Instruction for installing and using the icsneoAPI shared library for Linux

- Install the libFTDI shared library, version 0.18. This is an open source FTDI driver that sits above libusb. You can either download and install the compiled library using the software center on your Linux installation, or download and build it from here:

  http://www.intra2net.com/en/developer/libftdi/download.php

*** *Note: you will need to have root access to perform the following actions* ***

- Copy the shared library to /usr/local/lib:

  cp libicsneoAPI.so.0.1.3 /usr/local/lib

- Change directory to /usr/lib:

  cd /usr/lib

- Make a symbolic link to the shared library:

  ln -s /usr/local/lib/libicsneoAPI.so.0.1.3 libicsneoAPI.so

Setting USB Device permissions

Most systems will require a change to the permissions on USB devices for your
applications to access the ports. This version of the icsneoAPI for Linux was developed using Ubuntu Linux, version 10.4.1,
Linux kernel version 2.6.32-24-generic.

Your system should have a similar configuration for changing the permissions on the USB ports:

- I had to modify the following file:

/lib/udev/rules.d/50-udev-default.rule (you will need to have root access

to modify this file)

- Find the entry:

# libusb devices nodes
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device",
MODE="0664"

- Change the "0664" to "0666".

Using the icsneoAPI Shared Library to Write Applications

- The file icsneoLinuxAPI.h contains prototypes for the icsneoAPI functions supported. The LinuxAPIHelp.pdf describes each of the API functions.
- A sample C++ project (GeneralTestProject) is provided inside the tar file. It was a created using the codelite v2.5.2.4031. It shows basic functionality for using a neoVI device and the API.
- Your project will need to include the following files. They can be found in the directory "Include_files" in the tar file:

  - icsneoLinuxAPI.h

  - icsnVC40.h

  - icsSpyData.h

  - icsSpyDataCommon.h

  - icsSpyDataCStyle.h
- Your project will need link the following libraries:

  - rt (run-time library for Linux)

  - icsneoAPI (symbolic link should be located at /usr/lib/libicsneoAPI.so)

Instruction for installing and using the icsneoAPI shared library for Linux

- Install the libFTDI shared library, version 0.18. This is an open source FTDI driver that sits above libusb. You can either download and install the compiled library using the software center on your Linux installation, or download and build it from here:

  http://www.intra2net.com/en/developer/libftdi/download.php

*** *Note: you will need to have root access to perform the following actions* ***

- Copy the shared library to /usr/local/lib:

  cp libicsneoAPI.so.0.1.3 /usr/local/lib

- Change directory to /usr/lib:

  cd /usr/lib

- Make a symbolic link to the shared library:

  ln -s /usr/local/lib/libicsneoAPI.so.0.1.3 libicsneoAPI.so

Setting USB Device permissions

Most systems will require a change to the permissions on USB devices for your
applications to access the ports. This version of the icsneoAPI for Linux was developed using Ubuntu Linux, version 10.4.1,
Linux kernel version 2.6.32-24-generic.

Your system should have a similar configuration for changing the permissions on the USB ports:

- I had to modify the following file:

/lib/udev/rules.d/50-udev-default.rule (you will need to have root access

to modify this file)

- Find the entry:

# libusb devices nodes
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device",
MODE="0664"

- Change the "0664" to "0666".

Using the icsneoAPI Shared Library to Write Applications

- The file icsneoLinuxAPI.h contains prototypes for the icsneoAPI functions supported. The LinuxAPIHelp.pdf
  describes each of the API functions.
- A sample C++ project (GeneralTestProject) is provided inside the tar file. It was a created using the codelite v2.5.2.4031.
  It shows basic functionality for using a neoVI device and the API.
- Your project will need to include the following files. They can be found in the directory "Include_files" in the tar file:

  - icsneoLinuxAPI.h

  - icsnVC40.h

  - icsSpyData.h

  - icsSpyDataCommon.h

  - icsSpyDataCStyle.h
- Your project will need link the following libraries:

  - rt (run-time library for Linux)

  - icsneoAPI (symbolic link should be located at
  /usr/lib/libicsneoAPI.so)

Release Notes for version 0.1.3 (Release Date: 10/6/2010)

- The API was updated to use the open source shared library libFTDI. Previous versions relied on the libftd2xx library available from FTDI, but that version has not been updated to reflect changes in the USB subsystem in the latest versions of the Linux kernel. You will need to have the libftdi.so in your /usr/lib directory. You can either download and install the the libftdi shared library using your Linux desktop software center, or download and build it from here:

  http://www.intra2net.com/en/developer/libftdi/download.php
- The API was developed and tested using Ubuntu 10.4.1, with Linux kernel version 2.6.32.24 - generic.
- neoVI Yellow firmware update is now working
-

  This release supports the following Intrepid devices:

  neoVI Fire (all hardware revisions)
  ValueCAN 3
  neoVI Yellow (hardware version 1.2 only)

Release Notes for version 0.1.2 (Release Date: 01/20/2010)

-

  This release supports the following Intrepid devices:

  neoVI Fire (all hardware revisions)
  ValueCAN 3
  neoVI Yellow (hardware version 1.2 only)
-

  neoVI Yellow firmware update will fail in this version. I still need to track down this bug. If you have a neoVI Yellow and can't connect to it, it's likely that it's firmware does not match the version required by this release. I have included a Windows application called neoVI3GExplorer.exe. Try running it on a Windows PC and connecting to the neoVI Yellow. neoVI3GExplorer will update the

Yellow to the proper firmware version. In fact, I recommend doing this with any supported neoVI device that you are having trouble connecting to.

** Note - In order for neoVI3GExplorer to connect to a neoVI device on the Windows PC, you must have Intrepid's signed USB drivers installed. You can download the latest version from www.intrepidsupport.com. On that page look for the link "Drivers (auto install)" under the Product Drivers bullet.

Release Notes for version 0.1.1 (Release Date: 08/21/2009)

- This release adds support for updating the firmware on a neoVI Fire (board rev 1.1). neoVI Fire 1.2 support will be in the next release.
- CoreMini scripting functions have been added
- Documentation updated. Release and installation notes added to PDF documentation
- The CoreMiniScriptExample demonstrates how to load a script into the neoVI or ValueCAN3 and run it
- Firmed up the timing issues for connecting to devices and updating their firmware. I am still occasionally seeing firmware updates time-out half-way through the update. If this happens, try removing/replacing the USB cable from the device and power-cycling.

Release Notes for version 0.1.0 (Release Date: 07/20/2009)

- The first release of this Linux shared library is for evaluation

purposes only

- This initial release supports the ValueCAN3 and neoVI Fire (board rev 1.1)

- CoreMini scripting is not supported for the first release. This functionality will be included in a later release

- A neoVI device can only be opened by one application at a time. A future version of the library will provide the ability to allow multiple applications to share a device

- The LinuxAPIHelp.pdf file outlines the supported API functions and their usage

- The icsnAPI.pdf file covers the Windows version of the icsneoAPI but is included because it contains more information regarding neoVI devices

- The ReadMe.txt file describes how to build an application to utilize the icsneoAPI

- This release of the icsneoAPI for Linux requires specific firmware to be present on the neoVI Fire. The ability to flash update the neoVI Fire from the shared library for Linux will not be supported until the next release of the API. The firmware versions : MPIC = 1.84 UPIC = 1.2, LPIC = 1.24 JPIC = 1.5

- If you are using a neoVI Fire and cannot open it, or you are not able to read/transmit message, contact me at the email address below. We may need to have you reflash your neoVI to a different version of firmware using a Windows PC. Please contact me at the email address below if this occurs

Firmware updating for the ValueCAN3 is supported in this version and will happen on connect if the firmware in your device does not match that expected by the icsneoAPI. The GeneralTestProject demonstrates how to set callback functions to trap and display the output of a flash update to the user

Linux API Overview - intrepidcs API

icsneoAPI for Linux may be downloaded from icsneoAPI For Linux.
The current release version is 0.1.3

API Function List

Name	Description
InitializeAPI	Initializes the API and all of it's variables.
ShutdownAPI	Cleans up resources used by the API.
FindNeoDevices	Used to locate connected neoVI devices.
OpenNeoDevice	Used to open a communication link with a specific neoVI device.
ClosePort	Closes the communication link with the neoVI device.
GetMessages	Reads messages from the neoVI device.
WaitForRxMessagesWithTimeOut	Waits a specified amount of time for a received message.
TxMessages	Transmits messages to vehicle networks using a neoVI device.
SetBitRate	Sets the baud or bit rate for a specific neoVI network.
SetReflashDisplayCallbacks	Sets callback function pointers for flashing a neoVI device.
GetTimeStampForMsg	Calculates and returns the timestamp for a message.
FreeObject	Releases system resources used by the neoVI device.
GetHWFirmwareInfo	Gets the firmware version stored in a neoVI device.
GetDLLFirmwareInfo	Gets the firmware version stored in the API.
GetStoredFirmwareInfo	Gets the firmware version stored in the API for a specified type of neoVI device.
GetLastAPIError	Returns the error generated by the last API call.
GetErrorInfo	Returns a text description of an API error.
GetErrorInfoW	Returns a text description of an API error, in wide character format.
GetErrorMessages	Returns the API error message queue.
EnableNetworkRXQueue	Enables and disables the receive queue for network messages.
GetVCAN3Settings	Gets device and network parameters for a ValueCAN3 device.
SetVCAN3Settings	Sets device and network parameters for a ValueCAN3 device.
GetFireSettings	Gets device and network parameters for a neoVI Fire device.
SetFireSettings	Sets device and network parameters for a neoVI Fire device.
GetDeviceParameters	Gets individual parameters for a neoVI device.
SetDeviceParameters	Sets individual parameters for a neoVI device.
ScriptStart	Starts execution of a script that has been downloaded to a neoVI device
ScriptStop	Stops execution of a script running on a neoVI device
ScriptLoad	Downloads a script to a connected neoVI device into a specified location
ScriptClear	Clears a script from a specific location on a neoVI device
ScriptStartFBlock	Starts a function block within a script on a neoVI device
ScriptGetFBlockStatus	Returns the run status of a function block within a script on a neoVI device

ScriptStopFBlock	Stops the execution of a function block within a script on a neoVI device
ScriptGetScriptStatus	returns the run status of a specified function block within a script
ScriptReadAppSignal	Read an application signal from a script running on a neoVI device
ScriptWriteAppSignal	Set the value of an application signal in a script running on a neoVI device
ScriptReadRxMessage	Reads parameters for a receive message defined in a script on a neoVI device
ScriptReadTxMessage	Reads parameters for a transmit message defined within a script on a neoVI device
ScriptWriteRxMessage	Alter a receive message defined within script on a neoVI device
ScriptWriteTxMessage	Alter a transmit message defined within a script on a neoVI device
ScriptReadISO15765TxMessage	Read parameters of an ISO15765-2 long transmit message defined within a script on a device
ScriptWriteISO15765TxMessage	Change the parameters for an ISO15765-2 long transmit message defined within a scri neoVI device
GetRTC	Returns the value of the real-time clock on a connected neoVI device
SetRTC	Set the value of the real-time clock on a connected neoVI device

Last Updated : Tuesday, October 05, 2010

InitializeAPI Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method initializes the API and prepares it for use

C/C++ Declare

int icsneoInitializeAPI(void);

Parameters

None

Return Values

None

Remarks

Must be called before any other API calls are made.

Example

C/C++ Example

icsneoInitializeAPI();

Last Updated : Tuesday, January 19, 2010

ShutdownAPI Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method shuts down the API and releases any resources allocated during its use

C/C++ Declare

int icsneoShutdownAPI(void);

Parameters

None

Return Values

None

Remarks

Must be called when the application is done using the API.

Example

C/C++ Example

icsneoShutdownAPI();

Last Updated : Tuesday, January 19, 2010

FindNeoDevices Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the neoVI hardware devices connected to the PC.

C/C++ Declare

```
 int icsneoFindNeoDevices(unsigned long DeviceTypes,  NeoDevice *pNeoDevices, int
*pNumberOfDevices);
```

Parameters

DeviceTypes
[in] Specifies the types of neoVI devices to find. Currently supported values are:

NEODEVICE_FIRE 8
NEODEVICE_VCAN3 16
NEODEVICE_YELLOW 32
NEODEVICE_ALL 65535

These values are defined in the icsnVC40.h file.

You may use logical OR to choose which devices to look for or use NEODEVICE_ALL to specify all devices.

pNeoDevices
[out] This is the address of the first element of an array of NeoDevice structures (defined in icsnVC40.h). This array should big enough to hold 255 devices. You must specify the size of the pNeoDevices array in the pNumberOfDevices parameter. The number of devices found will be limited to the value of pNumberofDevices or 255, whichever is lower. Each returned NeoDevice structure will contain information for each device such as its type, device 'handle' and serial number.

pNumberOfDevices
[in/out] In: Specifies the size of the pNeoDevices array. Must be in the range 0 to 255.
Out: Specifies the number of neo devices that were found. This can be in the range 0 to 255.

Return Values

1 if the function succeeded. 0 if it failed for any reason. If the function succeeds but no devices are found 1 will still be returned and pNumberOfDevices will equal 0.

Remarks

The NeoDevice array elements that are returned with this function may be passed to OpenNeoDevice so that individual neoVI devices can be opened.

Example

C/C++ Example:

```
 NeoDevice Devices[255];
unsigned long lDevTypes = NEODEVICE_VCAN3 | NEODEVICE_FIRE;
int iNumDevices = 255;
int iRetVal = 0;

//Search for just ValueCAN3 and FIRE
```

```
iRetVal = icsneoFindNeoDevices(lDevTypes, Devices, &iNumDevices);

//Search for all supported neoVI types
iRetVal = icsneoFindNeoDevices(NEODEVICE_ALL, Devices, &iNumDevices);
```


Last Updated : Tuesday, January 19, 2010

OpenNeoDevice Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method opens a communication link the a neoVI device.

C/C++ Declare

int icsneoOpenNeoDevice(NeoDevice *pNeoDevice, int *hObject, unsigned char *bNetworkIDs, int bConfigRead, int bSyncToPC);

Parameters

pNeoDevice
[in] A valid NeoDevice structure filled with information about a specific neoVI device. This must be obtained by calling FindNeoDevices.

hObject
[out] The address of an int value. This will be set to the handle of the neoVI driver object that is created. It is needed as an input parameter to other API function calls. Every time you create a new neoVI object you must call ClosePort and FreeObject to avoid creating a memory and resource leak.

bNetworkIDs
[in] This is an array of number IDs which specify the NetworkID parameter of each network. This allows you to assign a custom network ID to each network. Normally, you will assign consecutive IDs to each of the networks. See NetworkIDList for a list of current network ID's. ***You may also set this parameter to NULL (zero) and the default network ID's will be used.***

bConfigRead
[in] Specifies whether the DLL should read the neoVI's device configuration before enabling the device. It is recommended that this value be set to 1.

bSyncToPC
[in] Not supported

Return Values

If the port has been opened successfully, the return value will be 1. Otherwise the return value will be zero.

Remarks

Each successful call to OpenNeoDevice should be matched with a call to the ClosePort and FreeObject methods.

Example

C/C++ Example

```
int hObject = 0;  // holds a handle to the neoVI object
int iRetVal;
int iCount;
NeoDevice *pDevice = pParmIn; //created previously

    iRetVal = icsneoOpenNeoDevice(pDevice, &hObject, NULL, 1, 0);
```

Last Updated : Tuesday, January 19, 2010

ClosePort Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method closes the communication link with the neoVI hardware.

C/C++ Declare

```
int icsneoClosePort(int hObject, int * pNumberOfErrors);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pNumberOfErrors
[out] Specifies the number of errors in the neoVI DLL error queue. You can read out the errors by calling the GetErrorMessages method.

Return Values

If the port has been closed successfully the return value will be 1. Otherwise, it will return zero. It will also return zero if the port is already closed.

Remarks

Must be called once for each successful call to OpenNeoDevice or memory and resource leaks will occur.

Example

C/C++ Example

```
int lNumberOfErrors;    // used to get the number of errors
int iResult;

// Close Communication
iResult = icsneoClosePort(hObject, &iNumberOfErrors);
```


Last Updated : Tuesday, January 19, 2010

GetMessages Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method reads messages from the neoVI device.

C/C++ Declare

```
int icsneoGetMessages(int hObject, icsSpyMessage *pMsg, int *pNumberOfMessages, int *pNumberOfErrors);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pMsg
[out] This is the address of the first element of an array of icsSpyMessage structures. This array will be loaded with messages received by the hardware. This array must be sized to fit 20,000 icsSpyMessage structures.

pNumberOfMessages
[out] Specifies the number of messages the driver has loaded in the pMsg array. This number can be up to 20,000 messages.

pNumberOfErrors
[out] Specifies the number of errors in the neoVI DLL error queue. Errors are obtained using GetErrorMessages.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. This function will return 1 even if no messages were received, provided there are no errors. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

The driver object will hold 20,000 received messages before it will generate an rx buffer overflow error (indicated by a NEOVI_ERROR_DLL_RX_MSG_BUFFER_OVERFLOW error message in the error queue). It is the job of the application software to read this buffer at regular intervals. The rate that the application needs to read these messages is dependant on the rate messages are received on the bus. For example, a high bandwidth CAN bus can generate 5000 messages per second. In this case you must read out the messages at least every four seconds or overflow errors will result.

Example

C/C++ Example

```
int hObject = 0;                  // holds a handle to the neoVI object
icsSpyMessage stMessages[20000]; // holds the received messages
int iResult;
int iNumberOfErrors;
int iNumberOfMessages;

// read out the messages
iResult = icsneoGetMessages(hObject,stMessages,&iNumberOfMessages,&iNumberOfErrors);
```

Last Updated : Tuesday, January 19, 2010

WaitForRxMessagesWithTimeOut Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to wait a specified amount of time for received messages from the neoVI hardware.

C/C++ Declare

```
int icsneoWaitForRxMessagesWithTimeOut(int hObject, unsigned int iTimeOut);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iTimeOut
[in] Specifies the amount of time in milliseconds that the function will wait for a received message before returning.

Return Values

0 if no message was received during the wait period. 1 if a message was received. -1 will be returned if there is an error condition. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

This function allows an application to avoid 'polling' for received messages. It will return as soon as a message is received or when the timeout specified has been reached.

Example

C/C++ Example

```
 int hObject = 0; // holds a handle to the neoVI object
icsSpyMessage stMessages[20000]; // holds the received messages
int iResult;
int iNumberOfErrors;
int iNumberOfMessages;
unsigned int iTimeOut = 5; //milliseconds
bool bDone = false;

while(!bDone)
{
   iResult = icsneoWaitForRxMessagesWithTimeOut(hObject, iTimeOut);

       if(iResult == 0)
        continue; //no messages received

   iResult = icsneoGetMessages(hObject,stMessages,&iNumberOfMessages,&iNumberOfErrors);

   if(iResult == 0)
       printf("Problem Reading Messages\n");
   else
       printf("Read %d Messages\n", iNumberOfMessages);
}
```

```
 return 0;
```


Last Updated : Tuesday, January 19, 2010

TxMessages Method - intrepidcs API

C/C++ declare - Parameters - Return Value - Remarks - C/C++ example

This method transmits messages asynchronously to vehicle networks using the neoVI hardware.

C/C++ Declare

```
int icsneoTxMessages(int hObject, icsSpyMessage *pMsg, int lNetworkID, int
lNumMessages);
```

Parameters

hObject
[in] Handle which specifies the driver object created by OpenNeoDevice

pMsg
[in] This is the address of the first element of an array of icsSpyMessage structures. This array will be loaded by the application software with messages that are to be transmitted by the hardware.

lNetworkID
[in] Specifies the network to transmit the message on. See NetworkID List for a list of valid Network ID values. Network support varies by neoVI device. NETID_DEVICE transmits on to the neoVI Device Virtual Network (see users manual).

lNumMessages
[in] Specifies the number of messages to be transmitted. This parameter should always be set to one unless you are transmitting a long Message.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_ISOTX_DATA_BUFFER_ALLOC = 13
NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_ILLEGAL_TX_NETWORK= 90
NEOVI_ERROR_DLL_3G_DEVICE_LICENSE_NEEDS_TO_BE_UPGRADED = 190

Remarks

This function call adds a transmit message to the transmit queue. The message will be transmitted when the network is free and all previously transmitted messages have been transmitted.

Transmit Report

After the messages has been transmitted there will be a transmit report message returned from the device. The transmit report will be read out with GetMessages. Any message read which has the SPY_STATUS_TX_MSG (icsSpyStatusTx) bit set in the status bitfield is a transmit report.

You can also identify a particular transmitted message with DescriptionID field. This two byte field (only 14 bits are used) allows the programmer to assign an arbitrary number to a message. This number is then returned in the transmit report.

The transmit report does not necessarily mean the message was transmitted successfully. For example, the Ford SCP network will return a Transmit Report if it had tried to send a message. Therefore, the programmer should always check the GlobalError Flag in the status bitfield.

To transmit different messages, set the appropriate bits in the status bitfields. For example, there are bits for init waveforms, extended identifiers and remote frames.

Example

C/C++ Example

```
 int hObject = 0; // holds a handle to the neoVI object
icsSpyMessage stMsg;
int iResult;

// Load the message to be transmitted ArbID = FF standard data 0x22 0x52 0x12 0x28
stMsg.ArbIDOrHeader = 0xFF;
stMsg.NumberBytesData = 4;
stMsg.Data[0] = 0x22;
stMsg.Data[1] = 0x52;
stMsg.Data[2] = 0x12;
stMsg.Data[3] = 0x28;

 // Status Bitfield standard ID no remote frame
stMsg.StatusBitField = 0;
stMsg.StatusBitField2 = 0;

// Transmit the message on high speed can
iResult = icsneoTxMessages(hObject,&stMsg,NETID_HSCAN,1);
```


Last Updated : Tuesday, January 19, 2010

SetBitRate Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method sets bit rates for networks on neoVI devices

C/C++ Declare

```
int icsneoSetBitRate(int hObject, int iBitRate, int iNetworkID);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iB*itRate*
[in] Specifies bit rate setting. Valid values depend on the network specified.

For the networks NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_FIRE_HSCAN2, NETID_HSCAN3, NETID_LSFTCAN,
valid bit rates are 2000, 33333, 50000, 62500, 83333, 100000, 125000, 250000, 500000, 800000, 1000000

For the networks NETID_LIN, NETID_ISO2, NETID_FIRE_LIN2, NETID_FIRE_LIN3, NETID_FIRE_LIN4,
valid bit rates are

For the network NETID_FIRE_CGI valid bit rates are 625000 and 115200

iNetworkID
[in] Specifies the network. The valid values are:

NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_FIRE_HSCAN2, NETID_HSCAN3, NETID_LSFTCAN, NETID_LIN,
NETID_ISO2, NETID_FIRE_LIN2, NETID_FIRE_LIN3, NETID_FIRE_LIN4, NETID_FIRE_CGI

These values are defined in the icsnVC40.h file

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_INVALID_NETID = 8
NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_RED_INVALID_BAUD_SPECIFIED = 122
NEOVI_ERROR_DLL_SEND_DEVICE_CONFIG_ERROR = 229
NEOVI_ERROR_DLL_GET_DEVICE_CONFIG_ERROR = 230
NEOVI_ERROR_DLL_UNKNOWN_NEOVI_TYPE = 231

Remarks

The specified network must exist on the connected neoVI device.

Example

C/C++ Example:

```
 int iRetVal;

iRetVal = icsneoSetBitrate(hObject,  500000, NETID_HSCAN);
if(iRetVal == 0)
{
    printf("\nFailed to set the bit rate");
}
else
{
    printf("\nSuccessfully set the bit rate");
}
```


Last Updated : Tuesday, January 19, 2010

SetReflashDisplayCallbacks Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to set 'call back' functions that will be called by the intrepidcs API when flashing a neoVI device. The use of call back functions allows the client application to receive the status messages and display them as desired.

C/C++ Declare

```
 int icsneoSetReflashDisplayCallbacks(void (*OnPrompt)(unsigned long), void
(*OnReflashUpdate)(const wchar_t *, unsigned long));
```

Parameters

*void (*OnPrompt)(unsigned long)*

[in] Specifies a function pointer that will be called when a message must be displayed instructing the user to disconnect and then re-connect the neoVI from the USB port. The function receives an unsigned long that will contain the serial number of the neoVI device being flash updated. Before returning from this function call the user of the client application must be prompted to unplug the neoVI from the USB port and then re-connect it before continuing. This is to put the USB chip in the neoVI into bootloader mode so that flashing can begin. This function will not be called when flashing a ValueCAN3 device.

*void (*OnReflashUpdate)(const wchar_t *, unsigned long)*

[in] Specifies a function pointer that will be called when a flashing status message is ready to be displayed. The function receives a pointer to a wide character string that contains the status message to display. It also receives an unsigned long that contains the percentage complete for the current chip being flashed. The percentage value will reset to 0 for each new chip. For example, the neoVI Fire has four chips to flash while the ValueCAN3 has only two.

Return Values

1 if successful, 0 if either function pointer is NULL.

Remarks

Once the callbacks have been set they are valid and active until the the DLL is unloaded or until the ClearReflashDisplayCallbacks function is called.

Example

C/C++ Example:

```
void ReflashStatus(const wchar_t* status, unsigned long percent)
{
    printf("%ls %i % \r\n", status, percent);
}

void BootLoaderCallBack(unsigned long SerialNumber)
{
    printf("Flashing neoVI #%d\r\nDisconnect USB, reconnect USB and then hit any
key....\r\n", SerialNumber);

       //process users key press here
}

void MyFunction(void)
{
```

```
    icsneoSetReflashCallbacks(BootLoaderCallBack, ReflashStatus);
}
```


Last Updated : Tuesday, January 19, 2010

GetTimeStampForMsg Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method calculates the timestamp for a message, based on the connected hardware type, and converts it to a usable variable.

C/C++ Declare

```
int icsneoGetTimeStampForMsg(int hObject, icsSpyMessage *pMsg, icsSpyMessage *pMsg,
double *pTimeStamp);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pMsg
[in] The message to be used for calculating timestamp.

pTimeStamp
[out] The calculated timestamp.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

Different models of neoVI devices have different time resolutions. This function uses the proper formula to calculate a timestamp based on the connected device type.

Example

C/C++ Example

```
 int hObject;
int iResult;
double dTimeStamp;
icsSpyMessage Msg;

iResult = icsneoGetTimeStampForMsg(m_hObject, &Msg, &dTimeStamp);
```


Last Updated : Tuesday, January 19, 2010

FreeObject Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method releases system resources used by the neoVI device.

C/C++ Declare

```
void icsneoFreeObject(int hObject);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

Return Values

None.

Remarks

This method is used to release any resources that were allocated by OpenNeoDevice. Applications that create neoVI handles should release them using this method, however, the intrepidCS API will release any resources that it created for the client application when the client application ends and the API is unloaded.

Example

C/C++ Example

```
icsneoFreeObject(hObject);
```


Last Updated : Tuesday, January 19, 2010

GetHWFirmwareInfo Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the firmware version of the open neoVI device.

C/C++ Declare

```
int icsneoGetHWFirmwareInfo(int hObject, stAPIFirmwareInfo *pInfo);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pInfo
[out] Pointer to an stAPIFirmwareInfo structure.

Return Values

Returns 1 if successful, 0 if an error occurred.

Remarks

This method returns the firmware version stored in the open neoVI device.

Example

C/C++ Example

```
stAPIFirmwareInfo FirmwareInfo = new stAPIFirmwareInfo();
int iResult;

iResult = icsNeoDll.icsneoGetHWFirmwareInfo(m_hObject, ref FirmwareInfo);

if(iResult == 0)
{
    printf("Problem getting the neoVI's firmware information");
    return;
}
```


Last Updated : Tuesday, January 19, 2010

GetDLLFirmwareInfo Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the firmware version stored within the DLL API.

C/C++ Declare

```
int icsneoGetDLLFirmwareInfo(int hObject, stAPIFirmwareInfo *pInfo);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pInfo
[out] Pointer to an stAPIFirmwareInfo structure.

Return Values

Returns 1 if successful, 0 if an error occurred.

Remarks

This method returns the version information for the neoVI firmware stored within the neoVI DLL API.

Example

C# Example

```
 stAPIFirmwareInfo FirmwareInfo = new stAPIFirmwareInfo();
int iResult;

iResult = icsNeoDll.icsneoGetDLLFirmwareInfo(m_hObject, ref FirmwareInfo);
if(iResult == 0)
{
    printf("Problem getting the version of the firmware stored within the DLL API");
    return;
}
```


Last Updated : Tuesday, January 19, 2010

GetStoredFirmwareInfo Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the firmware version stored in the DLL for a specific type of neoVI device.

C/C++ Declare

```
int icsneoGetStoredFirmwareInfo(unsigned long NeoDeviceType, char *pInfo);
```

Parameters

NeoDeviceType

[in] Specifies the type of neoVI device. Currently supported values are:

NEODEVICE_FIRE 8
NEODEVICE_VCAN3 16
NEODEVICE_YELLOW 32

pInfo

[out] An array where the firmware information will be stored. This array must be at least 50 bytes in size.

Return Values

Returns 1 if successful, 0 if an error occurred.

Remarks

This method returns the firmware version stored in the DLL for a specific type of neoVI device.

The string returned will be the version number of each programmable chip on the device:

"chip name:version,chipname:version,..."

Examples:

Fire: "MPIC:1.84,UPIC:1.2,LPIC:1.24,JPIC:1.5"
ValueCAN3: "MPIC:0.38"
Yellow: "MPIC:0.2,UPIC:1.2"

Example

C/C++ Example

```
 char FirmwareInfo[50];
int iResult;

iResult = icsneoGetStoredFirmwareInfo(NEODEVICE_FIRE, FirmwareInfo);

if(iResult == 0)
{
    printf("Problem getting the stored firmware information");
    return;
}
```


Last Updated : Tuesday, January 19, 2010

GetLastAPIError Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the error generated by the last API call.

C/C++ Declare

```
int icsneoGetLastAPIError(int hObject, int *piErrorNumber);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

piErrorNumber
[out] The value of the error generated by the previous API call will be returned. The text description of the error can then be obtained by calling GetErrorInfo.

Return Values

If an error was generated and stored during the last API call then 1 will be returned. 0 will be returned if no error was generated since the port was opened or the last time that GetLastAPIError was called. The stored error will be cleared after this call. API errors are generated and stored on a 'per-thread' basis. The calling thread will only receive errors generated within it's own context. If a new API error is generated before the previous error has been retrieved, the previous error will be lost. All errors generated can still be retrieved using GetErrorMessages. However, GetErrorMessages will return errors generated in all threads, not just the current thread.

Remarks

Example

C/C++ Example:

```
unsigned long ulErrorNumber;

if(iResult != 1)
{
    icsneoGetLastAPIError(&ulErrorNumber);
}
```


Last Updated : Tuesday, January 19, 2010

GetErrorInfo Method - intrepidcs API

C/C++ declare - Parameters - Return Value - Remarks - C/C++ example

This method returns a text description of an intrepidcs API error number.

C/C++ Declare

```
int icsneoGetErrorInfo(int lErrorNumber,
                       char *szErrorDescriptionShort,
                         char *szErrorDescriptionLong,
                       int *lMaxLengthShort,
                       int *lMaxLengthLong,
                       int *lErrorSeverity,
                       int *lRestartNeeded);
```

Parameters

lErrorNumber
[in] This is the number of the error message returned from GetErrorMessages.

sErrorDescriptionShort
[out] This is short description of the error. This parameter should be sized to include up to 255 characters including the NULL terminator.

sErrorDescriptionLong
[out] This is longer more detailed description of the error. This parameter should be sized to include up to 255 characters including the NULL terminator.

lMaxLengthShort
[in] This is the size in characters of the *sErrorDescriptionShort* array that is being passed in. This value must be 255 or less.

lMaxLengthLong
[in] This is the size in characters of the *sErrorDescriptionLong* array that is being passed in. This value must be 255 or less.

lErrorSeverity
[out] This indicates the error severity. This is estimated severity for the application and doesn't have significant meaning. See Table 1 below for more information.

lRestartNeeded
[out] If 1 it is recommend that the application close communications with the DLL and reopen it.

Return Values

If the error number was found successfully the return value will be non-zero.

Remarks

None.

Table 1 - Descriptions of Error Severity

Error Severity	Description
const unsigned long icsspyErrCritical=0x10;	A critical error which affects operation or accuracy
const unsigned long icsspyErrExclamation=0x30;	An important error which may be critical depending on the application.
const unsigned long icsspyErrInformation=0x40;	An error which probably does not need attention.

const unsigned long icsspyErrQuestion=0x20;	An error which is not understood.

Example

C/C++ Example

```
lResult = icsneoGetErrorMessages(hObject,iErrors, &lNumberOfErrors);

 if(lResult == 0)
   printf("Problem Reading errors");

 // dump the neoVI errors
if (lNumberOfErrors > 0)
{
     for (lCount = 0; lCount < lNumberOfErrors; lCount++)
  {
        wprintf(szOut, "Error %d - ", iErrors[lCount]);


icsneoGetErrorInfo(iErrors[lCount],szDescriptionShort,szDescriptionLong,

&lMaxLengthShort,&lMaxLengthLong,&lErrorSeverity,&lRestartNeeded);

         printf("%s" szDescriptionShort);
        printf("\n");
  }
}
```


Last Updated : Tuesday, October 05, 2010

GetErrorInfoW Method - intrepidcs API

C/C++ declare - Parameters - Return Value - Remarks - C/C++ example

This method returns a text description of an intrepidcs API error number, in wide character format.

C/C++ Declare

```
int icsneoGetErrorInfoW(int lErrorNumber,
                        wchar_t *szErrorDescriptionShort,
                          wchar_t *szErrorDescriptionLong,
                        int *lMaxLengthShort,
                        int *lMaxLengthLong,
                        int *lErrorSeverity,
                        int *lRestartNeeded);
```

Parameters

lErrorNumber
[in] This is the number of the error message returned from GetErrorMessages.

sErrorDescriptionShort
[out] This is short description of the error. This parameter should be sized to include up to 255 characters including the NULL terminator.

sErrorDescriptionLong
[out] This is longer more detailed description of the error. This parameter should be sized to include up to 255 characters including the NULL terminator.

lMaxLengthShort
[in] This is the size in characters of the *sErrorDescriptionShort* array that is being passed in. This value must be 255 or less.

lMaxLengthLong
[in] This is the size in characters of the *sErrorDescriptionLong* array that is being passed in. This value must be 255 or less.

lErrorSeverity
[out] This indicates the error severity. This is estimated severity for the application and doesn't have significant meaning. See Table 1 below for more information.

lRestartNeeded
[out] If 1 it is recommend that the application close communications with the DLL and reopen it.

Return Values

If the error number was found successfully the return value will be non-zero.

Remarks

None.

Table 1 - Descriptions of Error Severity

Error Severity	Description
const unsigned long icsspyErrCritical=0x10;	A critical error which affects operation or accuracy
const unsigned long icsspyErrExclamation=0x30;	An important error which may be critical depending on the application.

const unsigned long icsspyErrInformation=0x40;	An error which probably does not need attention.
const unsigned long icsspyErrQuestion=0x20;	An error which is not understood.

Example

C/C++ Example

```
lResult = icsneoGetErrorMessages(hObject,iErrors, &lNumberOfErrors);

 if(lResult == 0)
   printf("Problem Reading errors");

 // dump the neoVI errors
if (lNumberOfErrors > 0)
{
     for (lCount = 0; lCount < lNumberOfErrors; lCount++)
  {
        wprintf(szOut, "Error %d - ", iErrors[lCount]);


icsneoGetErrorInfoW(iErrors[lCount],szDescriptionShort,szDescriptionLong,

&lMaxLengthShort,&lMaxLengthLong,&lErrorSeverity,&lRestartNeeded);

         printf("%S", szDescriptionShort);
        printf("\n");
  }
}
```


Last Updated : Tuesday, October 05, 2010

GetErrorMessages Method - intrepidcs API

C/C++ declare - Parameters - Return Value - Remarks - C/C++ example

This method reads the neoVI DLL error message queue.

C/C++ Declare

```
int icsneoGetErrorMessages(int hObject, int *pErrorMsgs, int *pNumberOfErrors);
```

Parameters

hObject
[in] Specifies the driver object created with OpenNeoDevice.

pErrorMsgs
[out] This is the address of the first element of an array of long variables of at least 600 elements. This array will be loaded with the current error queue. The error queue will contain errors generated by all threads, not just the current thread. You can get a text description of this error using GetErrorInfo.

pNumberOfErrors
[out] Specifies the number of errors copied into the pErrorMsgs buffer. The maximum value will be 600.

Return Values

Returns 1 if successful, 0 on failure.

Remarks

The error queue will be reset after this method is called.

Example

C/C++ Example

```
 int hObject = 0; // holds a handle to the neoVI object
int iErrors[599];
int lResult;
int lNumberOfErrors;
wchar_t szOut[200];
long lCount;

 // Read the errors from the DLL
lResult = icsneoGetErrorMessages(hObject,iErrors,&lNumberOfErrors);

// dump the neoVI errors to the debug window
if(lNumberOfErrors > 0)
{
    for(lCount = 0;lCount <lNumberOfErrors; lCount++)
    {
        wsprintf(szOut, "Error %d\n", iErrors[lCount]);

          }
}
else
    printf("No Errors to report\n");
```


Last Updated : Tuesday, January 19, 2010

EnableNetworkRXQueue Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method enables or disables the received messages queue for a specific application connected to a neoVI device.

C/C++ Declare

```
int icsneoEnableNetworkRXQueue(int hObject, int lEnable);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

lEnable
[in] 1 to enable network receive, 0 to disable network receive.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

This function will enable and disable network traffic for a specific client application connected to the neoVI. Other applications connected to the same neoVI device will not be affected.

Example

C/C++ Example

```
icsneoEnableNetworkRXQueue(hObject, 0); //disable the incoming message queue
```


Last Updated : Tuesday, January 19, 2010 Monday, January 24, 2005

GetVCAN3Settings Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method reads the configuration settings from a ValueCAN3 device.

C/C++ Declare

```
int icsneoGetVCAN3Settings(int hObject, SVCAN3Settings *pSettings, int *iNumBytes);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pSettings
[out] Pointer to a SVCAN3Settings structure.

iNumBytes
[in] This value is always the size, in bytes, of the SVCAN3Settings structure.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

After getting the current settings, you may change the parameters defined in the SVCAN3Settings structure and write the settings back to the ValueCAN3 using SetVCAN3Settings.

Example

C/C++ Example

```
 SVCAN3Settings VCANReadSettings;
int iNumberOfBytes;
int iResult;

//Get the settings
iNumberOfBytes = sizeof(VCANReadSettings );

iResult = icsneoGetVCAN3Settings(m_hObject, &VCANReadSettings , iNumberOfBytes);

if(iResult == 0)
{
    printf("Problem reading VCAN3 configuration");
    return;
}
```


Last Updated : Tuesday, January 19, 2010

SetVCAN3Settings Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method writes configuration settings to a ValueCAN3 device.

C/C++ Declare

```
int icsneoSetVCAN3Settings(int hObject, SVCAN3Settings *pSettings, int iNumBytes, int
bSaveToEEPROM);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pSettings
[in] The address of an allocated SVCAN3Settings structure.

iNumBytes
[in] This value is always the size, in bytes, of the SVCAN3Settings structure.

bSaveToEEPROM
[in] If set to 0, the settings changes will revert to the values stored in EEPROM when the ValueCAN3 is power-cycled. If set to 1, the values will overwrite the EEPROM settings and become persistent across power-cycles of the ValueCAN3.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

Before using this function, the SVCAN3Settings structure must be initialized with the current neoVI settings using GetVCAN3Settings.

Example

C/C++ Example

```
SVCAN3Settings VCANReadSettings;
int iNumberOfBytes;
int iResult;

//################################
//VCANReadSettings struct is read
//and changed as needed before
//Setting the new values
//################################

iNumberOfBytes=sizeof(VCANReadSettings );
iResult = icsneoSetVCAN3Settings(m_hObject, &VCANReadSettings , iNumberOfBytes, 1);
if(iResult == 0)
{
    printf("Problem Sending VCAN configuration");
    return;
```

```
}
```


Last Updated : Tuesday, January 19, 2010

GetFireSettings Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method reads the configuration settings from a neoVI Fire device.

C/C++ Declare

```
int icsneoGetFireSettings(int hObject, SFireSettings *pSettings, int iNumBytes);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pSettings
[out] Pointer to an SFireSettings structure.

iNumBytes
[in] This value is always the size, in bytes, of the SFireSettings structure.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

After getting the current settings, you may change the parameters defined in the SFireSettings structure and write the settings back to the neoVI Fire using SetFireSettings.

Example

C/C++ Example

```
 SFireSettings FireReadSettings;
int iNumberOfBytes;
int iResult;

//Get the settings
iNumberOfBytes = sizeof(SFireSettings);
iResult = icsneoGetFireSettings(m_hObject, &FireReadSettings, iNumberOfBytes);

if(iResult == 0)
{
    printf("Problem reading FIRE configuration");
    return;
}
```


Last Updated : Tuesday, January 19, 2010

SetFireSettings Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method writes configuration settings to a neoVI Fire device.

C/C++ Declare

```
int icsneoSetFireSettings(int hObject, SFireSettings *pSettings, int iNumBytes, int
bSaveToEEPROM);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pSettings
[out] Pointer to an SFireSettings structure.

iNumBytes
[in] This value is always the size, in bytes, of the SFireSettings structure.

bSaveToEEPROM
[in] If set to 0, the settings changes will revert to the values stored in EEPROM when the neoVI is power-cycled. If set to 1, the values will overwrite the EEPROM settings and become persistent across power-cycles of the neoVI.

Return Values

Returns 1 if successful, 0 if an error occurred. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

Before using this function, the SFireSettings structure must be initialized with the current neoVI settings using GetFireSettings.

Example

C/C++ Example

```
 SFireSettings FireReadSettings;
int iNumberOfBytes;
int iResult;

//################################
//FireReadSettings struct is read
//and changed as needed before
//Setting the new values
//################################

iNumberOfBytes=sizeof(SFireSettings);

iResult = icsneoSetFireSettings(m_hObject, &FireReadSettings, iNumberOfBytes, 1);

if(iResult == 0)
{
    printf("Problem Sending FIRE configuration");
```

```
    return;
}
```


Last Updated : Tuesday, January 19, 2010

GetDeviceParameters Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method reads individual neoVI device parameters.

C/C++ Declare

```
int icsneoGetDeviceParameters(int hObject, char *pParameters, char *pValues, short ValuesLength);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pParameters
[in] This is an array containing parameter names. Each parameter is separated by a comma. The parameter names are matched without regard to case. All spaces are ignored. The size of this array must be 1024 bytes or less. The format of the array is:

ParameterName,ParameterName, , . . .

See Valid Parameters for a list of parameter names for each device and supported network.

See examples below on how to build a parameter string.

pValues
[out] This array will contain the values requested in the pParameters array. The values will be separated by comma's and in the order of the parameter names specified in the pParameters array. If a parameter name is not recognized the word "Error" will be placed in that value's location. If the pValues array length (specified by the ValuesLength parameter) is not long enough to store all of the values, the retrieval of parameter values will end and only a portion of the values will have be read and stored. The return value of the function, if greater than 0, will indicate the number of values read.

Return Values

-1 if there was an error while reading parameter values from the device. A return value greater than 0 indicates the total number of parameters read. A return value of 0 indicates that ValueLength was greater than 1024. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

It is ineffecient to use this function to read one parameter value at a time. If multiple parameters need to be read, combine them into a long string and call this function once.

Example

C/C++ Example

```
char pGetFireParms[] = "network_enables,can1/Mode,can1/Baudrate";
char Values[500];
int iRetVal;

iRetVal = icsneoGetDeviceParameters(hObject, pGetFireParms, Values, 499);
```

Last Updated : Tuesday, January 19, 2010

SetDeviceParameters Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method changes neoVI device parameters.

C/C++ Declare

```
int icsneoSetDeviceParameters(int hObject, char *pParmValue, int *pErrorIndex, int bSaveToEERPROM);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pParmValue
[in] This is an array containing parameter names and values. Each parameter and value is separated by a equal sign, and each parameter/value pairing is separated by a comma. The parameter names are matched without regard to case. All spaces are ignored. The size of this array must be 1024 bytes or less. The format of the array is:

ParameterName=Value,ParameterName=Value, . . .

See Valid Parameters for a list of parameter names for each device and supported network.

See examples below on how to build a parameter/value string.

pErrorIndex
[out] If there are any errors detected within the pParmValue parameter, this value will indicate index of the parameter where the first error was found. The index is zero-based.

bSaveToEEPROM
[in] This value determines if the parameter changes are permanent or will be lost when the device is power-cycled. Set the value to 1 to write the changes to EEPROM, 0 to keep the changes restricted to RAM.

Return Values

1 if the changes are successful. -1 if there was an error while writing the changes to the device. 0 if there is an error detected within the pParmValue array. If the return value is 0, indicating an error, check the pErrorIndex to get the index of the first error detected within the pParmValue array. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

It is ineffecient to use this function to write one parameter change at a time. If multiple parameters need to be changed, combine them into a long string and call this function once.

Example

C/C++ Example

```
char SetFireParms[100];
char Values[500];
int iRetVal;
int iErrorIndex;
```

```
unsigned short NetworkEnables = 0xFFFF;

sprintf(SetFireParms, "network_enables=%d,can1/Baudrate=9,can1/Mode=0", NetworkEnables);

 iRetVal = icsneoSetDeviceParameters(hObject, SetFireParms, &iErrorIndex, 1);
```


Last Updated : Tuesday, January 19, 2010

ScriptStart Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method starts the execution of a script that has been downloaded to a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptStart(int hObject, int iLocation);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iLocation
[in] Specifies the physical location of the script to be executed on the neoVI device. Valid values are:

SCRIPT_LOCATION_FLASH_MEM = 0 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_SDCARD = 1 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_VCAN3_MEM = 2 (Valid only on a ValueCAN 3 device)

These values are defined in the icsnVC40.h file

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_INVALID_SCRIPT_LOCATION = 213
NEOVI_ERROR_DLL_SDCARD_NOT_INSERTED = 214
NEOVI_ERROR_DLL_SCRIPT_START_ERROR = 218

Remarks

The script must have been successfully downloaded to the neoVI using LoadScript. Use ScriptStop to suspend execution of the script. If the connected device is a ValueCAN 3 and a location other than SCRIPT_LOCATION_VCAN3_MEM will generate an error.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;

printf("Attempting to start the script\n");
iRetVal = icsneoScriptStart(hObject, DefaultScriptLocation);
if(iRetVal == 0)
{
    printf("Failed to start the script API Error\n");
}
else
{
    printf("Successfully started the script\n");
}
```

Last Updated : Tuesday, January 19, 2010

ScriptStop Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method stops the execution of a script that is running on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptStop(int hObject);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error.

Remarks

If a script is executing on the neoVI calling this method will stop it. The script will still be present on the device and can be started again by ScriptStart. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;

iRetVal = icsneoScriptStop(m_hObject);
if(iRetVal == 0)
{
    printf("Failed to Stop the script API Error\n");
}
else
{
    printf("Successfully stopped the script\n");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptLoad Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method downloads a script to a connected neoVI device into a specified location.

C/C++ Declare

```
int _stdcall icsneoScriptLoad(int hObject, const unsigned char *bin, unsigned long
len_bytes, int iLocation);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

bin
[in] An array of bytes that represent a compiled script. These bytes are contained in a header file called cmvspy.h. This file is created by Vehicle Spy when a script is compiled. Please see Vehicle Spy documentation for details.

len_bytes
[in] Specifies the number of bytes represented by the *bin* parameter

iLocation
[in] Specifies the physical location to where the script will be loaded on the neoVI device. Valid values are as follows:

SCRIPT_LOCATION_FLASH_MEM = 0 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_SDCARD = 1 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_VCAN3_MEM = 2 (Valid only on a ValueCAN 3 device)

These values are defined in the icsnVC40.h file

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_INVALID_SCRIPT_LOCATION = 213
NEOVI_ERROR_DLL_SDCARD_NOT_INSERTED = 214
NEOVI_ERROR_DLL_SDCARD_WRITE_ERROR = 216
NEOVI_ERROR_DLL_SCRIPT_ERROR_DOWNLOADING_SCRIPT = 220
NEOVI_ERROR_DLL_SDCARD_READ_ERROR = 217

Remarks

The script will be stored on the device in the specified location. If the location is SCRIPT_LOCATION_FLASH_MEM or SCRIPT_LOCATION_SDCARD the script will persist in the device in that location until cleared by ScriptClear. If the neoVI device is booted (plugged in to power) without being connected to a USB port the stored script will execute. The location SCRIPT_LOCATION_VCAN3_MEM applies only to the ValueCAN 3 device and the storage will be cleared when power is removed from the device.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;
unsigned long NumBinBytes;
NumBinBytes = CM_EXE_SIZE; //from the cmvspy.h file. The length of the compiled script
//ucharConfigurationArrayPM is defined in cmvspy.h.
//It is a pointer to the array of compiled script bytes

 iRetVal = icsneoScriptLoad(hObject, ucharConfigurationArrayPM, NumBinBytes,
DefaultScriptLocation);
if(iRetVal == 0)
{
    printf("\nFailed to load the script into the neo device);
}
else
{
    printf("\nSuccessfully loaded the script into the neoVI");
}
```

ScriptClear Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method clears a script from a specific location on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptClear(int hObject, int iLocation);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iLocation
[in] Specifies the physical location of the script to be cleared on the neoVI device. Valid values are:

SCRIPT_LOCATION_FLASH_MEM = 0 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_SDCARD = 1 (Valid only on a neoVI Fire or neoVI Red)
SCRIPT_LOCATION_VCAN3_MEM = 2 (Valid only on a ValueCAN 3 device)

These values are defined in the icsnVC40.h file

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_INVALID_SCRIPT_LOCATION = 213
NEOVI_ERROR_DLL_SDCARD_NOT_INSERTED = 214
NEOVI_ERROR_DLL_SDCARD_WRITE_ERROR = 216
NEOVI_ERROR_DLL_SCRIPT_ERROR_CLEARING_SCRIPT = 221

Remarks

If a script exists in the specified location it will be erased from that location. If the script is running it will be stopped. Any function blocks that are running will be stopped.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;
//Clear the script from it's storage location.
//Both SCRIPT_LOCATION_FLASH_MEM and SCRIPT_LOCATION_SDCARD are persistent.
//On a ValueCAN 3, SCRIPT_LOCATION_VCAN3_MEM will clear when the device
//loses power or resets.

iRetVal = icsneoScriptClear(hObject, DefaultScriptLocation);
if(iRetVal == 0)
{
    printf("\nFailed to clear the script);
}
else
```

```
{
    printf("\nSuccessfully cleared the script");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptStartFBlock Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method starts the specified function block within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptStartFBlock(int hObject, unsigned int iFunctionBlockIndex);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iFunctionBlockIndex
[in] The index value of the function block to start

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_FUNCBLOCK_INDEX = 219
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Remarks

The script containing the specified function block must have been successfully downloaded to the neoVI using LoadScript. The valid index values for a function blocks within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;

iRetVal = icsneoScriptStartFBlock(hObject, Function_Block_1);
if(iRetVal == 0)
{
    printf("\nFailed to start the function block);
}
else
{
    printf("\nSuccessfully started the function block");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptGetFBlockStatus Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the run status of a specified function block within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptGetFBlockStatus(int hObject, unsigned int iFunctionBlockIndex,
int *piStatus);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iFunctionBlockIndex
[in] The index value of the function block to start

piStatus
[out] 0 = stopped 1 = running

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_FUNCBLOCK_INDEX = 219
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING= 226

Remarks

The script containing the specified function block must have been successfully downloaded to the neoVI using ScriptLoadScript. Execution of the script must have been started by using ScriptStartScript. The valid index values for function blocks within a script can be found in the cmvspy.vs3cmb.h file (Produced by Vehicle Spy. Please see Vehicle Spy documentation).

Example

C/C++ Example:

```
 int iRetVal;
int iRunStatus;
unsigned long lLastErrNum;

iRetVal = icsneoScriptGetFBlockStatus(hObject, Function_Block_1, &iRunStatus);
if(iRetVal == 0)
{
    printf("\nFailed to check function block status);
}
else
{
    printf("\nFunction block status = %s\r\n", iRunStatus == 0 ? "Stopped" : "Running"
);
}
```

ScriptStopFBlock Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method stops the execution of a specified function block within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptStopFBlock(int hObject, unsigned int iFunctionBlockIndex);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iFunctionBlockIndex
[in] The index value of the function block to stop

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_FUNCBLOCK_INDEX = 219
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING= 226

Remarks

The script containing the specified function block must have been successfully downloaded to the neoVI using LoadScript. Execution of the script must have been started by StartScript. Execution of the function block must have been started using StartFBlock. The valid index values for function blocks within a script can be found in the cmvspy.vs3cmb.h file (Produced by Vehicle Spy. Please see Vehicle Spy documentation).

Example

C/C++ Example:

```
int iRetVal;
unsigned long lLastErrNum;

iRetVal = icsneoScriptStopFBlock(hObject, Function_Block_1);
if(iRetVal == 0)
{
    printf("\nFailed to stop the function block.);
}
else
{
    printf("\nSuccessfully stopped the function block");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptGetScriptStatus Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the status of the script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptGetScriptStatus(int hObject, unsigned int iFunctionBlockIndex,
int *piStatus);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iFunctionBlockIndex
[in] The index value of the function block

piStatus
[out] 0 = Stopped 1 = Running

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Remarks

The script must have been successfully downloaded to the neoVI using ScriptLoadScript.

Example

C/C++ Example:

```
 int iRetVal;
int iStatus;
unsigned long lLastErrNum;

iRetVal = icsneoScriptGetScriptStatus(hObject, &iStatus);
if(iRetVal == 0)
{
    printf("\nFailed to get the script status. API Error = %d\r\n", lLastErrNum);
}
else
{
    printf("\nScript status = %s\r\n", iStatus == 0 ? "Stopped" : "Running");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptReadAppSignal Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to read an application signal from a script running on a neoVI device.

C/C++ Declare

```
 int _stdcall icsneoScriptReadAppSignal(int hObject, unsigned int iIndex, double
*dValue);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

unsigned int iIndex
[in] The index value of the transmit message to read

double *dValue
[in] Contains the current value of the application signal.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_APPSIG_INDEX = 225
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Remarks

The script containing the specified application signal must have been successfully downloaded to the neoVI using ScriptLoadScript. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for application signals within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;
double dValue = 0;

iRetVal = icsneoScriptReadAppSignal(hObject, App_Signal_1, &dValue);
if(iRetVal == 0)
{
    printf("\nFailed to read the application signal.);
}
else
{
    printf("\nApplication signal = %f\r\n", dValue);
}
```


Last Updated : Tuesday, January 19, 2010

ScriptWriteAppSignal Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to set the value of an application signal in a script running on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptWriteAppSignal(int hObject, unsigned int iIndex, double
dValue);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the application signal.

dValue
[in] The new value of the application signal.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_APPSIG_INDEX = 225

Remarks

The script containing the specified application signal must have been successfully downloaded to the neoVI using ScriptLoad. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for application signals within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;
double dValue;

dValue = 999;
iRetVal = icsneoScriptWriteAppSignal(hObject, App_Signal_1, dValue);
if(iRetVal == 0)
{
    printf("\nFailed to write the application signal. API Error = %d", lLastErrNum);
}
else
{
    printf("\nApplication signal write succeeded\r\n");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptReadRxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to read the parameters for a receive message defined with a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptReadRxMessage(int hObject, unsigned int iIndex, icsSpyMessage
*pRxMessageMask, icsSpyMessage *pRxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the transmit message to read

pTxMessageMask
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be loaded with the requested receive message mask.

pTxMessage
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be loaded with the requested receive message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Remarks

The script containing the specified receive message must have been successfully downloaded to the neoVI using LoadScript. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for receive messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
int iRetVal;
int i;
unsigned long lLastErrNum;
icsSpyMessage MsgMask, Msg;


iRetVal = icsneoScriptReadRxMessage(hObject, Receive_Msg_1, &MsgMask, &Msg);
if(iRetVal == 0)
{
```

```
    printf("\nFailed to read the receive message.");
}
else
{
    printf("\nRead the receive message from the script.");
    printf("ArbID = %X Data bytes: ", Msg.ArbIDOrHeader);
    for(i = 0; i < 8; i++)
    {
        printf("%02X ", Msg.Data[i]);
    }
}
```


Last Updated : Tuesday, January 19, 2010

ScriptReadTxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to read the parameters for a transmit message defined within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptReadTxMessage(int hObject, unsigned int iIndex, icsSpyMessage
*pTxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

unsigned int iIndex
[in] The index value of the transmit message to read

icsSpyMessage *pTxMessage
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be loaded with the requested transmit message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING= 226

Remarks

The script containing the specified transmit message must have been successfully downloaded to the neoVI using LoadScript. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for transmit messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
 int iRetVal;
int i;
unsigned long lLastErrNum;
icsSpyMessage Msg;
iRetVal = icsneoScriptReadTxMessage(hObject, TestMessage1, &Msg);

if(iRetVal == 0)
{
    printf("\nFailed to read the transmit message.);
}
else
{
    printf("\nRead the transmit message from the script:");
    printf("ArbID = %X Data bytes: ", Msg.ArbIDOrHeader);
```

```
    for(i = 0; i < 8; i++)
    {
        printf("%02X ", Msg.Data[i]);
    }
}
```


Last Updated : Tuesday, January 19, 2010

ScriptWriteRxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to alter a receive message defined within script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptWriteRxMessage(int hObject, unsigned int iIndex, icsSpyMessage
*pRxMessageMask, icsSpyMessage *pRxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the transmit message to read

pRxMessageMask
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be used to change the specified receive message mask.

pRxMessage
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be used to change the specified receive message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224

Remarks

The script containing the specified receive message must have been successfully downloaded to the neoVI using ScriptLoad. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for receive messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

```
 int iRetVal;
unsigned long lLastErrNum;
icsSpyMessage MsgMask, Msg;

//first read the existing message to get a baseline
iRetVal = icsneoScriptReadRxMessage(hObject, Receive_Msg_1, &MsgMask, &Msg);

if(iRetVal == 0)
{
    iRetVal = icsneoGetLastAPIError(hObject, &lLastErrNum);
    printf("\nFailed to read the receive message before writing it);
    printf("\nPress a key to continue");
```

```
}
Msg.ArbIDOrHeader = 0x030;
memset(Msg.Data, 0x03, 8);
iRetVal = icsneoScriptWriteRxMessage(hObject, Receive_Msg_1, &MsgMask, &Msg);

if(iRetVal == 0)
{
    printf("\nFailed to write the receive message.);
}
else
{
    printf("\nWrote the transmit message to the script");
}
```


Last Updated : Tuesday, January 19, 2010

ScriptWriteTxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to alter a transmit message defined within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptWriteTxMessage(int hObject, unsigned int iIndex, icsSpyMessage
*pTxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the transmit message to read

pTxMessage
[in] This is the address of an instance of an allocated icsSpyMessage structure. The structure will be used to change the specified transmit message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226

Remarks

The script containing the specified transmit message must have been successfully downloaded to the neoVI using ScriptLoad. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for transmit messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see the Vehicle Spy documentation.

Example

C/C++ Example:

Last Updated : Tuesday, January 19, 2010

ScriptReadISO15765 _2_TxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to read the parameters for an ISO15765-2 long transmit message defined within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptReadISO15765_2_TxMessage(int hObject, unsigned int iIndex,
stCM_ISO157652_TxMessage *pTxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the transmit message to read

pTxMessage
[out] An instance of an allocated stCM_ISO157652_TxMessageStructure structure. The structure will be loaded with the requested transmit message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224

Remarks

The script containing the specified transmit message must have been successfully downloaded to the neoVI using LoadScript. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for transmit messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see Vehicle Spy documentation.

Example

C/C++ Example:

Last Updated : Tuesday, January 19, 2010

ScriptWriteISO15765_2_TxMessage Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method is used to change the parameters for an ISO15765-2 long transmit message defined within a script on a neoVI device.

C/C++ Declare

```
int _stdcall icsneoScriptWriteISO15765_2_TxMessage(int hObject, unsigned int iIndex,
stCM_ISO157652_TxMessage *pTxMessage);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

iIndex
[in] The index value of the transmit message to read

pTxMessage
[in] This is the address of an instance of an allocated stCM_ISO157652_TxMessageStructure structure. The structure will be used to change the specified transmit message.

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224

Remarks

The script containing the specified transmit message must have been successfully downloaded to the neoVI using ScriptLoad. The script must also have been started using ScriptStart. This function will fail if ScriptStop has been called. The valid index values for transmit messages within a script can be found in the cmvspy.vs3cmb.h file that is produced by Vehicle Spy. Please see the Vehicle Spy documentation.

Examples

C/C++ Example:

Last Updated : Tuesday, January 19, 2010

GetRTC Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method returns the value of the real-time clock on a connected neoVI device.

C/C++ Declare

```
int icsneoGetRTC(int hObject, icsSpyTime *pTime);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pTime
[in] The address of a icsSpyTime structure. This structure is defined in the file icsSpyDataCommon.h

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

Example

C/C++ Example:

Last Updated : Tuesday, January 19, 2010

SetRTC Method - intrepidcs API

C/C++ declare - Parameters - Return Values - Remarks - C/C++ example

This method sets the value of the real-time clock on a connected neoVI device.

C/C++ Declare

```
int icsneoSetRTC(int hObject, icsSpyTime *pTime);
```

Parameters

hObject
[in] Specifies the driver object created by OpenNeoDevice.

pTime
[in] The address of a icsSpyTime structure. This structure is defined in the file icsSpyDataCommon.h

Return Values

1 if the function succeeded. 0 if it failed for any reason. GetLastAPIError must be called to obtain the specific error. The errors that can be generated by this function are:

NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE = 75

Remarks

Examples

C/C++ Example:

Last Updated : Tuesday, January 19, 2010