an array of bytes to be send.

bParameter

A Byte providing several parameters, defining the frame to be sent:

Parameter byte							
D7	D6	D5	D4	D3	D2	D1	D0
Breaklength	Reserved			M2S	Number of bytes in S2M message		
1 = send break of 18 bittimes 0 = send break of 36 bittimes				1 = M2S message 0 = S2M message	M2S = 0: Number of bytes that a slave will send M2S = 1: Data is ignored		

isCRC2

A flag showing the type of the checksum:

TRUE – the frame_id will be included in the checksum.

FALSE – the frame_id won't be included in the checksum.

This parameter also affects the kind of the checksum over the received frame.

pbtDiag

An address of **Byte** variable, which will receive the diagnostic byte returned from PTC04. This byte indicates if an error has occurred:

D7	D6	D5	D4	D3	D2	D1	D0
Vmaster	-	-	-	Error message			
0 = master supply OK 1 = no master supply				0x00: no error 0x01: no slave answer on an S2M message 0x02: busy, a message is travelling over the bus			

ResponseTimeout

A **long** specifying the time (in ms) to wait the PTC04 to answer a single command. In case this value is 0 (default), this parameter will be set to the value of **PTC04.ResponseTimeout** property.

pvResult

An address of **VARIANT** variable that gets received frame as array of bytes.

Return value

Visual Basic:

A **VARIANT** containing an array of bytes representing received frame.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pvResult contains a valid value.
Any other error code	The operation failed. * pvResult is empty.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.5 Exchange Method

Sends a frame to and receives a response from the LIN bus. This method couldn't be called from Visual Basic.

Syntax

C++:
HRESULT _Exchange(*/*[in]*/ unsigned char * pSend, /*[in,out]*/ unsigned char * pbLength, /*[in]*/ unsigned char bParameter, /*[in]*/ unsigned char isCRC2, /*[out]*/ unsigned char* pbDiag, /*[in, defaultvalue(0)]*/ long lResponseTimeout, /*[out, retval]*/ unsigned char ** ppRecv);*

Parameters

pSend

An array of bytes to be send.

pbLength

An address of **Byte** variable containing the length of the **pSend** array. After the method execution the variable will be set to the length of ***ppRecv** array.

bParameter

A Byte providing several parameters, defining the frame to be sent. See [Exchange](#) method description for details.

isCRC2

A flag showing the type of the checksum:

non zero – the frame_id will be included in the checksum.

0 – the frame_id won't be included in the checksum.

This parameter also affects the kind of the checksum over the received frame.

pbDiag

An address of **Byte** variable, which will receive the diagnostic byte returned from the PTC04. This byte indicates if an error has occurred. See [Exchange](#) method description for details.

ResponseTimeout

A **long** specifying the time (in ms) to wait the PTC04 to answer a single command. In case this value is 0 (default), this parameter will be set to the value of **PTC04.ResponseTimeout** property.

ppRecv

An address of array of bytes that gets received frame. Before calling this method, the address must contain NULL. After successful execution, the address will contain a valid address of the returned array. The caller is responsible to free the used memory by calling **LocalFree** API function.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *ppRecv and *pbLength contain a valid value.
Any other error code	The operation failed. *ppRecv and *pbLength contain zero.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.6 **GetLinBusState Method**

Returns a byte representing current state of the LIN bus. This state can be one of the following:

State value	Explanation
1	Normal state: the LIN bus is in normal operation, the bus is recessive and ready to transport messages
2	Wake up: At least one wake up pulse is seen since the previous state request
3	Dominant: The bus is stuck at the dominant level for more that the maximal wake up period
4	Transceiving: LIN messages are travelling on the bus
5	Sending wake up: the master is sending a wake up pulse to the slaves

Syntax

Visual Basic:

Function GetLinBusState() As Byte

C++:

HRESULT GetLinBusState([out, retval] unsigned char * pVal);

Parameters

pVal

An address of a Byte which will get the status information.

Return value

Visual Basic:

A Byte containing status information received from PTC04.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pVal contains a valid value.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.7 **GetSetting Method**

Returns the value of a particular setting.

Syntax

Visual Basic:

Function GetSetting(settingID as SettingCodes)

C++:

HRESULT GetSetting([in] SettingCodes settingID, [out,retval] TVariant* pVal);

Parameters

settingID

A [SettingCodes](#) constant specifying the ID of the setting.

pVal

An address of **VARIANT** variable that will receive the return value of the method. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:

A **Variant** containing the value of a setting.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pVal contains valid value.
Any other error code	The operation failed. *pVal is Empty .

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.8 OpenProfile Method

Opens the specified file and updates the settings.

Syntax

Visual Basic:

Sub OpenProfile(FileName as String)

C++:

HRESULT OpenProfile([in] BSTR FileName);

Parameters

FileName

A **String** specifying the path of the file to open.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.9 SaveProfile Method

Saves the settings into a previously opened profile. This function fails if there is not a profile in use.

Syntax

Visual Basic:

Sub SaveProfile()

C++:

HRESULT SaveProfile();

Parameters

none

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.10 SaveProfileAs Method

Saves the settings into the specified file.

Syntax

Visual Basic:

Sub SaveProfileAs(FileName as String)

C++:

HRESULT SaveProfileAs([in] BSTR FileName);

Parameters

FileName

A **String** specifying the path of the file.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.11 SendWakeUpPulse Method

This method sends a wake-up pulse to the slaves.

Syntax

Visual Basic:

Sub SendWakeUpPulse()

C++:

HRESULT SendWakeUpPulse ();

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.12 SetSetting Method

Changes the value of a particular setting. Sets an associated internal variable. The setting is also sent immediately to the firmware.

NOTE: If necessary, the changes can be saved in the profile with a subsequent call to [SaveProfile](#) or [SaveProfileAs](#) methods.

Syntax

Visual Basic:

Sub SetSetting(settingID as SettingCodes, Value)

C++:

HRESULT SetSetting([in] SettingCodes settingID, [in] TVariantInParam Value);

Parameters

settingID

A [SettingCodes](#) constant specifying the ID of the setting to modify.

Value

A VARIANT containing new value for the setting.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.13 **ValidateId Method**

Validates an identifier byte according to the LIN specification. Input byte must be in the range from 0 to 63.

Syntax

Visual Basic:

Function ValidateId(bId As Byte) As Byte

C++:

HRESULT ValidateId ([in] unsigned char bId, [out, retval] unsigned char * pbNewId);

Parameters

bId

A **Byte** containing an identifier to be validated.

pbNewId

An address of the result byte.

Return value

Visual Basic:

A **Byte** containing validated identifier.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully. *pbNewId contains a valid value.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.14 **SetVbatConnected Method**

This method connects or disconnects a programmable power supply to Vbat line. The level of the PPS can be set by [SetVbatLevel](#) method. The order of calling both methods is not important.

Syntax

Visual Basic:

Sub SetVbatConnected(Status As Boolean)

C++:

HRESULT SetVbatConnected([in] VARIANT_BOOL Status);

Parameters

Status

A **Boolean** specifying what to be done – True to connect or False to disconnect PPS and Vbat.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

5.15 *SetVbatLevel Method*

This method sets the level of the PPS, which can be connected to Vbat line. The doesn't always set directly the Vbat line. Connection or disconnection to Vbat is controlled by [SetVbatConnected](#) method. The order of calling both methods is not important.

Syntax

Visual Basic:

Sub SetVbatLevel(Voltage As Single)

C++:

HRESULT SetVbatLevel([in] float Voltage);

Parameters

Voltage

A **Single** voltage level to set on the PPS for Vbat line.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6 MelexCM loader methods of PSFPTCLINAAMLXDevice

These methods are members of **PSFPTCLINAAMLXDevice** object and are separated only for clarity. Their aim is to upload and download memory of MelexCM slave device, connected to LIN bus.

Prior to calling any Download, Update or Verify method it's necessary to have configured the corresponding protocol(s) and the loader. The methods for that are:

[ConfigureLinLoader](#) - must be called when LIN or Fast protocol is used

[ConfigureFastLoader](#) - must be called when Fast, FastStandalone or SlowStandalone protocol is used

[ConfigureMlxcmLoader](#) – always

Also for the LIN or Fast protocols it's necessary to set **SettingLinSpeed** setting of the MLXLINPSFDevice object to the required baudrate in LIN mode.

These configurations have to be done once and can be used for many subsequent loader method calls.

Another possibility is to use several settings instead of ConfigureXXX method calls. Related setting enumeration constants are **SettingLoaderProtocol**, **SettingLoaderFilename**, **SettingLinSpeed**, **SettingLoaderLinNad** and **SettingFastLoaderBaudrate**.

6.1 How to upload a HEX file

The following Visual Basic code marks the necessary steps to pass in order to upload a file.

```
' MyDev is an object of PSFPTCLINAAMLXDevice
' Prior to running this procedure MyDev must be created and connected to PTC04 (see p.5.1)

' Configure LIN network address
MyDev.ConfigureLinLoader Parameter:=0, NAD:=127
' Configure LIN baudrate
MyDev.SetSetting settingId:=SettingLinSpeed, Value:=9600

' Configure Fast protocol baudrate
MyDev.ConfigureFastLoader Parameter:=0, BaudRate:=100000

' Configure the protocol and specify the path to Melexis loader code HEX file
MyDev.ConfigureMlxcmLoader Protocol:=LoaderProtocolFast, LoaderFilename:="loader.hex"

' Upload the file firmware.hex (full path must be specified); do not sink progress events
MyDev.MlxcmUploadHexFile Filename:="firmware.hex", Progress:=Nothing, Hint:=Empty

' Compare the file firmware.hex(full path must be specified) with the device's memory
' do not sink progress events
MyDev.MlxcmVerifyHexFile Filename:="firmware.hex", Progress:=Nothing, Hint:=Empty
```

6.2 How to download a HEX file

The following Visual Basic code marks the necessary steps to pass in order to download full Flash contents to a HEX file.

```
' MyDev is an object of PSFPTCLINAAMLXDevice
' Prior to running this procedure MyDev must be created and connected to PTC04 (see p.5.1)

' Configure LIN node address (NAD); Parameter is reserved (not used)
MyDev.ConfigureLinLoader Parameter:=0, NAD:=127
' Configure LIN baudrate
MyDev.SetSetting settingId:=SettingLinSpeed, Value:=9600

' Configure Fast protocol baudrate; Parameter is reserved (not used)
MyDev.ConfigureFastLoader Parameter:=0, BaudRate:=100000

' Configure the protocol and specify the path to Melexis loader code HEX file
MyDev.ConfigureMlxcmLoader Protocol:=LoaderProtocolFast, LoaderFilename:="loader.hex"

' Download area [0; 0x8000) to file downloaded.hex (full path must be specified); do not sink progress events
MyDev.MlxcmDownloadHexFile Filename:="downloaded.hex", StartAddr:=0, Len:=32768,
Progress:=Nothing, Hint:=Empty
```

6.3 MlxcmUploadHexFile Method

This method uploads data from a .HEX file into a MelexCM device.

Syntax

Visual Basic:

Sub MlxcmUploadHexFile(Filename as String, Progress as Object, Hint as Variant)

C++:

HRESULT MlxcmUploadHexFile([in] BSTR Filename, [in] LPDISPATCH Progress, [in] TVariantInParam Hint);

Parameters

Filename

A **String** specifying the full path to .HEX file.

Progress

Object that implements IMPTProgressCallback interface. It should have implementation of methods OnStart, OnProgress and OnEnd. Nothing (NULL) can be passed if the callback is not needed.

Hint

A **Variant** that is sent back to callback object as parameter in Onxxx methods.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.4 MlxcmVerifyHexFile Method

This method compares data from a .HEX file with that into a MelexCM device. Note that the range [0x2000; 0x2400) is not verified.

Syntax

Visual Basic:

Sub MlxcmVerifyHexFile (Filename as String, Progress as Object, Hint as Variant, [StartAddr As Long], [EndAddr As Long = &H7FFFFFFF&])

C++:

HRESULT MlxcmVerifyHexFile ([in] BSTR Filename, [in] LPDISPATCH Progress, [in] TVariantInParam Hint, [in,defaultvalue(0)] long StartAddr, [in,defaultvalue(0x7FFFFFFF)] long EndAddr);

Parameters

Filename

A **String** specifying the full path to .HEX file.

Progress

Object that implements IMPTProgressCallback interface. It should have implementation of methods OnStart, OnProgress and OnEnd. Nothing (NULL) can be passed if the callback is not needed.

Hint

A **Variant** that is sent back to callback object as parameter in Onxxx methods.

StartAddr

A **Long** value, specifying the start address (**inclusive**) of the region to be uploaded. The data from the HEX file, which is outside the region, will be ignored.

EndAddr

A **Long** value, specifying the end address (**exclusive**) of the region to be uploaded. The data from the HEX file, which is outside the region, will be ignored.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.5 *MlxcmDownloadHexFile Method*

This method reads data from certain area of a MelexCM device and saves it into a .HEX file.

Syntax

Visual Basic:

Sub MlxcmDownloadHexFile (Filename as String, StartAddr as Long, EndAddr as Long, Progress as Object, Hint as Variant)

C++:

HRESULT MlxcmDownloadHexFile ([in] BSTR Filename, [in] long StartAddr, [in] long EndAddr, [in] LPDISPATCH Progress, [in] TVariantInParam Hint);

Parameters

Filename

A **String** specifying the full path to .HEX file.

StartAddr

A **Long** specifying the start address (**inclusive**) of memory area to be read.

EndAddr

A **Long** specifying the end address (**exclusive**) of the area to be read.

Progress

Object that implements IMPTProgressCallback interface. It should have implementation of methods OnStart, OnProgress and OnEnd. Nothing (NULL) can be passed if the callback is not needed.

Hint

A **Variant** that is sent back to callback object as parameter in Onxxx methods.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.6 *MlxcmUpload Method*

This method uploads data into a MelexCM device.

Syntax

Visual Basic:

Sub MlxcmUpload (StartAddr as Long, Format as Long, Data as Variant)

C++:

HRESULT MlxcmUpload([in] long StartAddr, [in] long Format, [in] TVariantInParam Data);

Parameters

StartAddr

A **Long** specifying the start address of memory area to be uploaded.

Format

A **Long** specifying the format of the data in Data. Possible values are:

Value	Format
1	Data is an array of bytes. This is the preferred format for Visual Basic applications.
2	Data is an ANSI string packed in bstrVal member of *pvarID. This is the preferred format for C++ applications because of the best performance. It is a binary data so the string can contain zeroes and may not be zero terminated. Callers can get its real length by calling SysStringByteLen API on bstrVal member.

Data

A **Variant** containing the data to be uploaded.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.7 *MlxcmDownload Method*

This method reads data from a MelexCM device.

Syntax

Visual Basic:

Function MlxcMDownload (StartAddr as Long, EndAddr as Long, Format as Long) as Variant

C++:

HRESULT MlxcMDownload ([in] long StartAddr, [in] long EndAddr, [in] long Format, [out,retval] TVariant* pData);

Parameters

StartAddr

A **Long** specifying the start address (**inclusive**) of the memory area to be read.

EndAddr

A **Long** specifying the end address (**exclusive**) of the memory area to be read.

Format

A **Long** specifying the format of the returned data in pData. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications.
2	Return value is an ANSI string packed in bstrVal member of *pvarID. This is the preferred format for C++ applications because of the best performance. It is a binary data so the string can contain zeroes and may not be zero terminated. Callers can get its real length by calling SysStringByteLen API on bstrVal member.

pData

An address of **VARIANT** variable that will receive the data read from the device. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:

A **Variant** containing the data read from the device.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.8 ConfigureMlxcMLoader Method

This method configures parameters used in uploading and downloading functions.

Note: These configuration parameters can be get/set via [GetSetting](#) and [SetSetting](#) methods, and specifying **SettingLoaderProtocol** or **SettingLoaderFilename** enumeration constants. This allows them to be loaded from and stored to a file profile using the corresponding methods.

Syntax

Visual Basic:

Sub ConfigureMlxcmLoader (Protocol as LoaderProtocolCodes, LoaderFilename as String)

C++:

HRESULT ConfigureMlxcmLoader ([in] LoaderProtocolCodes Protocol,
[in] BSTR LoaderFilename);

Parameters

Protocol

A [LoaderProtocolCodes](#) constant specifying the protocol to be used while uploading and downloading. By default library uses Fast protocol.

LoaderFilename

A **String** specifying the full path to .HEX file, containing the code of the loader program. This parameter must be set once before uploading.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.9 ConfigureLinLoader Method

This method configures parameters relevant to LIN protocol used for uploading and downloading.

Note: NAD parameter can be get/set via [GetSetting](#) and [SetSetting](#) methods and specifying **SettingLoaderLinNad** enumeration constant. This allows it to be loaded from and stored to a file profile using the corresponding methods.

Syntax

Visual Basic:

Sub ConfigureLinLoader (Parameter as Long, NAD as Byte)

C++:

HRESULT ConfigureLinLoader ([in] long Parameter, [in] unsigned char NAD);

Parameters

Parameter

Reserved.

NAD

A **Byte** specifying NAD of the LIN node which will be used for uploading or downloading memory.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.10 *ConfigureFastLoader Method*

This method configures parameters relevant to fast protocol used for uploading and downloading.

Note: Fast loader baudrate can be get/set via [GetSetting](#) and [SetSetting](#) methods and specifying **SettingFastLoaderBaudrate** enumeration constant. This allows it to be loaded from and stored to a file profile using the corresponding methods.

Syntax

Visual Basic:

Sub ConfigureFastLoader (Parameter as Long, Baudrate as Long)

C++:

HRESULT ConfigureFastLoader ([in] long Parameter, [in] long Baudrate);

Parameters

Parameter

Reserved.

Baudrate

A **Long** specifying the required baudrate for fast protocol. Possible values are between 5000 and 200000 [bps]. The default value is 125000 [bps].

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.11 *MlxcmAbortLoader Method*

This method stops uploading or downloading process.

Syntax

Visual Basic:

Sub MlxcMAbortLoader()

C++:
HRESULT MlxcMAbortLoader ();

Parameters

none

Return value

C++:
The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.12 MlxcMSaveHexFileInXram Method

This method uploads data from a .HEX file into the PTC04's XRAM. Then the methods MlxcMUploadFromXram and MlxcMVerifyWithXram can be used to perform operations with a MelexCM device.

Syntax

Visual Basic:

Sub MlxcMSaveHexFileInXram(Filename As String, Progress As Object, Hint as Variant, ByRef LoaderPtr As Long, ByRef MlxPtr As Long, ByRef UserPtr As Long, [StartAddr As Long = 0], [EndAddr As Long = &H7FFFFFFF])

C++:

HRESULT MlxcMSaveHexFileInXram ([in] BSTR Filename, [in] LPDISPATCH Progress, [in] TVariantInParam Hint, [out] long* LoaderPtr, [out] long* MlxPtr, [out] long* UserPtr, [in,defaultvalue(0)] long StartAddr, [in,defaultvalue(0x7FFFFFFF)] long EndAddr);

Parameters

Filename

A **String** specifying the full path to .HEX file.

Progress

Object that implements IMPTProgressCallback interface. It should have implementation of methods OnStart, OnProgress and OnEnd. Nothing (NULL) can be passed if the callback is not needed.

Hint

A **Variant** that is sent back to callback object as parameter in Onxxx methods.

LoaderPtr

MlxPtr

UserPtr

An address of a **Long** variable that will receive the XRAM starting address of the corresponding code part. These variables are required later in **MlxcmUploadFromXram** and **MlxcmVerifyWithXram** methods.

StartAddr

A **Long** value, specifying the start address (**inclusive**) of the region to be uploaded. The data from the HEX file, which is outside the region, will be ignored.

EndAddr

A **Long** value, specifying the end address (**exclusive**) of the region to be uploaded. The data from the HEX file, which is outside the region, will be ignored.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.13 MlxcmUploadFromXram Method

This method uploads the data stored in PTC04's XRAM into a MelexCM device.

Syntax

Visual Basic:

Sub MlxcmUploadFromXram(LoaderPtr As Long, MlxPtr As Long, UserPtr As Long)

C++:

HRESULT MlxcmUploadFromXram ([in] long LoaderPtr, [in] long MlxPtr, [in] long UserPtr);

Parameters

LoaderPtr

MlxPtr

UserPtr

A **Long** value, specifying the start address of the XRAM where the data is saved. These values should be taken as a result from **MlxcmSaveHexFileInXram** method.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.14 **MlxcMVerifyWithXram Method**

This method compares the data stored in PTC04's XRAM with that into a MelexCM device. Note that the range [0x2000; 0x2400) is not verified.

Syntax

Visual Basic:

Sub MlxcMVerifyWithXram (Mem1Ptr As Long, Mem2Ptr As Long)

C++:

HRESULT MlxcMVerifyWithXram ([in] long Mem1Ptr, [in] long Mem2Ptr);

Parameters

Mem1Ptr

Mem2Ptr

A **Long** value, specifying the start address of the XRAM where the data is saved. These values should be taken as a result from **MlxcMSaveHexFileInXram** method and normally are the MlxPtr and UserPtr.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.15 **LoaderLog Property**

This is a boolean property, which enables or disables logging of MelexCM loader commands. The default value for this property is **False**.

Syntax

Visual Basic:

Property LoaderLog as Boolean

C++:

HRESULT get_LoaderLog(/*[out][retval]*/ VARIANT_BOOL * pValue);
HRESULT set_LoaderLog(/*[in]*/ VARIANT_BOOL Value);

Parameters

pValue

An address of **Boolean** variable that receives current value of the property.

Value

A **Boolean** specifying new value for the property.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.16 *MlxcmNvramUploadHexFile Method*

This method uploads NVRAM data from a .HEX file into a MelexCM device. The valid address range, specified in the HEX file is within 0 to 127. All data outside that region will be ignored.

Syntax

Visual Basic:

Sub MlxcmNvramUploadHexFile(Filename as String, Options as NvramOptionCodes)

C++:

HRESULT MlxcmNvramUploadHexFile([in] BSTR Filename,
[in] NvramOptionCodes Options);

Parameters

Filename

A **String** specifying the full path to .HEX file.

Options

A combination (sum or bitwise OR) of [NvramOptionCodes](#), setting the required options.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.17 *MlxcmNvramDownloadHexFile Method*

This method reads data from NVRAM of a MelexCM device and saves it into a .HEX file.

Syntax

Visual Basic:

Function MlxcmNvramDownloadHexFile(Filename as String) as Boolean

C++:

HRESULT MlxcnNvramDownloadHexFile([in] BSTR Filename,
 [out,retval] VARIANT_BOOL* pRes);

Parameters

Filename

A **String** specifying the full path to .HEX file.

pRes

An address of **VARIANT_BOOL** variable, receiving the result of NVRAM validity check. If the checksum of data is correct, the result is **true** otherwise it is **false**.

Return value

Visual Basic:

A Boolean value, containing the result of NVRAM validity check. If the checksum of data is correct, the result is **true** otherwise it is **false**.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.18 MlxcnNvramUpload Method

This method uploads NVRAM data into a MelexCM device.

Syntax

Visual Basic:

Sub MlxcnNvramUpload (Format as Long, Data as Variant, Options as NvramOptionCodes)

C++:

HRESULT MlxcnNvramUpload([in] long Format, [in] TVariantInParam Data, [in] NvramOptionCodes Options);

Parameters

Format

A **Long** specifying the format of the data in Data. Possible values are:

Value	Format
1	Data is an array of bytes. This is the preferred format for Visual Basic applications.
2	Data is an ANSI string packed in bstrVal member of *pvarID. This is the preferred format for C++ applications because of the best performance. It is a binary data so the string can contain zeroes and may not be zero terminated. Callers can get its real length by calling SysStringByteLen API on bstrVal member.

Data

A **Variant** containing the data to be uploaded. Only the first 128 bytes are meaningful.

Options

A combination (sum or bitwise OR) of [NvramOptionCodes](#), setting the required options.

Return value

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

6.19 *MlxcmNvramDownload Method*

This method reads NVRAM data from a MelexCM device.

Syntax

Visual Basic:

Function MlxcmNvramDownload (Format as Long, Byref pData as Variant) as Boolean

C++:

HRESULT MlxcmNvramDownload [in] long Format, [out] TVariant* pData, [out,retval] VARIANT_BOOL* pRes);

Parameters

Format

A **Long** specifying the format of the returned data in pData. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications.
2	Return value is an ANSI string packed in bstrVal member of *pvarID. This is the preferred format for C++ applications because of the best performance. It is a binary data so the string can contain zeroes and may not be zero terminated. Callers can get its real length by calling SysStringByteLen API on bstrVal member.

pData

An address of **VARIANT** variable that will receive the data read from the device. The caller is responsible to call VariantClear on that variable when it is no longer needed.

pRes

An address of **VARIANT_BOOL** variable, receiving the result of NVRAM validity check. If the checksum of data is correct, the result is **true** otherwise it is **false**.

Return value

Visual Basic:

A Boolean value, containing the result of NVRAM validity check. If the checksum of data is correct, the result is **true** otherwise it is **false**.

C++:

The return value obtained from the returned HRESULT is one of the following:

Return value	Meaning
S_OK	The operation completed successfully.
Any other error code	The operation failed.

Quick Info

Header: Declared in PSFPTCLINAAMLXModule_TLB.h.

7 Enumeration constants

7.1 SettingCodes enumeration

The following constants specify different settings. They are used by [GetSetting](#) and [SetSetting](#) methods.

Constant	Value	Type	Default value	Description
SettingVbatGnd	1	float (Single)	0 [V]	Vbat ground voltage
SettingVbatNom	2	float (Single)	12.0 [V]	Vbat nominal voltage
SettingMeasureDelay	3	long (Long)	1000 [µs]	Delay before measurement
SettingMeasureFilter	4	short (Integer)	100	Measure filter
SettingLinSpeed	5	long (Long)	9600 [bps]	LIN protocol baudrate
SettingBusyTimeout	6	long (Long)	1000 [µs]	Timeout for getting a non-busy response from PTC04, while waiting a response from a slave node.
SettingLoaderProtocol	7	long (Long)	2 (Fast protocol)	Loader protocol (see Loader protocol codes)
SettingLoaderFilename	8	BSTR (String)	"LoaderA.hex"	The full name of the Intel HEX file, containing the loader code
SettingLoaderLinNad	9	unsigned char (Byte)	0x7F	The network address of the LIN node, which will be used for uploading or downloading
SettingFastLoaderBaudrate	10	long (Long)	125000 [bps]	Baudrate of the Fast protocol
SettingTFastToLin	11	long (Long)	40000 [us]	Delay before and after switching from Fast to LIN mode.
SettingTLinToFast	12	long (Long)	40000 [us]	Delay before and after switching from LIN to Fast mode.
SettingTLinSingleCmd	13	long (Long)	1000 [us]	Delay after each command in LIN mode.
SettingTFastSingleCmd	14	long (Long)	1000 [us]	Delay after each command in Fast mode.
SettingTWriteFlash	15	long (Long)	20000 [us]	Time to let the MelexCM write a flash page.
SettingTRestart	16	long (Long)	50000 [us]	Delay after restart command.
SettingTFastMultiFrame	17	long (Long)	100 [us]	Delay between the subframes within a long frame (valid for Fast mode).

7.2 **LoaderProtocolCodes enumeration**

The following constants specify encoding types of the protocol to be used when uploading and downloading memory. They are used in [ConfigureMlxcmLoader](#) method.

Constant	Value	Description
LoaderProtocolNone	0	Protocol not specified
LoaderProtocolLIN	1	Use LIN protocol (Default)
LoaderProtocolFast	2	Use Fast protocol (Default)
LoaderProtocolFastStandalone	3	For devices with Fast but w/o LIN support
LoaderProtocolSlowStandalone	4	For devices with PWM input

7.3 **NvramOptionCodes enumeration**

The following constants specify different options used by NVRAM upload functions ([MlxcmNvramUploadHexFile](#) and [MlxcmNvramUpload](#)).

Constant	Value	Description
nvrNone	0	None of below listed options is active
nvrResetCounter	1	The counter will be reset
nvrUpdateCrc	2	NVRAM checksum will be updated
nvrVerify	4	Uploaded data will be verified instantly

8 Disclaimer

Devices sold by Melexis are covered by the warranty and patent indemnification provisions appearing in its Term of Sale. Melexis makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Melexis reserves the right to change specifications and prices at any time and without notice. Therefore, prior to designing this product into a system, it is necessary to check with Melexis for current information. This product is intended for use in normal commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or life-sustaining equipment are specifically not recommended without additional processing by Melexis for each application.

The information furnished by Melexis is believed to be correct and accurate. However, Melexis shall not be liable to recipient or any third party for any damages, including but not limited to personal injury, property damage, loss of profits, loss of use, interrupt of business or indirect, special incidental or consequential damages, of any kind, in connection with or arising out of the furnishing, performance or use of the technical data herein. No obligation or liability to recipient or any third party shall arise or flow out of Melexis' rendering of technical or other services.

© 2004 Melexis NV. All rights reserved.

website at:

www.melexis.com

Or for additional information contact Melexis Direct:

Europe and Japan:

Phone: +32 13 67 04 95

E-mail: sales_europe@melexis.com

All other locations:

Phone: +1 603 223 2362

E-mail: sales_usa@melexis.com

QS9000, VDA6.1 and ISO14001 Certified