

Unmanaged MCP2210 DLL Documentation

Contents

| | |
|----------------------------------|----|
| Document Revision History | 5 |
| DLL Requirements | 6 |
| Unmanaged DLL API Overview | 6 |
| DlIConstants:..... | 8 |
| DLL Error Codes | 9 |
| Detailed API Function List..... | 11 |
| DlIInit..... | 11 |
| DlICleanUp | 11 |
| WriteEEProm | 11 |
| ReadEEProm | 12 |
| GetGpioPinDir | 12 |
| SetGpioPinDir..... | 12 |
| GetGpioPinVal..... | 13 |
| SetGpioPinVal | 13 |
| EnterAccessPasswd..... | 13 |
| GetPasswdAttmptCnt | 14 |
| GetPasswdAccessStatus..... | 14 |
| RequestSpiBusRel | 15 |
| TxferSpiData..... | 15 |

| | |
|--------------------------------|----|
| CancelSpiTxfer | 15 |
| GetEvtCntFromIntPin..... | 16 |
| GetSpiBusRelExtReqStatus | 16 |
| GetSpiBusCurOwner | 16 |
| GetVid | 18 |
| GetPid | 18 |
| GetVidPid | 18 |
| SetVidPid..... | 19 |
| GetSerialNumber | 19 |
| GetConnectionStatus..... | 19 |
| GetPwrConfigSrc..... | 20 |
| GetPwrConfigRmtWkupCpbl | 20 |
| GetPwrConfig..... | 20 |
| SetPwrConfig | 21 |
| GetReqdCurrentLd..... | 21 |
| SetReqdCurrentLd..... | 21 |
| GetStringManufacturer | 22 |
| SetStringManufacturer | 22 |
| GetStringDescriptor | 22 |
| SetStringDescriptor..... | 23 |
| GetPermanentDevLockStatus..... | 23 |
| SetPermanentDevLock..... | 23 |
| GetPasswdEnStatus | 24 |
| SetPasswdEnStatus | 24 |
| SetNewPasswd..... | 24 |

| | |
|--------------------------------|----|
| GetAllChipSettings | 25 |
| SetAllChipSettings..... | 26 |
| GetGpioDfltOutput | 26 |
| GetGpioDfltDirection | 27 |
| GetGpioDesignations | 27 |
| GetGpioConfig | 28 |
| SetGpioConfig | 28 |
| GetRmtWkupEnStatus | 29 |
| SetRmtWkupEnStatus..... | 29 |
| GetInterruptPinMode | 30 |
| SetInterruptPinMode..... | 30 |
| GetSpiBusReleaseEnStatus | 31 |
| SetSpiBusReleaseEnStatus..... | 31 |
| GetAllSpiSettings..... | 32 |
| SetAllSpiSettings | 32 |
| GetSpiBitRate..... | 33 |
| SetSpiBitRate | 33 |
| GetSpiCsIdleValue..... | 33 |
| GetSpiCsActiveValue..... | 34 |
| GetSpiCsValues | 34 |
| SetSpiCsValues | 35 |
| GetSpiDelayCsToData | 35 |
| GetSpiDelayDataToData | 35 |
| GetSpiDelayDataToCs | 36 |
| GetSpiDelays | 36 |

| | |
|--------------------------|----|
| SetSpiDelays..... | 37 |
| GetSpiTxferSize | 37 |
| SetSpiTxferSize..... | 37 |
| GetSpiMode | 38 |
| SetSpiMode..... | 38 |
| GetDevCount | 40 |
| GetSelectedDevNum..... | 40 |
| GetSelectedDevInfo | 40 |
| SelectDevice..... | 41 |

Document Revision History

| Version | Release Date | Description |
|---------|--------------|---|
| V1.0 | 06/27/2014 | Initial release |
| V1.1 | 09/19/2014 | Added error code for SPI transfer timeout (new feature) |
| V1.1.1 | 09/24/2014 | Corrected function details for both DllInit() and DllCleanup(). Changed Visual C++ redist requirement for managed DLL to reflect recent update. |

DLL Requirements

This is the key difference is between the managed and unmanaged DLL versions. The managed DLL runs within the managed Microsoft .NET environment and hence requires the **.NET framework (v2.0 or higher)** to be installed while the unmanaged DLL does not have this requirement. Hence, the undamaged DLL works best in environments where Python or other languages not supported in Microsoft Visual Studio are to be used. See below for a summary of the DLL requirements for each version.

| DLL Version: | Requirements: |
|--------------|--|
| Managed | 1. .NET framework (v2.0 or higher) 2. Microsoft Visual C++ 2010 Redistributable Package |
| Unmanaged | 1. Microsoft Visual C++ 2010 Redistributable Package |

Unmanaged DLL API Overview

The DLL is to be initialized using the **DllInit()** function and the function **DllCleanup()** is to be called before the DLL is closed.

```
void  DevIO::DllInit(UINT32 VendorID, UINT32 ProductID)
void  DevIO::DllCleanup()

    /////  EEPROM Operations
int    DevFunctions::WriteEEProm(BYTE address, BYTE content)
int    DevFunctions::ReadEEProm(BYTE address)

    /////  GPIO
int    DevFunctions::GetGpioPinDir()
int    DevFunctions::SetGpioPinDir(WORD gpioDir)
int    DevFunctions::GetGpioPinVal()
int    DevFunctions::SetGpioPinVal(WORD gpioVal)

    /////  Password Operations
int    DevFunctions::EnterAccessPasswd(String^ passwd)
int    DevFunctions::GetPasswdAttmpCnt()
int    DevFunctions::GetPasswdAccessStatus()

    /////  SPI Bus Operations
int    DevFunctions::RequestSpiBusRel()
int    DevFunctions::TxferSpiData(BYTE* txData, BYTE* rxData)
int    DevFunctions::CancelSpiTxfer()

    /////  Other Chip Status Information
```

```

int    DevFunctions::GetEvtCntFromIntPin()
int    DevFunctions::GetSpiBusRelExtReqStatus()
int    DevFunctions::GetSpiBusCurOwner()

    ////  USB Related
INT64  DevSettings::GetVid()
INT64  DevSettings::GetPid()
int    DevSettings::GetVidPid(UINT32* vid, UINT32* pid)
int    DevSettings::SetVidPid(UINT32 vid, UINT32 pid)
int    DevSettings::GetSerialNumber(WCHAR* serialNumber)
bool   DevSettings::GetConnectionStatus()
int    DevSettings::GetPwrConfigSrc()
int    DevSettings::GetPwrConfigRmtWkupCpbl()
int    DevSettings::GetPwrConfig(int* pwrSrc, bool* isRmtWkupCpble)
int    DevSettings::SetPwrConfig(int pwrSrc, bool isRmtWkupCpble)
int    DevSettings::GetReqdCurrentLd()
int    DevSettings::SetReqdCurrentLd(int current)
int    DevSettings::GetStringManufacturer(WCHAR*)
int    DevSettings::SetStringManufacturer(WCHAR* newString)
int    DevSettings::GetStringDescriptor(WCHAR*)
int    DevSettings::SetStringDescriptor(WCHAR* newString)

    ////  Chip Access Settings
int    DevSettings::GetPermanentDevLockStatus()
int    DevSettings::SetPermanentDevLock()
int    DevSettings::GetPasswdEnStatus()
int    DevSettings::SetPasswdEnStatus(bool onOff, char* newPasswd)
int    DevSettings::SetNewPasswd(String^ curPasswd, char* newPasswd)

    ////  Chip Settings
int    DevSettings::GetAllChipSettings(int whichToGet, BYTE* gpioPinDes,
                                       int* gpioDfltOutput, int* gpioDfltDir,
                                       bool* rmtWkupEn, BYTE* intPinMd,
                                       bool* spiBusRelEn)
int    DevSettings::SetAllChipSettings(int whichToSet, BYTE* gpioPinDes,
                                       int gpioDfltOutput, int gpioDfltDir,
                                       bool rmtWkupEn, BYTE intPinMd,
                                       bool spiBusRelEn)
int    DevSettings::GetGpioDfltOutput(int whichToGet)
int    DevSettings::GetGpioDfltDirection(int whichToGet)
int    DevSettings::GetGpioPinDesignations(int whichToGet,
                                       BYTE* gpioPinDes)
int    DevSettings::GetGpioConfig(int whichToSet, BYTE* gpioPinDes,
                                       int* gpioOutputDflt, int* gpioDir)
int    DevSettings::SetGpioConfig(int whichToSet, BYTE* gpioPinDes,
                                       int gpioOutputDflt, int gpioDir)
int    DevSettings::GetRmtWkupEnStatus(int whichToGet)

```

```

int    DevSettings::SetRmtWkupEnStatus(int whichToSet, bool enStatus)
int    DevSettings::GetInterruptPinMode(int whichToGet)
int    DevSettings::SetInterruptPinMode(int whichToSet, int intPinMode)
int    DevSettings::GetSpiBusReleaseEnStatus(int whichToGet)
int    DevSettings::SetSpiBusReleaseEnStatus(int whichToSet, bool spiBusRelEn)

    ////   SPI Settings
int    DevSettings::GetAllSpiSettings(int whichToGet, UINT32* bitRate,
                                     WORD* idleCsVal, WORD* activeCsVal,
                                     WORD* csToDataDly, WORD* dataToDataDly,
                                     WORD* dataToCsDly, WORD* txferSize,
                                     BYTE* spiMd)
int    DevSettings::SetAllSpiSettings(int whichToSet, UINT32 bitRate,
                                     WORD idleCsVal, WORD activeCsVal,
                                     WORD csToDataDly, WORD dataToDataDly,
                                     WORD dataToCsDly, WORD txferSize,
                                     BYTE spiMd)

int    DevSettings::GetSpiBitRate(int whichToGet)
int    DevSettings::SetSpiBitRate(int whichToSet, UINT32 bitRate)
int    DevSettings::GetSpiIdleCsValue(int whichToGet)
int    DevSettings::GetSpiActiveCsValue(int whichToGet)
int    DevSettings::GetSpiCsValues(int whichToGet, WORD* csIdleCsVal, WORD* csActiveVal)
int    DevSettings::SetSpiCsValues(int whichToSet, WORD csIdleCsVal, WORD csActiveVal)
int    DevSettings::GetSpiDelayCsToData(int whichToGet)
int    DevSettings::GetSpiDelayDataToData(int whichToGet)
int    DevSettings::GetSpiDelayDataToCs(int whichToGet)
int    DevSettings::GetSpiDelays(int whichToGet, WORD* csToDataDly, WORD* dataToDataDly,
                                WORD* dataToCsDly)
int    DevSettings::SetSpiDelays(int whichToSet, WORD csToDataDly, WORD dataToDataDly,
                                WORD dataToCsDly)

int    DevSettings::GetSpiTxferSize(int whichToGet)
int    DevSettings::SetSpiTxferSize(int whichToSet, WORD txferSize)
int    DevSettings::GetSpiMode(int whichToGet)
int    DevSettings::SetSpiMode(int whichToSet, BYTE spiMode)

    ////   Multiple Device Support
int    DevSpecial::GetDeviceCount()
int    DevSpecial::GetSelectedDevNum()
String^ DevSpecial::GetSelectedDevInfo()
int    DevSpecial::SelectDev(int devNum)

```

DllConstants:

```

//The two constants below are the same - user chooses which is more intuitive
static const int OFF = 0;
static const int DISABLED = 0;
//The two constants below are the same - user chooses which is more intuitive
static const int ON = 1;

```



```

static const int ENABLED = 1;
//Constants to be used for all whichTo(Get/Set) variables in DLL functions
static const int CURRENT_SETTINGS_ONLY = 0;
static const int PWRUP_DEFAULTS_ONLY = 1;
static const int BOTH = 2;

```

DLL Error Codes

Nearly every function within the DLL will return an error code should something go wrong in the operation. Use this table to decipher what went wrong and what action to take in order to resolve the error.

Error Code Range and Category:

- 0 – 99 - Device Error codes;
- 100 - 999 - Error originated from DLL or was further interpreted by DLL

Error Code Table

| Error Code | Error Description | Recommended Resolution |
|-------------|---|--|
| 0 | No error (Success) | N/A |
| -2 | Device is busy | N/A |
| -3 | Wrong password entered | N/A |
| -4 | NVRAM locked | N/A |
| -8 | SPI transfer in progress | N/A |
| -9 | SPI Bus not available | N/A |
| -10 | SPI transfer timeout | N/A |
| -101 | Board not connected | Check connection and device enumeration |
| -106 | Writing to the device failed | Ensure DLL was initialized properly |
| -107 | Reading the device failed | Ensure DLL was initialized properly |
| -201 | Invalid parameter given (1 st parameter) | 1 st parameter was invalid, verify parameters |
| -202 | Invalid parameter given (2 nd parameter) | 2 nd parameter was invalid, verify parameters |
| -203 | Invalid parameter given (3 rd parameter) | 3 rd parameter was invalid, verify parameters |
| -204 | Invalid parameter given (4 th parameter) | 4 th parameter was invalid, verify parameters |
| -205 | Invalid parameter given (5 th parameter) | 5 th parameter was invalid, verify parameters |
| -206 | Invalid parameter given (6 th parameter) | 6 th parameter was invalid, verify parameters |
| -207 | Invalid parameter given (7 th parameter) | 7 th parameter was invalid, verify parameters |

| | | |
|-------------|---|---|
| -208 | Invalid parameter given (8 th parameter) | 8 th parameter was invalid, verify parameters |
| -209 | Invalid parameter given (9 th parameter) | 9 th parameter was invalid, verify parameters |
| -300 | USB transfer in progress | N/A |
| -301 | Invalid USB power configuration found | Re-configure device with a valid USB power config |
| -501 | Chip access is password protected | You must enter access password to access settings |
| -502 | Chip NVRAM is permanently locked | No modifications can be made to NVRAM |
| -503 | Password attempt failed, <5 tries remain | Password failed, password mechanism still open for additional attempts. |
| -504 | Password attempt failed, ≥5 tries | Password mechanism is now temporarily blocked. Must power cycle the device in order to unlock for further attempts at accessing the device. |
| -601 | EEPROM write failed | N/A |
| -602 | EEPROM read failed | N/A |

Detailed API Function List

DllInit

Name:

DllInit (UINT32 vendorID, UINT32 productID)

Purpose:

Sets the Vendor and Product ID used for the project. THIS MUST BE DONE IN ORDER TO BEGIN USING THE DLL!

Parameters:

Inputs:

vendorID - Assigned by USB IF (www.usb.org)

productID - Assigned by the Vendor ID Holder

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

Call this before any other function calls!

DllCleanup

Name:

DllCleanup ()

Purpose:

Cleans up internal variables within DLL when finished using the DLL.

Parameters:

none

Returns:

none

Notes:

Call this when exiting your application.

WriteEEProm

Name:

WriteEEProm

Purpose:

Write to the EEPROM

Parameters:

Inputs:

address (BYTE) - Location in EEPROM where to write the data

content (BYTE) - Content to put in the EEPROM at specified address.

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:
None

ReadEEProm

Name:
ReadEEProm
Purpose:
Read the EEPROM at the specified address
Parameters:
Inputs:
address (BYTE) - Location in EEPROM where to write the data
Outputs:
none
Returns:
int - Contains error code or EEPROM data value. Negative value if error,
otherwise the function was successful and value returned is from EEPROM.
Notes:
none

GetGpioPinDir

Name:
GetGpioPinDir
Purpose:
Get GPIO Pin Direction
Parameters:
Inputs:
none
Outputs:
none
Returns:
int - If successful, return GPIO pin values in lower 9 bits. Other = failed
Notes:
none

SetGpioPinDir

Name:
SetGpioPinDir
Purpose:
Set GPIO Pin Direction
Parameters:
Inputs:
gpioDir (WORD) - What to set the GPIO pin directions as
Mapping(MSB to LSB - only lower 9 bits used):
GP8DIR GP7DIR GP6DIR GP5DIR GP4DIR GP3DIR GP2DIR GP1DIR GP0DIR

Outputs:
 none
Returns:
 int - If successful, 0 is returned. Anything else means operation failed
Notes:
 none

GetGpioPinVal

Name:
 GetGpioPinVal
Purpose:
 Get GPIO Pin Values
Parameters:
 Inputs:
 none
 Outputs:
 none
Returns:
 int - If successful, return GPIO pin values in lower 9 bits. Negative = failed
Notes:
 This command will have an effect only on those GPs that are previously configured as GPIOs.

SetGpioPinVal

Name:
 SetGpioPinVal
Purpose:
 Set GPIO Pin Values
Parameters:
 Inputs:
 gpioVal (WORD) - values to set the GPIOs to
 Mapping(MSB to LSB - only lower 9 bits used):
 GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL
 Outputs:
 none
Returns:
 int - If successful, 0 is returned. Anything else means operation failed
Notes:
 The GPIO pin value will have an effect only on those GPs that are previously configured as GPIOs.

EnterAccessPasswd

Name:
 EnterAccessPasswd
Purpose:

Send the chip access password to the device in attempt to unlock the device for modifying boot-up settings.

Parameters:

Inputs:

 strPasswd (char*) – Password to submit to the device (must be 8 characters long)

Outputs:

 none

Returns:

 int - If successful, 0 will be returned to indicate chip access granted. If a negative value is returned, there was an error.

Notes:

The device will only allow for five attempts prior to self-locking itself. For more information on the chip access protection settings, see the device datasheet on microchip.com

GetPasswdAttmptCnt

Name:

 GetPasswdAttmptCnt

Purpose:

 Get the number of password attempts up to this point

Parameters:

Inputs:

 none

Outputs:

 none

Returns:

 int - Number of password access attempts so far. If negative, there was an error.

Notes:

 none

GetPasswdAccessStatus

Name:

 GetPasswdAccessStatus

Purpose:

 Get the password access status (password access granted or not granted)

Parameters:

Inputs:

 none

Outputs:

 none

Returns:

 int - If 0, password has not been guessed correctly and access is not granted.
 If 1, password has been guessed correctly and access is granted.
 If less than zero, an error occurred.

Notes:

 none

RequestSpiBusRel

Name:
RequestSpiBusRel

Purpose:
Request chip to release the SPI bus

Parameters:

Inputs:
none

Outputs:
none

Returns:
int - If successful, return GPIO pin values in lower 16 bits. If error, value is negative.

Notes:
none

TxferSpiData

Name:
TxferSpiData

Purpose:
Transfer the specified SPI data

Parameters:

Inputs:
dataTx (BYTE*) - Array of SPI data to send to device

Outputs:
dataRx (BYTE*) - Array that will contain the SPI data received from device

Returns:
int - Return value: negative = error
 0 = success

Notes:
none

CancelSpiTxfer

Name:
CancelSpiTxfer

Purpose:
Cancel the current SPI transfer

Parameters:

Inputs:
none

Outputs:
none

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetEvtCntFromIntPin

Name:

GetEvtCntFromIntPin

Purpose:

Get the current number of events from the interrupt pin.

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - Current event count. If the value is negative, an error occurred.

Notes:

none

GetSpiBusRelExtReqStatus

Name:

GetSpiBusRelExtReqStatus

Purpose:

Get status regarding whether or not there is an external request for SPI bus release

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - If 0, no external request for SPI bus release
If 1, pending external request for SPI bus release
If less than zero, an error occurred.

Notes:

none

GetSpiBusCurOwner

Name:

GetSpiBusCurOwner

Purpose:

Get information regarding who is the current SPI bus owner

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - If 0, no owner
If 1, USB Bridge is the owner
If 2, an external master is the owner
If less than zero, an error occurred.

Notes:

none

GetVid

Name:

GetVid

Purpose:

Get the vendor ID (VID) of the part.

Parameters:

Inputs:

none

Outputs:

none

Returns:

INT64 - VID value is returned. If error, a negative value is returned.

Notes:

none

GetPid

Name:

GetPid

Purpose:

Get the product ID (PID) of the part.

Parameters:

Inputs:

none

Outputs:

none

Returns:

INT64 - PID value is returned. If error, a negative value is returned.

Notes:

none

GetVidPid

Name:

GetVidPid

Purpose:

Get the vendor ID (VID) and product ID (PID) of the part.

Parameters:

Inputs:

none

Outputs:

UINT32* - VID value of the part

UINT32* - PID value of the part

Returns:

none

Notes:

This function is only supported in the unmanaged version of the MCP2210 DLL.

SetVidPid

Name:
SetVidPid

Purpose:
Set the VID and PID of the part.

Parameters:

Inputs:
vid (UINT32) - Vendor ID value to set
pid (UINT32) - Product ID value to set

Outputs:
none

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:
none

GetSerialNumber

Name:
GetSerialNumber

Purpose:
Get the device serial number.

Parameters:

Inputs:
none

Outputs:
WCHAR* - Serial number string is returned through output parameter

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:
none

GetConnectionStatus

Name:
GetConnectionStatus

Purpose:
Retrieve the connection status of the device

Parameters:

Inputs:
none

Outputs:
none

Returns:
bool - True if connected, false if device is unconnected

Notes:
none

GetPwrConfigSrc

Name:
GetPwrConfigSrc

Purpose:
Get the USB power configuration setting: Power source (host or self).

Parameters:

Inputs:
none

Outputs:
none

Returns:
int - Return code that indicates USB device power configuration value. If less than zero, there was an error.
(0 = bus-powered, 1 = Self-powered)

Notes:
none

GetPwrConfigRmtWkupCpbl

Name:
GetPwrConfigRmtWkupCpbl

Purpose:
Get the USB power configuration setting: remote wakeup capable.

Parameters:

Inputs:
none

Outputs:
none

Returns:
int - Will be either TRUE(1) or FALSE (0). If less than zero, there was an error

Notes:
none

GetPwrConfig

Name:
SetPwrConfig

Purpose:
Set the USB power configuration settings for the part.

Parameters:

Inputs:
none

Outputs:
out_pwrSrc (int*) - USB power mode (0 = Host-Powered, 1 = Self-Powered)
out_rmtWkup (int*) - Will be either TRUE(1) or FALSE (0).

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:
none

SetPwrConfig

Name:
SetPwrConfig

Purpose:
Set the USB power configuration settings for the part.

Parameters:

Inputs:

- pwrOptn (int) - USB power mode (0 = Host-Powered, 1 = Self-Powered)
- rmtWkup (bool)- Specify if the device is remote wakeup capable (false = no, true = yes)

Outputs:
none

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:
none

GetReqdCurrentLd

Name:
GetReqdCurrentLd

Purpose:
Gets the amount of current (mA) that the device will request from the USB host

Parameters:

Inputs:
none

Outputs:
none

Returns:
int - If positive, USB device current requested value

Notes:
The value returned is equal to the actual value for this setting (i.e. a return value of 100 means the setting is set to 100mA).

SetReqdCurrentLd

Name:
SetReqdCurrentLd

Purpose:
Sets the amount of current (mA) that the device will request from the USB host

Parameters:

Inputs:

- newVal (int) - The value to be set.

Outputs:
none

Returns:
int - Contains error code. 0 = successful. Other = failed

Notes:
This function will adjust the value to meet the USB spec (1 = 2mA). So if you want

this setting to be set to 100mA, simply use 100 as input to this function (rather than 50 as the USB spec would require).

GetStringManufacturer

Name:
 GetStringManufacturer

Purpose:
 Retrieve the manufacturer string from the device.

Parameters:

 Inputs:
 none

 Outputs:
 manString (WCHAR*) - If successful, the current device USB manufacturer string is returned. If an error occurred, null is returned.

Returns:
 int - Contains error code. 0 = successful. Other = failed

Notes:
 The string returned can be up to 29 Unicode characters in length.

SetStringManufacturer

Name:
 SetStringManufacturer

Purpose:
 Set the USB manufacturer string for the device.

Parameters:

 Inputs:
 manString (WCHAR*) - Holds Unicode or ANSI string of desired manufacturer string

 Outputs:
 none

Returns:
 int - Contains error code. 0 = successful. Other = failed

Notes:
 The USB manufacturer string can be up to 29 Unicode characters long.

GetStringDescriptor

Name:
 GetStringDescriptor

Purpose:
 Retrieve the string descriptor from the device.

Parameters:

 Inputs:
 none

 Outputs:
 descString (WCHAR*) - If successful, the current device USB string descriptor is returned. If an error occurred, null is returned.

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

The string returned can be up to 29 Unicode characters in length.

SetStringDescriptor

Name:

SetStringDescriptor

Purpose:

Set the USB string descriptor for the device.

Parameters:

Inputs:

descString (WCHAR*) - Holds Unicode or ANSI string of desired string descriptor

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

The USB string descriptor can be up to 29 Unicode characters long.

GetPermanentDevLockStatus

Name:

GetPermanentDevLockStatus

Purpose:

Get the enable/disable status of the permanent device lock protection

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - Value will be 1 if enabled, 0 if disabled, and less than 0 if there was an error.

Notes:

none

SetPermanentDevLock

Name:

LockDevice

Purpose:

Permanently lock the part (THIS CANNOT BE UNDONE - USE WITH EXTREME CAUTION!)

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

!! WARNING WARNING !!! -- USE THIS FUNCTION WITH GREAT CAUTION. THE CHIP NON-VOLATILE MEMORY CANNOT BE CONFIGURED AFTER THIS FUNCTION HAS BEEN INVOKED!!

GetPasswdEnStatus

Name:

GetPasswdEnStatus

Purpose:

Get the enable/disable status of the chip password protection mechanism

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - Value will be 1 if enabled, 0 if disabled, and less than 0 if there was an error.

Notes:

none

SetPasswdEnStatus

Name:

SetPasswdEnStatus

Purpose:

Enable/disable the password protection

Parameters:

Inputs:

enStatus (bool)- True if you want to enable password protection, and false if not

passwd (char*) - Must be the desired password when enabling password protection.

The length of this password MUST be 8 characters long. When disabling password protection, this parameter must be NULL.

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

If you are disabling the password protection, ensure that the access password has already been entered and the device has no protection enabled. If this is not done prior to an attempt to disable the password mechanism, the attempt will fail.

SetNewPasswd

Name:

SetNewPasswd

Purpose:

Change the current password for the device.

Parameters:

Inputs:

- oldPasswd (char*) - Should contain current password if previously set. Otherwise use null in this parameter's place.
- newPasswd (char*) - The new password to set for the device.

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

This function is nearly the same as using SetPasswdEnStatus(). Use NULL in the oldPasswd parameter if there is currently no password protection enabled and you want to enable password protection using the password specified in the newPasswd parameter. Use NULL in the newPasswd parameter and supply the current password in the oldPasswd parameter if you want to disable password protection. Remember that any password chosen must have a length of 8 characters.

GetAllChipSettings

Name:

GetAllChipSettings

Purpose:

Get all the chip settings with one function.

Parameters:

Inputs:

- whichToGet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1,
- gpPinDes (BYTE*) - Array of 9 elements specifying each GPIO pin designation
(0 = GPIO, 1 = Chip-selects, 2 = Dedicated pin function)
- dfltGpioOutput (int*) - Default GPIO output.
Mapping (MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL
- dfltGpioDir (int*) - Default GPIO direction, where 0 = output and 1 = input
Mapping (MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL
- rmtWkupEn (bool*) - Enable/disable remote wake-up
- intPintMd (BYTE*) - Dedicated Pin function (interrupt pin mode).
See below for different modes:
 - > b100 - count high pulses
 - > b011 - count low pulses
 - > b010 - count rising edges
 - > b001 - count falling edges
 - > b000 - no interrupt counting
- spiBusRelEn (bool*) - SPI Bus release enable.
0 = SPI bus is released between transfer,
1 = SPI bus is not released by MCP2210 between transfers

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

This function is only available in unmanaged DLL version.

SetAllChipSettings

Name:

SetAllChipSettings

Purpose:

Set all the chip settings with one function.

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1,
BOTH = 2

gpPinDes (BYTE*) - Array of 9 elements specifying each GPIO pin designation
(0 = GPIO, 1 = Chip-selects, 2 = Dedicated pin function)

dfltGpioOutput (int) - Default GPIO output.
Mapping (MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

dfltGpioDir (int) - Default GPIO direction, where 0 = output and 1 = input
Mapping (MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

rmtWkupEn (bool) - Enable/disable remote wake-up

intPintMd (BYTE) - Dedicated Pin function (interrupt pin mode).
See below for different modes:
-> b100 - count high pulses
-> b011 - count low pulses
-> b010 - count rising edges
-> b001 - count falling edges
-> b000 - no interrupt counting

spiBusRelEn (bool) - SPI Bus release enable.
0 = SPI bus is released between transfer,
1 = SPI bus is not released by MCP2210 between transfers

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

None

GetGpioDfltOutput

Name:

GetGpioDfltOutput

Purpose:

Get GPIO configuration default output

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:
none

Returns:

int - If less than zero, there was an error. Otherwise this value is the default
GPIO output value. More info:
Mapping(MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

Notes:

none

GetGpioDfltDirection

Name:

GetGpioDfltDirection

Purpose:

Get GPIO configuration default direction

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:
none

Returns:

int - If less than zero, there was an error. Otherwise this value is the
dfltGpioDir value. More info:
Default GPIO direction, where 0 = output and 1 = input
Mapping(MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

Notes:

none

GetGpioDesignations

Name:

GetGpioPinDesignations

Purpose:

Get GPIO Pin designations

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

gpPinDes (BYTE*) - Array of 9 elements specifying each GPIO pin
designation

Returns:

int - If the output Byte array is NULL, then an error occurred. Otherwise it was successful.

Notes:

Since the output of this function is through the Byte array given as a parameter, the contents of this array does not need to be assigned to anything.

GetGpioConfig

Name:

SetGpioConfig

Purpose:

Get all GPIO configuration settings

Parameters:

Inputs:

None

Outputs:

whichToGet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1

gpPinDes (BYTE*) - Array of 9 elements specifying each GPIO pin designation
(0 = GPIO, 1 = Chip-selects, 2 = Dedicated pin function)

dfltGpioOutput (int*)- Default GPIO output.

Mapping(MSB to LSB - only lower 9 bits used):

GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

dfltGpioDir (int*) - Default GPIO direction, where 0 = output and 1 = input

Mapping(MSB to LSB - only lower 9 bits used):

GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

SetGpioConfig

Name:

SetGpioConfig

Purpose:

Allow the user to adjust GPIO configuration settings

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1,

BOTH = 2

gpPinDes (BYTE*) - Array of 9 elements specifying each GPIO pin designation
(0 = GPIO, 1 = Chip-selects, 2 = Dedicated pin function)

dfltGpioOutput (int)- Default GPIO output.

Mapping(MSB to LSB - only lower 9 bits used):

GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL
dfltGpioDir (int) - Default GPIO direction, where 0 = output and 1 = input
Mapping(MSB to LSB - only lower 9 bits used):
GP8VAL GP7VAL GP6VAL GP5VAL GP4VAL GP3VAL GP2VAL GP1VAL GP0VAL

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetRmtWkupEnStatus

Name:

GetRmtWkupEnStatus

Purpose:

Get the enable state of the remote wakeup setting

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - The status of the remote wakeup enable(1)/disabled(0). If negative, this
is an error code.

Notes:

none

SetRmtWkupEnStatus

Name:

SetRmtWkupEnStatus

Purpose:

Set the enable state of the remote wakeup setting

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1,
BOTH = 2

rmtWkupEn (bool) - Enable(true)/disable(false) remote wake-up

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetInterruptPinMode

Name:

GetInterruptPinMode

Purpose:

Get interrupt pin mode settings

Parameters:

Inputs:

 whichToGet (int) - Use static constants defined in DllConstants class.
 Cannot use BOTH in any Get... functions. Use one of the following:
 CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

 none

Returns:

 int - Dedicated Pin function (interrupt pin mode). If the return value is negative, the operation failed. See below for different modes:
 -> 4 - count high pulses
 -> 3 - count low pulses
 -> 2 - count rising edges
 -> 1 - count falling edges
 -> 0 - no interrupt counting

Notes:

 none

SetInterruptPinMode

Name:

SetInterruptPinMode

Purpose:

Allow the user to adjust GPIO configuration settings

Parameters:

Inputs:

 whichToSet (int) - Use static constants defined in DllConstants class.
 Use one of the following:
 CURRENT_SETTINGS_ONLY = 0,
 PWRUP_DEFAULTS_ONLY = 1,
 BOTH = 2

 intPintMd (int) - Dedicated Pin function (interrupt pin mode). See below for different modes:
 -> 4 - count high pulses
 -> 3 - count low pulses
 -> 2 - count rising edges
 -> 1 - count falling edges
 -> 0 - no interrupt counting

Outputs:

 none

Returns:

int - Contains error code. 0 = successful. Other = failed
Notes:
none

GetSpiBusReleaseEnStatus

Name:
GetSpiBusReleaseEnStatus
Purpose:
Get the enable state of the SPI bus release setting
Parameters:
Inputs:
 whichToGet (int) - Use static constants defined in DllConstants class.
 Cannot use BOTH in any Get... functions. Use one of the following:
 CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1
Outputs:
 none
Returns:
 int - The status of the remote wakeup enable(1)/disabled(0). If not these values,
 it is an error code.
Notes:
 none

SetSpiBusReleaseEnStatus

Name:
SetSpiBusReleaseEnStatus
Purpose:
Set the enable state of the SPI bus release setting
Parameters:
Inputs:
 whichToSet (int) - Use static constants defined in DllConstants class.
 Use one of the following:
 CURRENT_SETTINGS_ONLY = 0,
 PWRUP_DEFAULTS_ONLY = 1,
 BOTH = 2
 spiBusRelEn (bool)- SPI Bus release enable.
 false = SPI bus is released between transfer,
 true = SPI bus is not released by MCP2210 between transfers
Outputs:
 none
Returns:
 int - Contains error code. 0 = successful. Other = failed
Notes:
 none

GetAllSpiSettings

Name:

GetAllSpiSettings

Purpose:

Get all the SPI settings with one function.

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1

baudRate (UINT32*) - SPI bit rate speed

idleCsVal (WORD*) - IDLE CS (chip select) value

activeCsVal (WORD*) - ACTIVE CS value

csToDataDly (WORD*) - CS to data delay

dataToDataDly(WORD*)- Delay between subsequent data bytes

dataToCsDly (WORD*) - Last data byte to CS

txferSize (WORD*) - Bytes per SPI transaction

spiMd (BYTE*) - SPI mode (Possible values: 0, 1, 2, or 3)

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

This function is available in the unmanaged DLL version only.

SetAllSpiSettings

Name:

SetAllSpiSettings

Purpose:

Set all the SPI settings with one function.

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1,

BOTH = 2

baudRate (UINT32) - SPI bit rate speed

idleCsVal (WORD) - IDLE CS (chip select) value

activeCsVal (WORD) - ACTIVE CS value

csToDataDly (WORD) - CS to data delay

dataToDataDly(WORD)- Delay between subsequent data bytes

dataToCsDly (WORD) - Last data byte to CS

txferSize (WORD) - Bytes per SPI transaction

spiMd (BYTE) - SPI mode (Possible values: 0, 1, 2, or 3)

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetSpiBitRate

Name:

GetSpiBitRate

Purpose:

Get SPI bit rate value.

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - The value of the SPI bit rate

Notes:

none

SetSpiBitRate

Name:

SetSpiBitRate

Purpose:

Set SPI bit rate value.

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1,
BOTH = 2

bitRate (UINT32) - Value of desired bit rate

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetSpiCsIdleValue

Name:

GetSpiCsIdleValue

Purpose:

Get the SPI chip select idle value

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:
none

Returns:

int - The value of the SPI chip select idle value. If the return value is negative,
the operation failed.

Notes:

none

GetSpiCsActiveValue

Name:

GetSpiCsActiveValue

Purpose:

Get the SPI chip select active value

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:
none

Returns:

int - The value of the SPI chip select active value. If the return value is negative,
the operation failed.

Notes:

none

GetSpiCsValues

Name:

GetSpiCsValues

Purpose:

Get the SPI chip select values

Parameters:

Inputs:

None

Outputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1
idleCsVal (WORD*) - IDLE CS (chip select) value
activeCsVal (WORD*)- ACTIVE CS value

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

This function is only available in the unmanaged version of the DLL.

SetSpiCsValues

Name:

SetSpiCsValues

Purpose:

Set the SPI chip select values

Parameters:

Inputs:

 whichToSet (int) - Use static constants defined in DllConstants class.

 Use one of the following:

 CURRENT_SETTINGS_ONLY = 0,

 PWRUP_DEFAULTS_ONLY = 1,

 BOTH = 2

 idleCsVal (WORD) - IDLE CS (chip select) value

 activeCsVal (WORD)- ACTIVE CS value

Outputs:

 none

Returns:

 int - Contains error code. 0 = successful. Other = failed

Notes:

 none

GetSpiDelayCsToData

Name:

GetSpiDelayCsToData

Purpose:

Get CS to data SPI delay

Parameters:

Inputs:

 whichToGet (int) - Use static constants defined in DllConstants class.

 Cannot use BOTH in any Get... functions. Use one of the following:

 CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

 none

Returns:

 int - Returns CS to data SPI delay value. If return code is less than zero, an error occurred.

Notes:

 none

GetSpiDelayDataToData

Name:

GetSpiDelayDataToData

Purpose:

Get data to data SPI delay

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - Returns data to data SPI delay value. If return code is less than zero, an error occurred.

Notes:

none

GetSpiDelayDataToCs

Name:

GetSpiDelayDataToCs

Purpose:

Get data to CS SPI delay

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Cannot use BOTH in any Get... functions. Use one of the following:
CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - Returns data to CS SPI delay value. If return code is less than zero, an error occurred.

Notes:

none

GetSpiDelays

Name:

GetSpiDelays

Purpose:

Get the SPI delays

Parameters:

Inputs:

None

Outputs:

whichToGet (int) - Use static constants defined in DllConstants class.
Use one of the following:
CURRENT_SETTINGS_ONLY = 0,
PWRUP_DEFAULTS_ONLY = 1

csToDataDly (WORD*) - CS to data delay

dataToDataDly (WORD*) - Delay between subsequent data bytes

dataToCsDly (WORD*) - Last data byte to CS

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

This function is only available in the unmanaged version of the DLL.

SetSpiDelays

Name:

SetSpiDelays

Purpose:

Set the SPI delays

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1,

BOTH = 2

csToDataDly (WORD)- CS to data delay

dataToDataDly (WORD) - Delay between subsequent data bytes

dataToCsDly (WORD- Last data byte to CS

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetSpiTxferSize

Name:

GetSpiTxferSize

Purpose:

Get the SPI transfer size.

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.

Cannot use BOTH in any Get... functions. Use one of the following:

CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - The SPI transfer size is returned. If return code is less than zero, an error occurred.

Notes:

none

SetSpiTxferSize

Name:

SetSpiTxfSize

Purpose:

Set the SPI transfer size.

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1,

BOTH = 2

txferSize (WORD) - Bytes per SPI transfer (can range from 0 to 65535 inclusive)

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetSpiMode

Name:

GetSpiMode

Purpose:

Get the SPI mode

Parameters:

Inputs:

whichToGet (int) - Use static constants defined in DllConstants class.

Cannot use BOTH in any Get... functions. Use one of the following:

CURRENT_SETTINGS_ONLY = 0, PWRUP_DEFAULTS_ONLY = 1

Outputs:

none

Returns:

int - The value of the SPI Mode (0, 1, 2, or 3). If return code is less than zero, an error occurred.

Notes:

none

SetSpiMode

Name:

SetSpiMode

Purpose:

Set the SPI mode

Parameters:

Inputs:

whichToSet (int) - Use static constants defined in DllConstants class.

Use one of the following:

CURRENT_SETTINGS_ONLY = 0,

PWRUP_DEFAULTS_ONLY = 1,

BOTH = 2

spiMd (BYTE)- Specify SPI mode 0, 1, 2, or 3.

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

none

GetDevCount

Name:

GetDevCount

Purpose:

Gets the total number of attached devices

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - The total number of attached MCP2210 devices.

Notes:

IMPORTANT: You MUST use the GetConnectionStatus() function prior to calling this function since the device count is refreshed by doing so.

GetSelectedDevNum

Name:

GetSelectedDevNum

Purpose:

Gets the unique index number that indicates with MCP2210 device is selected.

Parameters:

Inputs:

none

Outputs:

none

Returns:

int - The unique index number identifying the selected MCP2210 device.

Notes:

Numbering of devices starts with 0. Hence, the first device will be indicated by the number 0 and the 2nd device by 1 and so on.

GetSelectedDevInfo

Name:

GetSelectedDevInfo

Purpose:

Get the information for the currently selected device

Parameters:

Inputs:

none

Outputs:

none

Returns:

String^ - String that gives information regarding the currently selected device

Notes:

In order to uniquely identify each device from another, you can use this string and/or the device serial number.

SelectDevice

Name:

SelectDev

Purpose:

Select the device to which the DLL will communicate

Parameters:

Inputs:

devNum (int) - The device number to be selected

Outputs:

none

Returns:

int - Contains error code. 0 = successful. Other = failed

Notes:

Numbering of devices starts with 0. Hence, the first device will be indicated by the number 0 and the 2nd device by 1 and so on. Be sure to get the total count of devices available before using this function in order to be sure that you are selecting a valid device number (max valid number is devCount-1). ***IMPORTANT: You MUST use the GetConnectionStatus() function after switching your selected device to allow proper operation and manipulation of that device.***
