PTC04 PROGRAMMER PRODUCT SPECIFIC FUNCTIONS

SOFTWARE LIBRARY

1 Overview

2 Introduction

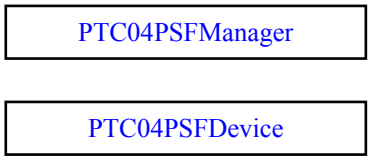
PTC04 PSF is MS Windows software library, which meets the requirements for a Product Specific Functions (PSF) module, defined in Melexis Programming Toolbox (MPT) object model. The library implements in-process COM objects for interaction with Melexis PTC04 programmers. It is designed primarily to be 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 Object Model

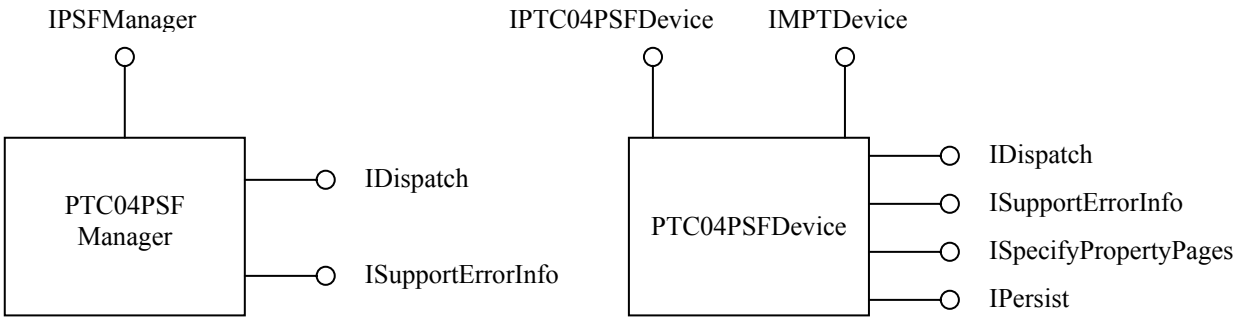
MPT object model specifies that a PSF module must expose two COM objects which implement certain COM interfaces. PTC04 PSF implements these two objects.

- **PTC04PSFManager 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.
- **PTC04PSFDevice object** – implements IPTC04PSFDevice specific interface. However, this interface derives from IMPTDevice standard MPT interface and therefore PTC04PSFDevice also implements the functionality of MPTDevice standard MPT object. In addition to standard IMPTDevice methods, IPTC04PSFDevice 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.

3.1 Object Hierarchy



3.2 Objects with interfaces



4 PTC04PSFManager Object

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.

PTC04PSFManager 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_PTC04PSFManager**, or ProgID **"MPT. PTC04PSFManager"**:

```
hRes = ::CoCreateInstance(CLSID_PTC04PSFManager, NULL, CLSCTX_INPROC,
                          IID_IPSFManager, (void**) &pPSFMan);
```

Visual Basic applications should call CreateObject function to instantiate PTC04PSFManager:

```
Set PSFMan = CreateObject("MPT. PTC04PSFManager")
```

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 PTC04PSFDevice objects, one for each connected PTC-04 programmer. The collection is empty if there are no connected programmers.

5 PTC04PSFDevice Object

This object implements standard MPT category **CATID_MLXMPTPSFSerialDevice** as well as library specific **CATID_MLXMPTPTC04Device** category. It also declares a required specific category **CATID_MLXMPTPTC04UIModule** for identification of required user interface modules.

This object can be created directly with CoCreateInstance/CreateObject or by calling the device scanning procedure ScanStandalone of PTC04PSFManager object. The following Visual Basic subroutine shows how to instantiate PTC04PSFDevice object by performing device scan on the system:

```
Sub CreateDevice()
   Dim PSFMan As PTC04PSFManager, DevicesCol As ObjectCollection, I As Long
   On Error GoTo lError

   Set PSFMan = CreateObject("MPT.PTC04PSFManager")
   Set DevicesCol = PSFMan.ScanStandalone(dtSerial)
   If DevicesCol.Count <= 0 Then
      MsgBox ("No PTC04 programmers found!")
      2Exit Sub
   End If

   ' Dev is a global variable of type PTC04PSFDevice
   ‘ 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

lError:
   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 PTC04PSFManager, DevicesCol As ObjectCollection, I As Long
   Dim CommMan As CommManager, Chan As MPTChannel
   On Error GoTo lError

   If bAutomatic Then
      ' Automatic device scanning begins here
      Set PSFMan = CreateObject("MPT.PTC04PSFManager")
      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
      ' Manual connection begins here
      Set CommMan = CreateObject("MPT.CommManager")
      Set MyDev = CreateObject("MPT.PTC04PSFDevice")
      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

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

PTC04PSFDevice 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 PTC04PSFDevice object implements IPTC04PSFDevice library specific interface, which derives from IMPTDevice. The following is a description of its properties and methods.

5.1 Global Functions

5.1.1 ResetHardware Method

Description

Resets the PTC04 programmer. Also exits from the bootloader mode.

Syntax

Visual Basic:

Sub ResetHardware()

C++:

HRESULT ResetHardware();

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 PTC04PSFModule_TLB.h.

5.1.2 GetMainHardwareID Method

Description

Sends GetHardwareID_Main command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetMainHardwareID([Format as Long = 1])

C++:

HRESULT GetMainHardwareID (/*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarID);

Parameters

Format
A **long** specifying the format of the returned data in pvarID. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

pvarID
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 hardware ID.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.1.3 GetModuleHardwareID Method

Description

Sends GetHardwareID_Module command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:
Function GetModuleHardwareID([Format as Long = 1])

C++:
HRESULT GetModuleHardwareID (/*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarID);

Parameters

Format
A **long** specifying the format of the returned data in pvarID. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

pvarID
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 hardware ID.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.1.4 GetSoftwareID Method

Description

Sends GetSoftwareID command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:
Function GetSoftwareID([Format as Long = 1])

C++:
HRESULT GetSoftwareID (/*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarID);

Parameters

Format
A **long** specifying the format of the returned data in pvarID. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

pvarID
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 software ID.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.1.5 GetCPUConfiguration Method

Description

Sends GetCPUConfiguration command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:
Sub GetCPUConfiguration(FB0 As Byte, FB1 As Byte, FB2 As Byte, FB3 As Byte)

C++:
HRESULT GetCPUConfiguration(unsigned_char* FB0/*[out]*/, unsigned_char* FB1/*[out]*/, unsigned_char* FB2/*[out]*/, unsigned_char* FB3/*[out]*/);

Parameters

FB0, FB1, FB2, FB3
Addresses of **Byte** variables which on return will contain the corresponding CPU configuration bytes.

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 PTC04PSFModule_TLB.h.

5.1.6 SendCommand Method

Description

Sends the requested command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function SendCommand(Cmd as Byte, [vParameters], [Format as Long = 1], [bCheck as Boolean = True])

C++:

HRESULT SendCommand(/*[in]*/ unsigned char Cmd, /*[in][optional]*/ VARIANT vParameters, /*[in]*/ long Format, /*[in]*/ VARIANT_BOOL bCheck, /*[out][retval]*/ VARIANT* pvRes);

Parameters

Cmd

A **Byte** specifying the code of the command to send.

vParameters

A **VARIANT** containing optional command parameters. In case the command does not have parameters it must be an empty variant. The value of the *Format* parameter is ignored in the latter case. Optional, the default is an empty variant.

Format

A **long** specifying the format of the data in vParameters and the returned data in pvRes. Possible values are:

Value	Format
1	Data is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

bCheck

A **Boolean** controlling the execution of the method. If it is True (this is the default), the method will first call standard CheckSetup method to check the connection with PTC-04 programmer and, if necessary, to exit the bootloader mode. If it is False, the method will call a lightweight implementation of CheckSetup, just to check if there is a communication channel. The programmer will be left in its current mode.

pvRes
An address of VARIANT variable that will receive the response from PTC-04 programmer. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:
A **Variant** containing the response from PTC-04 programmer.

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

Return value	**Meaning**
S_OK	The operation completed successfully. *pvRes contains valid value.
Any other error code	The operation failed. *pvRes contains an empty variant.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.1.7 SetPortE Method

Description

Overwrites data direction register (DDRE) as well as the PortE register of the PTC04 programmer and returns the data read after the setting. For a detailed description of the PortE, see Atmega128.pdf document.

Syntax

Visual Basic:
Function SetPortE(DDRE as Byte, Data as Byte) as Byte

C++:
HRESULT SetPortE(/*[in]*/ unsigned char DDRE, /*[in]*/ unsigned char Data,
/*[out][retval]*/ unsigned char* pDataIn);

Parameters

DDRE
A **Byte** specifying the data direction of the PortE. Each bit may have independent direction (1=output ; 0 =input).

Data
A **Byte** specifying the data to be set in the PortE. If a particular bit is an input, it affects the Pull-up activation.

pDataIn
An address of **Byte** variable which will receive the data read from the port after the setting.

Return value

Visual Basic:
A **Byte** containing the data read from the port after the setting.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

Available in: version 1.54 or higher.

5.1.8 SetPortF Method

Description

Overwrites data direction register (DDRF) as well as the PortF register of the PTC04 programmer and returns the data read after the setting. For a detailed description of the PortF, see Atmega128.pdf document.

Syntax

Visual Basic:

Function SetPortF(DDRF as Byte, Data as Byte) as Byte

C++:

```
HRESULT SetPortF(/*[in]*/ unsigned char DDRF, /*[in]*/ unsigned char Data,
                 /*[out][retval]*/ unsigned char* pDataIn);
```

Parameters

DDRE
A **Byte** specifying the data direction of the PortF. Each bit may have independent direction (1=output ; 0 =input).

Data
A **Byte** specifying the data to be set in the PortF. If a particular bit is an input, it affects the Pull-up activation.

pDataIn
An address of **Byte** variable which will receive the data read from the port after the setting.

Return value

Visual Basic:

A **Byte** containing the data read from the port after the setting.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

Available in: version 1.54 or higher.

5.2 EEPROM Functions

5.2.1 GetContentsFromEEPROM Method

Description

Sends GetContentsFromEEPROM command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetContentsFromEEPROM(Addr as Integer, NrBytes as Integer, [Format as Long = 1])

C++:

HRESULT GetContentsFromEEPROM (/*[in]*/ short Addr, /*[in]*/ short NrBytes, /*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarContents);

Parameters

Addr
An **Integer** specifying 12-bit address in EEPROM to start reading from.

NrBytes
An **Integer** specifying how much bytes to read. Valid values are in the range [1 – 250].

Format
A **long** specifying the format of the returned data in pvarContents. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
2	Return value is an ANSI string packed in bstrVal member of *pvarContents. 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.

pvarContents
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 contents of the EEPROM cells.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.2.2 SetContentsToEEPROM Method

Description

Sends SetContentsToEEPROM command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetContentsToEEPROM(Addr as Integer, vData, [Format as Long = 1])

C++:

HRESULT SetContentsToEEPROM (/*[in]*/ short Addr, /*[in]*/ VARIANT vData, /*[in]*/ long Format);

Parameters

Addr
An **Integer** specifying 12-bit address in EEPROM to start writing from.

vData
A **VARIANT** containing the data to be written.

Format
A **long** specifying the format of the data in vData. Possible values are:

Value	Format
1	vData is an array of bytes. This is the default value.
2	vData is an ANSI string packed in bstrVal member. 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.
3	vData is an Unicode string.

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 PTC04PSFModule_TLB.h.

5.2.3 GetTextFromEEPROM Method

Description

Sends GetTextFromEEPROM command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetTextFromEEPROM(Addr as Integer) as String

C++:
HRESULT GetTextFromEEPROM (/*[in]*/ short Addr, /*[out][retval]*/ BSTR* pTxt);

Parameters

Addr
An **Integer** specifying 12-bit address in EEPROM to start reading from.

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

Return value

Visual Basic:
A **String** containing the contents of the EEPROM cells.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.2.4 SetTextToEEPROM Method

Description

Sends SetTextToEEPROM command to the PTC04 programmer.

Syntax

Visual Basic:
Sub SetTextToEEPROM(Addr as Integer, Txt as String)

C++:
HRESULT SetTextToEEPROM (/*[in]*/ short Addr, /*[in]*/ BSTR Txt);

Parameters

Addr
An **Integer** specifying 12-bit address in EEPROM to start writing from.

Txt
A **String** containing the data to be written.

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 PTC04PSFModule_TLB.h.

5.3 I²C Functions

5.3.1 I2CSendCommand Method

Description

Sends Comm_I2C command to the PTC04 programmer.

Syntax

Visual Basic:

Sub I2CSendCommand(vOp, [Format as Long = 1])

C++:

HRESULT I2CSendCommand(/*[in]*/VARIANT vOp, /*[in]*/long Format)

Parameters

vOp
Specifies the command sequence.

Format
A **long** specifying the format of the data in **vOp**. Possible values are:

Value	Format
1	vOp is an array of bytes. This is the default value.
2	vOp is an ANSI string packed in bstrVal member. 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.

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 PTC04PSFModule_TLB.h.

5.3.2 I2CGetContentsFromEE Method

Description

Sends GetContentsFrom_I2C_EE command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function I2CGetContentsFromEE(I2C_ID As Long, Addr As Long, n As Long, [Format as Long = 1])

C++:

HRESULT I2CGetContentsFromEE(/*[in]*/long I2C_ID, /*[in]*/long Addr, /*[in]*/long n, /*[in]*/long Format, /*[out,retval]*/VARIANT* pvarContents);

Parameters

I2C_ID
I2C device number * 2.

Addr
An address in I2C EEPROM to start reading from.

n
Number of bytes to read. Valid values are in the range [1 – 250].

Format
A **long** specifying the format of the returned data in pvarContents. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
2	Return value is an ANSI string packed in bstrVal member of *pvarContents. 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.

pvarContents
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 contents of the EEPROM cells.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.3.3 I2CSetContentsToEE Method

Description

Sends SetContentsTo_I2C_EE command to the PTC04 programmer.

Syntax

Visual Basic:

Sub I2CSetContentsToEE(I2C_ID As Long, Addr As Long, vData, [Format as Long = 1])

C++:

HRESULT I2CSetContentsToEE(/*[in]*/long I2C_ID, /*[in]*/long Addr,
/*[in]*/VARIANT vData, /*[in]*/long Format)

Parameters

I2C_ID
I2C device number * 2.

Addr
An address in I2C EEPROM to start writing from.

vData
A **VARIANT** containing the data to be written.

Format
A **long** specifying the format of the data in vData. Possible values are:

Value	Format
1	vData is an array of bytes. This is the default value
2	vData is an ANSI string packed in bstrVal member. 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.
3	vData is an Unicode string.

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 PTC04PSFModule_TLB.h.

5.4 RAM Functions

5.4.1 GetContentsFromCoreRAM Method

Description

Sends GetContentsFromCoreRAM command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetContentsFromCoreRAM(Addr as Long, NrBytes as Byte, [Format as Long = 1])

C++:

HRESULT GetContentsFromCoreRAM(/*[in]*/ long Addr, /*[in]*/ unsigned char NrBytes, /*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarContents);

Parameters

Addr
A **Long** specifying an address in core RAM to start reading from.

NrBytes
A **Byte** specifying how many bytes to read. Valid values are in the range [1 – 250].

Format
A **long** specifying the format of the returned data in pvarContents. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
2	Return value is an ANSI string packed in bstrVal member of *pvarContents. 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.

pvarContents
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 contents of the core RAM cells.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.4.2 SetContentsToCoreRAM Method

Description

Sends SetContentsToCoreRAM command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetContentsToCoreRAM(Addr as Long, vData, [Format as Long = 1])

C++:

HRESULT SetContentsToCoreRAM(/*[in]*/ long Addr, /*[in]*/ VARIANT vData, /*[in]*/ long Format);

Parameters

Addr
A **Long** specifying an address in core RAM to start writing from.

vData
A **VARIANT** containing the data to be written.

Format
A **long** specifying the format of the data in vData. Possible values are:

Value	Format
1	vData is an array of bytes. This is the default value.
2	vData is an ANSI string packed in bstrVal member. 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.
3	vData is an Unicode string.

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 PTC04PSFModule_TLB.h.

5.4.3 GetContentsFromXRAM Method

Description

Sends GetContentsFromXRAM command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetContentsFromXRAM(Addr as Long, NrBytes as Byte, [Format as Long = 1])

C++:

HRESULT GetContentsFromXRAM(/*[in]*/ long Addr, /*[in]*/ unsigned char NrBytes, /*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarContents);

Parameters

Addr
A **Long** specifying an address in extended RAM to start reading from.

NrBytes
A **Byte** specifying how many bytes to read. Valid values are in the range [1 – 250].

Format
A **long** specifying the format of the returned data in pvarContents. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
2	Return value is an ANSI string packed in bstrVal member of *pvarContents. 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.

pvarContents
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 contents of the extended RAM cells.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.4.4 SetContentsToXRAM Method

Description

Sends SetContentsToXRAM command to the PTC04 programmer.

Syntax

Visual Basic:
Sub SetContentsToXRAM(Addr as Long, vData, [Format as Long = 1])

C++:
HRESULT SetContentsToXRAM(/*[in]*/ long Addr, /*[in]*/ VARIANT vData, /*[in]*/ long Format);

Parameters

Addr
A **Long** specifying an address in extended RAM to start writing from.

vData
A **VARIANT** containing the data to be written.

Format
A **long** specifying the format of the data in vData. Possible values are:

Value	Format
1	vData is an array of bytes. This is the default value.
2	vData is an ANSI string packed in bstrVal member. 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.
3	vData is a Unicode string.

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 PTC04PSFModule_TLB.h.

5.5 Pattern Functions

5.5.1 SetTiming Method

Description

Sends SetTiming command to the PTC04 programmer. It fills the timing array of the programmer.

Syntax

Visual Basic:

Sub SetTiming(TimingNr as Byte, Value as Long)

C++:

HRESULT SetTiming (/*[in]*/ unsigned char TimingNr, /*[in]*/ long Value);

Parameters

TimingNr
A **Byte** specifying an index in the timing array where to set the new value [0-31].

Value
A **Long** specifying the new value to set in the timing array.

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 PTC04PSFModule_TLB.h.

5.5.2 SetLevel Method

Description

Sends SetLevel command to the PTC04 programmer. It fills the levels array of the specified channel.

Syntax

Visual Basic:

Sub SetLevel(ChanNr as Byte, LevelNr as Byte, Value as Single)

C++:

HRESULT SetLevel (/*[in]*/ unsigned char ChanNr, /*[in]*/ unsigned char LevelNr, /*[in]*/ float Value);

Parameters

ChanNr
A **Byte** specifying channel number [0-7]. A value from the predefined constants **PPS1**, **PPS2** or **PPS3** can also be used (this is only for the voltage channels). See Chapter Channel Numbers

LevelNr
A **Byte** specifying an array number [0-7].

Value
A **float** specifying the level in Volt or mA (depending on the channel).

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 PTC04PSFModule_TLB.h.

5.5.3 RunSinglePattern Method

Description

Sends RunSinglePattern command to the PTC04 programmer. It runs a single pattern by using the indexed TimingLevels and the indexed VoltageLevels.

Syntax

Visual Basic:
Sub RunSinglePattern(ChanNr as Byte, vPattern, [Format as Long = 1])

C++:
HRESULT RunSinglePattern (/*[in]*/ unsigned char ChanNr, /*[in]*/ VARIANT vPattern, /*[in]*/ long Format);

Parameters

ChanNr
A **Byte** specifying channel number [0-7].

vPattern
A **VARIANT** containing an array of bytes that specify indexed TimingLevels and indexed VoltageLevels.
The pattern contains both indexes in every byte:
T4 T3 T2 T1 T0 L2 L1 L0, where L0-L2 is level index [0-7] in level array and T4-T0 is time index [0-31] in timing array

Format
A **long** specifying the format of the data in vPattern. Possible values are:

Value	Format
1	vPattern is an array of bytes. This is the default value.
2	vPattern is an ANSI string packed in bstrVal member. 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.

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 PTC04PSFModule_TLB.h.

5.5.4 RunRAMSinglePattern Method

Description

Sends RunRAMSinglePattern command to the PTC04 programmer. It runs single pattern of *Size* bytes of the external RAM staring from the assigned address *Addr*.

Syntax

Visual Basic:
Sub RunRAMSinglePattern(Addr as Long, Size as Long)

C++:
HRESULT RunRAMSinglePattern(/*[in]*/ long Addr, /*[in]*/ long Size);

Parameters

Addr
A **Long** specifying the address of the pattern in the external RAM.

Size
A **Long** specifying the pattern size in bytes.

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 PTC04PSFModule_TLB.h.

5.5.5 RunRAMMultiPattern Method

Description

Sends RunRAMMultiPattern command to the PTC04 programmer. It runs multi-pattern of *Size* bytes of the external RAM, starting from the assigned address *Addr*.

Syntax

Visual Basic:

Sub RunRAMMultiPattern(Addr as Long, Size as Long)

C++:

HRESULT RunRAMMultiPattern(/*[in]*/ long Addr, /*[in]*/ long Size);

Parameters

Addr
A **Long** specifying the address of the pattern in the external RAM.

Size
A **Long** specifying the pattern size in bytes.

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 PTC04PSFModule_TLB.h.

5.5.6 RunMultiPattern Method

Description

Sends RunMultiPattern command to the PTC04 programmer. It runs immediate pattern of N bytes (N is the size of the array in *vPattern*).

Syntax

Visual Basic:

Sub RunMultiPattern(vPattern, Format as Long)

C++:

HRESULT RunMultiPattern(/*[in]*/ VARIANT vPattern, /*[in]*/ long Format);

Parameters

vPattern
A **VARIANT** containing the pattern. An array of bytes that specify indexed TimingLevels and indexed VoltageLevels.
Unlike the single pattern, the multi pattern contains one index per byte:

Pattern Byte Structure

Command Name	Command #		Data					
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Set DAC Value:	1	1	Dch2	Dch1	Dch0	L2	L1	L0
Set Fast DAC:	1	0	D5	D4	D3	D2	D1	D0
Measure ADC Channel:	0	1	0	0	CH3	CH2	CH1	CH0
Measure Fast ATmega:	0	1	1	1	x	x	CH1	CH0
Compare Fast ATmega:	0	1	1	0	Expec	Res	CH1	CH0
Wait / Pause:	0	0	X	T4	T3	T2	T1	T0

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

Value	Format
1	vPattern is an array of bytes. This is the default value.
2	vPattern is an ANSI string packed in bstrVal member. 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.

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 PTC04PSFModule_TLB.h.

5.6 MMF Pattern Functions

5.6.1 MmfOpenPattern Method

Description

This method loads a pattern from the specified MMF file. Returned ID number can be used in the other MMF methods to specify this pattern.

Syntax

Visual Basic:

Function MmfOpenPattern(Filename as String, Pattern as String) as Long

C++:

HRESULT MmfOpenPattern(/*[in]*/ BSTR Filename, /*[in]*/ BSTR Pattern,
/*[out,retval]*/ long* PatternId);

Parameters

Filename
A **String** specifying the full path to the file containing referenced pattern.

Pattern
A **String** specifying the name of the required pattern. If this is an empty string, the global pattern of the MMF file is loaded.

PatternId
An address of **Long** variable, which will receive the value of unique ID of the loaded pattern.

Return value

Visual Basic:

A **Long**, containing the value of unique ID of the loaded pattern.

C++:

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

Return value	**Meaning**
S_OK	The operation completed successfully. PatternId **is not 0**.
Any other error code	The operation failed. PatternId is 0.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.2 MmfClosePattern Method

Description

This method closes a previously opened pattern. All resources (parameters, buffers, etc.) are released.

Syntax

Visual Basic:

Sub MmfClosePattern(PatternId as Long)

C++:

HRESULT MmfClosePattern(/*[in]*/ long PatternId);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern 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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.3 MmfGetParameter Method

Description

This method gets the value of the specified parameter.

Syntax

Visual Basic:

Function MmfGetParameter(PatternId as Long, Name as String, Format as Long) as Variant

C++:

HRESULT MmfGetParameter(/*[in]*/ long PatternId, /*[in]*/ BSTR Name,
/*[in]*/ long Format, /*[out,retval]*/ VARIANT* pData);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

Name
A **String** specifying the name of the parameter.

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. This is the default value.

2	Return value is an ANSI string packed in bstrVal member of *pData. 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 parameter's value. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:
A **Variant** containing the parameter's 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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.4 MmfSetParameter Method

Description

This method sets the value of the specified parameter.

Syntax

Visual Basic:
Sub MmfSetParameter(PatternId as Long, Name as String, Data as Variant, Format as Long)

C++:
HRESULT MmfSetParameter(/*[in]*/ long PatternId, /*[in]*/ BSTR Name,
/*[in]*/ VARIANT Data, /*[in,opt,defaultvalue(1)]*/ long Format);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

Name
A **String** specifying the name of the parameter.

Data
A **VARIANT** containing the data to be set as parameter's value.

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. This is the default value.

2	Data is an ANSI string packed in bstrVal member of Data. 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. Such a string can be allocated by SysAllocStringByteLen 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.
Any other error code	The operation failed.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.5 MmfSetReadBuffer Method

Description

This method allocates and fills data in a buffer that can be read by the pattern.

Syntax

Visual Basic:

```
Sub MmfSetReadBuffer(PatternId as Long, Data as Variant, Format as Long)
```

C++:

```
HRESULT MmfSetReadBuffer(/*[in]*/ long PatternId, /*[in]*/ VARIANT Data,
                         /*[in,opt,defaultvalue(1)]*/ long Format);
```

Parameters

PatternId

A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

Data

A **VARIANT** containing the data to be set as a read buffer for the pattern.

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. This is the default value.
2	Data is an ANSI string packed in bstrVal member of Data. 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. Such a string can be allocated by SysAllocStringByteLen 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.
Any other error code	The operation failed.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.6 MmfSetWriteBuffer Method

Description

This method allocates a buffer that can be written by the pattern.

Syntax

Visual Basic:

Sub MmfSetWriteBuffer(PatternId as Long, NBytes as Long)

C++:

HRESULT MmfSetWriteBuffer(/*[in]*/ long PatternId, /*[in]*/ long NBytes);

Parameters

PatternId

A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

NBytes

A **long** specifying the size of the buffer in bytes.

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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.7 MmfRunPattern Method

Description

This method runs the specified pattern. At the end it returns a Boolean, corresponding to the SUCCESS flag.

Syntax

Visual Basic:

Function MmfRunPatter(PatternId as Long) as Boolean

C++:

HRESULT MmfRunPattern(/*[in]*/ long PatternId,
/*[out,retval]*/ VARIANT_BOOL* pResult);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

pResult
An address of VARIANT_BOOL variable that will receive the status of SUCCESS flag after execution. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:

A **Boolean** containing the status of SUCCESS flag after execution.

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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.8 MmfGetWriteBuffer Method

Description

This method gets the contents of the write buffer.

Syntax

Visual Basic:

Function MmfGetWriteBuffer(PatternId as Long, Format as Long) as Variant

C++:

HRESULT MmfGetWriteBuffer(/*[in]*/ long PatternId, /*[in]*/ long Format,
/*[out,retval]*/ VARIANT* pData);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

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. This is the default value.
2	Return value is an ANSI string packed in bstrVal member of *pData. 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 contents of the write buffer. The caller is responsible to call VariantClear on that variable when it is no longer needed.

Return value

Visual Basic:
A **Variant** containing the contents of the write buffer.

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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.9 MmfSetVectorDefinition Method

Description

This method defines the MUST/MICE vector like in the beginning of a TVC file.
Note: This method changes the shape of the MUST/MICE bit, so it's not recommended to use it unless it's confirmed by a Melexis engineer.

Syntax

Visual Basic:
Sub MmfSetVectorDefinition(PatternId as Long, Definition as String)

C++:
HRESULT MmfSetVectorDefinition(/*[in]*/ long PatternId, /*[in]*/ BSTR Definition);

Parameters

PatternId
A **Long**, specifying the pattern. Such an ID is obtained by MmfOpenPattern method.

Definition
A **String** of 24 chars, specifying shape of the MUST/MICE bit. The first three are MUST1, MUST0 and MICE for the first 1/8 of the bittime, next three - for the second 1/8 and so forth. The default vector definition is:

MUST1	MUST0	MICE
0	1	X
0	1	X
0	1	X
D	1	X
D	D	X
D	D	X
D	1	D
0	1	X

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 PTC04PSFModule_TLB.h.

Available in: version 1.59 or higher.

5.6.10 MmfFileLogEnabled Property

Description

This is a boolean property, which enables or disables logging during loading (not running) of patterns, i.e. during MmfOpenPattern method execution.
By default this property is **False**.

Syntax

Visual Basic:

Property MmfFileLogEnabled as Boolean

C++:

HRESULT get_MmfFileLogEnabled(/*[out][retval]*/ VARIANT_BOOL * pValue);
HRESULT set_MmfFileLogEnabled(/*[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 PTC04PSFModule_TLB.h.

Available in: version 1.62 or higher.

5.6.11 MmfProcessLogEnabled Property

Description

This is a boolean property, which enables or disables logging running of patterns, i.e. during MmfRunPattern method execution.
By default this property is **False**.

Syntax

Visual Basic:

Property MmfProcessLogEnabled as Boolean

C++:

HRESULT get_MmfProcessLogEnabled(/*[out][retval]*/ VARIANT_BOOL * pValue);
HRESULT set_MmfProcessLogEnabled(/*[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 PTC04PSFModule_TLB.h.

Available in: version 1.62 or higher.

5.7 Timing Functions

5.7.1 GenerateFrequency Method

Description

Sends GenerateFrequency command to the PTC04 programmer. It will start generating PWM signal with the specified frequency and duty cycle. The frequency is generated on pin PE3 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.

Syntax

Visual Basic:

Sub GenerateFrequency(Frequency as Single, [DutyCycle as Single = 50])

C++:

HRESULT GenerateFrequency(/*[in]*/ float Frequency, /*[in,def,opt]*/ float DutyCycle);

Parameters

Frequency
A **Single** specifying the frequency in Hz [245..1000000].

DutyCycle
A **Single** specifying duty cycle of the resulting signal in [%]. Default value is 50.

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 PTC04PSFModule_TLB.h.
Available in: Version 1.53 or higher.

5.8 Timing Measurements Functions

5.8.1 SetTimeoutPWM Method

Description

Sends SetTimeoutPWM command to the PTC04 programmer. It sets the timeout for PWM measurements.

Syntax

Visual Basic:

Sub SetTimeout(Time as Integer)

C++:

HRESULT SetTimeout(/*[in]*/ short Time);

Parameters

Time

An **Integer** specifying the timeout in [ms]. It must be a value between 4 and 1000.

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 PTC04PSFModule_TLB.h.

5.8.2 GetDataPWM Method

Description

Sends GetDataPWM command to the PTC04 programmer. It measures PWM signal on specified channel and returns total period and the high part of the period. The measurement is done on pin PE4…7 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.
The limits for the period of the PWM signal that can be measured by the method are:

- Minimum – 26 µs (37500 Hz). In this case minimum duty cycle is 8 µs.
- Maximum - 556 ms (1.8 Hz)

Syntax

Visual Basic:

Sub GetDataPWM(Channel as Byte, Edge as Byte, PeriodHigh as Single, PeriodTotal as Single)

C++:

HRESULT GetDataPWM(/*[in]*/ unsigned char Channel, /*[in]*/ unsigned char Edge, /*[out]*/ float* PeriodHigh, /*[out]*/ float* PeriodTotal);

Parameters

Channel
A **Byte** specifying channel number [0...3] → [PE4...7]

Edge
A **Byte** specifying falling or rising edge to trigger the measurement.

PeriodHigh
An address of **Single** variable that will take the high part of the period.

PeriodTotal
An address of **Single** variable that will take the total period.

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 PTC04PSFModule_TLB.h.

5.8.3 GetValuePWM Method

Description

Sends GetValuePWM command to the PTC04 programmer. It measures PWM signal on specified channel and returns pulse width and period as floating point values. The measurement is done on pin PE4…7 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.
The limits for the period of the PWM signal that can be measured by the method are:
- Minimum – 26 µs (37500 Hz). In this case minimum duty cycle is 8 µs.
- Maximum - 556 ms (1.8 Hz)

Syntax

Visual Basic:
Sub GetValuePWM(Channel As Byte, Edge as Byte, PeriodHigh as Single, PeriodTotal as Single)

C++:
HRESULT GetValuePWM(/*[in]*/ unsigned char Channel, /*[in]*/ unsigned char Edge, /*[out]*/ float* PeriodHigh, /*[out]*/ float* PeriodTotal);

Parameters

Channel
A **Byte** specifying channel number [0…3] → [PE4…7]

Edge
A **Byte** specifying falling or rising edge to trigger the measurement.

PeriodHigh
An address of **Single** variable that will take the high part of the period.

PeriodTotal
An address of **Single** variable that will take the total period.

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 PTC04PSFModule_TLB.h.

5.8.4 GetFilteredValuePWM Method

Description

Sends GetValuePWM command to the PTC04 programmer. It measures "Filter" times PWM signal on specified channel and returns average pulse width and period as floating point values. The measurement is done on pin PE4…7 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.
The limits for the period of the PWM signal that can be measured by the method are:
- Minimum – 26 µs (37500 Hz). In this case minimum duty cycle is 8 µs.
- Maximum - 556 ms (1.8 Hz) for filter 1 and 91 ms (11 Hz) for filter 10 and above.

Syntax

Visual Basic:
Sub GetFilteredValuePWM(Channel As Byte , Edge as Byte, PeriodHigh as Single, PeriodTotal as Single, Filter As Long)

C++:
HRESULT GetFilteredValuePWM(/*[in]*/ unsigned char Channel , /*[in]*/ unsigned char Edge, /*[out]*/ float* PeriodHigh, /*[out]*/ float* PeriodTotal, /*[out]*/ long* Filter);

Parameters

Channel
A **Byte** specifying channel number [0...3] → [PE4…7]

Edge
A **Byte** specifying falling or rising edge to trigger the measurement.

PeriodHigh
An address of **Single** variable that will take the high part of the period.

PeriodTotal
An address of **Single** variable that will take the total period.

Filter
An address of **Long** variable that will take the filter used for PWM measurements.

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 PTC04PSFModule_TLB.h.

5.8.5 GetFilteredPeriod Method

Description

Sends GetFilteredPeriod command to the PTC04 programmer. It measures multiple times PWM signal on specified channel and returns average period in [us] as floating point value. Number of measurements must be set in advance by SetMeasureFilter method. The measurement is done on pin PE4…7 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.
The limits for the period of the PWM signal that can be measured by the method are:

- Minimum – 26 µs (37500 Hz). In this case minimum duty cycle is 8 µs.
- Maximum - 556 ms (1.8 Hz) for filter 1 and 91 ms (11 Hz) for filter 10 and above.

Syntax

Visual Basic:

Function GetFilteredPeriod(Channel As Byte, Edge as Byte) as Single

C++:

HRESULT GetFilteredPeriod(/*[in]*/ unsigned char Channel, /*[in]*/ unsigned char Edge, /*[out, retval]*/ float* Period);

Parameters

Channel
A **Byte** specifying channel number [0..3] → [PE4…7]

Edge
A **Byte** specifying falling or rising edge to trigger the measurement.

Period
An address of **float** variable that will take the period value in [us].

Return value

Visual Basic:

A **Single** containing the period value in [us].

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.
Available in: Version 1.53 or higher.

5.8.6 GetFilteredFrequency Method

Description

Calls GetFilteredPeriod method and returns average frequency of the signal in [Hz] as floating point value. Number of measurements must be set in advance by SetMeasureFilter method. The measurement is done on pin PE4…7 of the PortE. For a detailed description of the PortE, see Atmega128.pdf document.
The limits for the period of the PWM signal that can be measured by the method are:

- Minimum – 26 µs (37500 Hz). In this case minimum duty cycle is 8 µs.
- Maximum - 556 ms (1.8 Hz) for filter 1 and 91 ms (11 Hz) for filter 10 and above.

Syntax

Visual Basic:

Function GetFilteredFrequency(Channel As Byte, Edge as Byte) as Single

C++:

```
HRESULT GetFilteredFrequency(/*[in]*/ unsigned char Channel, /*[in]*/ unsigned char Edge,
        /*[out, retval]*/ float* Frequency);
```

Parameters

Channel
A **Byte** specifying channel number [0…3] → [PE4…7]

Edge
A **Byte** specifying falling or rising edge to trigger the measurement.

Frequency
An address of **float** variable that will take the frequency value in [Hz].

Return value

Visual Basic:

A **Single** containing the frequency value in [Hz].

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.
Available in: Version 1.53 or higher.

5.9 Drivers Functions

5.9.1 SetDAC Method

Description

Sends SetDAC command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetDAC(DACNr as Byte, Code as Long)

C++:

HRESULT SetDAC (/*[in]*/ unsigned char DACNr, /*[in]*/ long Code);

Parameters

DACNr
A **Byte** specifying DAC number [0-7].
See Chapter Channel Numbers.

Code
A **Long** value to set DAC.

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 PTC04PSFModule_TLB.h.

5.9.2 SetFastDAC Method

Description

Sends SetFastDAC command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetFastDAC(DACNr as Byte, Code as Long)

C++:

HRESULT SetFastDAC (/*[in]*/ unsigned char DACNr, /*[in]*/ long Code);

Parameters

DACNr
A **Byte** specifying DAC number [0].

Code
A **Long** value to set DAC.

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 PTC04PSFModule_TLB.h.

5.9.3 SetPPS Method

Description

Sends SetPPS command to the PTC04 programmer.

Syntax

Visual Basic:
Sub SetPPS(ChanNr as Long, Vout as Single)

C++:
HRESULT SetPPS (/*[in]*/ long ChanNr, /*[in]*/ float Vout);

Parameters

ChanNr
A **Long** specifying PPS number [0-7]. A value from the predefined constants **PPS1**, **PPS2** or **PPS3** can also be used to specify the PPS number (this is only for the voltage channels).
See Chapter Channel Numbers

Vout
A **float** in Volt for the channels 0, 1, 2, 3 or in mA for the channels 4, 5, 6.

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 PTC04PSFModule_TLB.h.

5.9.4 SetRelays Method

Description

Sends SetRelays command to the PTC04 programmer. Set Relay is the command used to activate the PPS drivers.

Syntax

Visual Basic:

Sub SetRelays(Relays As Byte, Status As Byte)

C++:

HRESULT SetRelays(/*[in]*/ unsigned char Relays, /*[in]*/ unsigned char Status);

Parameters

Relays
All bits equal to 1 point a relay to be changed (not all relays must be activated).
See Chapter Channel Numbers

Status
Every corresponding bit will set the activated relay to the status of the bit.

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.

Example

When Relays = 0010 0001 and Status = 0000 1111
Then relay 0 will be closed and relay 5 will be opened. The other entire are not touched.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.9.5 GetRelayStatus Method

Description

Sends GetRelayStatus command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetRelayStatus() As Byte

C++:

HRESULT GetRelayStatus(/*[out, retval]*/ unsigned char* pbtRelays);

Parameters

C++:

pbtRelays
An address of byte variable that will receive the status of all relays. Each bit specifies the status of the corresponding relay: 0 – OFF, 1 - ON.

Return value

Visual Basic:

A **Byte** which is the status of all relays. Each bit specifies the status of the corresponding relay: 0 – OFF, 1 - ON.

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 PTC04PSFModule_TLB.h.

5.9.6 SetCurrentLimitPPS Method

Description

Calls SetPPS method in order to set the current limit for particular PPS.

Syntax

Visual Basic:

Sub SetPPS(PPS as Byte, Level as Single)

C++:

HRESULT SetCurrentLimitPPS (/*[in]*/ unsigned char PPS, /*[in]*/ float Level);

Parameters

PPS
A **Byte** specifying PPS number [1-3]. A value from the predefined constants **PPS1**, **PPS2** or **PPS3** can also be used to specify the PPS number.
See chapter Channel Numbers.

Level
A **Single** specifying the desired current limit in mA.

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 PTC04PSFModule_TLB.h.

5.10 Measure Functions

5.10.1 GetADC Method

Description

Sends GetADC command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetADC(ChanNr as Byte) as Long

C++:

HRESULT GetADC (/*[in]*/ unsigned char ChanNr, /*[out, retval]*/ long* plVal);

Parameters

ChanNr
A **Byte** specifying channel number [0-31]. See chapter Channel Numbers.

plVal
An address of **long** variable that will receive the return value in the low order word. High order word is not used and is always zero.

Return value

Visual Basic:

A **Long** containing the result of ADC in the low order word. High order word is not used and is always zero.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.10.2 GetFilteredADC Method

Description

Sends GetFilteredADC command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetFilteredADC(ChanNr as Byte) as Single

C++:

HRESULT GetFilteredADC (/*[in]*/ unsigned char ChanNr, /*[out, retval]*/ float* pResult);

Parameters

ChanNr
A **Byte** specifying channel number [0-31]. See chapter Channel Numbers.

pResult
An address of **float** variable that will receive the return value.

Return value

Visual Basic:
A **Single** containing the result of ADC.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.10.3 GetLevel Method

Description

Sends GetLevel command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:
Function GetLevel(ChanNr as Byte) as Single

C++:
HRESULT GetLevel (/*[in]*/ unsigned char ChanNr, /*[out, retval]*/ float* fResult);

Parameters

ChanNr
A **Byte** specifying channel number [0-7]. A value from the predefined constants **PPS1**, **PPS2** or **PPS3** can also be used (this is only for the voltage channels). See chapter Channel Numbers.

fResult
An address of **float** variable that will receive the return value.

Return value

Visual Basic:
A **Single** containing the result.

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.10.4 GetCurrent Method

Description

Sends GetCurrent command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetCurrent(ChanNr as Byte) as Single

C++:

HRESULT GetCurrent (/*[in]*/ unsigned char ChanNr, /*[out, retval]*/ float* fResult);

Parameters

ChanNr
A **Byte** specifying channel number [0-7]. A value from the predefined constants **PPS1**, **PPS2** or **PPS3** can also be used to specify a PPS to measure its current. See chapter Channel Numbers.

fResult
An address of **float** variable that will receive the return value.

Return value

Visual Basic:

A **Single** containing the result.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.10.5 SetMeasureDelay Method

Description

Sends SetMeasureDelay command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetMeasureDelay(Delay as Long)

C++:

HRESULT SetMeasureDelay(/*[in]*/ long Delay);

Parameters

Delay
A **Long** specifying the new value for measure delay.

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 PTC04PSFModule_TLB.h.

5.10.6 SetMeasureFilter Method

Description

Sends SetMeasureFilter command to the PTC04 programmer.

Syntax

Visual Basic:
Sub SetMeasureFilter(Nmeas as Integer)

C++:
HRESULT SetMeasureDelay(/*[in]*/ short Nmeas);

Parameters

Nmeas
An **Integer** specifying the new value for number of measurements.

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 PTC04PSFModule_TLB.h.

5.10.7 SetSampleDelay Method

Description

Sends SetSampleDelay command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetSampleDelay(Delay as Long)

C++:

HRESULT SetSampleDelay(/*[in]*/ long Delay);

Parameters

Delay
A **Long** specifying the new value for sample delay.

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 PTC04PSFModule_TLB.h.

5.10.8 SelectChannel Method

Description

Sends SelectChannel command to the PTC04 programmer.
Note: This method is not intended for independent calling by the user. It is called internally by measuring methods when necessary.

Syntax

Visual Basic:

Sub SelectChannel (NrChannel as Byte)

C++:

HRESULT SelectChannel(/*[in]*/ unsigned char NrChannel);

Parameters

NrChannel
A **Byte** specifying channel number.

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 PTC04PSFModule_TLB.h.

5.11 Channel Numbers

5.11.1 Sensing lines

Voltage		
Name	***ADC channel***	***ADC High sensitivity channel***
Sensing Line 1	12	28
Sensing Line 2	13	29
Sensing Line 3	14	30
Sensing Line 4	15	31
Use for the Functions	GetADC GetFilteredADC GetLevel SelectChannel	GetADC GetFilteredADC GetLevel SelectChannel

5.11.2 Drivers and there channels

	Voltage				Current			Extra	
Driver	***Name***	***DAC channel***	***ADC channel***	***ADC High sensitivity channel***	***Name***	***DAC channel***	***ADC channel***	***Shut Down PPS***	***Constant***
1	PPS1	0	0	16	PPS5	4	1	1	PPS1 = 100
2	PPS2	1	2	18	PPS6	5	3	2	PPS2 = 101
3	PPS3	2	4	20	PPS7	6	5	4	PPS3 = 102
4	PPS4	3	6	22	/	/	7	8	
Use for the Functions		SetDAC SetPPS SetLevel	GetADC GetFilteredADC GetLevel SelectChannel	GetADC GetFilteredADC GetLevel SelectChannel		SetDAC SetPPS SetLevel	GetADC GetFilteredADC GetLevel SelectChannel GetCurrent	SetRelays GetRelayStatus	SetCurrentLimitPPS GetCurrent SetLevel SetPPS GetLevel

5.12 Extension Support Functions

5.12.1 SetDBIO Method

Description

Sends SetDBIO command to the PTC04 programmer.

Syntax

Visual Basic:

Sub SetDBIO(Data as Byte)

C++:

HRESULT SetDBIO(/*[in]*/ unsigned char Data);

Parameters

Data
A **Byte** specifying the new value for signals F8-FF to the Daughterboard.

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 PTC04PSFModule_TLB.h.

5.12.2 GetDBIO Method

Description

Sends GetDBIO command to the PTC04 programmer and returns the response.

Syntax

Visual Basic:

Function GetDBIO() as Byte

C++:

HRESULT GetDBIO(/*[out, retval]*/ unsigned char* pData);

Parameters

pData
An address of **Byte** variable that will receive the state of the eight DB-IO bits.

Return value

Visual Basic:

A **Byte** containing the state of the eight DB-IO bits.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.12.3 WriteToDBExtension Method

Description

Sends WriteToDBExtension command to the PTC04 programmer. It modifies the external address area F800 till FFFF.

Syntax

Visual Basic:

Sub WriteToDBExtension(Addr as Integer, Data as Byte)

C++:

HRESULT WriteToDBExtension(/*[in]*/ short Addr, /*[in]*/ unsigned char Data);

Parameters

Addr

An **Integer** specifying the address to write at.

Data

A **Byte** specifying the new value for extended address area.

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 PTC04PSFModule_TLB.h.

5.12.4 ReadFromDBExtension Method

Description

Sends ReadFromDBExtension command to the PTC04 programmer. It reads data from from the extended address area F800 till FFFF.

Syntax

Visual Basic:

Function ReadFromDBExtension(Addr as Integer) as Byte

C++:

HRESULT ReadFromDBExtension(/*[in]*/ short Addr, /*[out, retval]*/ unsigned char *pbData);

Parameters

Addr
An **Integer** specifying the address to read from.

pbData
An address of **Byte** variable that will receive the value from the extended address area.

Return value

Visual Basic:

A **Byte** containing the value from the extended address area.

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 PTC04PSFModule_TLB.h.

5.13 BootLoader Functions

5.13.1 EnterBootLoader Method

Description

Sends Goto_BootLoader command to the PTC04 programmer.

Syntax

Visual Basic:

Function EnterBootLoader() As Byte

C++:

HRESULT EnterBootLoader(/*[out, retval]*/ unsigned char* pbtMode);

Parameters

pbtMode

An address of byte variable that will receive the mode of the bootloader software after executing the command: 1 – start-up mode, 2 – programming mode, 0 – an error has occurred.

Return value

Visual Basic:

A **Byte** containing the mode of the bootloader software after executing the command: 1 – start-up mode, 2 – programming mode, 0 – an error has occurred.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.13.2 ExitBootLoader Method

Description

Sends Exit_BootLoader command to the PTC04 programmer.

Syntax

Visual Basic:

Sub ExitBootLoader()

C++:

HRESULT ExitBootLoader();

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 PTC04PSFModule_TLB.h.

5.13.3 BLGetHardwareID Method

Description

Sends GetHardwareID_Main command to the PTC04 programmer in bootloader mode and returns the response. The method automatically sends Goto_BootLoader command to ensure it is in bootloader mode. The caller is responsible to call ExitBootLoader after that if it is necessary.

Syntax

Visual Basic:

Function BLGetHardwareID([Format As Long = 1])

C++:

HRESULT BLGetHardwareID(/*[in,def]*/long Format, /*[out, retval]*/ VARIANT* pvarID);

Parameters

Format

A **long** specifying the format of the returned data in pvarID. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

pvarID

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 hardware ID.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.13.4 BLGetSoftwareID Method

Description

Sends GetSoftwareID command to the PTC04 programmer in bootloader mode and returns the response. The method automatically sends Goto_BootLoader command to ensure it is in bootloader mode. The caller is responsible to call ExitBootLoader after that if it is necessary.

Syntax

Visual Basic:

Function BLGetSoftwareID([Format As Long = 1])

C++:

HRESULT BLGetSoftwareID (/*[in]*/ long Format, /*[out][retval]*/ VARIANT* pvarID);

Parameters

Format

A **long** specifying the format of the returned data in pvarID. Possible values are:

Value	Format
1	Return value is an array of bytes. This is the preferred format for Visual Basic applications. This is the default value.
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.

pvarID

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 software ID.

C++:

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

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

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.13.5 BLUploadIntelHexFile Method

Description

Uploads Hex file with firmware into the PTC04 programmer. The programmer is left in bootloader mode if this function returns with success.

Syntax

Visual Basic:

Sub BLUploadIntelHexFile(FileName As String, Progress As Object, [vHint])

C++:

```
HRESULT BLUploadIntelHexFile(/*[in]*/BSTR FileName,
                             /*[in]*/LPDISPATCH Progress,
                             /*[in,opt]*/VARIANT vHint);
```

Parameters

FileName
Specifies full path name of the 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.

vHint
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 PTC04PSFModule_TLB.h.

5.13.6 BLSendIntelHexLine Method

Description

Sends SendIntelHexFile command to the PTC04 programmer. The programmer must be in bootloader programming mode in order to execute this command properly.

Syntax

Visual Basic:

Sub BLSendIntelHexLine(vHLine, [Format As Long = 1])

C++:
HRESULT BLSendIntelHexLine(/*[in]*/VARIANT vHLine, /*[in]*/long Format);

Parameters

vHLine
Specifies one line of a Hex file to be programmed.

Format
A **long** specifying the format of the data in **vHLine**. Possible values are:

Value	Format
1	vHLine is an array of bytes. This is the default value.
2	vHLine is an ANSI string packed in bstrVal member. 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.
3	vHLine is an Unicode string.

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 PTC04PSFModule_TLB.h.

5.13.7 BLVerifyIntelHexFile Method

Description

Compares firmware that is currently in the programmer with one in selected hex file.

Syntax

Visual Basic:
Sub BLVerifyIntelHexFile(FileName As String, Progress As Object, [vHint])

C++:
HRESULT BLVerifyIntelHexFile (/*[in]*/BSTR FileName,
/*[in]*/LPDISPATCH Progress,
/*[in,opt]*/VARIANT vHint);

Parameters

FileName
Specifies full path name of the 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.

vHint
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 PTC04PSFModule_TLB.h.

5.13.8 BLVerifyIntelHexLine Method

Description

Compares one line from the hex file with corresponding data of the firmware currently present in the programmer.

Syntax

Visual Basic:

Sub BLVerifyIntelHexLine(vHLine, [Format As Long = 1])

C++:

HRESULT BLVerifyIntelHexLine(/*[in]*/VARIANT vHLine, /*[in]*/long Format);

Parameters

vHLine

Specifies one line of a Hex file to be compared.

Format

A **long** specifying the format of the data in **vHLine**. Possible values are:

Value	Format
1	vHLine is an array of bytes. This is the default value.
2	vHLine is an ANSI string packed in bstrVal member. 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.
3	vHLine is an Unicode string.

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 PTC04PSFModule_TLB.h.

5.14 Properties

5.14.1 ResponseTimeout Property

Description

This property gets/sets the time (in ms) that is allowed to elapse before signaling timeout for a particular operation. During this period PTC-04 programmer should start sending the response/acknowledge for the operation or else the communication layer will generate an error.
The following methods utilize this parameter:

SetContentsToEEPROM
SetTextToEEPROM
GetADC
GetFilteredADC
GetLevel
GetCurrent
I2CSendComand
I2CSetContentsToEE
RunSinglePattern
RunRAMMultiPattern
RunMultiPattern
RunRAMSinglePattern
GetDataPWM
GetValuePWM
GetFilteredValuePWM

The default value for this property is 2000 (2 sec).

Syntax

Visual Basic:

```
Property ResponseTimeout as Long
```

C++:

```
HRESULT get_ResponseTimeout(/*[out][retval]*/ long* pValue);
HRESULT set_ResponseTimeout(/*[in]*/ long Value);
```

Parameters

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

Value
A **long** 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. *pValue contains a valid pointer.
Any other error code	The operation failed. *pValue contains NULL.

Quick Info

Header: Declared in PTC04PSFModule_TLB.h.

5.14.2 CommunicationLog Property

Description

This is a boolean property, which enables or disables logging of data transmitted through the communication channel. The default value for this property is **False**.

Syntax

Visual Basic:

Property CommunicationLog as Boolean

C++:

HRESULT get_CommunicationLog(/*[out][retval]*/ VARIANT_BOOL * pValue);
HRESULT set_CommunicationLog(/*[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 PTC04PSFModule_TLB.h.

6 Disclamer

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.

For the latest version of this document, go to our website at
www.melexis.com

Or for additional information contact Melexis Direct:

Europe, Africa, Asia:
Phone: +32 13 670 495
E-mail: sales_europe@melexis.com

America:
Phone: +1 603 223 2362
E-mail: sales_usa@melexis.com

ISO/TS 16949 and ISO14001 Certified