

# intrepidcs API - Intrepid Control Systems, Inc.

Overview .....	1
Basics & Win32 Examples .....	2
Visual Studio 32/64 bit Apps .....	3
Visual Basic 6 .....	6
Visual Basic .Net 2008 .....	8
VC++ .....	11
Visual C++ 6 .....	13
Visual C++ 2003 .....	14
Visual C++ 2008 (GUI) .....	15
C# .Net 2008 .....	17
Borland C++ Builder .....	20
Borland Delphi .....	23
LabVIEW .....	26
LabWindows CVI .....	29
Unity Graphics Engine .....	32
Excel (VBA) .....	35
neoVI Hardware Information .....	38
neoVI Network Compatibility Table .....	39
2nd Generation Hardware .....	41
Device Descriptions .....	41
ValueCAN Supported Networks .....	43
neoVI Blue Supported Networks .....	44
3rd Generation Hardware .....	45
Device Descriptions .....	45
ValueCAN3 Supported Networks .....	47
neoVI Fire Supported Networks .....	48
WIN32 API Functions and Types .....	50
Basic Functions .....	55
FindNeoDevices .....	56
OpenNeoDevice .....	60
ClosePort .....	64
FreeObject .....	67

Message Functions .....	69
GetMessages .....	70
TxMessages .....	74
WaitForRxMessagesWithTimeOut .....	79
GetTimeStampForMsg .....	83
EnableNetworkRXQueue .....	86
GetISO15765Status .....	88
SetISO15765RxParameters .....	95
Transmiting Long Messages .....	100
Device Settings Functions .....	102
neoVI Blue and ValueCAN .....	104
GetConfiguration .....	105
SendConfiguration .....	109
neoVI Fire .....	115
GetFireSettings .....	116
SetFireSettings .....	120
ValueCAN3 .....	124
GetVCAN3Settings .....	125
SetVCAN3Settings .....	129
General Device Settings .....	133
SetBitRate .....	134
GetHWFirmwareInfo .....	138
GetDLLFirmwareInfo .....	141
ForceFirmwareUpdate .....	144
GetDeviceParameters .....	146
SetDeviceParameters .....	149
SetReflashDisplayCallbacks .....	152
ClearReflashDisplayCallbacks .....	154
GetRTC .....	156
SetRTC .....	158
Error Functions .....	160
GetLastError .....	161
GetErrorMessages .....	163
GetErrorInfo .....	166
Error Messages .....	173
General Utility Functions .....	185

ValidateHObject .....	186
GetDLLVersion .....	189
StartSockServer .....	191
StopSockServer .....	193
GetPerformanceParameters .....	195
<b>CoreMini Script Functions .....</b>	<b>198</b>
ScriptStart .....	199
ScriptStop .....	202
ScriptLoad .....	205
ScriptClear .....	211
ScriptStartFBlock .....	215
ScriptGetFBlockStatus .....	218
ScriptStopFBlock .....	222
ScriptGetScriptStatus .....	225
ScriptReadAppSignal .....	229
ScriptWriteAppSignal .....	232
<b>Deprecated Functions .....</b>	<b>236</b>
OpenPortEx .....	237
OpenPortEx Hardware type .....	243
FindAllCOMDevices .....	244
FindAllUSBDevices .....	250
EnableNetworkCom .....	251
<b>Structures, Types, and Defines .....</b>	<b>253</b>
Message Structures .....	255
Array vs. message structure .....	265
Message status bit field values .....	267
Intrepid API Data Types .....	276
Valid Device Parameters .....	277
SFireSettingsStructure .....	281
SVCAN3Settings Structure .....	294
neoVI Blue and ValueCAN Configuration Array .....	299
CAN_SETTINGS Structure .....	304
SWCAN_SETTINGS Structure .....	308
LIN_SETTINGS Structure .....	312
ISO9141_KEYWORD2000_SETTINGS Structure .....	314
TextAPI_SETTINGS Structure .....	317

UART_SETTINGS Structure .....	321
ISO9141_KEYWORD2000_INIT_STEPS Structure .....	324
stCM_ISO157652_TxMessage_Structure .....	326
neoVI Network ID List .....	328
NeoDevice Structure .....	329
stAPIFirmwareInfo Structure .....	332
<b>Development FAQ's .....</b>	<b>336</b>
How do I detect and handle disconnects? .....	337
How do I set parameters on a neoVI device? .....	339
How do I open more than one channel on a single piece of hardware? .....	340
How do I communicate on LIN? .....	341
How do I send a Extended Frame or a High Voltage Wakeup or ISO9141/KW2K Init? .....	344
How do I use CoreMini Scripts in an application? .....	345
Part 1: Create the CoreMini Script .....	347
Part 2: Putting cmvspy.vs3cmb to use .....	349
Part 3: Conclusion .....	351
<b>Raw API .....</b>	<b>352</b>
Starting Communications .....	353
Comm Packet .....	355
Device Status Packet (Rx) .....	357
Interface Switch (Tx) .....	359
No Data Message .....	360
CAN Overview .....	361
CAN Rx Packet (Rx) .....	362
CAN Tx Packet (Tx) .....	365
CAN Rx Buffer Overflow (Rx) .....	367
CAN Error State Message (Rx) .....	368
CAN Tx Complete (Rx) .....	369
ISO/KW2K/LIN Tx Packet (Tx) .....	370
ISO/KW2K/LIN Tx Complete or Error Status (Rx) .....	371
ISO/KW2K/LIN Rx Packet (Rx) .....	373
Raw J1850 PWM .....	375
MainPic Buffer Overflow (Rx) .....	376
Main51 Tx FIFO Overflow (Rx) .....	377
Raw Command Summary .....	378
neoVI Green USB Interface .....	381

Unix-like Operating System Support .....	383
J2534 Support .....	384
Vehicle Spy Text API .....	385
neoVI PRO User Interface .....	396
ECU Object .....	400
Labview to Vehicle Spy 3 .....	403
Vehicle Spy Binary File Buffer File Spec .....	404
Contact .....	412

## The intrepidcs API - Create your own software applications

### Overview

The neoVI API provides a simple way to access the neoVI hardware with WIN32 development tools. This documentation describes how to use the API for custom applications. Each API has an example targeted for both C/C++, C#, and Visual Basic (VB). Operational examples are included for [Microsoft](#) Visual C++, [National Instruments](#) LabVIEW, [National Instruments](#) LabWindows CVI, [Borland](#) C++ Builder, [Borland](#) Delphi, and [Microsoft](#) Visual Basic.

Included with neoVI is the "icsneo40.dll" DLL. This DLL is a high performance multi-threaded DLL, capable of supporting many neoVI devices simultaneously. The DLL can be used through static or dynamic linkage.

For applications using older neoVI devices (neoVI **Blue** and ValueCAN 2) which do not use Windows (such as Linux or an embedded system) we specify a [Raw Communications API](#). This allows a non-windows device to operate a neoVI. For newer hardware devices, like neoVI **FIRE/RED** or ValueCAN 3, a [Linux interface](#) is available.

### Getting Started

To get started, review the [Basic Operation](#) topic and the topics describing how to use the API in [Visual Basic](#), [Visual Basic .NET](#), [Visual C++](#), [Visual C#](#), [LabWindows CVI](#), [LabVIEW](#), [Borland C++ Builder](#), [Unity Engine](#), and [Borland Delphi](#).

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, June 29, 2015*

## Basic Operation - intrepidcs API

When you use the intrepidcs API you will do the following:

- 1) Start your application.
- 2) Open the driver and create (if necessary) the neoVI object using the [OpenPort](#) or [OpenNeoDevice](#) method.
- 3) Transmit messages using the [TxMessages](#) method.
- 4) Read messages on the network using the [GetMessages](#) method.
- 5) Optionally readout any errors using the [GetErrorMessages](#) method.
- 6) Repeat steps 3 through 5 while your application is monitoring the network.
- 7) Close the driver when you are not monitoring by calling the [ClosePort](#) method.
- 8) If you want to start monitoring again go back to step 2.
- 9) When your application exits, you must free (destroy) your neoVI object by calling the [FreeObject](#) method.

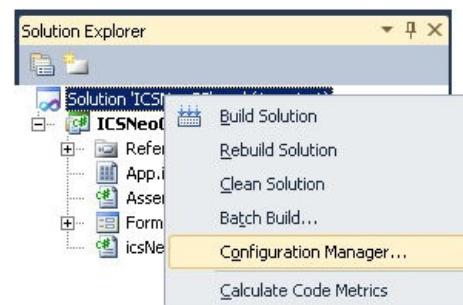
**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, May 27, 2009*

## Using the intrepidcs API in Visual Studio: 32 bit DLL and 64 bit OS - intrepidcs API

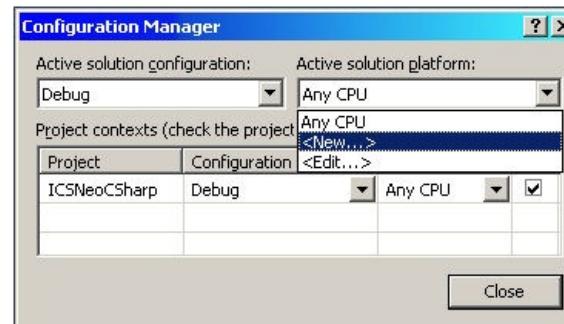
Visual Studio 2005, 2008, and 2010 have different options to compile an application. By default, new programs or programs upgraded from earlier versions of Visual Studio, will be configured as "ANY CPU". This option will run as a 32 bit program on a 32 bit version of Windows and as a 64 bit program in a 64 bit version of Windows. The icsneo40.dll is a 32 bit dll. To ensure an application will function on a 32 bit (x86) OS and a 64 bit (x64) OS, set the Configuration Manager in Visual Studio to x86. The steps to change your configuration and compile options are different depending on the version of Visual Studio, level installed, and your settings. The steps below show how to make this change in Visual Studio 2010 Professional using General settings. For help with your version and level of Visual Studio consult Visual Studio help or [MSDN](#).

The first step is to open the "Configuration Manager". This can be found by right clicking on your Solution and choosing "Configuration Manager" (figure 1).

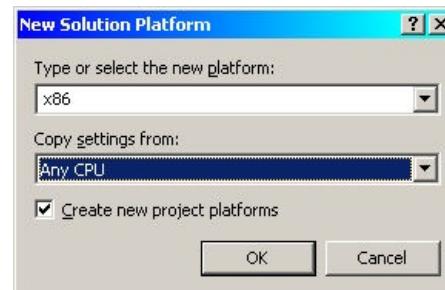
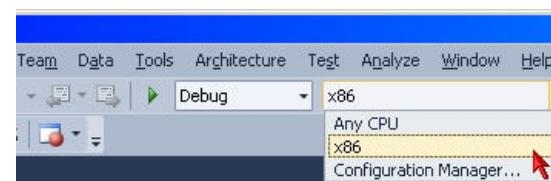


**Figure 1 - Configuration Manager can be found in the Solution Explorer.**

Once in Configuration Manager, select x86 for the Active solution platform. If you do not have this option select "<New...>" (figure 2).

**Figure 2 - Configuration Manager**

For the type, select "x86" and copy the settings from "Any CPU" (figure 3). After a new Platform has been created, it can be selected from the Standard tool bar (figure 4)

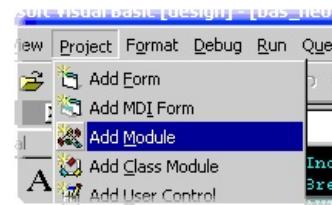
**Figure 3 - Creating a new Platform****Figure 4 - Standard Tool bar selecting platform**



## Using the intrepidcs API in Visual Basic 6 - intrepidcs API

### Setup - Example

To use the intrepidcs API in Visual Basic 6 add the "[bas\\_neoVI.bas](#)" module into your VB project (figure 1). Then, call the methods as defined in the [Basic Operation](#) document.

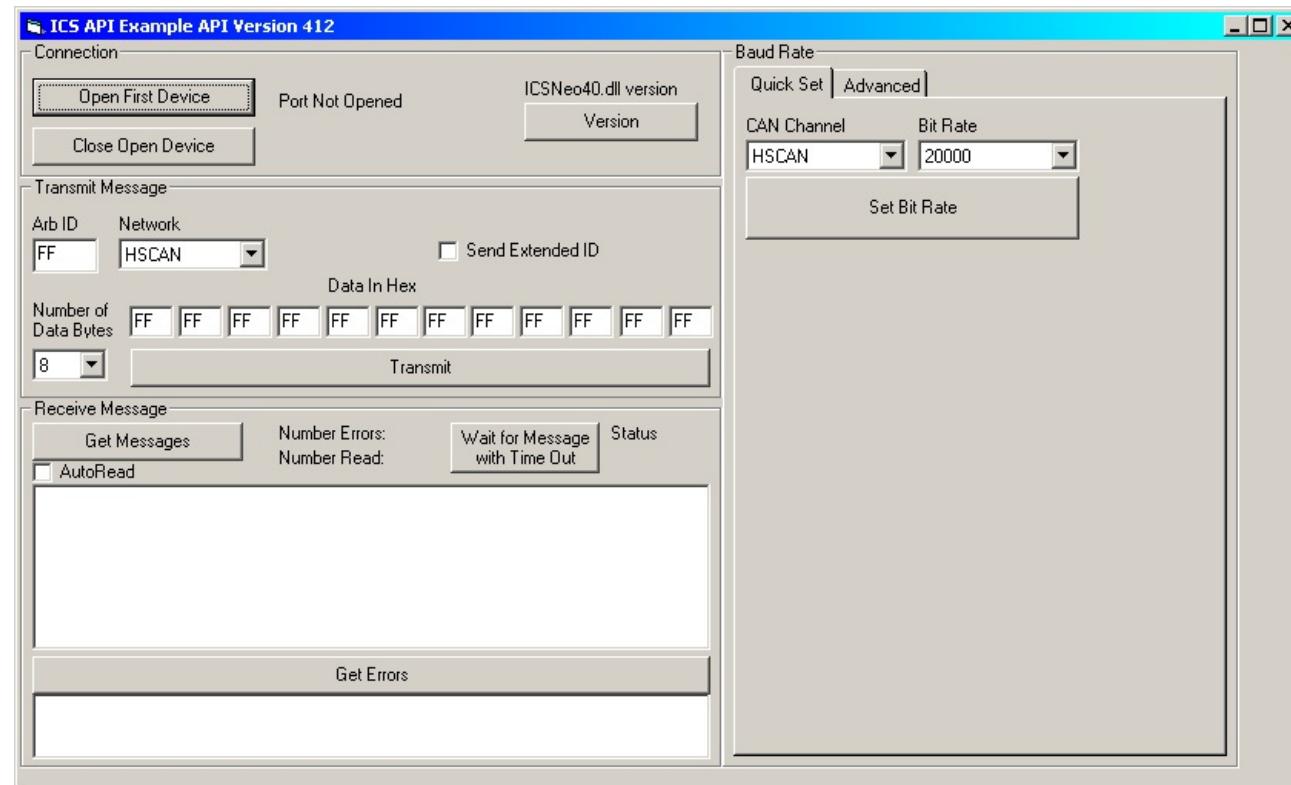


**Figure 1 - Add Module Command From the VB6 Menu.**

### **Example**

A Visual Basic example (Figure 1) is included to show how the API all works together. The project consists of three files: 1) the project file: prjNeoExample.vbp 2) the form file : frmNeoVIExample.frm, and 3) the neoVI module : bas\_neoVI.bas. These are included in the following file: [VBneoVI.zip](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 2 - The Visual Basic Example.**

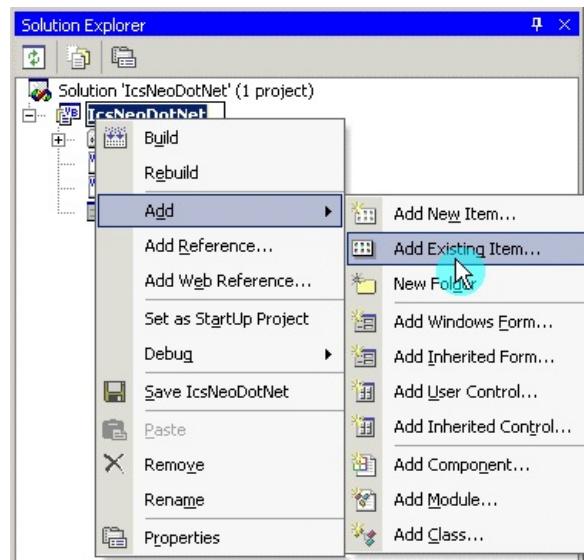
**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Thursday, December 23, 2010

## Using the intrepidcs API in Visual Basic - intrepidcs API

### Setup - Example

To use the intrepidcs API in Visual Basic add the "[bas\\_neoVI.vb](#)" module into your VB project (figure 1) by right clicking on the Solution and selecting Add Existing Item from the Add menu. Open the bas\_neoVI.vb. Then, call the methods as defined in the [WIN32 API Functions and Types](#) Section of this document.



**Figure 1 - Add Module Command From the VB.NET Menu.**

### Example

A Visual Basic Dot Net 2008 example (Figure 1) is included to show how the API all works together. The main project files are as follows: 1) the project file: IcsNeoDotNet.sln 2) the form file : Form1.vb, and 3) the neoVI module : bas\_neoVI.vb. All project files are included in the following file: [VBnet2008.zip](#). This project will open in Visual Studio 2008, 2010, 2012, 2013, and 2015.

The example shows how to open and close communication to the driver, send messages and read messages on the networks.

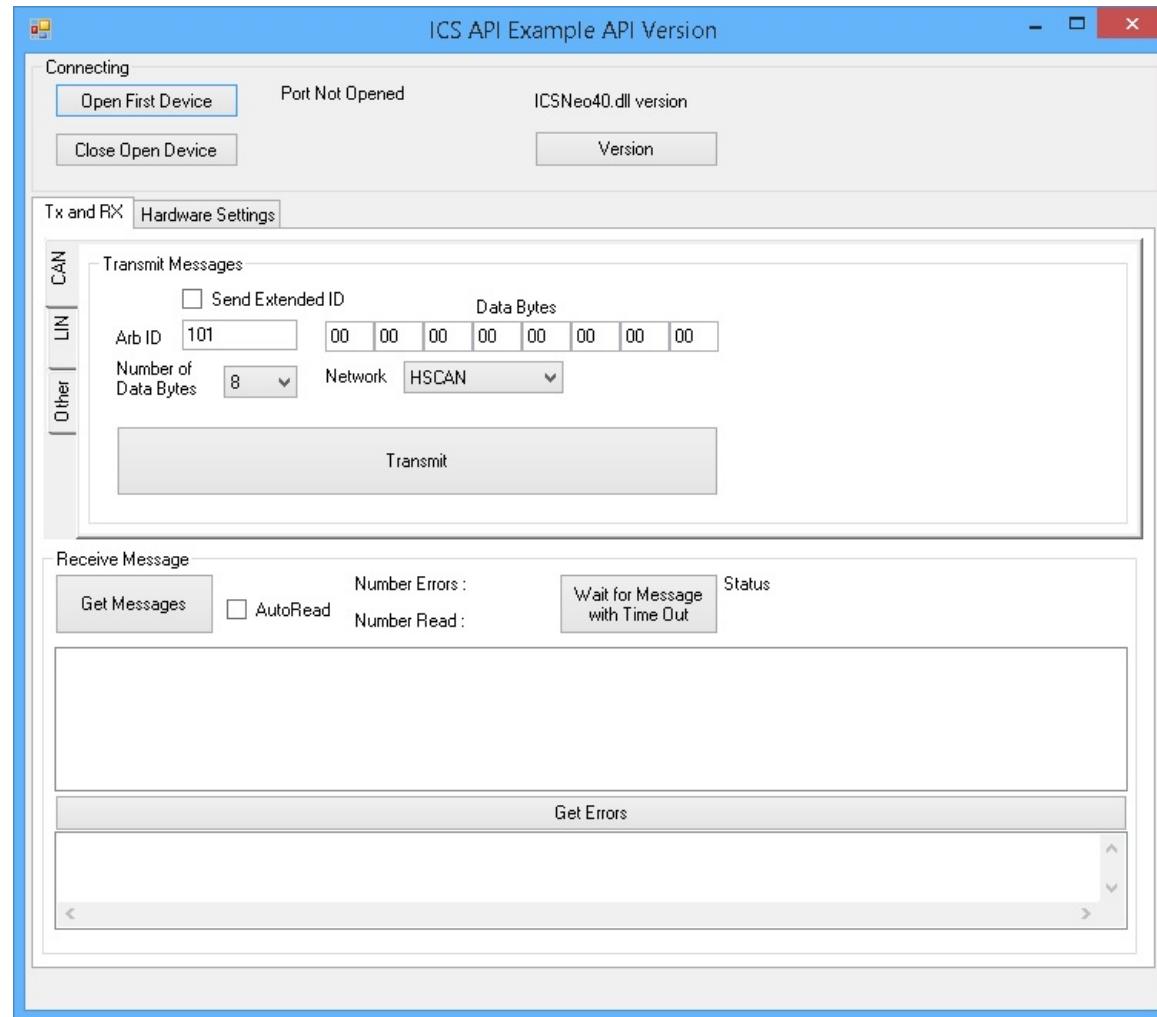


Figure 2 - The Visual Basic Dot Net 2008 Example.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Monday, September 07, 2015



## Using the intrepidcs API in Visual C++ - intrepidcs API

[Setup](#) - [Example](#)

### Do the following steps to use neoVI in Visual C++:

- 1) Start your new project and add the [Dynamic link helper files](#) to your project.
- 2) Add a `#include "icsneo40DLLAPI.h"` to your project

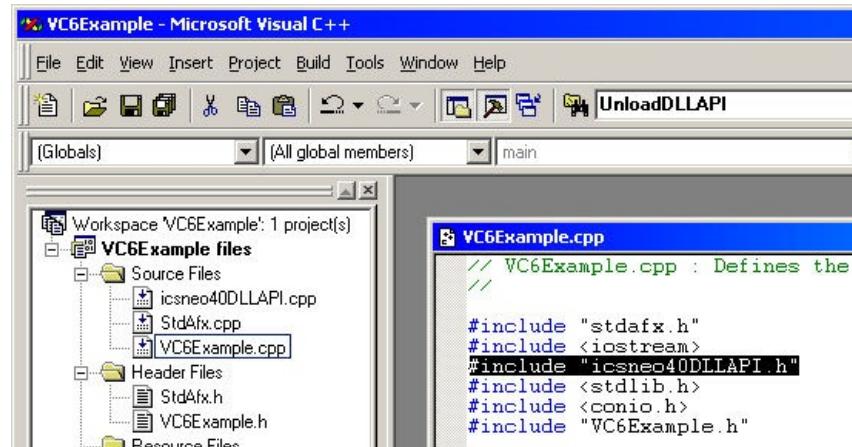


Figure 1 - Link to the "icsnVC40.lib"

- 3) Use the Functions "LoadDLLAPI" to load the functions and "UnloadDLLAPI" to unload the functions. Examples are below.

```
HINSTANCE hDLL;

//----Load the DLL
if(!LoadDLLAPI(hDLL))
{
    //problem, close the application
    printf("Problem loading Library\r\nMake sure icsneo40.dll is installed and accessable\r\nPress any key to Exit");
}

//----Unload the DLL
UnloadDLLAPI(hDLL);
```

- 4) Finally, call the methods as defined in the [Basic Operation](#) document.

Please see the Visual C++ example for more information on using the neoVI API with Visual C++.

VC++ 6 - [Visual C++ 6 example](#)

VC++ 2003 - 2010 - [Visual C++ 2003-2010 example](#)

VC++ 2008 GUI Example - [Visual C++ 2008 example](#)

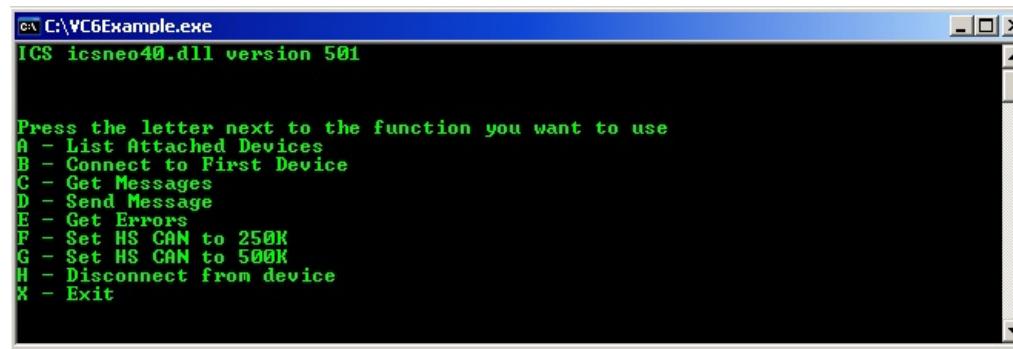
**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 23, 2010*

## Visual C++ Example - intrepidcs API

A Visual C++ example (Figure 1) is included to show how the API all works together. The example files are included in the following file: [VCneoVI.zip \(31kB\)](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 1 - The Visual C++ Example.**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 23, 2010*

## Visual C++ 2003 to 2010 Example - intrepidcs API

A Visual C++ example (Figure 1) is included to show how the API all works together. The example files are included in the following file: [VCNewneoVI.zip](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.

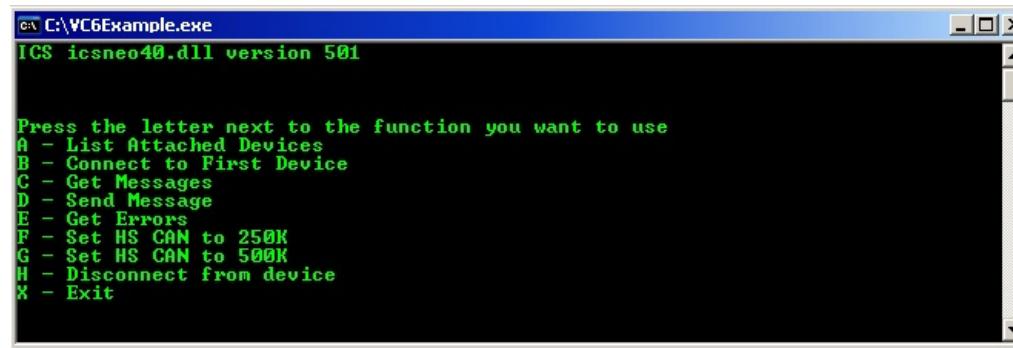


Figure 1 - The Visual C++ Example.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Tuesday, February 15, 2011

## Visual C++ 2008 Example - intrepidcs API

A Visual C++ 2008 example (Figure 1) is included to show how the API all works together. The main project files are as follows: 1) the project file: VS8CNeoExample.sln 2) the form file : Form1.h, and 3) the neoVI module : icsnVC40.h. All the needed project files are included in the following file: [CPnet2008.zip](#). This project will open in Visual Studio 2008, 2010, 2012, 2013, and 2015.

The example shows how to open and close communication to the driver, send messages and read messages on the networks.

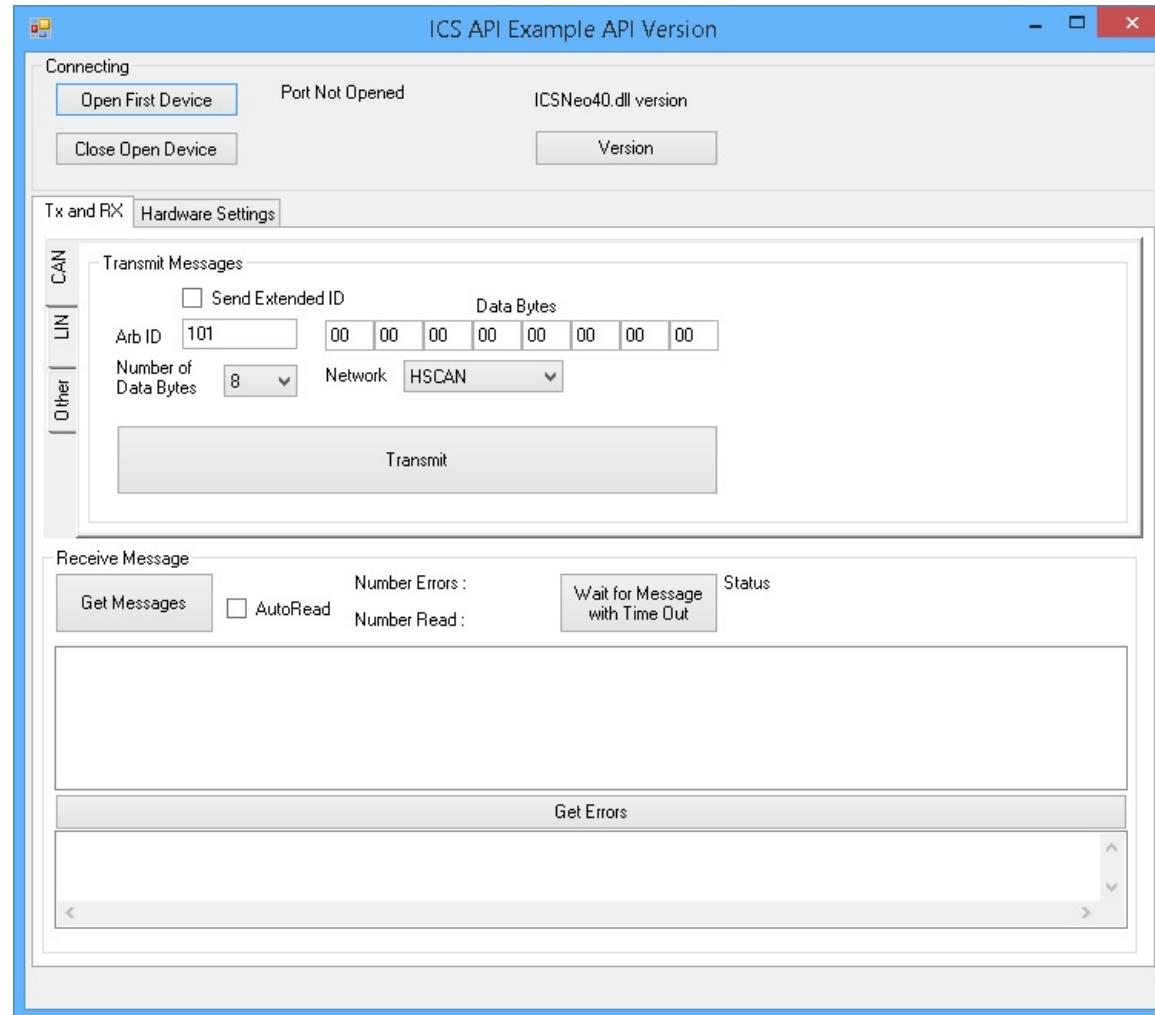


Figure 1 - The Visual C++ 2008 Example.

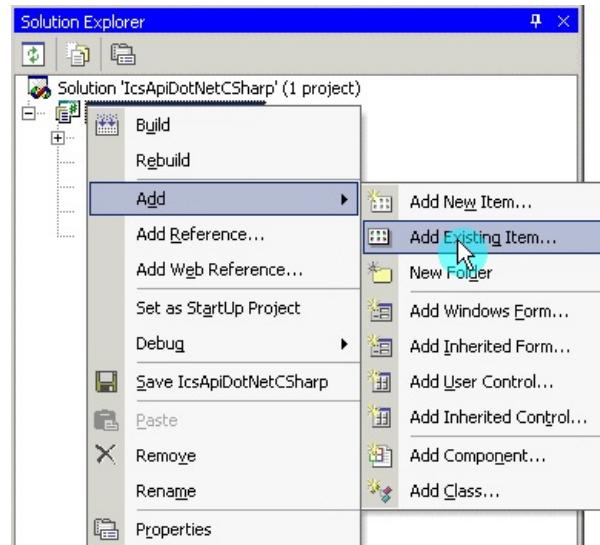
**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, September 07, 2015*

## Using the intrepidcs API in C# - intrepidcs API

### Setup - Example

To use the intrepidcs API in C# add the "[icsNeoClass.cs](#)" Class into your C# project (figure 1). Right click on the solution and select "Add Existing Item" from the "Add" menu. Then, call the methods as defined in the [WIN32 API Functions and Types](#) Section of this document.

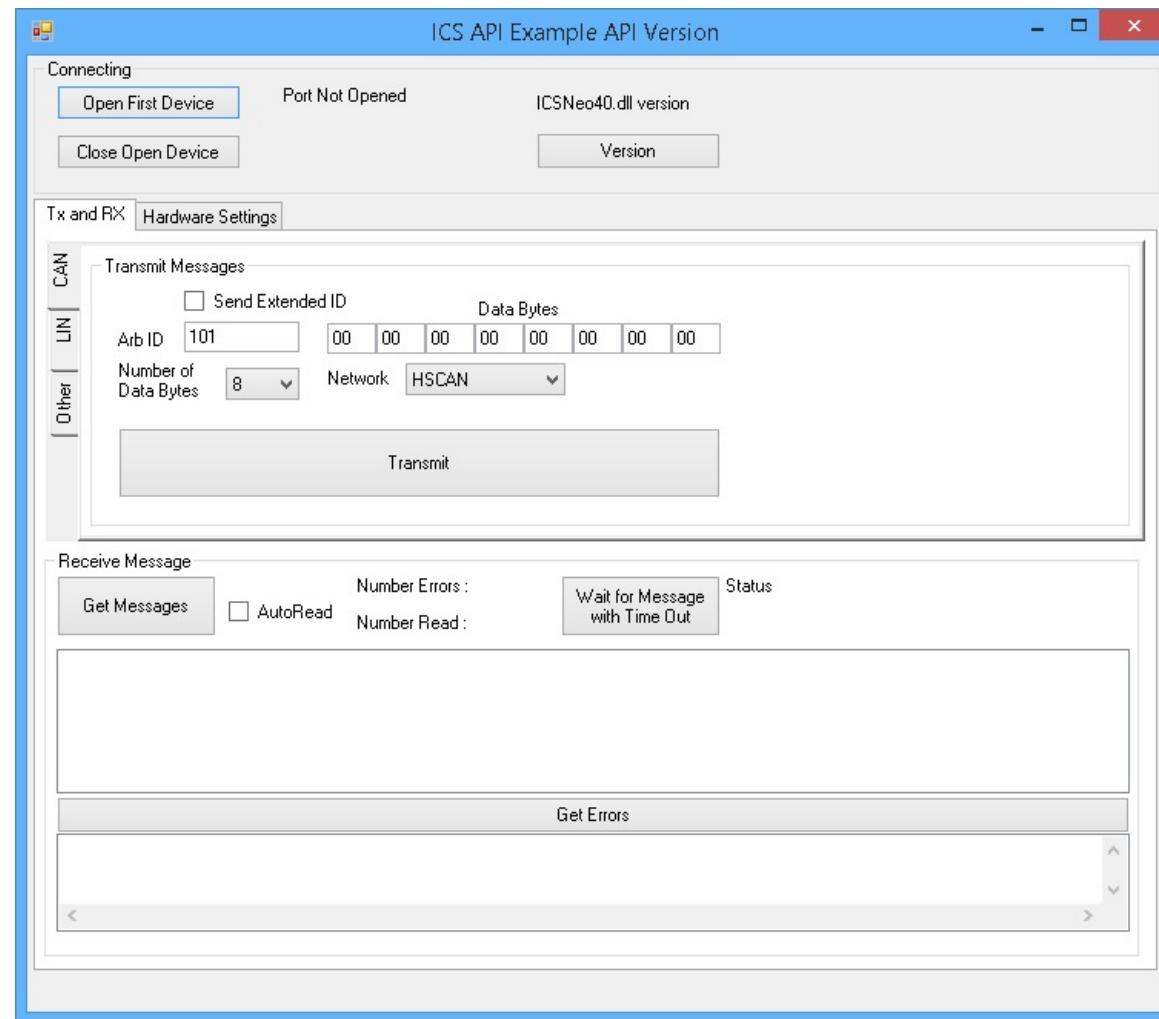


**Figure 1 - Add Existing Item From the Add menu C#.NET Menu.**

### Example

A C# Dot Net 2008 example (Figure 1) is included to show how the API all works together. The main project files are as follows: 1) the project file: IcsApiDotNetCSharp.sln 2) the form file : Form1.cs, and 3) the neoVI module : icsNeoClass.cs. All project files are included in the following file: [CSnet2008.zip](#). This project will open in Visual Studio 2008, 2010, 2012, 2013, and 2015.

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 2 - The C# Dot Net 2008 Example.**

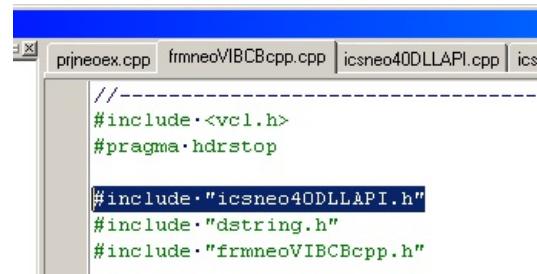
**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Monday, September 07, 2015



**Using the intrepidcs API in Borland C++ Builder - intrepidcs API**[Setup](#) - [Example](#)**Do the following steps to use neoVI in Borland C++ Builder:**

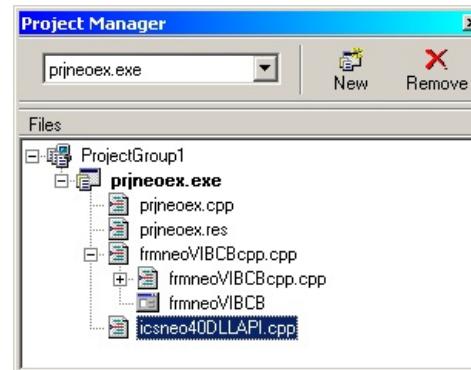
- 1) Start your new project and add the [Dynamic link helper files](#) to your project.
- 2) Add a `#include "icsneo40DLLAPI.h"` to your project



A screenshot of the Borland C++ Builder code editor. The current file is `frmneoVIBCB.cpp`. The code window contains the following lines of code:

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "icsneo40DLLAPI.h"
#include "dstring.h"
#include "frmneoVIBCB.h"
```

**Figure 1 - include for "icsneo40DLLAPI.h"****Figure 2 - "icsneo40DLLAPI.cpp" added to project**

- 3) Use the Functions "LoadDLLAPI" to load the functions and "UnloadDLLAPI" to unload the functions. Examples are below.

```
HINSTANCE hDLL;

//----Load the DLL
if (!LoadDLLAPI (hDLL))
{
    //problem, close the application
```

```
printf("Problem loading Library\r\nMake sure icsneo40.dll is installed and accessable\r\nPress any key to  
Exit");  
}  
  
//-----Unload the DLL  
UnloadDLLAPI(hDLL);
```

4) Finally, call the methods as defined in the he [Basic Operation](#) document.

### Example

A Borland C++ Builder example (Figure 1) is included to show how the API all works together. The example files are included in the following file: [BCBneoVI.zip \(31kB\)](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.

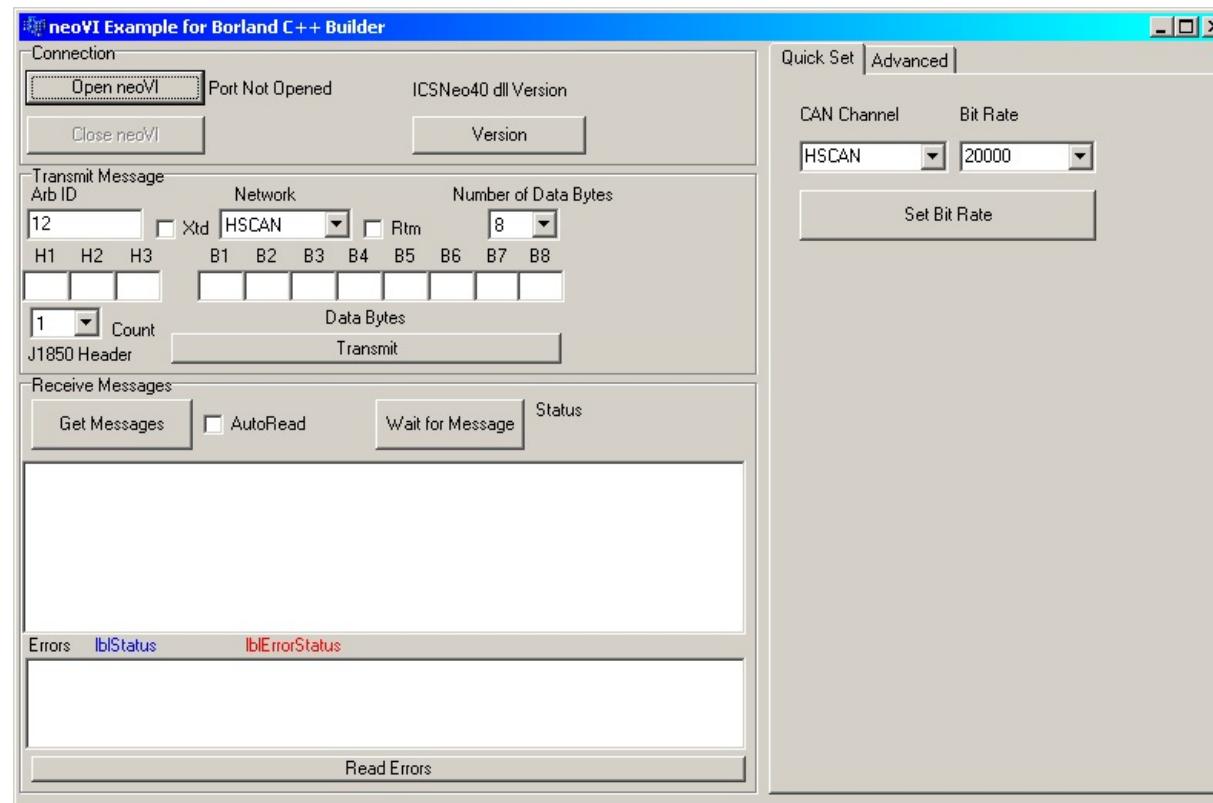


Figure 1 - The Borland C++ Builder Example.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

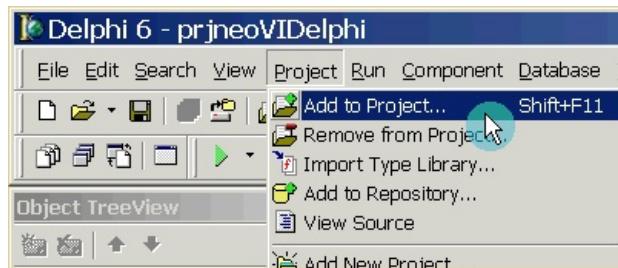
*Last Updated : Thursday, December 23, 2010*

## Using the intrepidcs API in Delphi - intrepidcs API

[Setup](#) - [Example](#)

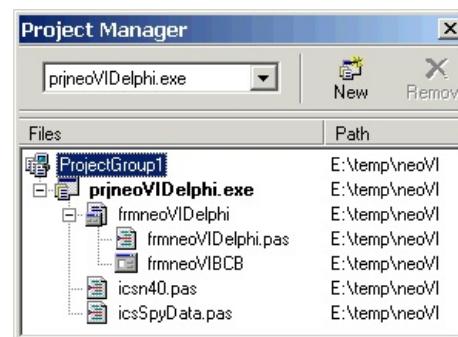
**Do the following steps to use neoVI in Borland Delphi:**

- 1) Copy the import library "[icsn40.pas](#)", and data structure file "[icsSpyData.pas](#)" to your project directory.
- 2) Link to icsn40.pas import library via the "Add to Project..." option (Figure 1). This dialog is accessible via the "Project" pull down menu in Delphi. When the file dialog appears, select the icsn40.pas.



**Figure 1 - Add the Link to the "icsn40.pas"**

- 3) Your project manager will now show the import library icsn40.pas (Figure 2).



**Figure 2 - The import library "icsn40.pas" is loaded.**

- 3) Include icsn40 and icsSpyData in your interface Uses (Figure 3).



The screenshot shows the Borland Delphi IDE with the file 'frmneoVIDelphi.pas' open. The code defines a unit named 'frmneoVIDelphi' with an interface section. The 'uses' section includes standard Windows components like Messages, SysUtils, Variants, Classes, Forms, Graphics, Controls, Dialogs, StdCtrls, ExtCtrls, and specific driver components like icsn40 and icsSpyData. The 'type' section defines a class 'TfrmneoVIBCB' that inherits from TForm, containing two button components: cmdOpen and cmdClose.

```
unit frmneoVIDelphi;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Forms,
  Graphics, Controls, Dialogs, StdCtrls, ExtCtrls,
  icsn40, icsSpyData;

type
  TfrmneoVIBCB = class(TForm)
    cmdOpen: TButton;
    cmdClose: TButton;
  end;
```

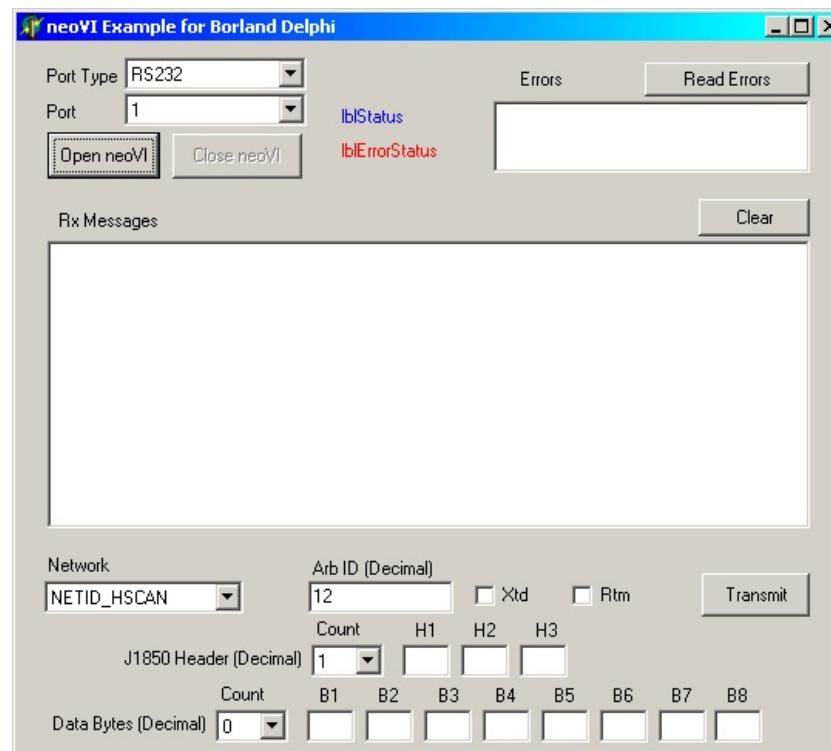
**Figure 3 - Adding to Interface Uses**

5) Finally, call the methods as defined in the [Basic Operation](#) document.

### Example

A Borland Delphi example (Figure 1) is included to show how the API all works together. The example files are included in the following file: [icsnDelphiSample.zip \(14kB\)](#)

The example shows how to open and close communication to the driver, send messages, and read messages on the networks.



**Figure 4 - The Borland Delphi Example.**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 23, 2010*

## Using the API in LabVIEW - intrepidcs API

[Overview](#) - [Steps to use in LabVIEW](#) - [Example](#)

### Overview

The intrepidcs API packages all of the WIN32 methods into LabVIEW LLB. Inside the LLB the sub VIs can be accessed to make it easy for you to use neoVI with National Instruments LabVIEW.

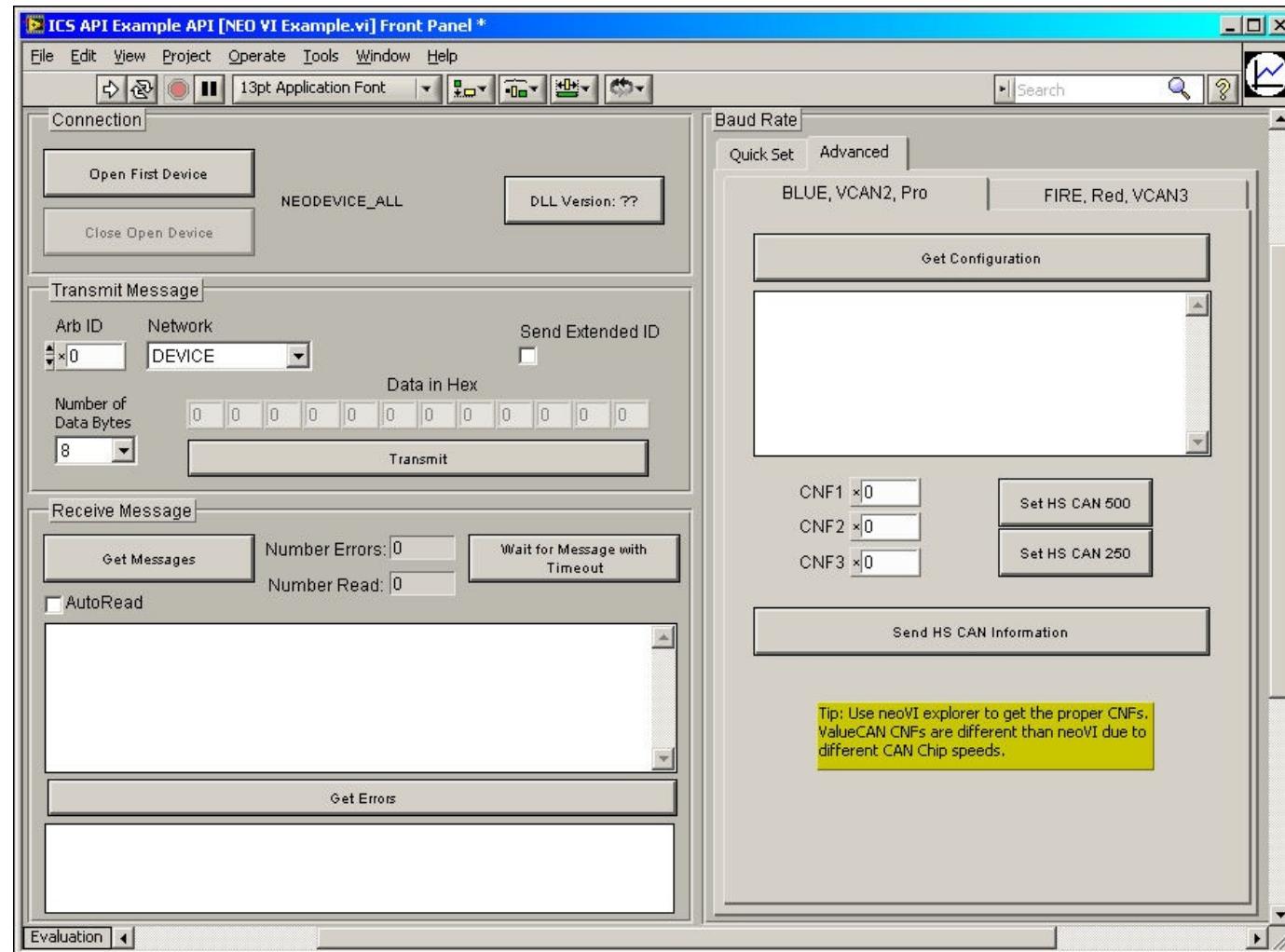
### Steps to use neoVI in LabVIEW

Open the "NEO VI Example.llb" found in the [Nlabv\\_neo.zip \(546kB\)](#) file. When you first access the main VI or sub VI's you may receive a warning regarding a Dependency missing. Make sure that your project is pointing to the icsneo40.dll. This is normally installed to the system32 or SysWOW64 directories.

### Example

A LabVIEW example (Figure 1) is included to show how the API all works together. An example project can be found here: [Nlabv\\_neo.zip \(546kB\)](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 1 - The LabVIEW Example.**



## Using the intrepidcs API in LabWindows CVI - intrepidcs API

[Setup](#) - [Example](#)

### Do the following steps to use neoVI in LabWindows CVI:

- 1) Copy the import library "[icsnLW40.lib](#)", header file "[icsnLW40.h](#)", and data structure file "[icsspyData.h](#)" to your project directory.
- 2) Link to icsnLW40.lib import library by selecting Add Files To Project from the Edit pull down menu. In the "Add Files To Project" dialog select the icsnLW40.lib file (figure 1). You can also the headers from step 1 if you wish. Your project should now list the files you added (figure 2).

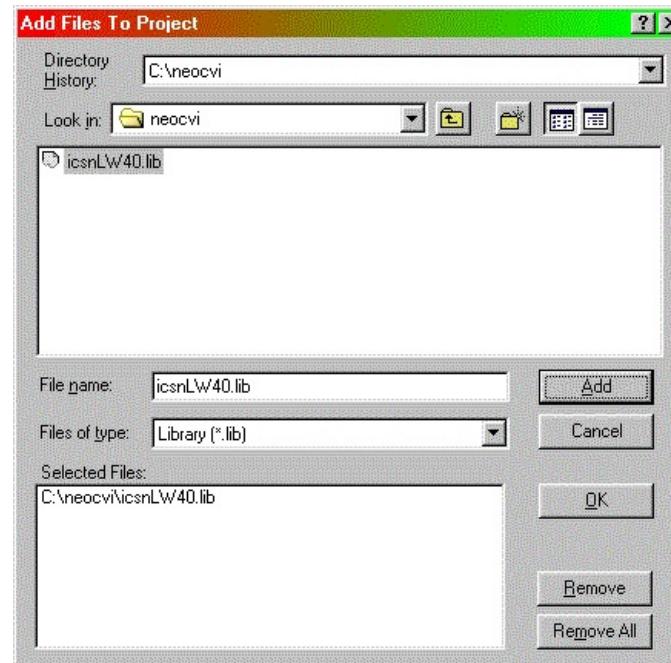


Figure 1 - Link to the "icsnLW40.lib"

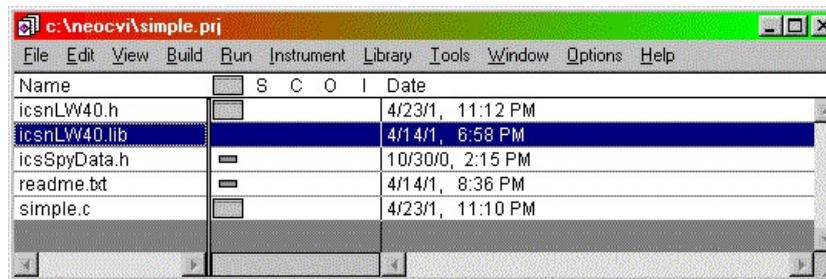


Figure 2 - The project window with the library added.

- 3) Include the header `icsnLW40.h` in your C module (Figure 3).

```
#include <ansi_c.h>
/* This is a simple project that w:
 * functions from an external DLL */

#include <cvirte.h>    /* Needed for
#include <formatio.h>
#include <userint.h>
#include "icsnLW40.h"
#include <stdio.h>

icsSpyMessage stMsgs[20000];  //
icsSpyMessageJ1850 stTxMsg;   //
icsSpyMessage stSTxMsg;      //

long hObject; // handle to the

```

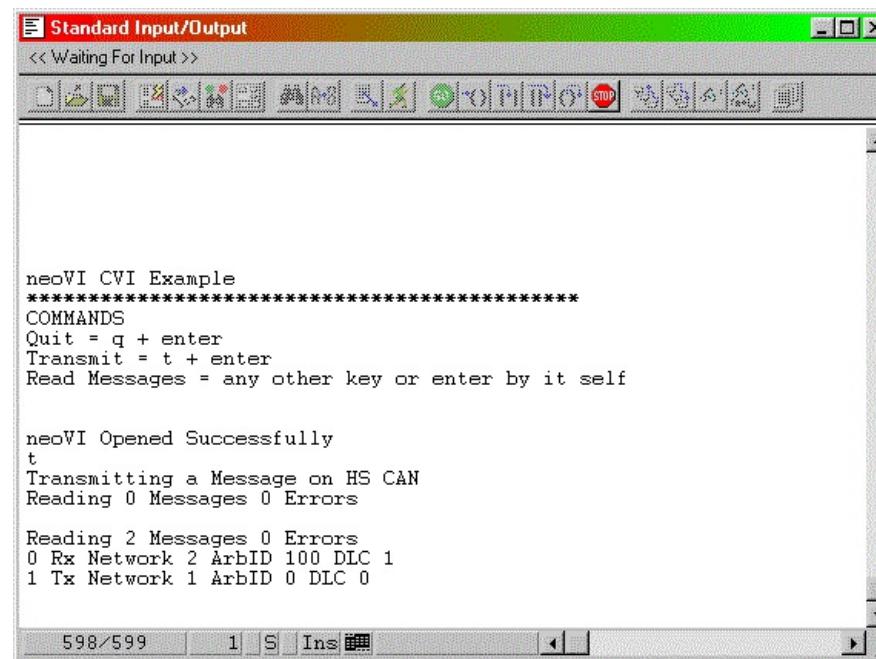
Figure 3 - The `#include` statement in the C module.

- 4) Finally, call the methods as defined in the [Basic Operation](#) document.

## Example

A National Instruments LabWindows example (Figure 1) is included to show how the API all works together. The example files are included in the following file: [LWneoVI.zip \(10kB\)](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 4 - The LabWindows CVI Example.**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 23, 2010*

## Unity3D Graphic Display API

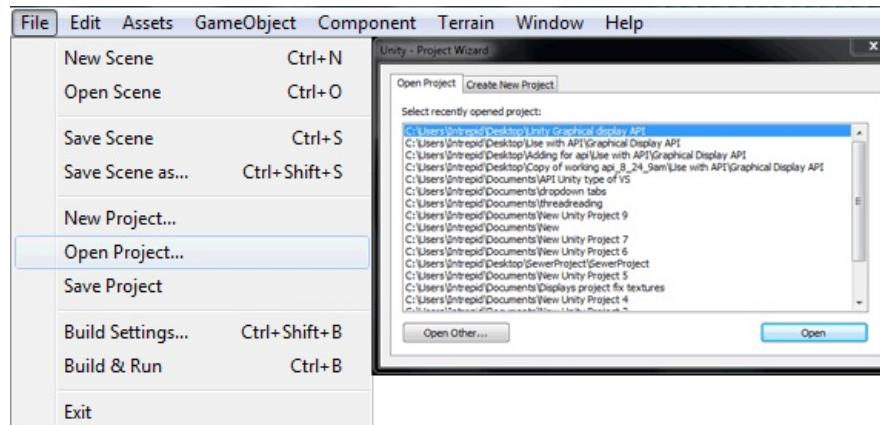
### Example

#### Do the following steps to use Unity3D Graphic Display:

First set the scene for running an .exe file, then an explanation of how the scripts work.

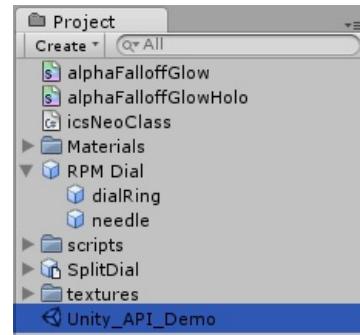
### Example

This is the Unity3D Demo in its entirety included to show how the API all works together; The example files are included in the following file: [Unity3d Graphics Panel Demo \(1537kB\)](#)



**Figure 1 - Opening the Unity 3D Graphics Panel Demo**

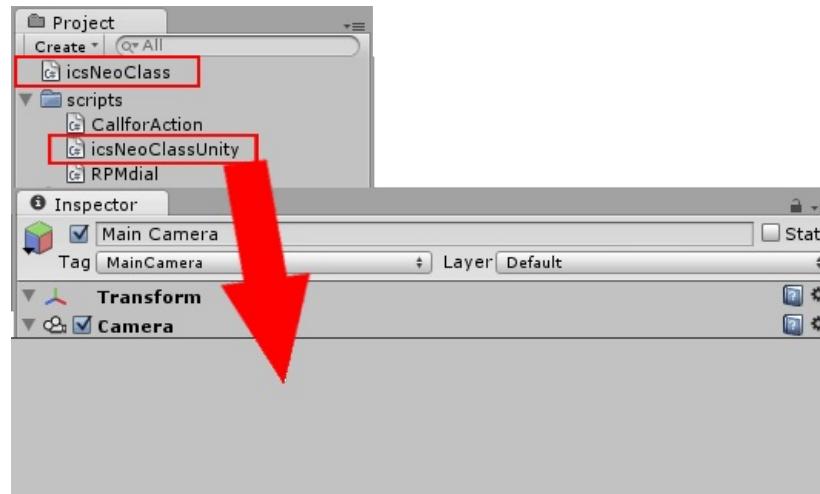
Open Unity3D. Locate and open the project folder called "**Unity Graphic display API**". (Figure 1)



**Figure 2 - Checking to see if the correct scene is open**

In the project panel open the Unity scene called **Unity\_API\_Demo**. Once opened, in the game view is a place holder for the graphical display. This is where all the data will be visually displayed upon execution of the application. (Figure 2)

Next, locate the **icsNeoClass.cs** and make sure it is in the correct folder, this imports the **.DLL** into the project panel (cannot be in any other folder, other then the main "**project asset folder**").



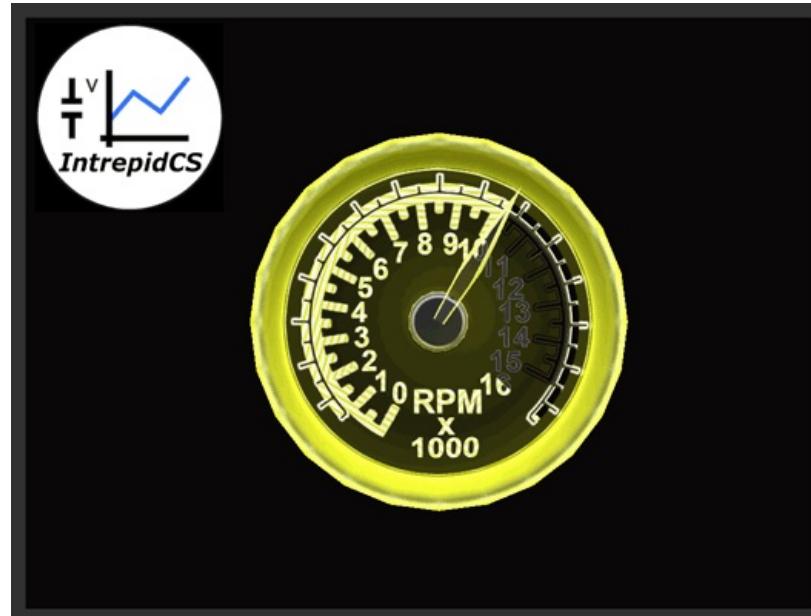
**Figure 3 - Placing scripts in the correct window**

Locate **IcsNeoClassUnity.cs**, located in the Scripts folder. This example script handles the functions of the Intrepid Hardware; open, close, transmit, receive, ect... Then attach the **IcsNeoClassUnity.cs** on the *Main Camera*. (refer to figure 3)

The demo should be all set up and ready to play out at this point. Press the play button near the top of the Unity3D windows to enter play mode and begin the demo. (Figure 4)



**Figure 4 - The play button in Unity 3D when in play mode**

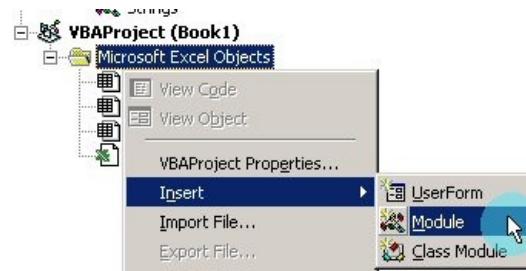


**Figure 5 - This is how your Graphics Panel should look in play mode**

## Using the intrepidcs API in Excel - intrepidcs API

### Setup - Example

To use the intrepidcs API in Excel or other VBA supported application add the "[bas\\_neoVI.vb](#)" module into your project (figure 1) by right clicking on the Project in the VBA Editor and selecting Insert and then Module. Open the bas\_neoVI.vb. Then, call the methods as defined in the [Basic Operation](#) document. The function calls for use in VBA are the same as the calls in Visual Basic 6.

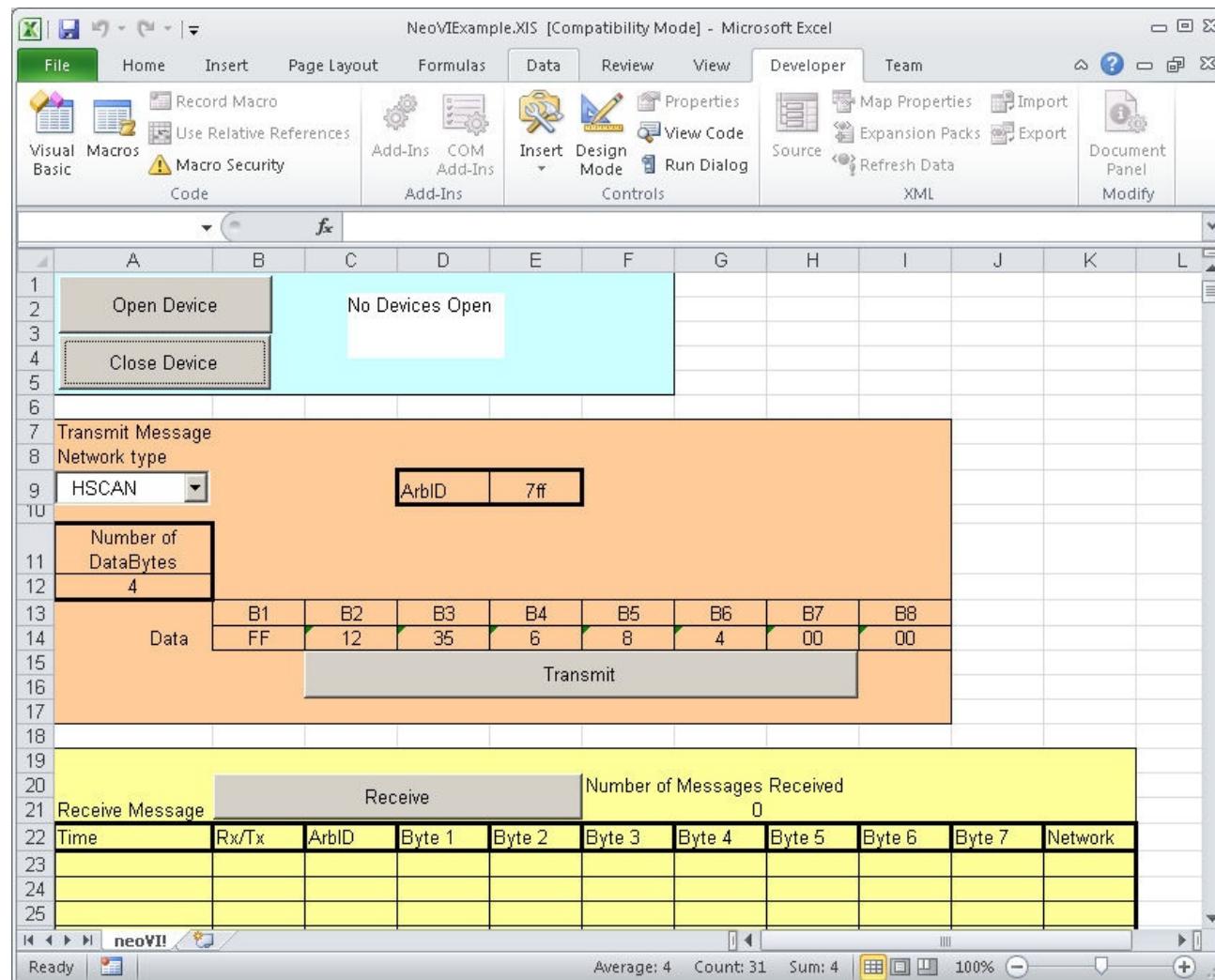


**Figure 1 - Add Module Command From the VB.NET Menu.**

### Example

A VBA Excel example (Figure 1) is included to show how the API all works together. This project only has 1 file, NeoVIEexample.XIS. Make sure macros are enabled to run this example. All the needed project files are included in the following file: [ExcelVBA.zip](#)

The example shows how to open and close communication to the driver, send messages and read messages on the networks.



**Figure 2 - The VBA Excel Example.**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Thursday, December 23, 2010



## **neoVI Hardware Information**

This section describes the supported neoVI devices.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, November 06, 2008*

## Network Compatibility List for neoVI Devices

This table lists currently supported neoVI devices and their compatible network capabilities. The network ID in the first column is used in the [icsSpyMessage](#) structure used by [GetMessages](#) and [TxMessages](#).

For example, if your application was written for a ValueCAN and you are sending and receiving messages on the CAN network using NETID\_HSCAN, you can upgrade to a neoVI Fire or ValueCAN3 and your application does not need to change. NETID\_HSCAN is a network that is supported by all neoVI devices.

The 'Network Name' for each device corresponds directly to the names listed on the pin diagram on the bottom of the device.

Network ID	Default Value	ValueCAN Network Name	neoVI Blue Network Name	neoVI Fire/Red Network Name	ValueCAN3 Network Name	neoVI Yellow Network Name
NETID_HSCAN	1	CAN	HSCAN	HS CAN 1	CAN 1	J1939/CAN 1
NETID_MSCAN	2		MSCAN	MS CAN	CAN 2	J1939/CAN 2
NETID_SWCAN	3		SW CAN	SW CAN		
NETID_LSFTCAN	4		LSFT CAN	LSFT CAN		
NETID_J1708	6		J1708			J1708
NETID_JVPW	8		J1850 VPW	J1850 VPW		J1850
NETID_ISO	9		ISO	ISO K		ISO K
NETID_LIN	16		LIN	LIN1		

*Last Updated : Thursday, October 23, 2008*

## 2nd Generation neoVI Devices

<b>neoVI Blue</b> 	<b>Networks:</b> 2 dual-wire CAN/J1939, single-wire CAN, low-speed fault tolerant CAN, J1850 PWM, J1850 VPW, LIN, J1708, ISO9141/KWP2000. Digital and analog I/O channels.  <b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API
<b>ValueCAN</b> 	<b>Networks:</b> 1 dual-wire CAN/J1939  <b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API  * Also referred to as ValueCAN2
<b>neoVI PRO</b> 	<b>Networks:</b> 2 dual-wire CAN/J1939, single-wire CAN, low-speed fault tolerant CAN, J1850 PWM, J1850 VPW, LIN, J1708, ISO9141/KWP2000. Digital and analog I/O channels.  <b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API

*Last Updated : Thursday, October 30, 2008*

## Network ID List for the ValueCAN

The chart below lists the valid network ID for the ValueCAN. The network ID is part of the [icsSpyMessage](#) structure and indicates the network a transmitted or received message is associated with.

Network ID	Default Value	Network Name
NETID_HSCAN	1	CAN

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, October 23, 2008*

## Network ID List for the neoVI Blue

The chart below lists the valid network ID's for the neoVI Blue. The network ID is part of the [icsSpyMessage](#) structure and indicates the network a transmitted or received message is associated with.

Network ID	Default Value	Network Name
NETID_HSCAN	1	HS CAN
NETID_MSCAN	2	MS CAN
NETID_SWCAN	3	SW CAN
NETID_LSFTCAN	4	LSFT CAN
NETID_FORDSCP	5	J1850 PWM
NETID_J1708	6	J1708
NETID_JVPW	8	J1850 VPW
NETID_ISO	9	ISO (K and L)
NETID_LIN	16	LIN

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, October 23, 2008*

### 3rd Generation neoVI Devices

<b>neoVI Fire</b> 	<p><b>Networks:</b> 4 dual wire CAN/J1939, single wire CAN, low-speed fault tolerant CAN, 4 LIN, 4 ISO9141/KWP2000</p> <p><b>Other Features:</b> SD Card for onboard data storage and stand-alone logging, embedded scripting, and simulation. Digital inputs and outputs.</p> <p><b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API</p>
<b>neoVI Red</b> 	<p><b>Networks:</b> 4 dual wire CAN/J1939, single wire CAN, low-speed fault tolerant CAN, 4 LIN. With the Red you can access any 2 CAN channels and any 2 LIN channels at one time.</p> <p><b>Other Features:</b> SD Card for onboard data storage and stand-alone logging, embedded scripting, and simulation. Digital inputs/outputs.</p> <p><b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API</p> <p>Also comes in a lower-cost 'Limited' version. The limited version allows access to either 2 CAN channels or 2 LIN channels, but not both simultaneously.</p>
<b>ValueCAN3</b> 	<p><b>Networks:</b> 2 dual wire CAN/J1939</p> <p><b>Other Features:</b> Embedded scripting and simulation.</p> <p><b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API</p>
<b>neoVI Yellow</b> 	<p><b>Networks:</b> 2 dual wire CAN/J1939, J1708, ISO9141/KWP2000, J1850</p> <p><b>Other Features:</b> SD Card for onboard data storage and stand-alone logging, embedded scripting, and Simulation. Digital inputs/outputs.</p> <p><b>Interfaces:</b> Vehicle Spy, J2534, RP1210, intrepidcs API</p>

Also comes in a lower-cost 'Limited' version.

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, October 30, 2008*

## Network ID List for the ValueCAN 3

The chart below lists the valid network ID's for the ValueCAN3. The network ID is part of the [icsSpyMessage](#) structure and indicates the network a transmitted or received message is associated with.

Network ID	Default Value	Network Name
NETID_HSCAN	1	HS CAN 1
NETID_MSCAN	2	MS CAN

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, October 23, 2008*

## Network ID List for the neoVI Fire

The chart below lists the valid network ID's for the neoVI Fire. The network ID is part of the [icsSpyMessage](#) structure and indicates the network a transmitted or received message is associated with.

Network ID	Default Value	Network Name
NETID_HSCAN	1	HS CAN 1
NETID_MSCAN	2	MS CAN
NETID_SWCAN	3	SW CAN
NETID_LSFTCAN	4	LSFT CAN
NETID_JVPW	8	J1850 VPW
NETID_ISO	9	ISO
NETID_ISO2	14	ISO 2 *
NETID_LIN	16	LIN 1
NETID_ISO3	41	ISO 3 *
NETID_HSCAN2	42	HS CAN 2
NETID_HSCAN3	44	HS CAN 3
NETID_ISO4	47	ISO4 *
NETID_LIN2	48	LIN 2
NETID_LIN3	49	LIN 3
NETID_LIN4	50	LIN 4

\* Note: ISO 2, 3 and 4 are only available when LIN is not enabled on the device. The pin inputs for ISO 2, 3 and 4 are the same as LIN 2, 3 and 4, listed on the pin-out diagram on the bottom of the neoVI Fire.



## WIN32 API Overview - intrepidcs API

### Basic Operations

Name	Description
<a href="#">FindNeoDevices</a>	Used to locate connected neoVI devices.
<a href="#">OpenNeoDevice</a>	Used to open a communication link with a specific neoVI device.
<a href="#">ClosePort</a>	Closes the communication link with the neoVI device.
<a href="#">FreeObject</a>	Releases system resources used by the neoVI device.

### Message Functions

Name	Description
<a href="#">GetMessages</a>	Reads messages from the neoVI device.
<a href="#">TxMessages</a>	Transmits messages to vehicle networks using a neoVI device.
<a href="#">WaitForRxMessagesWithTimeOut</a>	Waits a specified amount of time in milliseconds for a received message
<a href="#">GetTimeStampForMsg</a>	Calculates and returns the timestamp for a message
<a href="#">EnableNetworkRXQueue</a>	Enables and disables the receive queue for network messages
<a href="#">GetISO15765Status</a>	Gets and clears the current status of the CAN ISO15765-2 network layer receive.
<a href="#">SetISO15765RxParameters</a>	Sets parameters for CAN ISO15765-2 network layer message reception

### Device Settings Functions

Name	Description

<a href="#">GetConfiguration</a>	Reads the configuration bytes for a neoVI Blue or ValueCAN device
<a href="#">SendConfiguration</a>	Sends configuration bytes to a neoVI Blue or ValueCAN device
<a href="#">GetFireSettings</a>	Gets device and network parameters for a neoVI Fire device
<a href="#">SetFireSettings</a>	Sets device and network parameters for a neoVI Fire device
<a href="#">GetVCAN3Settings</a>	Gets device and network parameters for a ValueCAN3 device
<a href="#">SetVCAN3Settings</a>	Sets device and network parameters for a ValueCAN3 device
<a href="#">SetBitRate</a>	Set the baud or bit rate for a specific neoVI network
<a href="#">GetHWFirmwareInfo</a>	Gets the firmware version of a neoVI device
<a href="#">GetDLLFirmwareInfo</a>	Gets the firmware version stored in the DLL API
<a href="#">ForceFirmwareUpdate</a>	Forces the firmware to updated on a neoVI device
<a href="#">GetDeviceParameters</a>	Gets individual parameters for a neoVI device
<a href="#">SetDeviceParameters</a>	Sets individual parameters for a neoVI device
<a href="#">SetReflashDisplayCallbacks</a>	Sets callback function pointers for flashing a neoVI
<a href="#">ClearReflashDisplayCallbacks</a>	Clears callback function pointers for flashing a neoVI
<a href="#">GetRTC</a>	Gets the current real-time clock value from a connect neoVI device
<a href="#">SetRTC</a>	Sets the current real-time clock value in a connected neoVI device

## Error Functions

Name	Description
<a href="#">GetLastAPIError</a>	Returns the error generated by the last intrepidcs API call
<a href="#">GetErrorMessages</a>	Returns the intrepidcs API error message queue
<a href="#">GetErrorInfo</a>	Returns a text description of an intrepidcs API error

## General Utility Functions

Name	Description
<a href="#">ValidateHObject</a>	Used to determine if an hObject reference is valid
<a href="#">GetDLLVersion</a>	Returns DLL version information
<a href="#">StartSockServer</a>	Starts the TCP/IP socket server at a specified port.
<a href="#">StopSockServer</a>	Stops the TCP/IP socket server

## CoreMini Functions

Function	Description
<a href="#">ScriptStart</a>	Starts execution of a script that has been downloaded to a neoVI device
<a href="#">ScriptStop</a>	Stops execution of a script running on a neoVI device
<a href="#">ScriptLoad</a>	Downloads a script to a connected neoVI device into a specified location
<a href="#">ScriptClear</a>	Clears a script from a specific location on a neoVI device
<a href="#">ScriptStartFBlock</a>	Starts a function block within a script on a neoVI device
<a href="#">ScriptGetFBlockStatus</a>	Returns the run status of a function block within a script on a

	neoVI device
<a href="#">ScriptStopFBlock</a>	Stops the execution of a function block within a script on a neoVI device
<a href="#">ScriptGetScriptStatus</a>	Stops the execution of a function block within a script on a neoVI device
<a href="#">ScriptReadAppSignal</a>	Read an application signal from a script running on a neoVI device
<a href="#">ScriptWriteAppSignal</a>	Set the value of an application signal in a script running on a neoVI device
<a href="#">ScriptReadRxMessage</a>	Reads parameters for a receive message defined in a script on a neoVI device
<a href="#">ScriptReadTxMessage</a>	Reads parameters for a transmit message defined within a script on a neoVI device
<a href="#">ScriptWriteRxMessage</a>	Alter a receive message defined within script on a neoVI device
<a href="#">ScriptWriteTxMessage</a>	Alter a transmit message defined within a script on a neoVI device
<a href="#">ScriptReadISO15765TxMessage</a>	Read parameters of an ISO15765-2 long transmit message in a script on a neoVI device
<a href="#">ScriptWriteISO15765TxMessage</a>	Change the parameters for an ISO15765-2 long transmit message defined in a script on a neoVI device

## Deprecated Functions

Name	Description
<a href="#">OpenPortEx</a>	Use <a href="#">OpenNeoDevice</a> instead

OpenPort	Use <a href="#">OpenNeoDevice</a> instead
EnableNetworkCom	It is no longer necessary to call this before and after calling SendConfiguration
FindAllUSBDevices	No longer supported. It is present in the API but will always return 0

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

## Basic Functions Overview - intrepidcs API

Name	Description
<a href="#">FindNeoDevices</a>	Used to locate connected neoVI devices.
<a href="#">OpenNeoDevice</a>	Used to open a communication link with a specific neoVI device.
<a href="#">ClosePort</a>	Closes the communication link with the neoVI device.
<a href="#">FreeObject</a>	Releases system resources used by the neoVI device.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, September 18, 2008*

**FindNeoDevices Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoFindNeoDevices Lib "icsneo40.dll" (ByVal DeviceTypes As Long, ByRef pNeoDevice As NeoDevice, ByRef pNumDevices As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoFindNeoDevices Lib "icsneo40.dll" (ByVal DeviceTypes As UInt32, ByRef pNeoDevice As NeoDevice, ByRef pNumDevices As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoFindNeoDevices(UInt32 DeviceTypes, ref NeoDevice pNeoDevice, ref Int32 pNumDevices);
```

**Parameters*****DeviceTypes***

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

```
NEODEVICE_BLUE = 1
NEODEVICE_DW_VCAN = 4
NEODEVICE_FIRE = 8
NEODEVICE_VCAN3 = 16
NEODEVICE_YELLOW = 32
NEODEVICE_RED = 64
NEODEVICE_ECU = 128
```

```
NEODEVICE_IEVB = 256  
NEODEVICE_PENDANT = 512  
NEODEVICE_ALL = &HFFFFFFF
```

Constants are defined in appropriate header or module.

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. 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.

---

### Examples

#### Visual Basic Example

```
Dim lResult As Long '//Holds the results from function call  
Dim ndNeoToOpen As NeoDevice '//Struct holding detected hardware information  
Dim iNumberOfDevices As Long '//Number of hardware devices to look for  
Dim lDevTypes As Long '//Holds the device types to look for  
  
'//Set the devices to look for  
lDevTypes = NEODEVICE_BLUE + NEODEVICE_FIRE  
'//Set the number of devices to find
```

```
iNumberOfDevices = 1
'//Search for connected hardware
lResult = icsneoFindNeoDevices(lDevTypes, ndNeoToOpen, iNumberOfDevices)
If (lResult = 0) Then MsgBox("Problem finding devices", vbOKOnly)
```

### C/C++ Example:

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

lDevTypes = NEODEVICE_BLUE + NEODEVICE_FIRE;

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

### C# Example:

```
int iResult;//Holds the results from function call
NeoDevice ndNeoToOpen = new NeoDevice(); ; //Struct holding detected hardware information
int iNumberOfDevices; //Number of hardware devices to look for
int lDevTypes; //Holds the device types to look for
lDevTypes = NEODEVICE_BLUE + NEODEVICE_FIRE;

//Set the number of devices to find
iNumberOfDevices = 1;
//Search for connected hardware
iResult = icsNeoDll.icsneoFindNeoDevices(65535, ref ndNeoToOpen, ref iNumberOfDevices);
if (iResult == 0)
{
    MessageBox.Show("Problem finding devices");
    return;
}
```

### Visual Basic .NET Example:

```
Dim iResult As Integer '//Holds the results from function call
```

```
Dim ndNeoToOpen As NeoDevice '//Struct holding detected hardware information
Dim iNumberOfDevices As Integer '//Number of hardware devices to look for
Dim lDevTypes As Integer '//Holds the device types to look for

'//Set the devices to look for
lDevTypes = NEODEVICE_BLUE + NEODEVICE_FIRE
'//Set the number of devices to find
iNumberOfDevices = 1
'//Search for connected hardware
iResult = icsneoFindNeoDevices(lDevTypes, ndNeoToOpen, iNumberOfDevices)
If (iResult = 0) Then MsgBox("Problem finding devices")
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, August 15, 2011*

**OpenNeoDevice Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method opens a communication link to the a neoVI device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoOpenNeoDevice Lib "icsneo40.dll" (ByRef pNeoDevice As NeoDevice, ByRef hObject As Long, ByRef bNetworkIDs As Byte, ByVal bConfigRead As Long, ByVal bSyncToPC As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoOpenNeoDevice Lib "icsneo40.dll" (ByRef pNeoDevice As NeoDevice, ByRef hObject As Int32, ByRef bNetworkIDs As Byte, ByVal bConfigRead As Int32, ByVal bSyncToPC As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoOpenNeoDevice(ref NeoDevice pNeoDevice, ref Int32 hObject, ref byte bNetworkIDs, Int32 bConfigRead, Int32 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] Specifies whether timestamps for messages should be adjusted to the PC's clock to compensate for time drift. This drift is caused by differences that occur over time between the clocks on the neoVI device and the PC.

**Return Values**

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

**Remarks**

To connect to more than one piece of hardware, multiple calls to [OpenNeoDevice](#) can be made. The key is to provide a different [NeoDevice](#) for each device you want to open. For other function calls, use the returned hObject to talk to that device.

Each successful call to [OpenNeoDevice](#) should be matched with a call to a [ClosePort](#) when you are finished accessing the hardware. [FreeObject](#) methods.

---

**Examples****Visual Basic Example**

```
Dim m_hObject As Long '//handle for device
Dim lResult As Long '//Holds the results from function call
Dim ndNeoToOpen As NeoDevice '//Struct holding detected hardware information
Dim bNetwork(0 to 63) As Byte '//List of hardware IDs
Dim iCount As Integer '//counter

'//File NetworkID array
For iCount = 0 To 63
    bNetwork(iCount) = CByte(iCount)
Next iCount
```

```
'//Open the first found device, ndNeoToOpen acquired from FindNeoDevices
iResult = icsneoOpenNeoDevice(ndNeoToOpen, m_hObject, bNetwork(0), 1, 0)
If CBool(iResult) Then
    MsgBox("Port Opened OK!", vbOKOnly)
Else
    MsgBox("Problem Opening Port", vbOKOnly)
    Exit Sub
End If
```

### 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);
```

### Visual Basic .NET Example

```
Dim m_hObject As Integer '//handle for device
Dim iResult As Integer '//Holds the results from function call
Dim ndNeoToOpen As NeoDevice '//Struct holding detected hardware information
Dim bNetwork(64) As Byte '//List of hardware IDs
Dim iCount As Integer '//counter

'//File NetworkID array
For iCount = 0 To 63
    bNetwork(iCount) = CByte(iCount)
Next iCount
'//Open the first found device, ndNeoToOpen acquired from FindNeoDevices
iResult = icsneoOpenNeoDevice(ndNeoToOpen, m_hObject, bNetwork(0), 1, 0)
If CBool(iResult) Then
    MsgBox("Port Opened OK!")
Else
    MsgBox("Problem Opening Port")
    Exit Sub
End If
```

**C# Example**

```
int m_hObject; //handle for device
int iResult; //Holds the results from function call
NeoDevice ndNeoToOpen = new NeoDevice(); //Struct holding detected hardware information
byte[] bNetwork = new byte[64]; //List of hardware IDs
int iCount; //counter

//File NetworkID array
for (iCount = 0; iCount < 64; iCount++)
{
    bNetwork[iCount] = Convert.ToByte(iCount);
}
//Open the first found device, ndNeoToOpen acquired from FindNeoDevices
iResult = icsNeoDll.icsneoOpenNeoDevice(ref ndNeoToOpen, ref m_hObject, ref bNetwork[0], 1, 0);
if (iResult == 1)
{
    MessageBox.Show("Port Opened OK!");
}
else
{
    MessageBox.Show("Problem Opening Port");
    return;
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, August 23, 2010*

**ClosePort Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method closes the communication link with the neoVI hardware.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoClosePort Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pNumberOfErrors As Long)  
As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoClosePort Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pNumberOfErrors As Int32)  
As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern Int32 icsneoClosePort(Int32 hObject, ref Int32 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.

---

## Examples

### Visual Basic Example

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim lResult As Long
Dim lNumberOfErrors As Long

'//close the port
lResult = icsneoClosePort(m_hObject, lNumberOfErrors)
If CBool(lResult) Then
    MsgBox("Port Closed OK!", vbOKOnly)
Else
    MsgBox("Problem Closing Port", vbOKOnly)
End If
```

### C/C++ Example

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

// Close Communication
iResult = icsneoClosePort(hObject, &iNumberOfErrors);
// Test the Result
if (iResult== 0)
    MessageBox(hWnd,TEXT("Problem Closing Port"),TEXT("neoVI Example"),0);
else
    MessageBox(hWnd,TEXT("Port Closed Successfully"),TEXT("neoVI Example"),0);
```

### C# Example

```
//Declared at form level and previously open with a call to OpenNeoDevice
int m_hObject; //handle for device,

int iResult;
int iNumberOfErrors = 0;
```

```
//close the port
iResult = icsNeoDll.icsneoClosePort(m_hObject, ref iNumberOfErrors);
if (iResult == 1)
{
    MessageBox.Show("Port Closed OK!");
}
else
{
    MessageBox.Show("Problem ClosingPort");
}
m_bPortOpen = false;
```

### Visual Basic .NET Example

```
Private m_hObject As Integer '// Declared at form level and previously open with a call to OpenNeoDevice

Dim iResult As Integer
Dim iNumberOfErrors As Integer

'//close the port
iResult = icsneoClosePort(m_hObject, iNumberOfErrors)
If CBool(iResult) Then
    MsgBox("Port Closed OK!")
Else
    MsgBox("Problem Closing Port")
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2008*

**FreeObject Method - intrepidCS API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method releases system resources used by the neoVI device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoFreeObject Lib "icsneo40.dll" (ByVal hObject As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoFreeObject Lib "icsneo40.dll" (ByVal hObject As Int32)
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern void icsneoFreeObject(Int32 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. The LabVIEW neoClosePort.vi will call the FreeObject API.

## Examples

### Visual Basic Example

```
Call icsneoFreeObject(m_hObject) '// free the memory associated with our driver object
```

### C/C++ Example

```
icsneoFreeObject(hObject); //Free the memory associated with our driver object
```

### C# Example

```
icsNeoDll.icsneoFreeObject(m_hObject); //Free the memory associated with our driver object
```

### Visual Basic .NET Example

```
Call icsneoFreeObject(m_hObject) '//Free the memory associated with our driver object
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2008*

## Message Functions Overview - intrepidcs API

Name	Description
<a href="#">GetMessages</a>	Reads messages from the neoVI device.
<a href="#">TxMessages</a>	Transmits messages to vehicle networks using a neoVI device.
<a href="#">WaitForRxMessagesWithTimeOut</a>	Waits a specified amount of time in milliseconds for a received message
<a href="#">GetTimeStampForMsg</a>	Calculates and returns the timestamp for a message
<a href="#">EnableNetworkRXQueue</a>	Enables and disables the receive queue for network messages
<a href="#">GetISO15765Status</a>	Gets and clears the current status of the CAN ISO15765-2 network layer receive.
<a href="#">SetISO15765RxParameters</a>	Sets parameters for CAN ISO15765-2 network layer message reception

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, September 18, 2008*

**GetMessages Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET delcare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method reads messages from the neoVI device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetMessages Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pMsg As icsSpyMessage, ByRef pNumberOfMessages As Long, ByRef pNumberOfErrors As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetMessages Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pMsg As icsSpyMessage, ByRef pNumberOfMessages As Int32, ByRef pNumberOfErrors As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetMessages(Int32 hObject, ref icsSpyMessage pMsg, ref Int32 pNumberOfMessages, ref Int32 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.

---

**Examples****Visual Basic Example**

```
'// Declared at form level
Private m_hObject As Long '// Holds the object for the state of the application
Private stMessages(0 To 19999) As icsSpyMessage '// Array of message structures to hold the received data

Dim lResult As Long
Dim lCount as Long
Dim lNumberOfMessages As Long
Dim lNumberOfErrors As Long

'// read the messages from the driver
lResult = icsneoGetMessages(m_hObject, stMessages(0), lNumberOfMessages, lNumberOfErrors)

'// was the read successful
If CBool(lResult) Then
    '// print all the received messages network ID to VB's debug window
    For lCount = 0 To lNumberOfMessages-1
        Debug.Print "Message from network " & stMessages(lCount).NetworkID
    Next lCount
```

```
Else
    MsgBox "Problem Reading Messages"
End If
```

### 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);
if (iResult == 0)
    MessageBox(hWnd,TEXT("Problem Reading Messages"),TEXT("neoVI Example"),0);
else
    MessageBox(hWnd, TEXT("Messages Read Successfully"),TEXT("neoVI Example"),0);
```

### Visual Basic .NET Example

```
Dim lResult As Long ''Storage for Result of Call
Dim iNumberOfErrors As Long ''Storage for number of errors
Dim lNumberOfMessages As Long ''Storage for number of messages
Dim stMessages(20000) As icsSpyMessage '// Array of message structures to hold the received data

lResult = icsneoGetMessages(m_hObject, stMessages(0), lNumberOfMessages, iNumberOfErrors) ''Call get message
function
If Not CBool(lResult) Then ''See if Call was successful
    MsgBox("Problem Getting Messages")
    Exit Sub
End If
```

### C# Example

```
long lResult; //Storage the Result for the function call
int lNumberOfErrors = 0; //Storage for the number of Errors
int lNumberOfMessages = 0; //Storage for the number of messages

//Call Get messages
lResult = icsNeoDll.icsneoGetMessages(m_hObject, ref stMessages[0],ref lNumberOfMessages,ref lNumberOfErrors);
if(lResult == 0) //Check the Results to see if there is a problem
{
```

```
    MessageBox.Show ("Problem Getting Messages");
    return;
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

**TxMessages Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Value](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET Example](#) - [C# Example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoTxMessages Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pMsg As icsSpyMessage, ByVal lNetworkID As Long, ByVal lNumMessages As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoTxMessages Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pMsg As icsSpyMessage, ByVal lNetworkID As Int32, ByVal lNumMessages As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoTxMessages(Int32 hObject, ref icsSpyMessage pMsg, Int32 lNetworkID, Int32 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).

### ***INumMessages***

[in] Specifies the number of messages to be transmitted. This parameter should always be set to one unless you are transmitting a long Message. Transmitting long messages on ISO or J1708 is described in a [different topic](#).

### **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. Transmitting long messages which are greater than 12 bytes on ISO or J1708 is described in a [different topic](#).

### **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.

## Examples

### Visual Basic Example

```
'// Declared at form level
Private m_hObject As Long '// Holds the object for the state of the application

Dim lResult As Long
Dim stMessagesTx As icsSpyMessage

'// Load the message to be transmitted ArbID = FF standard data 0x22 0x52 0x12 0x28
With stMessagesTx
    .ArbIDOrHeader = &HFF
    .NumberBytesData = 4
    .Data(1) = &H22
    .Data(2) = &H52
    .Data(3) = &H12
    .Data(4) = &H28
End With

'// Transmit the assembled message
lResult = icsneoTxMessages(m_hObject, stMessagesTx, NETID_HSCAN, 1)
'// Test the returned result
If Not CBool(lResult) Then
    MsgBox "Problem Transmitting Message"
End If
```

### 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.ArbitIDOrHeader = 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);
if (iResult == 0)
    MessageBox(hWnd,TEXT("Problem Transmitting Messages"),TEXT("neoVI Example"),0);
```

### Visual Basic .NET Example

```
Dim lResult As Long
Dim stMessagesTx As icsSpyMessage
Dim lNumberBytes As Long
Dim sdataArray(7) As String

sdataArray(0) = txtDataByte1.Text ''Put data from form in
sdataArray(1) = txtDataByte2.Text ''form of Array
sdataArray(2) = txtDataByte3.Text
sdataArray(3) = txtDataByte4.Text

stMessagesTx.ArbitIDOrHeader = Val("&H" & txtArbID.Text) ''Set ArbID in structure

stMessagesTx.NumberBytesData = 4 ''Set the number of Data bytes

stMessagesTx.Data1 = Val("&H" & sdataArray(0)) ''Set data bytes in structure
stMessagesTx.Data2 = Val("&H" & sdataArray(1))
stMessagesTx.Data3 = Val("&H" & sdataArray(2))
stMessagesTx.Data4 = Val("&H" & sdataArray(3))

lResult = icsneoTxMessages(m_hObject, stMessagesTx, 1, 1) ''Send message
If Not CBool(lResult) Then ''Check the status of sent message
    MsgBox("Problem Transmitting Message")
End If
```

### C# Example

```
int iResult; //Storage for the Result from function call
icsSpyMessage stMessagesTx = new icsSpyMessage(); //Storage for TXMessage

//Set the ArbID information for TX message
stMessagesTx.ArbitIDOrHeader = ConvertFromHex(txtArbID.Text);
```

```
//Set the number of Data Bytes and set the Data Bytes values
stMessagesTx.NumberBytesData = Convert.ToByte(4);
stMessagesTx.Data1=Convert.ToByte(ConvertFromHex(txtDataByte1.Text));
stMessagesTx.Data2=Convert.ToByte(ConvertFromHex(txtDataByte2.Text));
stMessagesTx.Data3=Convert.ToByte(ConvertFromHex(txtDataByte3.Text));
stMessagesTx.Data4=Convert.ToByte(ConvertFromHex(txtDataByte4.Text));

//Clear the Status BitFields
stMessagesTx.StatusBitField = 0;
stMessagesTx.StatusBitField2 = 0;

//Call the Tx funciton
iResult=icsNeoDll.icsneoTxMessages(m_hObject, ref stMessagesTx,1,0);

if(iResult== 0) //Check the status of the Function Call
{
    MessageBox.Show ("Problem Transmitting Message");
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2008*

**WaitForRxMessagesWithTimeOut Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET delcare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoWaitForRxMessagesWithTimeOut Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iTimeOut As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoWaitForRxMessagesWithTimeOut Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iTimeOut As UInt32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoWaitForRxMessagesWithTimeOut(Int32 hObject, UInt32 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.

## Examples

### Visual Basic Example

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim lResult As Long
Dim iTimeOut As Long
iTimeOut = 5000 '//Set timeout to 5 seconds

lblWaitForRxMessageWithTimeOutResult.Caption = "Status"
DoEvents()

'//This function will block until, A: A Message is received by the hardware, or B: the timeout is reached
lResult = icsneoWaitForRxMessagesWithTimeOut(m_hObject, iTimeOut)
If lResult = 1 Then
    'Message received before timeout
    lblWaitForRxMessageWithTimeOutResult.Caption = "Message received"
    '//Do something with the messages received
    Call cmdReceive_Click()
Else
    'Timeout reached and no messages received
    lblWaitForRxMessageWithTimeOutResult.Caption = "Message Not received"
    '//Take action if no messages were received
End If
```

### C/C++ Example

```
int hObject = 0; // holds a handle to the neoVI object
icsSpyMessage stMessages[19999]; // 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)
        MessageBox(hWnd, TEXT("Problem Reading Messages"), TEXT("neoVI Example"), 0);
    else
        MessageBox(hWnd, TEXT("Messages Read Successfully"), TEXT("neoVI Example"), 0);
}

return 0;
```

## Visual Basic .NET Example

```
Private m_hObject As Integer '// Declared at form level and previously open with a call to OpenNeoDevice

Dim iResult As Integer
Dim iTimeOut As UInt32

iTimeOut = Convert.ToInt32(5000)

lblWaitForRxMessageWithTimeOutResult.Text = "Status"
Application.DoEvents()

'//This function will block until, A: A Message is received by the hardware, or B: the timeout is reached
iResult = icsneoWaitForRxMessagesWithTimeOut(m_hObject, iTimeOut)
If iResult = 1 Then
    'Message received before timeout
    lblWaitForRxMessageWithTimeOutResult.Text = "Message received"
    Call cmdReceive_Click(sender, e)
    '//Do something with the messages received
Else
```

```
'Timeout reached and no messages received
lblWaitForRxMessageWithTimeOutResult.Text = "Message Not received"
'//Take action if no messages were received
End If
```

## C# Example

```
//Declared at form level and previously open with a call to OpenNeoDevice
int m_hObject; //handle for device

int iResult;
UInt32 iTimeOut = 5000; //Set timeout to 5 seconds

lblWaitForRxMessageWithTimeOutResult.Text = "Status";
Application.DoEvents();

//This function will block until, A: A Message is received by the hardware, or B: the timeout is reached
iResult = icsNeoDll.icsneoWaitForRxMessagesWithTimeOut(m_hObject, iTimeOut);
if (iResult == 1)
{
    //Message received before timeout
    lblWaitForRxMessageWithTimeOutResult.Text = "Message received";
    //Do something with the messages received
    cmdReceive_Click(sender, e);
}
else
{
    //Timeout reached and no messages received
    lblWaitForRxMessageWithTimeOutResult.Text = "Message Not received";
    //Take action if no messages were received
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2008*

**GetTimeStampForMsg Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET delcare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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 __stdcall icsneoGetTimeStampForMsg(int hObject, icsSpyMessage *pMsg, icsSpyMessage *pMsg, double *pTimeStamp);
```

**Visual Basic Declare**

```
Public Declare Function icsneoGetTimeStampForMsg Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pMsg As  
icsSpyMessage, ByRef pTimeStamp As Double) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetTimeStampForMsg Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pMsg As  
icsSpyMessage, ByRef pTimeStamp As Double) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetTimeStampForMsg(Int32 hObject, ref icsSpyMessage pMsg, ref 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.

---

## Examples

### Visual Basic Example

```
Dim lResult As Long
Dim dTimeStamp as Double
icsSpyMessage Msg;

lResult = icsneoGetTimeStampForMsg(m_hObject, Msg, dTimeStamp)
```

### C/C++ Example

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

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

### Visual Basic .NET Example

```
Dim iResult As Integer
Dim dTimeStamp As Double
icsSpyMessage Msg;

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

**C# Example**

```
int iResult;
long lTimeStamp;
icsSpyMessage Msg;

iResult = icsNeoDll.icsneoGetTimeStampForMsg(m_hObject, ref Msg, ref dTimeStamp);
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**EnableNetworkRXQueue Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

```
int __stdcall icsneoEnableNetworkRXQueue(int hObject, int iEnable);
```

**Visual Basic Declare**

```
Public Declare Function icsneoEnableNetworkRXQueue Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iEnable As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoEnableNetworkRXQueue Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iEnable As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoWaitForRxMessagesWithTimeOut(Int32 hObject, UInt32 iTimeOut);
```

**Parameters***hObject*

[in] Specifies the driver object created by [OpenNeoDevice](#).

*iEnable*

[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.

---

### Examples

#### Visual Basic Example

```
'// disable network communications
Call icsneoEnableNetworkRXQueue(m_hObject, 0)
```

#### C/C++ Example

```
// disable network communications
icsneoEnableNetworkRXQueue(m_hObject, 0);
```

#### Visual Basic .NET Example

```
'// disable network communications
Call icsneoEnableNetworkRXQueue(m_hObject, 0)
```

#### C# Example

```
// disable network communications
icsNeoDll.icsneoEnableNetworkRXQueue(m_hObject, 0);
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008 Monday, January 24, 2005*

**GetISO15765Status Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method gets and optionally clears the current status of the CAN ISO15765-2 network layer receive.**

**C/C++ Declare**

```
void __stdcall icsneoGetISO15765Status(int hObject, int iNetwork, int iReserved1, int iClearRxStatus, int *  
iReserved2, int * iRxStatus);
```

**Visual Basic Declare**

```
Public Declare Sub icsneoGetISO15765Status Lib "icsneo40.dll" (ByVal hObject As Long, _  
    ByVal lNetwork As Long, ByVal lReserved1 As Long,  
    ByVal lClearRxStatus As Long, ByRef lReseverd2 As Long, ByRef lRxStatus As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Sub icsneoGetISO15765Status Lib "icsneo40.dll" _  
(ByVal hObject As Integer, ByVal iNetwork As Integer, _  
    ByVal lReserved1 As Integer, ByVal iClearRxStatus As Integer, _  
    ByRef lReserved2 As Integer, ByRef lRxStatus As Integer)
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern void icsneoGetISO15765Status(int hObject, int iNetwork, int lReserved1, int iClearRxStatus,  
ref int lReserved2, ref int lRxStatus);
```

**Parameters*****hObject***

[in] Handle which specifies the driver object created with the [OpenPort](#) method.

***iNetwork***

[in] Specifies which CAN network the status is requested for. Can be one of the following values: NETID\_HSCAN = 1, NETID\_MSCAN = 2, NETID\_SWCAN = 3, and NETID\_LSFTCAN = 4.

***iReserved1***

[in] Set to zero.

#### *IClearRxStatus*

[in] If set to one this will clear the receive status and reset the receive state machine. If zero the status is not affected.

#### *Reserved2*

[out] A reserved bitfield.

#### *IRxStatus*

[out] A bitfield describing the progress of a receive operation. See Table 1 below for a definition of this bitfield.

### **Return Values**

None.

### **Remarks**

None.

**Table 1 - IRxStatus Bitfield Definition**

Flag	Description
INTREPIDCS_15765_RX_ERR_GLOBAL	At least one error occurred during the last reception
INTREPIDCS_15765_RX_ERR_CFRX_EXP_FF	A consecutive frame was received while the a first frame or single frame was expected
INTREPIDCS_15765_RX_ERR_FCRX_EXP_FF	A flow control frame was received while a first frame or single frame was expected.
INTREPIDCS_15765_RX_ERR_SFRX_EXP_CF	A single frame was received when a consecutive frame was expected. The rx operation is cancelled.
INTREPIDCS_15765_RX_ERR_FFRX_EXP_CF	A first frame was received when a consecutive frame was expected. The rx operation is cancelled.
INTREPIDCS_15765_RX_ERR_FCRX_EXP_CF	A flow control frame was received when a consecutive frame was expected. The rx operation is cancelled.
INTREPIDCS_15765_RX_ERR_CF_TIME_OUT	The ICFTimeOutMs Timeout parameter was exceeded while waiting for consecutive frames. The rx operation is cancelled if detected by a call to read messages or GetISO15765Status. If a consecutive frame

	is received and the timeout is detected it will continue the long message but will set this flag.
INTREPIDCS_15765_RX_COMPLETE	The rx operation received enough messages to contain all data as indicated in the first frame DL (data length) parameter. The rx operation is ready for next operation.
INTREPIDCS_15765_RX_IN_PROGRESS	There currently is a rx operation in progress.
INTREPIDCS_15765_RX_ERR_SEQ_ERR_CF	This indicates that a Consecutive frame had an improper sequence count.

## Examples

### Visual Basic Example

```

Dim lTxStatus As Long
Dim lRxStatus As Long

Call icsneoGetISO15765Status(m_hObject, NETID_HSCAN, 0, 0, lTxStatus, lRxStatus)

lstStatusItems.Clear

If (lRxStatus And icsspy15765RxErrGlobal) > 0 Then
    lstStatusItems.AddItem "Problem In Rx Status"
End If

If (lRxStatus And icsspy15765RxErrCFRX_EXP_FF) > 0 Then
    lstStatusItems.AddItem "Received a Consecutive Frame when expecting first frame"
End If

If (lRxStatus And icsspy15765RxErrFCRX_EXP_FF) > 0 Then
    lstStatusItems.AddItem "Received a Flow Control Frame when expecting first frame"
End If

If (lRxStatus And icsspy15765RxErrSFRX_EXP_CF) > 0 Then
    lstStatusItems.AddItem "Received a Single Frame when expecting Consecutive frame"
End If

If (lRxStatus And icsspy15765RxErrFFRX_EXP_CF) > 0 Then

```

```
lstStatusItems.AddItem "Received a First Frame when expecting Consecutive frame"
End If

If (lRxStatus And icsspy15765RxErrFCRX_EXP_CF) > 0 Then
    lstStatusItems.AddItem "Received a Flow Control Frame when expecting Consecutive frame"
End If

If (lRxStatus And icsspy15765RxErrCF_TIME_OUT) > 0 Then
    lstStatusItems.AddItem "Consecutive Timeout"
End If

If (lRxStatus And icsspy15765RxComplete) > 0 Then
    lstStatusItems.AddItem "Last Messaging Successful"
End If

If (lRxStatus And icsspy15765RxInProgress) > 0 Then
    lstStatusItems.AddItem "Rx In Progress"
End If

If (lRxStatus And icsspy15765RxErrSeqCntInCF) > 0 Then
    lstStatusItems.AddItem "Incorrect Sequence Count in Consecutive Frames"
End If
```

## C/C++ Example

```
int iTxStatus, iRxStatus;

icsneoGetISO15765Status(hObject, NETID_HSCAN, 0, 0, &iTxStatus, &iRxStatus);

// Output the rx status

if (iRxStatus & INTREPIDCS_15765_RX_ERR_GLOBAL)
    OutputDebugString(TEXT("Problem In Rx Status\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_CFRX_EXP_FF)
    OutputDebugString(TEXT("Received a Consecutive Frame when expecting first frame\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_FCRX_EXP_FF)
    OutputDebugString(TEXT("Received a Flow Control Frame when expecting first frame\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_SFRX_EXP_CF)
```

```
OutputDebugString(TEXT("Received a Single Frame when expecting Consecutive frame\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_FFRX_EXP_CF)
    OutputDebugString(TEXT("Received a First Frame when expecting Consecutive frame\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_FCRX_EXP_CF)
    OutputDebugString(TEXT("Received a Flow Control Frame when expecting Consecutive frame\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_CF_TIME_OUT)
    OutputDebugString(TEXT("Consecutive Timeout\n"));

if (iRxStatus & INTREPIDCS_15765_RX_COMPLETE)
    OutputDebugString(TEXT("Last Messaging Successful\n"));

if (iRxStatus & INTREPIDCS_15765_RX_IN_PROGRESS)
    OutputDebugString(TEXT("Rx In Progress\n"));

if (iRxStatus & INTREPIDCS_15765_RX_ERR_SEQ_ERR_CF)
    OutputDebugString(TEXT("Incorrect Sequence Count in Consecutive Frames\n"));
```

## C# Example

```
int lTxStatus = 0;
int lRxStatus = 0;

//Acquire the ISO 15765 Paramerts
icsNeoDll.icsneoGetISO15765Status(m_hObject, 1, 0, 0, ref lTxStatus, ref lRxStatus);

//Check for Problems in ISO 15765 Rx status
if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrGlobal)) > 0)
    lstStatusItems.Items.Add("Problem In Rx Status");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrCFRX_EXP_FF)) > 0 )
    lstStatusItems.Items.Add("Received a Consecutive Frame when expecting first frame");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrFCRX_EXP_FF)) > 0 )
    lstStatusItems.Items.Add("Received a Flow Control Frame when expecting first frame");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrSFRX_EXP_CF)) > 0 )
    lstStatusItems.Items.Add("Received a Single Frame when expecting Consecutive frame");
```

```
if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrFFRX_EXP_CF)) > 0 )
lstStatusItems.Items.Add("Received a First Frame when expecting Consecutive frame");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrFCRX_EXP_CF)) > 0 )
lstStatusItems.Items.Add("Received a Flow Control Frame when expecting Consecutive frame");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrCF_TIME_OUT)) > 0 )
lstStatusItems.Items.Add("Consecutive Timeout");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxComplete)) > 0 )
lstStatusItems.Items.Add("Last Messaging Successful");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxInProgress)) > 0 )
lstStatusItems.Items.Add("Rx In Progress");

if ((lRxStatus & Convert.ToInt32(icsspy15765RxBitfield.icsspy15765RxErrSeqCntInCF)) > 0 )
lstStatusItems.Items.Add("Incorrect Sequence Count in Consecutive Frames");
```

## Visual Basic .NET Example

```
Dim lTxStatus As Long
Dim lRxStatus As Long

'//Acquire the ISO 15765 Paramerts
Call icsneoGetISO15765Status(m_hObject, NETID_HSCAN, 0, 0, lTxStatus, lRxStatus)

'//Check for Problems in ISO 15765 Rx status
If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrGlobal) > 0 Then
    lstStatusItems.Items.Add("Problem In Rx Status")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrCFRX_EXP_FF) > 0 Then
    lstStatusItems.Items.Add("Received a Consecutive Frame when expecting first frame")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrFCRX_EXP_FF) > 0 Then
    lstStatusItems.Items.Add("Received a Flow Control Frame when expecting first frame")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrSFRX_EXP_CF) > 0 Then
    lstStatusItems.Items.Add("Received a Single Frame when expecting Consecutive frame")
```

```
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrFFRX_EXP_CF) > 0 Then
    lstStatusItems.Items.Add("Received a First Frame when expecting Consecutive frame")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrFCRX_EXP_CF) > 0 Then
    lstStatusItems.Items.Add("Received a Flow Control Frame when expecting Consecutive frame")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrCF_TIME_OUT) > 0 Then
    lstStatusItems.Items.Add("Consecutive Timeout")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxComplete) > 0 Then
    lstStatusItems.Items.Add("Last Messaging Successful")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxInProgress) > 0 Then
    lstStatusItems.Items.Add("Rx In Progress")
End If

If (lRxStatus And icsspy15765RxBitfield.icsspy15765RxErrSeqCntInCF) > 0 Then
    lstStatusItems.Items.Add("Incorrect Sequence Count in Consecutive Frames")
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, September 11, 2008*

**SetISO15765RxParameters Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method sets the parameters necessary to control the CAN ISO15765-2 network layer message reception.**

**C/C++ Declare**

```
void __stdcall icsneoSetISO15765RxParameters(int hObject,
    int lNetwork,
    int lEnable,
    spyFilterLong * pFF_CFMsgFilter,
    icsSpyMessage * pFlowCTxMsg,
    int lCFTimeOutMs,
    int lFlowCBlockSize,
    int lUsesExtendedAddressing,
    int lUseHardwareIfPresent);
```

**Visual Basic Declare**

```
Public Declare Sub icsneoSetISO15765RxParameters Lib "icsneo40.dll" _
    (ByVal hObject As Long, _
    ByVal lNetwork As Long, _
    ByVal lEnable As Long, _
    ByRef pFF_CFMsgFilter As spyFilterLong, _
    ByRef pFlowCTxMsg As icsSpyMessage, _
    ByVal lCFTimeOutMs As Long, _
    ByVal lFlowCBlockSize As Long, _
    ByVal lUsesExtendedAddressing As Long, _
    ByVal lUseHardwareIfPresent As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Sub icsneoSetISO15765RxParameters Lib "icsneo40.dll" _
    (ByVal hObject As Integer, _
    ByVal iNetwork As Integer, _
    ByVal iEnable As Integer, _
    ByRef pFF_CFMsgFilter As spyFilterLong, _
    ByRef pFlowCTxMsg As icsSpyMessage, _
    ByVal lCFTimeOutMs As Integer, _
    ByVal lFlowCBlockSize As Integer,
```

```
    ByVal lUsesExtendedAddressing As Integer, _  
    ByVal lUseHardwareIfPresent As Integer)
```

## C# Declare

```
[DllImport("icsneo40.dll")]  
public static extern void icsneoSetISO15765RxParameters (int hObject, int iNetwork, int iEnable, ref spyFilterLong  
pFF_CFMsgFilter ,ref icsSpyMessage pFlowCTxMsg, int lCFTimeOutMs, int lFlowCBlockSize, int lUsesExtendedAddressing,  
int lUseHardwareIfPresent);
```

## Parameters

### *hObject*

[in] Specifies the driver object created by [OpenNeoDevice](#).

### *iNetwork*

[in] Specifies which CAN neoVI network used for setup parameters. Can be one of the following values: NETID\_HSCAN = 1, NETID\_MSCAN = 2, NETID\_SWCAN = 3, and NETID\_LSFTCAN = 4.

### *iEnable*

[in] If iEnable is 1 ISO15765 services are enabled for this network. If zero they are disabled.

### *pFF\_CFMsgFilter*

[in] Specifies the filter used to identify both First Frame and Consecutive Frame messages. Fill in the ArbID only in the filter. If extended addressing is used, also fill in byte 1.

### *pFlowCTxMsg*

[in] Specifies the message transmitted as the Flow Control frame. This frame should be entirely filled in. This includes the FS (FlowStatus), BS (BlockStatus), and STMin (separation time minimum) parameters. The BS parameter should match what is supplied in the lFlowCBlockSize argument.

### *lCFTimeOutMs*

[in] Specifies how long to wait for CF (Consecutive Frames) from the transmitter of the message sequence before aborting the transmission.

### *lFlowCBlockSize*

[in] Indicates the interval at which the Flow control is sent out. For example, if set to three, the FC frame will be transmitted every three received CF frames. If set to zero, the only time the FC frame will be sent out will be when it receives the FF (First Frame).

### *lUsesExtendedAddressing*

[in] Indicates the messaging uses extended or mixed addressing. In this case, the first data byte includes address data and the

N\_PCI info (Network Protocol Control Information) starts at the second byte.

*IUseHardwareIfPresent*

[in] Currently not supported.

**Return Values**

None.

**Remarks**

This function will setup the hardware to accept receive multi-frame messages as defined in ISO15765-2. A call to this function will reset the receive engine to monitor for first frame messages. The receive engine can be monitored by calling the [GetISO15765Status](#) method.

---

**Examples**

**Visual Basic Example**

```
Dim stFilter As spyFilterLong
Dim stMsg As icsSpyMessage

'// filter
stFilter.Header = &H777
stFilter.HeaderMask = &FFFF

'// flow control frame
stMsg.ArbIDOrHeader = &H641
stMsg.NumberBytesData = 8
stMsg.StatusBitField = 2
stMsg.Data(1) = &H30 'flow control frame
stMsg.Data(2) = 3 'block size
stMsg.Data(3) = 0 'stmin =0

Call icsneoSetISO15765RxParameters(m_hObject, NETID_HSCAN, 1, stFilter, stMsg, 100, 0, 0, 0)
```

**C/C++ Example**

```
spyFilterLong FF_CFMMSGFilter;
icsSpyMessage FlowCTxMsg;
```

```
// reset the structures to zero  
  
memset(&FF_CFMMSGFilter, 0, sizeof(FF_CFMMSGFilter));  
memset(&FlowCTxMsg, 0, sizeof(FlowCTxMsg));  
  
// setup the filter  
  
FF_CFMMSGFilter.Header = 0x7E8;  
FF_CFMMSGFilter.HeaderMask = 0xFFF;  
FlowCTxMsg.StatusBitField=2;  
FlowCTxMsg.NumberBytesData =8;  
FlowCTxMsg.ArbitrationHeader = 0x641;  
FlowCTxMsg.Data[0] = 0x30; // flow control frame : FlowStatus=CTS  
FlowCTxMsg.Data[1] = 0x3; // Blocksize  
FlowCTxMsg.Data[2] = 0x0; // STMin=0  
  
// setup rx on HSCAN with 100 ms timeout and 3 block size  
SetISO15765RxParameters(hObject, NETID_HSCAN, true,&FF_CFMMSGFilter, &FlowCTxMsg,100 , 3,0,0);
```

## C# Example

```
spyFilterLong stFilter;  
icsSpyMessage stMsg;  
//Check to see if the port is open  
if(m_bPortOpen != true)  
{  
    MessageBox.Show("Port is not open");  
    return;  
}  
  
//Set the filter up  
stFilter.Header = ConvertFromHex(txtFirstFrame.Text);  
stFilter.HeaderMask = ConvertFromHex("FFF");  
  
//Set the Flow Control Frame Properties  
stMsg.ArbitrationHeader = ConvertFromHex(txtFlowControl.Text);  
stMsg.NumberBytesData = 8;  
stMsg.StatusBitField = 2;  
stMsg.Data1 = Convert.ToByte(ConvertFromHex("30")); //flow control frame  
stMsg.Data2 = 3; //block size  
stMsg.Data3 = 0; //stmin =0  
  
//Set the established parameters
```

```
icsNeoDll.icsneoSetISO15765RxParameters(m_hObject, 1, 1,out stFilter, out stMsg, 100, 0, 0, 0);
```

## Visual Basic .NET Example

```
Dim stFilter As spyFilterLong
Dim stMsg As icsSpyMessage

'//Check to see if the port is open
If Not m_bPortOpen Then
    MsgBox("Port is not open.")
    Exit Sub
End If

'//Set the filter up
stFilter.Header = "&H" & txtFirstFrame.Text
stFilter.HeaderMask = &HFFF

'//Set the Flow Control Frame Properties
stMsg.ArbIDOrHeader = "&H" & txtFlowControl.Text
stMsg.NumberBytesData = 8
stMsg.StatusBitField = 2
stMsg.Data1 = &H30 'flow control frame
stMsg.Data2 = 3 'block size
stMsg.Data3 = 0 'stmin =0

'//Set the established parameters
Call icsneoSetISO15765RxParameters(m_hObject, 1, 1, stFilter, stMsg, 100, 0, 0, 0)
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, September 17, 2008*

## Transmitting Long Messages - Intrepidcs API

[Main](#)

### Overview

For ISO and \*J1708 Networks a long message transmission option is allowed. Long message transmission is necessary when the message length exceeds twelve bytes (including checksum if used).

You can transmit a long message by calling the [TxMessages](#) method with modified arguments. First, you will pass an array of icsSpyMessage structures which contain the long message data. This data in the array must be set properly. Finally, you must set the INumMessages argument to the number of structures.

You must setup the long message data properly. For the first message, both the 3 byte Header and the 8 byte data section are filled in. For consecutive messages only the 8 byte data section is filled in.

### VB Example to Send a 16 byte message

```
'// Declared at form level
Private m_hObject As Long '// Holds the object for the state of the application

Dim stMsg(0 to 3) As icsSpyMessageJ1850
Dim stMsgJ1850 As icsSpyMessageJ1850
Dim lResult As Long

' Fill in first header
stMsgJ1850.Header(1) = bByte1
stMsgJ1850.Header(2) = bByte2
stMsgJ1850.Header(3) = bByte3
stMsgJ1850.NumberBytesHeader = 3
' Fill in data
stMsgJ1850 .Data(1) = bByte4
stMsgJ1850Data(2) = bByte5
stMsgJ1850.Data(3) = bByte6
stMsgJ1850.Data(4) = bByte7
stMsgJ1850.Data(5) = bByte8
stMsgJ1850.Data(6) = bByte9
stMsgJ1850.Data(7) = bByte10
stMsgJ1850.Data(8) = bByte11
stMsgJ1850.NumberBytesData = 8

LSet stMsg(0) = stMsgJ1850
```

```
' Fill in data
stMsgJ1850.NumberBytesHeader = 0
stMsgJ1850.Data(1) = bByte12
stMsgJ1850.Data(2) = bByte13
stMsgJ1850.Data(3) = bByte14
stMsgJ1850.Data(4) = bByte15
stMsgJ1850.Data(5) = bByte16
stMsgJ1850.NumberBytesData = 5

LSet stMsg(1) = stMsgJ1850

'// Transmit the assembled message
lResult = icsneoTxMessages(m_hObject, stMsg(0), NETID_ISO, 2)
'// Test the returned result
If Not CBool(lResult) Then
    MsgBox "Problem Transmitting Message"
End If
```

\* J1708 Networks on neoVI does not support transmit. Transmit support for neoVI PRO is planned please contact Intrepid Control Systems for availability.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Wednesday, May 28, 2003*

## Device Settings Functions Overview - intrepidcs API

Name	Description
<a href="#">GetConfiguration</a>	Reads the configuration bytes for a neoVI Blue or ValueCAN device
<a href="#">SendConfiguration</a>	Sends configuration bytes to a neoVI Blue or ValueCAN device
<a href="#">GetFireSettings</a>	Gets device and network parameters for a neoVI Fire device
<a href="#">SetFireSettings</a>	Sets device and network parameters for a neoVI Fire device
<a href="#">GetVCAN3Settings</a>	Gets device and network parameters for a ValueCAN3 device
<a href="#">SetVCAN3Settings</a>	Sets device and network parameters for a ValueCAN3 device
<a href="#">SetBitRate</a>	Set the baud or bit rate for a specific neoVI network
<a href="#">GetHWFirmwareInfo</a>	Gets the firmware version of a neoVI device
<a href="#">GetDLLFirmwareInfo</a>	Gets the firmware version stored in the DLL API
<a href="#">ForceFirmwareUpdate</a>	Forces the firmware to updated on a neoVI device
<a href="#">GetDeviceParameters</a>	Gets individual parameters for a neoVI device
<a href="#">SetDeviceParameters</a>	Sets individual parameters for a neoVI device
<a href="#">SetReflashDisplayCallbacks</a>	Sets callback function pointers for use when flashing a neoVI
<a href="#">ClearReflashDisplayCallbacks</a>	Clears callback function pointers for flashing a neoVI
<a href="#">GetRTC</a>	Gets the current real-time clock value from a connect neoVI device
<a href="#">SetRTC</a>	Sets the current real-time clock value in a

~~DEVICES~~

connected neoVI device

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Wednesday, August 05, 2009*

**Device Settings Functions Overview - neoVI Blue and ValueCAN - intrepidcs API**

Name	Description
<a href="#">GetConfiguration</a>	Reads the configuration bytes for a neoVI Blue or ValueCAN device
<a href="#">SendConfiguration</a>	Sends configuration bytes to a neoVI Blue or ValueCAN device

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Wednesday, December 17, 2008*

**GetConfiguration Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method reads the configuration from the hardware device.**

**Note:** This function is only to be used for neoVI Blue and ValueCAN. For neoVI Fire and neoVI Red use the [GetFireSettings](#) method. For ValueCAN3 use the [GetVCAN3Settings](#) method.

**C/C++ Declare**

```
int __stdcall icsneoGetConfiguration(int hObject, unsigned char *pData, int *pNumBytes);
```

**Visual Basic Declare**

```
Public Declare Function icsneoGetConfiguration Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pData As Byte, ByRef lNumBytes As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetConfiguration Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pData As Byte, ByRef lNumBytes As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetConfiguration(Int32 hObject, ref byte pData, ref Int32 lNumBytes);
```

**Parameters*****hObject***

[in] Specifies the driver object created with the [OpenNeoDevice](#) method.

***pData***

[out] Pointer to an array of 1024 bytes. Each index of the array corresponds to a configuration value. For a list of configuration values to change, please see the [Configuration Array](#) topic.

***pI NumBytes***

[out] This will return the number of bytes written to the array. For the current version of the API this will be 1024 bytes.

**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**

None.

---

**Examples****Visual Basic Example**

```
Dim lNumBytes As Long
Dim bConfigBytes(0 To 1024) As Byte
Dim lResult As Long

'// we must set 1024 bytes
lNumBytes = 1024

'// get the configuration
lResult = icsneoGetConfiguration(m_hObject, bConfigBytes(0), lNumBytes)

'// make sure the read was successful
If Not CBool(lResult) Then
    MsgBox "Problem reading configuration"
    Exit Sub
End If

'// load the cnf values into text boxes
txtCNF1 = Hex(bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF1))
txtCNF2 = Hex(bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF2))
txtCNF3 = Hex(bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF3))
```

**C/C++ Example**

## Displays the Value of CNF1 of ValueCAN/neoVI HSCAN

```
unsigned char bConfigBytes[1024];

int iNumConfigBytes = 1024;

lResult = icsneoGetConfiguration(hObject, bConfigBytes, &iNumConfigBytes);
if (lResult == 0)
    MessageBox(hWnd,TEXT("Problem Reading Configuration"),TEXT("neoVI Example"),0);
else
{
    wsprintf(szOut,TEXT("HSCAN CNF1 = %x"),bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF1]);
    MessageBox(hWnd,szOut,TEXT("neoVI Example"),0);
}
```

## C# Example

```
byte[] bConfigBytes = new byte[1024]; //Storage for Data Bytes from Device
int iNumBytes = 1204; //Storage for Number of Bytes
int lResult; //Storage for Result of Called Function
int Counter;

//Clear listbox
lstConfigInformation.Items.Clear();

//Call Get Configuration
lResult = icsNeoDll.icsneoGetConfiguration(m_hObject, ref bConfigBytes[0],ref iNumBytes);

//Fill ListBox with Data From function Call
for(Counter=0;Counter<1024;Counter++)
{
    lstConfigInformation.Items.Add("Byte Number-" + Counter + " Byte Data-" + bConfigBytes[Counter]);
}
```

## Visual Basic .NET Example

```
byte[] bConfigBytes = new byte[1024]; //Storage for Data Bytes from Device
int iNumBytes = 1204; //Storage for Number of Bytes
int lResult; //Storage for Result of Called Function
int Counter;
```

```
//Clear listbox
lstConfigInformation.Items.Clear();

//Call Get Configuration
lResult = icsNeoDll.icsneoGetConfiguration(m_hObject, ref bConfigBytes[0],ref iNumBytes);

//Fill ListBox with Data From function Call
for(Counter=0;Counter<1024;Counter++)
{
    lstConfigInformation.Items.Add("Byte Number-" + Counter + " Byte Data-" + bConfigBytes[Counter]);
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**SendConfiguration Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method sends configuration information to the hardware.**

**C/C++ Declare**

```
int __stdcall icsneoSendConfiguration(int hObject, unsigned char *pData, int lNumBytes);
```

**Visual Basic Declare**

```
Public Declare Function icsneoSendConfiguration Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pData As Byte, ByVal lNumBytes As Long) As Long
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetFireSettings(Int32 hObject, ref SFireSettings pSettings, Int32 iNumBytes, Int32 bSaveToEEPROM);
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSendConfiguration Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pData As Byte, ByVal lNumBytes As Int32) As Int32
```

**Parameters***hObject*

[in] Specifies the driver object created by [OpenNeoDevice](#).

*pData*

[in] This is an array configuration bytes. The format of this array is defined in the [Configuration Array](#) help topic. This data should be filled in with a call to [GetConfiguration](#) before calling SendConfiguration. The size of this array must always be 1024 bytes.

*lNumBytes*

[in] This must always be set to 1024.

## 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 method will only update the configuration defined in the [Configuration Array](#) topic. It will also apply checking to the data so that a neoVI is not programmed to an illegal state. For example, setting the CAN controller to an illegal operating mode.

---

## Examples

### Visual Basic Example

```
Private m_hObject As Long      '// Declared at form levelDim lResult As Long
Dim lNumBytes As Long
Dim bConfigBytes(0 To 1024) As Byte

'-----READ CONFIGURATION -----
// we must set 1024 bytes
lNumBytes = 1024

// get the configuration
lResult = icsneoGetConfiguration(m_hObject, bConfigBytes(0), lNumBytes)

// make sure the read was successful
If Not CBool(lResult) Then
    lblDevice = "Problem reading configuration"
    CloseDevice
    Exit Sub
End If

'-----UPDATE CONFIGURATION -----
// load the cnf values into text boxes
```

```

bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF1) = Val("&H" & txtCNF1)
bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF2) = Val("&H" & txtCNF2)
bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF3) = Val("&H" & txtCNF3)

'// can controller mode
Select Case (lstCANController.ListIndex)
    Case 0 'normal
        bConfigBytes(NEO_CFG_MPIC_HS_CAN_MODE) = 1
    Case 1 'listen
        bConfigBytes(NEO_CFG_MPIC_HS_CAN_MODE) = 2
    Case 2 'loop
        bConfigBytes(NEO_CFG_MPIC_HS_CAN_MODE) = 4
End Select

lResult = icsneoSendConfiguration(m_hObject, bConfigBytes(0), lNumBytes)

'// make sure the read was successful
If Not CBool(lResult) Then
    lblDevice = "Problem sending configuration"
    CloseDevice
    Exit Sub
End If

'-----SUCCESS -----
lblResults = "The update was successful"

```

## C/C++ Example

```

unsigned char bConfigBytes[1024];
int iNumConfigBytes = 1024;

if(m_bPortOpen)
{
    lResult = icsneoGetConfiguration(hObject, bConfigBytes, &iNumConfigBytes);
    if (lResult == 0)
        MessageBox(hWnd, TEXT("Problem Reading Configuration"), TEXT("neoVI Example"), 0);
    else
    {
        iOldCNF1=bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF1];
        iOldCNF2=bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF2];

```

```

iOldCNF3=bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF3];

// 250 K for Value CAN 500k for neoVI (neoVI and valuecan use different CAN controller rates)
bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF1] = 0x03;
bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF2] = 0xB8;
bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF3] = 0x05;

lResult = icsneoSendConfiguration(hObject, bConfigBytes, iNumConfigBytes);

if (lResult == 0)
    MessageBox(hWnd,TEXT("Problem Updating Configuration"),TEXT("neoVI Example"),0);
else
{
    wsprintf(szOut,TEXT("Old Values: HSCAN CNF1 = %x HSCAN CNF2 = %x HSCAN CNF3 = %x \n\nNew Values HSCAN
CNF1 = %x HSCAN CNF2 = %x HSCAN CNF3 = %x "),
        iOldCNF1,
        iOldCNF2,
        iOldCNF3,
        bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF1],
        bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF2],
        bConfigBytes[NEO_CFG_MPIC_HS_CAN_CNF3]);
    MessageBox(hWnd,szOut,TEXT("neoVI Example"),0);
}
}

else
    MessageBox(hWnd,TEXT("Port Not Open"),TEXT("neoVI Example"),0);

```

## Visual Basic .NET Example

```

Dim bConfigBytes(1024) As Byte ''Storage for Data bytes from device
Dim iNumBytes As Integer ''Storage for Number of Bytes
Dim lResult As Integer ''Storage for Result of Called Function
Dim Counter As Integer
Dim iNumberOfErrors As Long ''Storage for Number of Errors Received

''Clear ListBox
lstConfigInformation.Items.Clear()
''Call Get Configuration
lResult = icsneoGetConfiguration(m_hObject, bConfigBytes(0), iNumBytes)
''Fill Listbox with Data From Function Call

```

```

For Counter = 0 To 1024
    lstConfigInformation.Items.Add("Byte Number-" & Counter & " Byte Data-" & bConfigBytes(Counter))
Next Counter

'-----READ CONFIGURATION -----

''Set HS CAN Baud Rate Information
bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF1) = Val("&H" & txtCNF1.Text)
bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF2) = Val("&H" & txtCNF2.Text)
bConfigBytes(NEO_CFG_MPIC_HS_CAN_CNF3) = Val("&H" & txtCNF3.Text)

'-----SEND CONFIGURATION -----
''Call Sned configuration
lResult = icsneoSendConfiguration(m_hObject, bConfigBytes(0), iNumBytes)

'// make sure the read was successful
If Not CBool(lResult) Then
    MsgBox("Problem sending configuration")
    lResult = icsneoClosePort(m_hObject, iNumberOfErrors)
    Exit Sub
Else
    MsgBox("Configuration Successfull")
End If

```

**C# Example**

```

byte[] bConfigBytes= new byte[1024]; //Storage for Data bytes from device
int iNumBytes = 0; //Storage for Number of Bytes
int lResult = 0; //Storage for Result of Called Function
int Counter;
int lNumberOfErrors = 0; //Storage for Number of Errors Received

//Clear ListBox
lstConfigInformation.Items.Clear();

'-----READ CONFIGURATION -----

//Call Get Configuration
lResult = icsNeoDll.icsneoGetConfiguration(m_hObject, ref bConfigBytes[0],ref iNumBytes);

//Fill Listbox with Data From Function Call
for(Counter=0; Counter<1024;Counter++)

```

```
{  
    lstConfigInformation.Items.Add("Byte Number-" + Counter + " Byte Data-" + bConfigBytes[Counter]);  
  
}  
  
//Set HS CAN Baud Rate Information  
bConfigBytes[Convert.ToInt32(icsConfigSetup.NEO_CFG_MPIC_HS_CAN_CNF1)] =  
Convert.ToByte(ConvertFromHex(txtCNF1.Text));  
bConfigBytes[Convert.ToInt32(icsConfigSetup.NEO_CFG_MPIC_HS_CAN_CNF2)] =  
Convert.ToByte(ConvertFromHex(txtCNF2.Text));  
bConfigBytes[Convert.ToInt32(icsConfigSetup.NEO_CFG_MPIC_HS_CAN_CNF3)] =  
Convert.ToByte(ConvertFromHex(txtCNF3.Text));  
  
//-----SEND CONFIGURATION -----  
//Call Sned configuration  
lResult = icsNeoDll.icsneoSendConfiguration(m_hObject, ref bConfigBytes[0], iNumBytes);  
  
// make sure the read was successful  
if(lResult==0)  
{  
    MessageBox.Show("Problem sending configuration");  
    lResult = icsNeoDll.icsneoClosePort(m_hObject, ref lNumberOfErrors);  
}  
else  
{  
    MessageBox.Show("Configuration Successfull");  
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**Device Settings Functions Overview - neoVI Fire - intrepidcs API**

Name	Description
<a href="#">GetFireSettings</a>	Gets device and network parameters for a neoVI Fire device
<a href="#">SetFireSettings</a>	Sets device and network parameters for a neoVI Fire device

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Wednesday, December 17, 2008*

**GetFireSettings Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetFireSettings Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pSettings As SFireSettings, ByVal iNumBytes As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetFireSettings Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pSettings As SFireSettings, ByVal iNumBytes As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetFireSettings(Int32 hObject, ref SFireSettings pSettings, Int32 iNumBytes, Int32 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.

## 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](#).

---

## Examples

### Visual Basic Example

```
Dim FireReadSettings As SFireSettings
Dim lNumberOfBytes As Long
Dim lResult As Long

'//Get the setting
lResult = icsneoGetFireSettings(m_hObject, FireReadSettings, LenB(FireReadSettings))
If lResult = 0 Then
    MsgBox("Problem reading FIRE configuration", vbOKOnly)
    Exit Sub
End If
```

### C/C++ Example

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

//Get the settings
iNumberOfBytes=sizeof(SFireSettings);
iResult = icsneoGetFireSettings(m_hObject, &FireReadSettings, iNumberOfBytes);
if(iResult == 0)
{
    MessageBox::Show("Problem reading FIRE configuration");
    return;
```

```
}
```

### C# Example

```
//Declared at form level and previously open with a call to OpenNeoDevice
int m_hObject; //handle for device,
SFireSettings FireReadSettings= new SFireSettings();
int iNumberOfBytes;
int iResult;

//Get the settings
iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(FireReadSettings);
iResult = icsNeoDll.icsneoGetFireSettings(m_hObject,ref FireReadSettings, iNumberOfBytes);
if (iResult == 0)
{
    MessageBox.Show("Problem reading FIRE configuration");
    return;
}
```

### Visual Basic .NET Example

```
Private m_hObject As Integer '// Declared at form level and previously open with a call to OpenNeoDevice

Dim FireReadSettings As SFireSettings
Dim iNumberOfBytes As Integer
Dim iResult As Integer

'//Get the settings
iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(FireReadSettings)
iResult = icsneoGetFireSettings(m_hObject, FireReadSettings, iNumberOfBytes)
If iResult = 0 Then
    MsgBox("Problem reading FIRE configuration")
    Exit Sub
End If
```

*Last Updated : Wednesday, December 17, 2008*

**SetFireSettings Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method writes configuration settings to a neoVI Fire device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoSetFireSettings Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pSettings As SFireSettings, ByVal iNumBytes As Long, ByVal bSaveToEEPROM As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSetFireSettings Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pSettings As SFireSettings, ByVal iNumBytes As Int32, ByVal bSaveToEEPROM As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetFireSettings(Int32 hObject, ref SFireSettings pSettings, Int32 iNumBytes, Int32 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](#).

**Examples****Visual Basic Example**

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim FireReadSettings As SFireSettings
Dim lNumberOfBytes As Long
Dim lResult As Integer

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

lResult = icsneoSetFireSettings(m_hObject, FireReadSettings, LenB(FireReadSettings), 1)
If lResult = 0 Then
    MsgBox("Problem Sending FIRE configuration", vbOKOnly)
    Exit Sub
End If
```

**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)
{
    MessageBox::Show("Problem Sending FIRE configuration");
    return;
}
```

## C# Example

```
SFireSettings FireReadSettings = new SFireSettings();
int iNumberOfBytes;
int iResult;

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

iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VcanReadSettings);
iResult = icsNeoDll.icsneoSetFireSettings(m_hObject, ref FireReadSettings, iNumberOfBytes, 1);
if(iResult == 0)
{
    MessageBox.Show("Problem Sending FIRE configuration");
    return;
}
```

## Visual Basic .NET Example

```
Dim FireReadSettings As SFireSettings
```

```
Dim iNumberOfBytes As Integer
Dim iResult As Integer

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

iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VcanReadSettings)
iResult = icsneoSetFireSettings(m_hObject, FireReadSettings, iNumberOfBytes, 1)
If iResult = 0 Then
    MsgBox("Problem Sending FIRE configuration")
    Exit Sub
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, March 04, 2009*

**Device Settings Functions Overview - ValueCAN3 - intrepidcs API**

Name	Description
<a href="#">GetVCAN3Settings</a>	Gets device and network parameters for a ValueCAN3 device
<a href="#">SetVCAN3Settings</a>	Sets device and network parameters for a ValueCAN3 device

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Wednesday, December 17, 2008*

**GetVCAN3Settings Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method reads the configuration settings from a ValueCAN3 device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetVCAN3Settings Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pSettings As SVCAN3Settings, ByVal iNumBytes As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetVCAN3Settings Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pSettings As SVCAN3Settings, ByVal iNumBytes As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetVCAN3Settings(Int32 hObject, ref SVCAN3Settings pSettings, Int32 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](#).

---

## Examples

### Visual Basic Example

```
Dim VcanReadSettings As SVCAN3Settings
Dim lNumberOfBytes As Long
Dim lResult As Long

'//Get the setting
lResult = icsneoGetVCAN3Settings(m_hObject, VcanReadSettings, LenB(VcanReadSettings) )
If lResult = 0 Then
    MsgBox("Problem reading FIRE configuration", vbOKOnly)
    Exit Sub
End If
```

### C/C++ Example

```
SVCAN3Settings VcanReadSettings;
int iNumberOfBytes;
int iResult;

//Get the settings
iNumberOfBytes=sizeof(VcanReadSettings);
iResult = icsneoGetVCAN3Settings(m_hObject, &VcanReadSettings, iNumberOfBytes);
if(iResult == 0)
{
    MessageBox::Show("Problem reading VCAN configuration");
    return;
}
```

## C# Example

```
//Declared at form level and previously open with a call to OpenNeoDevice
int m_hObject; //handle for device,
SVCAN3Settings VcanReadSettings = new SVCAN3Settings();
int iNumberOfBytes;
int iResult;

//Get the settings
iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VcanReadSettings);
iResult = icsNeoDll.icsneoGetVCAN3Settings(m_hObject, ref VcanReadSettings, iNumberOfBytes);
if (iResult == 0)
{
    MessageBox.Show("Problem reading VCAN configuration");
    return;
}
```

## Visual Basic .NET Example

```
Private m_hObject As Integer '// Declared at form level and previously open with a call to OpenNeoDevice

Dim VcanReadSettings As SVCAN3Settings
Dim iNumberOfBytes As Integer
Dim iResult As Integer

'//Get the settings
iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VcanReadSettings)
iResult = icsneoGetVCAN3Settings(m_hObject, VcanReadSettings, iNumberOfBytes)
If iResult = 0 Then
    MsgBox("Problem reading VCAN configuration")
    Exit Sub
End If
```



**SetVCAN3Settings Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method writes configuration settings to a ValueCAN3 device.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoSetVCAN3Settings Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pSettings As SVCAN3Settings, ByVal iNumBytes As Long, ByVal bSaveToEEPROM As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSetVCAN3Settings Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pSettings As SVCAN3Settings, ByVal iNumBytes As Int32, ByVal bSaveToEEPROM As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetVCAN3Settings(Int32 hObject, ref SVCAN3Settings pSettings, Int32 iNumBytes,
Int32 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](#).

---

**Examples****Visual Basic Example**

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim VCANReadSettings As SVCAN3Settings
Dim lNumberOfBytes As Long
Dim lResult As Integer

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

lResult = icsneoSetVCAN3Settings(m_hObject, VCANReadSettings, LenB(VCANReadSettings), 1)
If lResult = 0 Then
    MsgBox("Problem Sending VCAN configuration", vbOKOnly)
    Exit Sub
End If
```

**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)
{
    MessageBox::Show("Problem Sending VCAN configuration");
    return;
}
```

## C# Example

```
SVCAN3Settings VCANReadSettings = new VCANReadSettings();
int iNumberOfBytes;
int iResult;

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

iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VCANReadSettings);
iResult = icsNeoDll.icsneoSetVCAN3Settings(m_hObject, ref VCANReadSettings , iNumberOfBytes, 1);

if(iResult == 0)
{
    MessageBox.Show("Problem Sending VCAN configuration");
    return;
}
```

## Visual Basic .NET Example

```
Dim VCANReadSettings As SVCAN3Settings
Dim iNumberOfBytes As Integer
Dim iResult As Integer

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

iNumberOfBytes = System.Runtime.InteropServices.Marshal.SizeOf(VCANReadSettings)
iResult = icsneoSetVCAN3Settings(m_hObject, VCANReadSettings, iNumberOfBytes, 1)

If iResult = 0 Then
    MsgBox("Problem Sending VCAN configuration")
    Exit Sub
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**Device Settings Functions Overview - General Device Settings - intrepidcs API**

**These functions are designed to work for both 2G and 3G devices**

Name	Description
<a href="#">SetBitRate</a>	Set the baud or bit rate for a specific neoVI network
<a href="#">GetHWFirmwareInfo</a>	Gets the firmware version of a neoVI device
<a href="#">GetDLLFirmwareInfo</a>	Gets the firmware version stored in the DLL API
<a href="#">ForceFirmwareUpdate</a>	Forces the firmware to updated on a neoVI device
<a href="#">GetDeviceParameters</a>	Gets individual parameters for a neoVI device (2G implementation pending)
<a href="#">SetDeviceParameters</a>	Sets individual parameters for a neoVI device (2G implementation pending)
<a href="#">SetReflashDisplayCallbacks</a>	Sets callback function pointers for use when flashing a neoVI
<a href="#">ClearReflashDisplayCallbacks</a>	Clears callback function pointers for flashing a neoVI

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, February 23, 2009*

**SetBitRate Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method sets bit rates for networks on neoVI devices**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoSetBitRate Lib "icsneo40.dll" (ByVal hObject As Long, ByVal BitRate As Long, ByVal NetworkID As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSetBitRate Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal BitRate As Int32, ByVal NetworkID As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetBitRate(Int32 hObject, Int32 BitRate, Int32 NetworkID);
```

**Parameters*****hObject***

[in] Specifies the driver object created by [OpenNeoDevice](#).

***iBitRate***

[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.

---

#### **Examples**

##### **Visual Basic Example**

```
Dim lResult As Long

'//Set the bit rate
lResult = icsneoSetBitRate(m_hObject, 500000, NETID_HSCAN)
If (lResult = 0) Then MsgBox("Problem setting bit rate")
```

**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");  
}
```

**C# Example:**

```
int iResult;  
  
//Set the bit rate  
iResult = icsNeoDll.icsneoSetBitRate(m_hObject, 500000, NETID_HSCAN);  
if (iResult == 0)  
{  
    MessageBox.Show("Problem setting bit rate");  
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Integer  
'//Set the bit rate  
iResult = icsneoSetBitRate(m_hObject, 500000, NETID_HSCAN)  
If (iResult = 0) Then MsgBox("Problem setting bit rate")
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*



**GetHWFirmwareInfo Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetHWFirmwareInfo Lib "icsneo40.dll" _  
(ByVal hObject As Long, ByRef pInfo As stAPIFirmwareInfo) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetHWFirmwareInfo Lib "icsneo40.dll" _  
(ByVal hObject As Integer, ByRef pInfo As stAPIFirmwareInfo) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoGetHWFirmwareInfo(int hObject, ref 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.

## Examples

### Visual Basic Example

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim FirmwareInfo As stAPIFirmwareInfo
Dim lResult As Integer

lResult = icsneoGetHWFirmwareInfo(m_hObject, FirmwareInfo)
If lResult = 0 Then
    MsgBox("Problem getting the neoVI's firmware information", vbOKOnly)
    Exit Sub
End If
```

### C/C++ Example

```
stAPIFirmwareInfo FirmwareInfo;
int iResult;

iResult = icsneoGetHWFirmwareInfo(m_hObject, &FirmwareInfo);
if(iResult == 0)
{
    MessageBox::Show("Problem getting the neoVI's firmware information");
    return;
}
```

### C# Example

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

iResult = icsNeoDll.icsneoGetHWFirmwareInfo(m_hObject, ref FirmwareInfo);
if(iResult == 0)
{
    MessageBox.Show("Problem getting the neoVI's firmware information");
    return;
}
```

**Visual Basic .NET Example**

```
Dim FirmwareInfo As stAPIFirmwareInfo
Dim iResult As Integer

iResult = icsneoGetHWFirmwareInfo(m_hObject, FirmwareInfo)
If iResult = 0 Then
    MsgBox("Problem getting the neoVI's firmware information")
    Exit Sub
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

**GetDLLFirmwareInfo Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetDLLFirmwareInfo Lib "icsneo40.dll" _  
(ByVal hObject As Long, ByRef pInfo As stAPIFirmwareInfo) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetDLLFirmwareInfo Lib "icsneo40.dll" _  
(ByVal hObject As Integer, ByRef pInfo As stAPIFirmwareInfo) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoGetDLLFirmwareInfo(int hObject, ref 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. This is the version that will be written

to the neoVI device by the [ForceFirmwareUpdate](#) method.

## Examples

### Visual Basic Example

```
Private m_hObject As Long '// Declared at form level and previously open with a call to OpenNeoDevice

Dim FirmwareInfo As stAPIFirmwareInfo
Dim lResult As Integer

lResult = icsneoGetDLLFirmwareInfo(m_hObject, FirmwareInfo)
If lResult = 0 Then
    MsgBox("Problem getting the version of the firmware stored within the DLL API", vbOKOnly)
    Exit Sub
End If
```

### C/C++ Example

```
stAPIFirmwareInfo FirmwareInfo;
int iResult;

iResult = icsneoGetDLLFirmwareInfo(m_hObject, &FirmwareInfo);
if(iResult == 0)
{
    MessageBox::Show("Problem getting the version of the firmware stored within the DLL API");
    return;
}
```

### C# Example

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

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

**Visual Basic .NET Example**

```
Dim FirmwareInfo As stAPIFirmwareInfo
Dim iResult As Integer

iResult = icsneoGetDLLFirmwareInfo(m_hObject, FirmwareInfo)
If iResult = 0 Then
    MsgBox("Problem getting the version of the firmware stored within the DLL API")
    Exit Sub
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**ForceFirmwareUpdate Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method forces the firmware on neoVI device to be updated.**

**C/C++ Declare**

```
void __stdcall icsneoForceFirmwareUpdate(int hObject);
```

**Visual Basic Declare**

```
Public Declare Function icsneoForceFirmwareUpdate Lib "icsneo40.dll" (ByVal hObject As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoForceFirmwareUpdate Lib "icsneo40.dll" (ByVal hObject As Integer)
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern void icsneoForceFirmwareUpdate(int hObject);
```

**Parameters***hObject*

[in] Specifies the driver object created by [OpenNeoDevice](#).

**Return Values**

None.

**Remarks**

This method is used to force the firmware on a neoVI device to be updated to the version stored in the DLL API.

**Examples****Visual Basic Example**

```
Call  icsneoForceFirmwareUpdate (m_hObject)
```

**C/C++ Example**

```
icsneoForceFirmwareUpdate (hObject);
```

**C# Example**

```
icsNeoDll.icsneoForceFirmwareUpdate (m_hObject);
```

**Visual Basic .NET Example**

```
Call  icsneoForceFirmwareUpdate (m_hObject)
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**GetDeviceParameters Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method reads individual neoVI device parameters.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetDeviceParameters Lib "icsneo40.dll" _  
    (ByVal hObject As Long, ByRef pParameters As Byte, ByRef pValues As byte, ByVal ValuesLength As int) As Long
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoGetDeviceParameters(int hObject, ref byte pParameters, ref byte pValues, short  
ValuesLength);
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetDeviceParameters Lib "icsneo40.dll" _  
    (ByVal hObject As Integer, ByRef pParameters As Byte, ByRef pValues As Byte, ByVal ValuesLength As short) As  
Integer
```

**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 been 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 inefficient 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.

---

### Examples

#### Visual Basic Example

#### C/C++ Example

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

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

#### Visual Basic .NET Example

## C# Example

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, November 17, 2008*

**SetDeviceParameters Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method changes neoVI device parameters.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoSetDeviceParameters Lib "icsneo40.dll" _  
    (ByVal hObject As Long, ByRef pParmValue As Byte, ByRef pErrorIndex As Long, ByVal bSaveToEEPROM As Long) As  
Long
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoSetDeviceParameters(int hObject, ref byte pParmValue, ref int pErrorIndex, int  
bSaveToEEPROM);
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSetDeviceParameters Lib "icsneo40.dll" _  
(ByVal hObject As Int32, ByVal pParmValue As String, ByRef pErrorIndex As Int32, ByVal bSaveToEEPROM As Int32) As  
Int32
```

**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 inefficient 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.

---

### Examples

#### Visual Basic 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);
```

### Visual Basic .NET Example

```
Dim ReturnVal As Int32
Dim Errors As Int32
Dim sCommand As String

sCommand = "network_enables=0"
ReturnVal = icsneoSetDeviceParameters(m_hObject, sCommand, Errors, 0)
```

### C# Example

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, June 08, 2010*

**SetReflashDisplayCallbacks Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method is used to set 'call back' functions that will be called by the intrepidcs API when flashing a neoVI device with the [ForceFirmwareUpdate](#) function. This overrides the Windows dialog box that normally displays the progress of a neoVI flash update. The use of call back functions allows the client application to receive the status messages and display them as desired.**

**C/C++ Declare**

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

**Visual Basic Declare****Visual Basic .NET Declare****C# Declare****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 USZB 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**

After calling this function you must call the [ForceFirmwareUpdate](#) function to cause the neoVI device firmware to be updated. Once the callbacks have been set they are valid and active until the the DLL is unloaded or until the [ClearReflashDisplayCallbacks](#) function is called.

---

**Examples****Visual Basic Example****C/C++ Example:****C# Example:****Visual Basic .NET Example:**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, January 22, 2009*

**ClearReflashDisplayCallbacks Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method is used to clear callback functions that were set using the [SetReflashDisplayCallback](#) method.**

**C/C++ Declare**

```
void __stdcall icsneoClearReflashDisplayCallbacks(void);
```

**Visual Basic Declare****Visual Basic .NET Declare****C# Declare****Parameters**

None

**Return Values**

None

**Remarks**

---

**Examples****Visual Basic Example****C/C++ Example:**

**C# Example:**

**Visual Basic .NET Example:**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, January 22, 2009*

**GetRTC Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetRTC Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pTime As icsSpyTime) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetRTC Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pTime As icsSpyTime) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetRTC(Int32 hObject, ref 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

### Visual Basic Example

### C/C++ Example:

### C# Example:

### Visual Basic .NET Example:

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

**SetRTC Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoSetRTC Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pTime As icsSpyTime) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoSetRTC Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pTime As icsSpyTime) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoSetRTC(Int32 hObject, ref 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

### Visual Basic Example

### C/C++ Example:

### C# Example:

### Visual Basic .NET Example:

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

## Error Functions Overview - intrepidcs API

Name	Description
<a href="#">GetLastAPIError</a>	Returns the error generated by the last intrepidcs API call
<a href="#">GetErrorMessages</a>	Returns the intrepidcs API error message queue
<a href="#">GetErrorInfo</a>	Returns a text description of an intrepidcs API error

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, September 18, 2008*

**GetLastAPIError Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetLastAPIError Lib "icsneo40.dll" (ByVal hObject As Long, ByRef piErrorNumber As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetLastAPIError Lib "icsneo40.dll" (ByVal hObject As Integer, ByRef piErrorNumber As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoGetLastAPIError(int hObject, ref 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

---

## Examples

### Visual Basic Example

### C/C++ Example:

### C# Example:

### Visual Basic .NET Example:

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, September 17, 2008*

**GetErrorMessages Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Value](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method reads the neoVI DLL error message queue.**

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetErrorMessages Lib "icsneo40.dll" (ByVal hObject As Long, ByRef pErrorMsgs As Long, ByRef pNumberOfErrors As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetErrorMessages Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef pErrorMsgs As Int32, ByRef pNumberOfErrors As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoGetErrorMessages(Int32 hObject, ref Int32 pErrorMsgs, ref Int32 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. A separate topic describes the possible values for [error messages](#). 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.

---

## Examples

### Visual Basic Example

```
'// Declared at form level
Private m_hObject As Long '// Holds the object for the state of the application

Dim lResult As Long
Dim lErrors(0 to 599) As Long
Dim lNumberOfErrors As Long

'// Read Out the errors
lResult = icsneoGetErrorMessages(m_hObject, lErrors(0), lNumberOfErrors)
'// Test the returned result
If Not CBool(lResult) Then
    MsgBox "Problem Reading Errors"
End If
```

### C/C++ Example

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

int iErrors[599];
int lResult;
int lNumberOfErrors;
TCHAR szOut[200];
long lCount;

// Read the errors from the DLL
lResult = icsneoGetErrorMessages(hObject, iErrors, &lNumberOfErrors);
if (lResult == 0)
    MessageBox(hWnd, TEXT("Problem Reading errors"), TEXT("neoVI Example"), 0);
```

```
// dump the neoVI errors to the debug window
if(lNumberOfErrors > 0)
{
    for(lCount=0;lCount <lNumberOfErrors; lCount++)
    {
        wsprintf(szOut,TEXT("Error %d\n"),iErrors[lCount]);
        OutputDebugString(szOut);
    }
}
else
    OutputDebugString(TEXT("No Errors to report\n"));
```

### Visual Basic .NET Example

```
Dim lResult As Integer '//Storage for result of Function Call
Dim lErrors(600) As Integer '//Array for Error information
Dim lNumberOfErrors As Integer '//Storage for Number of Errors

'// Read Out the errors
lResult = icsneoGetErrorMessages(m_hObject, lErrors(0), lNumberOfErrors)

'// Test the returned result
If Not CBool(lResult) Then
    MsgBox("Problem Reading Errors")
End If
```

### C# Example

```
int iResult = 0; //Storage for Result of Call
int[] iErrors = new int[600]; //Array for Error Numbers
int iNumberOfErrors = 0; // Storage for number of errors

// Read Out the errors
iResult = icsNeoDll.icsneoGetErrorMessages(m_hObject, ref iErrors[0], ref iNumberOfErrors);
```

**GetErrorInfo Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Value](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

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

**C/C++ Declare**

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

**Visual Basic Declare**

```
Public Declare Function icsneoGetErrorInfo Lib "icsneo40.dll" _  
    (ByVal lErrorNumber As Long, _  
    ByVal sErrorDescriptionShort As String, _  
    ByVal sErrorDescriptionLong As String, _  
    ByRef lMaxLengthShort As Long, _  
    ByRef lMaxLengthLong As Long, _  
    ByRef lErrorSeverity As Long, _  
    ByRef lRestartNeeded As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetErrorInfo Lib "icsneo40.dll" _  
    (ByVal lErrorNumber As Int32, _  
    ByVal sErrorDescriptionShort As String, _  
    ByVal sErrorDescriptionLong As String, _  
    ByRef lMaxLengthShort As Int32, _  
    ByRef lMaxLengthLong As Int32, _  
    ByRef lErrorSeverity As Int32, _  
    ByRef lRestartNeeded As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoGetErrorInfo(int iErrorNumber,
    StringBuilder sErrorDescriptionShort,
    StringBuilder sErrorDescriptionLong,
    ref int iMaxLengthShort,
    ref int iMaxLengthLong,
    ref int lErrorSeverity ,
    ref int lRestartNeeded);
```

## Parameters

### *iErrorNumber*

[in] This is the number of the error message returned from [GetErrorMessages](#). A separate topic describes the possible values for [error messages](#).

### *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.

### *iMaxLengthShort*

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

### *iMaxLengthLong*

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

### *iErrorSeverity*

[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.

### *iRestartNeeded*

[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.

**Examples****Visual Basic Example****This function assists with use of this function in Visual Basic**

```
Public Function icsneoGetDLLErrorInfo(ByVal lErrorNum As Long, ByVal sErrorShort As String, _
ByVal sErrorLong As String, ByVal lSeverity As Long, ByVal bRestart As Long) As Boolean

Dim lErrorLongLength As Long
Dim lErrorShortLength As Long
Dim lRestart As Long
Dim lResult As Long

sErrorLong = String(255, 0)
sErrorShort = String(255, 0)
lErrorLongLength = 255
lErrorShortLength = 255
lResult = icsneoGetErrorInfo(lErrorNum, sErrorShort, sErrorLong, _
lErrorShortLength, lErrorLongLength, lSeverity, lRestart)
sErrorShort = Left$(sErrorShort, lErrorShortLength)
sErrorLong = Left$(sErrorLong, lErrorLongLength)
bRestart = CBool(lRestart)
icsneoGetDLLErrorInfo = CBool(lResult)
```

---

End Function

### This function reads errors and loads them into a list box

```
Private Sub cmdGetErrors_Click()

Dim lResult As Long
Dim lErrors(0 To 599) As Long
Dim lNumberOfErrors As Long
Dim lCount As Long
Dim sErrorShort As String
Dim sErrorLong As String
Dim lSeverity As Long
Dim bRestart As Long

    // Read Out the errors
lResult = icsneoGetErrorMessages(m_hObject, lErrors(0), lNumberOfErrors)
    // Test the returned result
If Not CBool(lResult) Then
    MsgBox("Problem Reading Errors")
Else
    lstErrorHolder.Clear()
    For lCount = 1 To lNumberOfErrors
        Call icsneoGetDLLErrorInfo(lErrors(lCount - 1), sErrorShort, sErrorLong, lSeverity, bRestart)
        lstErrorHolder.AddItem(sErrorShort & " - Description" & sErrorLong & " - Errornum: " & lErrors(lCount - 1))
    Next lCount
End If
End Sub
```

### C/C++ Example

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

if (lResult == 0)
    MessageBox(hWnd, TEXT("Problem Reading errors"), TEXT("neoVI Example"), 0);

// dump the neoVI errors
if (lNumberOfErrors > 0)
{
```

```
for (lCount=0;lCount <lNumberOfErrors;lCount++)  
{  
  
    wsprintf(szOut,TEXT("Error %d - "),iErrors[lCount]);  
    OutputDebugString(szOut);  
    icsneoGetErrorInfo(iErrors[lCount],szDescriptionShort,szDescriptionLong,  
        &lMaxLengthShort,&lMaxLengthLong,&lErrorSeverity,&lRestartNeeded);  
  
    OutputDebugString(szDescriptionShort);  
    OutputDebugString(TEXT("\n"));  
}  
}  
else  
    OutputDebugString(TEXT("No Errors to report\n"));
```

### Visual Basic .NET Example

```
Public Function icsneoGetDLLErrorInfo(ByVal lErrorNum As Int32, ByRef sErrorShort As String, ByRef sErrorLong As  
String, ByRef lSeverity As Int32, ByRef bRestart As Int32) As Boolean  
  
Dim lErrorLongLength As Int32  
Dim lErrorShortLength As Int32  
Dim lRestart As Int32  
Dim lResult As Int32  
  
sErrorLong = New String(Chr(0), 255)  
sErrorShort = New String(Chr(0), 255)  
lErrorLongLength = 255  
lErrorShortLength = 255  
lResult = icsneoGetErrorInfo(lErrorNum, sErrorShort, sErrorLong, lErrorShortLength, lErrorLongLength,  
lSeverity, lRestart)  
  
sErrorShort = Left(sErrorShort, lErrorShortLength)  
sErrorLong = Left(sErrorLong, lErrorLongLength)  
bRestart = CBool(lRestart)  
icsneoGetDLLErrorInfo = CBool(lResult)  
End Function
```

**This function reads errors and loads them into a list box**

```

Private Sub cmdGetErrors_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdGetErrors.Click

Dim lResult As Integer '//Storage for result of Function Call
Dim lErrors(600) As Int32 '//Array for Error information
Dim lNumberOfErrors As Integer '//Storage for Number of Errors
Dim lCount As Integer '//Counter
Dim sErrorShort As String '//String Holding Short Error Name
Dim sErrorLong As String '//String Holding Long Error Name
Dim iSeverity As Int32 '//Storage for Level of error
Dim bRestart As Int32 '//flag for if Restart is required

'// Read Out the errors
lResult = icsneoGetErrorMessages(m_hObject, lErrors(0), lNumberOfErrors)
'// Test the returned result
If Not CBool(lResult) Then
    MsgBox("Problem Reading Errors")
Else
    '//List Error information
    lstErrorHolder.Items.Clear()
    For lCount = 1 To lNumberOfErrors
        Call icsneoGetDLLErrorInfo(lErrors(lCount - 1), sErrorShort, sErrorLong, iSeverity, bRestart)
        lstErrorHolder.Items.Add(sErrorShort + " - Description" +
                               sErrorLong + " - Errornum: " + Convert.ToString(lErrors(lCount - 1)))
    Next lCount
End If
End Sub

```

## C# Example

```

private void cmdGetErrors_Click(object sender, System.EventArgs e)
{
    int iResult = 0; //Storage for Result of Call
    int[] iErrors = new int[600]; //Array for Error Numbers
    int iNumberOfErrors = 0; // Storage for number of errors
    int iCount= 0; //Counter
    int iSeverity =0; //tells the Severity of Error
    int iMaxLengthShort = 0; //Tells Max length of Error String
    int iMaxLengthLong = 0; //Tells Max Length of Error String
    int lRestart = 0; //tells if a restart is needed
    StringBuilder sErrorShort = new StringBuilder(256); //String for Error

```

```
StringBuilder sErrorLong = new StringBuilder(256); //String for Error
iMaxLengthShort = 1; //Set initial conditions
iMaxLengthLong = 1; //Set initial conditions
// Read Out the errors
iResult = icsNeoDll.icsneoGetErrorMessages(m_hObject, ref iErrors[0], ref iNumberOfErrors);
// Test the returned result
if(iResult == 0)
{
    MessageBox.Show ("Problem Reading Errors");
}
else
{
    if(iNumberOfErrors != 0)
    {
        for(iCount=0;iCount< iNumberOfErrors;iCount++)
        {
            //Get Text Description of the Error
            iResult = icsNeoDll.icsneoGetErrorInfo(iErrors[iCount],
                sErrorShort, sErrorLong ,ref iMaxLengthShort , ref iMaxLengthLong, ref iSeverity,ref lRestart);
            lstErrorHolder.Items.Add (sErrorShort + " - Description " + sErrorLong + " - Errornum: " +
iErrors[iCount]);
        }
    }
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 17, 2008*

**Error Messages - intrepidcs API**[Main](#)

Table 1 lists the possible error messages returned in the [GetErrorMessages](#) API call. Some of these errors may also be returned by the [GetLastAPIError](#) API call.

**Table 1 - Error Messages**

<b>Error</b>	<b>Description</b>
NEOVI_ERROR_DLL_TX_BUFFER_OVERFLOW =0	The transmit buffer in the DLL has overflowed. This could occur if the USB connection was lost. It also could occur if you transmit more messages than can be sent on the vehicle networks.
NEOVI_ERROR_DLL_ERROR_BUFFER_OVERFLOW =1	This error occurs when the error queue overflows.
NEOVI_ERROR_DLL_USB_SEND_DATA_ERROR =2	This error occurs when there is a problem sending data on USB.
NEOVI_ERROR_DLL_ISO_DATA_BUFFER_ALLOC =3	Internal Driver Error.
NEOVI_ERROR_DLL_ISO_DATA_READ_BUFFER =4	Internal Driver Error.
NEOVI_ERROR_DLL_ISO_DATA_ZERO_PACKETS =5	Internal Driver Error.
NEOVI_ERROR_DLL_RX_MSG_BUFFER_OVERFLOW =6	This error occurs if the DLL overflows it's receive message buffer. Solve this error by calling <a href="#">GetMessages</a> at a faster interval.
NEOVI_ERROR_DLL_STOP_ISO_STREAM =7	Internal Driver Error.
NEOVI_ERROR_DLL_INVALID_NETID =8	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_RX_THREAD =9	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_TX_THREAD =10	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN_PIC_BUFFER_OVERFLOW =11	This error occurs if there is an overflow in the neoVI internal buffer.
NEOVI_ERROR_DLL_INVALID_DEVICE_RESPONSE =12	Internal Driver Error.
NEOVI_ERROR_DLL_ISOTX_DATA_BUFFER_ALLOC =13	Internal Driver Error.
NEOVI_ERROR_DLL_RX_CMD_BUFFER_OVERFLOW=14	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_RX_BUFFER_OVERFLOW=15	RS232 Buffer Overflow Error

NEOVI_ERROR_DLL_RS232_ERR_READCOMERR =16	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_ERR_READ=17	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_BUFFER_ALLOC=18	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_TX_BUFFER_OVERFLOW=19	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_MISC_ERROR=20	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_FIND_WRITE=21	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_FIND_BUFFER_ALLOC=22	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_FIND_CLEARCOMM=23	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_FIND_READCOMM=24	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_FIND_TIMEOUT=25	This error occurs if the neoVI DLL could not find the neoVI device on the specified RS232 port.
NEOVI_ERROR_DLL_RS232_ERR_BREAK=26	RS232 Break Error.
NEOVI_ERROR_DLL_RS232_ERR_FRAME=27	RS232 Framing Error.
NEOVI_ERROR_DLL_RS232_ERR_IOE=28	RS232 IOE Error.
NEOVI_ERROR_DLL_RS232_ERR_OVERRUN=29	RS232 Overrun Error.
NEOVI_ERROR_DLL_RS232_ERR_PARITY=30	RS232 Parity Error.
NEOVI_ERROR_DLL_RS232_TXBUFFER_ALLOC=31	Internal Driver Error.
NEOVI_ERROR_DLL_USB_TX_RS232_ERROR=32	Internal Driver Error. <small>Problem opening RS232 port. This is probably caused by a hardware problem or a driver issue.</small>

NEOVI_ERROR_DLL_RS232_CREATE_FILE=33	Problem opening RS232 port. This is probably caused by another application using the port or the port not being valid.
NEOVI_ERROR_DLL_RS232_GET_COMM_STATE=34	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_SET_COMM_STATE=35	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_START_COMM_RX_THREAD=36	Internal Driver Error.
NEOVI_ERROR_DLL_RS232_START_COMM_TX_THREAD=37	Internal Driver Error.
NEOVI_ERROR_DLL_SYNC_COUNT_ERR=38	A message rollover timestamp was missed. This is caused if the neoVI device is disconnected or powered down when in RS232 mode. It will also be caused if two neoVI applications open up the same USB port.
NEOVI_ERROR_DLL_RX_MSG_FRAME_ERR=39	A neoVI message packet was not properly formatted.
NEOVI_ERROR_DLL_RX_MSG_FIFO_OVER=40	The internal DLL FIFO used to store data received before parsing has overflowed.
NEOVI_ERROR_DLL_RX_MSG_CHK_SUM_ERR=41	A neoVI message packet was properly formatted but had an incorrect checksum.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_BULKIN_THREAD=42	Internal Driver Error.
NEOVI_ERROR_DLL_BULKIN_ERR_READ=43	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_RX_FIFO_OVERFLOW=44	The Rx FIFO used to store network data before it is sent to the PC has overflowed.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW=45	Each network has a FIFO on the Main51 controller for transmission. If you send messages to neoVI faster than neoVI can transmit them, you will receive this error.
NEOVI_ERROR_DLL_MAIN51_DEV_FIFO_OVERFLOW=46	This is a FIFO over error related to messages passed from the Main51 uController to the Main PIC uController
NEOVI_ERROR_DLL_RESET_STATUS_CHANGED=47	The neoVI reset status has changed. This error would occur if the neoVI had a watchdog reset.

NEOVI_ERROR_DLL_ISO_LONG_CACHE_OVERFLOW=48	Internal Driver Error.
NEOVI_ERROR_DLL_ISORX_LONG_BUFFER_ALLOC=49	Internal Driver Error.
NEOVI_ERROR_DLL_J1708_LONG_CACHE_OVERFLOW=50	Internal Driver Error.
NEOVI_ERROR_DLL_J1708_LONG_BUFFER_ALLOC=51	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_DEVICE=52	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_HSCAN=53	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_MSCAN=54	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_SWCAN=55	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_LSFTCAN=56	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_FORDSCP=57	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_J1708=58	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_AUX=5	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_JVPW=60	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_ISO=61	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_ISOPIC=62	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_MAIN51=63	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_HOST=64	Internal Driver Error.
NEOVI_ERROR_DLL_READ_ENTIRE_DEEPROM_ERROR=65	Internal Driver Error.
NEOVI_ERROR_DLL_WRITE_ENTIRE_DEEPROM_ERROR=66	Internal Driver Error.
NEOVI_ERROR_DLL_USB_PORT_ALREADY_OPEN=67	Internal Driver Error.
NEOVI_ERROR_DLL_JVPW_TX_REPORT_FIFO_ERR_IN=68	Internal Driver Error.
NEOVI_ERROR_DLL_ISOJ_TX_REPORT_FIFO_ERR_IN=69	Internal Driver Error.
NEOVI_ERROR_DLL_JVPW_TX_REPORT_FIFO_ERR_OUT=70	Internal Driver Error.
NEOVI_ERROR_DLL_ISOJ_TX_REPORT_FIFO_ERR_OUT=71	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_IN_FROM_HOST_FIFO=72	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_HOST_CHKSUM=73	Internal Driver Error.
NEOVI_ERROR_DLL_ISOJ_TX_HOST_MISSED_BYTE=74	Internal Driver Error.
NEOVI_ERROR_DLL_NEovi_NO_RESPONSE=75	Internal Driver Error.
NEOVI_ERROR_DLL_RX_SOCKET_FIFO_OVER=76	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_TXSOCKET_THREAD=77	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_RXSOCKET_THREAD=78	Internal Driver Error.

NEOVI_ERROR_DLL_PROBLEM_STOPPING_TXSOCKET_CLIENT_THREAD=78	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_STOPPING_RXSOCKET_CLIENT_THREAD=79	Internal Driver Error.
NEOVI_ERROR_DLL_TCP_CLIENT_TX=80	Internal Driver Error.
NEOVI_ERROR_DLL_TCP_CLIENT_RX=81	Internal Driver Error.
NEOVI_ERROR_DLL_TCP_CLIENT_RX_SOCK=82	Internal Driver Error.
NEOVI_ERROR_DLL_UNABLE_CONNECT_TO_SRVR=83	Internal Driver Error.
NEOVI_ERROR_DLL_UNABLE_CREATE_CLIENT SOCK=84	Internal Driver Error.
NEOVI_ERROR_DLL_UNABLE_WSASTARTUP=85	Internal Driver Error.
NEOVI_ERROR_DLL_SOCK_CL_RD_BUFFER_ALLOC=86	Internal Driver Error.
NEOVI_ERROR_DLL_SOCK_CL_TX_BUFFER_ALLOC=87	Internal Driver Error.
NEOVI_ERROR_DLL_SOCK_SRVR_RX_BUFFER_ALLOC=88	Internal Driver Error.
NEOVI_ERROR_DLL_SOCK_SRVR_TX_BUFFER_ALLOC=89	Internal Driver Error.
NEOVI_ERROR_DLL_ILLEGAL_TX_NETWORK=90	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_TX_HOST_OVERRUN=91	Internal Driver Error.
NEOVI_ERROR_DLL_OPEN_GET_COMM_TIMEOUT=92	Internal Driver Error.
NEOVI_ERROR_DLL_OPEN_SET_COMM_TIMEOUT=93	Internal Driver Error.
NEOVI_ERROR_DLL_OPEN_READ_DEVICE_TYPE=94	Internal Driver Error.
NEOVI_ERROR_DLL_OPEN_READ_DEVICE_TYPE_TOUT=95	Internal Driver Error.
NEOVI_ERROR_DLL_CLOSE_PURGE_COMM=96	Internal Driver Error.
NEOVI_ERROR_DLL_TX_COM_FIFO_OVERFLOW=97	Internal Driver Error.
NEOVI_ERROR_DLL_GET_USB SERIAL_DEVICES=98	Internal Driver Error.
NEOVI_ERROR_DLL_USB_TX_RS232_BCOUNT = 99	Internal Driver Error.
NEOVI_ERROR_DLL_OPEN_INBOOTLOADER = 100	Internal Driver Error.
NEOVI_ERROR_DLL_TOO_MANY_PERIODICS = 101	Internal Driver Error.
NEOVI_ERROR_DLL_PROBLEM_FIRMWARE_INFO = 102	Internal Driver Error.
NEOVI_ERROR_DLL_NRED_ODDNUMBYTES = 103	Internal Driver Error.
NEOVI_ERROR_DLL_NRED_UNKNOWN_RED_NETID = 104	Internal Driver Error.
NEOVI_ERROR_DLL_RED_NOT_SUPPORTED = 105	Internal Driver Error.
NEOVI_ERROR_DLL_RED_BL_START_INDEX = 106	Internal Driver Error.
NEOVI_ERROR_DLL_3G_BL_FAILURE = 107	Internal Driver Error.

NEOVI_ERROR_DLL_RED_BL_END_INDEX = 116	Internal Driver Error.
NEOVI_ERROR_DLL_RED_FAILED_TO_ENTER_BL = 117	Internal Driver Error.
NEOVI_ERROR_DLL_RED_REQ_SERIAL_NUMBER = 118	Internal Driver Error.
NEOVI_ERROR_DLL_RED_AUTHENTICATE = 119	Internal Driver Error.
NEOVI_ERROR_DLL_RED_APP_VERSION = 120	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SET_BAUD_COMM_FAILURE = 121	Internal Driver Error.
NEOVI_ERROR_DLL_RED_INVALID_BAUD_SPECIFIED = 122	Internal Driver Error.
NEOVI_ERROR_DLL_RED_INVALID_BAUD_SPECIFIED = 122	Internal Driver Error.
NEOVI_ERROR_DLL_RED_READ_BAUD_COMM_FAILURE = 123	Internal Driver Error.
NEOVI_ERROR_DLL_RED_FAILED_TO_SAVE_EEPROM = 124	Internal Driver Error.
NEOVI_ERROR_DLL_RED_FAILED_TO_UPDATE_WAVEFORM_CHANNEL = 125	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_UNKNOWN_NETWORK = 126	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_UNKNOWN_NETWORK = 127	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_UNKNOWN_NETWORK = 128	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_HSCAN1 = 129	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_HSCAN1 = 130	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_HSCAN1 = 131	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_HSCAN1 = 132	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_HSCAN2 = 133	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_HSCAN2 = 134	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_HSCAN2 = 135	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_HSCAN2 = 136	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_MSCAN1 = 137	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_MSCAN1 = 138	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_MSCAN1 = 139	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_MSCAN1 = 140	Internal Driver Error.

NEOVI_ERROR_DLL_RED_RX_MSG_FULL_HSCAN3 = 141	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_HSCAN3 = 142	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_HSCAN3 = 143	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_HSCAN3 = 144	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_SWCAN = 145	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_SWCAN = 146	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_SWCAN = 147	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_SWCAN = 148	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_LSFTCAN = 149	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_LSFTCAN = 150	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_LSFTCAN = 151	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_LSFTCAN = 152	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_LIN1 = 153	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_LIN1 = 154	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_LIN1 = 155	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_LIN1 = 156	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_LIN2 = 157	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_LIN2 = 158	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_LIN2 = 159	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_LIN2 = 160	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_LIN3 = 161	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_LIN3 = 162	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_LIN3 = 163	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_LIN3 = 164	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_LIN4 = 165	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_LIN4 = 166	Internal Driver Error.

NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_LIN4 = 167	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_LIN4 = 168	Internal Driver Error.
NEOVI_ERROR_DLL_USB_PURGE_FAILED = 169	Internal Driver Error.
NEOVI_ERROR_FIRE_COMM_BAD_PACKET = 170	Internal Driver Error.
NEOVI_ERROR_FIRE_CGI_COMM_BAD_PACKET = 171	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_HSCAN1 = 172	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_HSCAN2 = 173	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_HSCAN3 = 174	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_MSCAN = 175	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_SWCAN = 176	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_LSFTCAN = 177	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_LIN1 = 178	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_LIN2 = 179	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_LIN3 = 180	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_LIN4 = 181	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_UNKNOWN_NETWORK = 182	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_JVPW = 183	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_JVPW = 184	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_JVPW = 185	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_JVPW = 186	Internal Driver Error.
NEOVI_ERROR_DLL_INTERNAL_SERIAL_NO_DOES_NOT_MATCH_REGISTRY_SERIAL_NO = 187	Internal Driver Error.
NEOVI_ERROR_DLL_JVPW_LONG_CACHE_OVERFLOW = 188	Internal Driver Error.
NEOVI_ERROR_DLL_FAILED_TO_SET_LICENSE = 189	Internal Driver Error.
NEOVI_ERROR_DLL_3G_DEVICE_LICENSE_NEEDS_TO_BE_UPGRADED = 190	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_HSCAN = 191	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_MSCAN = 192	Internal Driver Error.

NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_HSCAN2 = 193	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_HSCAN3 = 194	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_LSFT = 195	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_SW = 196	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_LIN1 = 197	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_LIN2 = 198	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_LIN3 = 199	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_LIN4 = 200	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_JVPW = 201	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_KYW = 202	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_J1708 = 203	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_RTC_INVALID = 204	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_LOADED_DEFAULT_SETTINGS = 205	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_CGI = 206	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_CGI = 207	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_CGI = 208	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_CGI = 209	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_CGI = 210	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_CGI = 211	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_JVPW = 212	Internal Driver Error.
NEOVI_ERROR_DLL_INVALID_SCRIPT_LOCATION = 213	Internal Driver Error.
NEOVI_ERROR_DLL_SDCARD_NOT_INSERTED = 214	Internal Driver Error.
NEOVI_ERROR_DLL_SDCARD_NOT_FORMATTED = 215	Internal Driver Error.
NEOVI_ERROR_DLL_SDCARD_WRITE_ERROR = 216	Internal Driver Error.
NEOVI_ERROR_DLL_SDCARD_READ_ERROR = 217	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_START_ERROR = 218	Internal Driver Error.

NEOVI_ERROR_DLL_SCRIPT_INVALID_FUNCBLOCK_INDEX = 219	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_ERROR_DOWNLOADING_SCRIPT = 220	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_ERROR_CLEARING_SCRIPT = 221	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_ISO = 222	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_ISO = 223	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_INVALID_MSG_INDEX = 224	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_INVALID_APPSIG_INDEX = 225	Internal Driver Error.
NEOVI_ERROR_DLL_SCRIPT_NO_SCRIPT_RUNNING = 226	Internal Driver Error.
NEOVI_ERROR_DLL_COULD_NOT_SET_SETTINGS_FIRMWARE_MISMATCH = 227	Internal Driver Error.
NEOVI_ERROR_DLL_FIRE_CGI_TX_NOT_ENABLED = 228	Internal Driver Error.
NEOVI_ERROR_DLL_SEND_DEVICE_CONFIG_ERROR = 229	Internal Driver Error.
NEOVI_ERROR_DLL_GET_DEVICE_CONFIG_ERROR = 230	Internal Driver Error.
NEOVI_ERROR_DLL_UNKNOWN_NEovi_TYPE = 231	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_ISO2 = 232	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_ISO3 = 233	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_ISO4 = 234	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_ISO2 = 235	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_ISO2 = 236	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_ISO2 = 237	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_ISO2 = 238	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_ISO2 = 239	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_ISO3 = 240	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_ISO3 = 241	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_ISO3 = 242	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_ISO3 = 243	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_ISO3 = 244	Internal Driver Error.

NEOVI_ERROR_DLL_RED_RX_MSG_FULL_ISO4 = 245	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_ISO4 = 246	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_ISO4 = 247	Internal Driver Error.
NEOVI_ERROR_DLL_RED_DRIVER_OVERFLOW_ISO4 = 248	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_ISO4 = 249	Internal Driver Error.
NEOVI_ERROR_DLL_RED_FAILED_TO_CLEAR_LIN_SLAVE_DATA = 250	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_ENABLED_ISO1 = 251	Internal Driver Error.
NEOVI_ERROR_DLL_RED_RX_MSG_FULL_ISO1 = 252	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_MSG_FULL_ISO1 = 253	Internal Driver Error.
NEOVI_ERROR_DLL_RED_TX_REPORT_MSG_FULL_ISO1 = 254	Internal Driver Error.
NEOVI_ERROR_DLL_RED_SETTINGS_NOT_SET_ISO1 = 255	Internal Driver Error.
NEOVI_ERROR_DLL_DROPPED_RTC_CMD = 256	Internal Driver Error.
NEOVI_ERROR_DLL_J1850_TX_REQUESTS_FLUSHED = 257	Internal Driver Error.
NEOVI_ERROR_J1708_COMM_BAD_PACKET = 258	Internal Driver Error.
NEOVI_ERROR_DLL_NETWORK_NOT_SUPPORTED_BY_HARDWARE = 259	Internal Driver Error.
NEOVI_ERROR_DLL_FEATURE_NOT_UNLOCKED = 260	Internal Driver Error.
NEOVI_ERROR_DLL_DEVICE_NOT_POWERED = 261	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_OK = 262	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORBADINIT = 263	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORNOCANPIPE = 264	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORISONEG7FRESPONSE = 265	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORTOOLNOTSELECTED = 266	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORINVALIDDEVICESELECTED = 267	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORCOULDNOTOPENTOOL = 268	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORNOFLOWCONTROL = 269	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORUNSPECIFIC = 270	Internal Driver Error.

NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORCOREMININULLPTR = 271	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORCOREMINIZEROLEN = 272	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORRESERVED3 = 273	Internal Driver Error.
NEOVI_ERROR_DLL_3GCANDOWNLOADER_ERRORRESERVED4 = 274	Internal Driver Error.
NEOVI_ERROR_DLL_YELLOW_DEPRECATED = 275	Internal Driver Error.
NEOVI_ERROR_DLL_HARDWARE_FAILURE_SRAM = 276	Internal Driver Error.
NEOVI_ERROR_ACTIVE_CONNECTION_ALREADY_EXISTS = 277	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_RTC_FAILED_READ = 278	Internal Driver Error.
NEOVI_ERROR_DLL_MAIN51_RTC_AUTO_RTC = 279	Internal Driver Error.
NEOVI_ERROR_DLL_SEND_DEVICE_CONFIG_NOTPOSSIBLE = 287	Internal Driver Error.
NEOVI_ERROR_CHANNEL_LOCKED_BY_OTHER_CLIENT = 288	Occurs when an application tries to acquire a lock on a channel that has already been locked.
NEOVI_ERROR_NEOVISERVER_GENERAL_ERROR = 289	A general error has occurred with the neoVIServer.
NEOVI_ERROR_CHANNEL_LOCKING_NOT_SUPPORTED_FOR_DEVICE = 290	The connected device does not support channel locking.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Monday, March 25, 2013

**General Utility Functions Overview - intrepidcs API**

Name	Description
<a href="#">ValidateHObject</a>	Used to determine if an hObject reference is valid
<a href="#">GetDLLVersion</a>	Returns DLL version information
<a href="#">StartSockServer</a>	Starts the TCP/IP socket server at a specified port.
<a href="#">StopSockServer</a>	Stops the TCP/IP socket server

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Wednesday, December 17, 2008*

**ValidateHObject Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method is used to determine if a driver object is valid.**

**C/C++ Declare**

```
int __stdcall icsneoValidateHObject(int hObject);
```

**Visual Basic Declare**

```
Public Declare Function icsneoValidateHObject Lib "icsneo40.dll" (ByVal hObject As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoValidateHObject Lib "icsneo40.dll" (ByVal hObject As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoValidateHObject(int hObject);
```

**Parameters*****hObject***

[in] Specifies the driver object created by [OpenNeoDevice](#).

**Return Values**

1 if the hObject is valid. 0 if the object is invalid.

**Remarks**

A driver object will be invalid if it was never initialized by [OpenNeoDevice](#). Calling [ClosePort](#) will not invalidate a driver object; only [FreeObject](#) will do so.

---

**Examples****Visual Basic Example**

```
If (icsneoValidateHObject(m_hObject)) Then
    cmdCheckHardwareHandle.Caption = "Good"
Else
    cmdCheckHardwareHandle.Caption = "Lost"
End If
```

**C/C++ Example:**

```
if(Convert::ToBoolean(icsneoValidateHObject(m_hObject)))
{
    cmdCheckHardwareHandle->Text = "Good";
}
else
{
    cmdCheckHardwareHandle->Text = "Lost";
}
```

**C# Example:**

```
if (Convert.ToBoolean (icsNeoDll.icsneoValidateHObject(m_hObject)))
{
    cmdCheckHardwareHandle.Text = "Good";
}
else
{
    cmdCheckHardwareHandle.Text = "Lost";
}
```

**Visual Basic .NET Example:**

```
If (CBool(icsneoValidateHObject(m_hObject))) Then
    cmdCheckHardwareHandle.Text = "Good"
Else
    cmdCheckHardwareHandle.Text = "Lost"
End If
```



**GetDLLVersion Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method returns the software version of the DLL.**

**C/C++ Declare**

```
int __stdcall icsneoGetDLLVersion();
```

**Visual Basic Declare**

```
Public Declare Function icsneoGetDLLVersion Lib "icsneo40.dll" () As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetDLLVersion Lib "icsneo40.dll" () As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoGetDLLVersion();
```

**Parameters**

None.

**Return Values**

This function returns the version number of the API.

**Remarks**

None.

---

**Examples****Visual Basic Example**

```
'// get the DLL version information
```

```
MsgBox "The DLL version is: " & icsneoGetDLLVersion, vbInformation
```

### C/C++ Example

```
char szOut[200];  
  
// get the DLL version and report it  
wsprintf(szOut,TEXT("intrepidcs API DLL Version %d\n"), icsneoGetDLLVersion());  
MessageBox(hWnd,szOut,TEXT("Message"),MB_ICONINFORMATION);
```

### Visual Basic .NET Example

```
'// get the DLL version information and place in Button text  
Button1.Text = "Version " + Convert.ToString(icsneoGetDLLVersion)
```

### C# Example

```
'// get the DLL version information and place in Button text  
Button1.Text = "Version " + Convert.ToString(icsNeoDll.icsneoGetDLLVersion());
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, September 17, 2008*

**StartSocketServer Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method starts the TCP/IP socket server at a specified port.**

**C/C++ Declare**

```
int __stdcall icsneoStartSockServer(int hObject, int iPort);
```

**Visual Basic Declare**

```
Public Declare Function icsneoStartSockServer Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iPort As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoStartSockServer Lib "icsneo40.dll" (ByVal hObject As Integer, ByVal iPort As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoStartSockServer(int hObject, int iPort);
```

**Parameters***hObject*

[in] Handle which specifies the driver object created by [OpenNeoDevice](#)

*iPort*

[in] specifies the TCP/IP Port where the server will be established.

**Return Values**

If the server was started successfully the return value will be non-zero.

**Remarks**

This method creates a TCP/IP server in the DLL. This server can be attached to by any TCP/IP clients using the [RAW API](#) or using the DLL by specifying TCP/IP with [OpenNeoDevice](#). Only one server is allowed at a time.

## Examples

### Visual Basic Example

```
Call icsneoStartSockServer(m_hObject, iPort) '// start the socket server
```

### C/C++ Example

```
icsneoStartSockServer(hObject, iPort);
```

### Visual Basic .NET Example

```
bStatus = icsneoStartSockServer(m_hObject, iPort) '// start the socket server
```

### C# Example

```
iStatus = icsNeoDll.icsneoStartSockServer(m_hObject, Convert.ToInt32(txtServerPort.Text));
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, September 17, 2008*

**StopSocketServer Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method stops the TCP/IP socket server.**

**C/C++ Declare**

```
int __stdcall icsneoStopSockServer(int hObject);
```

**Visual Basic Declare**

```
Public Declare Function icsneoStopSockServer Lib "icsneo40.dll" (ByVal hObject As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoStopSockServer Lib "icsneo40.dll" (ByVal hObject As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoStopSockServer(int hObject);
```

**Parameters***hObject*

[in] Handle to the driver object created by [OpenNeoDevice](#).

**Return Values**

If the server has been stopped successfully the return value will be 1. If the function fails the return value will be zero.

**Remarks**

This method should be called when the server created with [StartSocketServer](#).

---

**Examples****Visual Basic Example**

---

```
Call icsneoStopSockServer(m hObject) '// stop the socket server
```

**C/C++ Example**

```
icsneoStopSockServer(hObject);
```

**Visual Basic .NET Example**

```
bStatus = icsneoStopSockServer(m_hObject) '// stop the socket server
```

**C# Example**

```
iStatus = icsNeoDll.icsneoStopSockServer(m_hObject); // stop the socket server
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, September 17, 2008*

**GetPerformanceParameters Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This method returns values indicating the performance of the DLL and/or device.**

**C/C++ Declare**

```
int __stdcall icsneoGetPerformanceParameters(int hObject, int * iBufferCount, int * iBufferMax, int *  
iOverFlowCount, int * iReserved1, int * iReserved2, int * iReserved3, int * iReserved4, int * iReserved5)
```

**Visual Basic Declare**

```
Private Declare Function icsneoGetPerformanceParameters Lib "icsneo40.dll" (ByVal hObject As Long, ByRef  
iBufferCount As Long, ByRef iBufferMax As Long, ByRef iOverFlowCount As Long, ByRef iReserved1 As Long, ByRef  
iReserved2 As Long, ByRef iReserved3 As Long, ByRef iReserved4 As Long, ByRef iReserved5 As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoGetPerformanceParameters Lib "icsneo40.dll" (ByVal hObject As Integer, ByRef  
iBufferCount As Integer, ByRef iBufferMax As Integer, ByRef iOverFlowCount As Integer, ByRef iReserved1 As Integer,  
ByRef iReserved2 As Integer, ByRef iReserved3 As Integer, ByRef iReserved4 As Integer, ByRef iReserved5 As Integer)  
As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoGetPerformanceParameters(int hObject, ref int iBufferCount, ref int iBufferMax, ref  
int iOverFlowCount, ref int iReserved1, ref int iReserved2, ref int iReserved3, ref int iReserved4, ref int  
iReserved5);
```

**Parameters*****hObject***

[in] Specifies the driver object created with the [OpenPort](#) method.

***iBufferCount***

[out] Specifies the driver object created with the [OpenPort](#) method.

***iBufferMax***

[out] Specifies the size of the buffer.

***iOverFlowCount***

[out] Indicates the the number of overflows that have occurred since the last successful [OpenPort](#) method was called.

***iReserved1***

[out] Reserved. Set to 0.

***iReserved2***

[out] Reserved. Set to 0.

***iReserved3***

[out] Reserved. Set to 0.

***iReserved4***

[out] Reserved. Set to 0.

***iReserved5***

[out] Reserved. Set to 0.

**Return Values**

This function returns the 1 when successful. 0 if otherwise.

**Remarks**

XX.

**Examples****Visual Basic Example**

```
lResult = icsneoGetPerformanceParameters(m_hObject, iBufferCount, iBufferMax, iOverFlowCount, iReserved1,  
iReserved2, iReserved3, iReserved4, iReserved5)  
  
If CBool(lResult) Then  
    txtBufferCount.Text = iBufferCount  
    txtBufferMax.Text = iBufferMax  
    txtOverFlow.Text = iOverFlowCount  
End If
```

**C/C++ Example**

```
icsneoGetPerformanceParameters(m_hObject, &iBufferCount, &iBufferMax, &iOverFlowCount, &iReserved1, &iReserved2,  
&iReserved3, &iReserved4, &iReserved5);
```

**Visual Basic .NET Example**

```
lResult = icsneoGetPerformanceParameters(m_hObject, iBufferCount, iBufferMax, iOverFlowCount, iReserved1,  
iReserved2, iReserved3, iReserved4, iReserved5)  
  
If CBool(lResult) Then  
    txtBufferCount.Text = iBufferCount  
    txtBufferMax.Text = iBufferMax  
    txtOverflowCount.Text = iOverFlowCount  
End If
```

**C# Example**

```
lResult = icsNeoDll.icsneoGetPerformanceParameters(m_hObject, ref iBufferCount, ref iBufferMax, ref iOverFlowCount,  
ref iReserved1, ref iReserved2, ref iReserved3, ref iReserved4, ref iReserved5);  
  
if(lResult=1)  
{  
    txtBufferCount.Text = Convert.ToString(iBufferCount);  
    txtBufferMax.Text = Convert.ToString(iBufferMax);  
    txtOverflowCount.Text = Convert.ToString(iOverFlowCount);  
}
```

## CoreMini Script interface Overview - intrepidcs API

The CoreMini Script Interface is a way to use functionality from Vehicle Spy in your own application. When a CoreMini file is created in Vehicle Spy 3 it creates header files for the functions in the file and giving access to basic parameters in the script.

This section includes commands to load and interact with CoreMini scripts for Intrepid Control Systems 3G hardware. CoreMini scripts are created using Vehicle Spy 3. For information on creating CoreMini scripts see the help documents for Vehicle Spy 3. The table below lists the different script interface commands and gives a short description of the command.

Function	Description
<a href="#">ScriptStart</a>	Starts execution of a script that has been downloaded to a neoVI device
<a href="#">ScriptStop</a>	Stops execution of a script running on a neoVI device
<a href="#">ScriptLoad</a>	Downloads a script to a connected neoVI device into a specified location
<a href="#">ScriptClear</a>	Clears a script from a specific location on a neoVI device
<a href="#">ScriptStartFBlock</a>	Starts a function block within a script on a neoVI device
<a href="#">ScriptGetFBlockStatus</a>	Returns the run status of a function block within a script on a neoVI device
<a href="#">ScriptStopFBlock</a>	Stops the execution of a function block within a script on a neoVI device
<a href="#">ScriptGetScriptStatus</a>	Stops the execution of a function block within a script on a neoVI device
<a href="#">ScriptReadAppSignal</a>	Read an application signal from a script running on a neoVI device
<a href="#">ScriptWriteAppSignal</a>	Set the value of an application signal in a script running on a neoVI device

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, June 29, 2015*

**ScriptStart Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptStart Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iLocation As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptStart Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iLocation As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptStart(Int32 hObject, Int32 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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long

'//Start CoreMini
iResult = icsneoScriptStart(m_hObject, CLng(SCRIPT_LOCATION_FLASH_MEM))

If iResult = 0 Then
    lblCMStatus.Caption = "CoreMini Failed to Start"
Else
    lblCMStatus.Caption = "CoreMini Started"
End If
```

### 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");
}
```

**C# Example:**

```
Int32 iResult;

//Start CoreMini
iResult = icsNeoDll.icsneoScriptStart(m_hObject, Convert.ToInt32(CoreMiniStoreLocation.SCRIPT_LOCATION_FLASH_MEM));

if(iResult == 0)
{
    lblCMStatus.Text = "CoreMini Failed to Start";
}
else
{
    lblCMStatus.Text = "CoreMini Started";
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32

'//Start CoreMini
iResult = icsneoScriptStart(m_hObject, SCRIPT_LOCATION_FLASH_MEM)

If iResult = 0 Then
    lblCMStatus.Text = "CoreMini Failed to Start"
Else
    lblCMStatus.Text = "CoreMini Started"
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptStop Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptStop Lib "icsneo40.dll" (ByVal hObject As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptStop Lib "icsneo40.dll" (ByVal hObject As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptStop(Int32 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

## Examples

### Visual Basic Example

```
Dim iResult As Long

'//Stop CoreMini
iResult = icsneoScriptStop(m_hObject)

If iResult = 0 Then
    lblCMStatus.Caption = "CoreMini Failed to Stop"
Else
    lblCMStatus.Caption = "CoreMini Stopped"
End If
```

### 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");
}
```

### C# Example:

```
Int32 iResult;

//Stop CoreMini
iResult = icsNeoDll.icsneoScriptStop(m_hObject);

if(iResult == 0)
{
    lblCMStatus.Text = "CoreMini Failed to Stop";
}
else
```

```
{  
    lblCMStatus.Text = "CoreMini Stopped";  
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32  
  
'//Stop CoreMini  
iResult = icsneoScriptStop(m_hObject)  
  
If iResult = 0 Then  
    lblCMStatus.Text = "CoreMini Failed to Stop"  
Else  
    lblCMStatus.Text = "CoreMini Stopped"  
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptLoad Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptLoad Lib "icsneo40.dll" (ByVal hObject As Long, ByRef bin As Byte, ByVal Len_Bytes As Long, ByVal iLocation As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptLoad Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef bin As Byte, ByVal Len_Bytes As UInt32, ByVal iLocation As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptLoad(Int32 hObject, ref byte bin, UInt32 len_bytes, Int32 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 = 4 (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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long
Dim DataLength As Long
Dim CoreMiniData() As Byte
Dim sNameOfFile As String
Dim ByteDataIn As Byte
Dim Counter As Integer

'//Open file Dialog
CommonDialog.ShowOpen()
sNameOfFile = CommonDialog.FileName
'// Check for the file to exist
```

```
If Dir(sNameOfFile) = "" Then
    lblCMStatus.Caption = "Problem reading file"
    Exit Sub
End If

Open sNameOfFile For Binary As 1
    DataLength = LOF(1)
    ReDim CoreMiniData(0 To DataLength - 1)
    For Counter = 0 To DataLength - 1
        Get 1, , ByteDataIn
        CoreMiniData(Counter) = ByteDataIn
    Next Counter
Close(1)

iResult = icsneoScriptLoad(m_hObject, CoreMiniData(0), DataLength, SCRIPT_LOCATION_FLASH_MEM)

If iResult = 0 Then
    lblCMStatus.Caption = "CoreMini Not Loaded"
Else
    lblCMStatus.Caption = "CoreMini loaded into hardware"
End If
```

### 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");
}
```

**C# Example:**

```
Int32 iResult;
UInt32 iLength=0;
byte[] CoreMiniData=new byte[1];

//Helper function
iResult = GetCoreMiniData(ref CoreMiniData,ref iLength);

if(iResult != 1)
{
    MessageBox.Show("Problem Loading CoreMini");
    return;
}

iResult = icsNeoDll.icsneoScriptLoad(m_hObject,ref CoreMiniData[0], iLength,
Convert.ToInt32(CoreMiniStoreLocation.SCRIPT_LOCATION_FLASH_MEM));

if(iResult == 0)
{
    lblCMStatus.Text = "CoreMini Not Loaded";
}
else
{
    lblCMStatus.Text = "CoreMini loaded into hardware";
}

//Helper function for reading the *.vs3cmbemini file

private Int32 GetCoreMiniData(ref byte[] DataArray,ref UInt32 DataLength)
{
string sNameOfFile;
System.IO.FileStream ReadFileStream;
System.IO.BinaryReader ReadBinaryData;

//Open file Dialog
OpenFileDialog.ShowDialog();
sNameOfFile = OpenFileDialog.FileName;
// Check for the file to exist
```

```
if (!System.IO.File.Exists(sNameOfFile)) return 0;
//Read data in
try
{
    ReadFileStream = new System.IO.FileStream(sNameOfFile, System.IO.FileMode.Open);
    ReadBinaryData = new System.IO.BinaryReader(ReadFileStream);
    DataLength = Convert.ToInt32(ReadFileStream.Length);
    DataArray = ReadBinaryData.ReadBytes(Convert.ToInt32(DataLength));
    ReadFileStream.Close();
    return 1;
}
catch(System.Exception ex)
{
    return 0;
}
}
```

### Visual Basic .NET Example:

```
Dim iResult As Int32
Dim iLength As UInt32
Dim CoreMiniData(1) As Byte

'//Helper function
iResult = GetCoreMiniData(CoreMiniData, iLength)

If iResult <> 1 Then MsgBox("Problem Loading CoreMini", MsgBoxStyle.OKOnly) : Exit Sub

Select Case iOpenDeviceType
    Case NEODEVICE_VCAN3
        iResult = icsneoScriptLoad(m_hObject, CoreMiniData(0), iLength, SCRIPT_LOCATION_VCAN3_MEM)
    Case NEODEVICE_FIRE
        iResult = icsneoScriptLoad(m_hObject, CoreMiniData(0), iLength, SCRIPT_LOCATION_FLASH_MEM)
    Case Else
        MsgBox("Hardware does not support CoreMini")
        iResult = 0
End Select

If iResult = 0 Then
    lblCMStatus.Text = "CoreMini Not Loaded"
Else
```

```
lblCMStatus.Text = "CoreMini loaded into hardware"  
End If  
  
  
'//Helper function for reading the *.vs3cmbeMini file  
Private Function GetCoreMiniData(ByRef DataArray() As Byte, ByRef DataLength As UInt32) As Int32  
Dim sNameOfFile As String  
Dim ReadFileStream As IO.FileStream  
  
'//Open file Dialog  
OpenFileDialog.ShowDialog()  
sNameOfFile = OpenFileDialog.FileName  
  
'// Check for the file to exist  
If Not IO.File.Exists(sNameOfFile) Then Return 0  
'//get the data length  
DataLength = Convert.ToInt32(FileLen(sNameOfFile))  
'//Read data in  
Try  
    ReadFileStream = New IO.FileStream(sNameOfFile, IO FileMode.Open)  
    Dim ReadBinaryData As New IO.BinaryReader(ReadFileStream)  
    DataArray = ReadBinaryData.ReadBytes(Convert.ToInt32(DataLength))  
    ReadFileStream.Close()  
Catch ex As Exception  
    ReadFileStream.Close()  
    Return 0  
End Try  
  
Return 1  
End Function
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, March 19, 2010*

**ScriptClear Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptClear Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iLocation As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptClear Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iLocation As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptClear(Int32 hObject, Int32 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 = 4  (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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long

'//clear CoreMini
iResult = icsneoScriptClear(m_hObject, SCRIPT_LOCATION_FLASH_MEM)

If iResult = 0 Then
    lblCMStatus.Caption = "CoreMini Failed to Clear"
Else
    lblCMStatus.Caption = "CoreMini Cleared"
End If
```

### 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");  
}
```

**C# Example:**

```
Int32 iResult;  
  
'//clear CoreMini  
iResult = icsNeoDll.icsneoScriptClear(m_hObject, Convert.ToInt32(CoreMiniStoreLocation.SCRIPT_LOCATION_FLASH_MEM));  
  
if(iResult == 0)  
{  
    lblCMStatus.Text = "CoreMini Failed to Clear";  
}  
else  
{  
    lblCMStatus.Text = "CoreMini Cleared";  
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32  
  
'//clear CoreMini  
iResult = icsneoScriptClear(m_hObject, SCRIPT_LOCATION_FLASH_MEM)  
  
If iResult = 0 Then  
    lblCMStatus.Text = "CoreMini Failed to Clear"  
Else  
    lblCMStatus.Text = "CoreMini Cleared"  
End If
```



**ScriptStartFBlock Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptStartFBlock Lib "icsneo40.dll" (ByVal hObject As Long, ByVal fb_index As Long)
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptStartFBlock Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal fb_index As UInt32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptStartFBlock(Int32 hObject, UInt32 fb_index);
```

**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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long

'//Start Function Block in CoreMini
iResult = icsneoScriptStartFBlock(m_hObject, cboFBToChange.ListIndex)

If iResult = 0 Then
    lblFBStatus.Caption = "Function Block failed to Start"
Else
    lblFBStatus.Caption = "Function Block Started"
End If
```

### 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");
}
```

---

### C# Example:

```
Int32 iResult;

//Start Function Block in CoreMini
iResult = icsNeoDll.icsneoScriptStartFBlock(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex));

if (iResult == 0)
{
    lblFBStatus.Text = "Function Block failed to Start";
}
else
{
    lblFBStatus.Text = "Function Block Started";
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32

'//Start Function Block in CoreMini
iResult = icsneoScriptStartFBlock(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex))

If iResult = 0 Then
    lblFBStatus.Text = "Function Block failed to Start"
Else
    lblFBStatus.Text = "Function Block Started"
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptGetFBlockStatus Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptGetFBlockStatus Lib "icsneo40.dll" (ByVal hObject As Long, ByVal fb_index As Long, ByRef piRunStatus As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptGetFBlockStatus Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal fb_index As UInt32, ByRef piRunStatus As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptGetFBlockStatus(Int32 hObject, UInt32 fb_index, ref Int32 piRunStatus);
```

**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).

---

## Examples

### Visual Basic Example

```
Dim iResult As Long
Dim iStatus As Long

'//Get Status of Function Block in CoreMini
iResult = icsneoScriptGetFBlockStatus(m_hObject, cboFBToChange.ListIndex, iStatus)

If iResult = 0 Then
    lblFBStatus.Caption = "Failed to get CoreMini Status"
Else
    Select Case iStatus
        Case SCRIPT_STATUS_RUNNING
            lblFBStatus.Caption = "CoreMini Function Block Running"
        Case SCRIPT_STATUS_STOPPED
            lblFBStatus.Caption = "CoreMini Function Block Stopped"
        Case Else
            lblFBStatus.Caption = "Unhandled State"
    End Select
End If
```

### 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");
}
```

**C# Example:**

```
Int32 iResult;
Int32 iStatus=0;

//Get Status of Function Block in CoreMini
iResult = icsNeoDll.icsneoScriptGetFBlockStatus(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex), ref
iStatus);

if(iResult == 0)
{
    lblFBStatus.Text = "Failed to get CoreMini Status";
}
else
{
    switch(iStatus)
    {
        case (int)ScriptStates.SCRIPT_STATUS_RUNNING:
            lblFBStatus.Text = "CoreMini Function Block Running";
            break;
        case (int)ScriptStates.SCRIPT_STATUS_STOPPED:
            lblFBStatus.Text = "CoreMini Function Block Stopped";
            break;
        default:
            lblFBStatus.Text = "Unhandled State";
            break;
    }
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32
Dim iStatus As Int32

'//Get Status of Function Block in CoreMini
iResult = icsneoScriptGetFBlockStatus(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex), iStatus)

If iResult = 0 Then
    lblFBStatus.Text = "Failed to get CoreMini Status"
Else
    Select Case iStatus
        Case SCRIPT_STATUS_RUNNING
            lblFBStatus.Text = "CoreMini Function Block Running"
        Case SCRIPT_STATUS_STOPPED
            lblFBStatus.Text = "CoreMini Function Block Stopped"
        Case Else
            lblFBStatus.Text = "Unhandled State"
    End Select
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptStopFBlock Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptStopFBlock Lib "icsneo40.dll" (ByVal hObject As Long, ByVal fb_index As Long)
As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptStopFBlock Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal fb_index As
UInt32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptStopFBlock(Int32 hObject, UInt32 fb_index);
```

**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).

## Examples

### Visual Basic Example

```
Dim iResult As Long

'//Stop Function Block in CoreMini
iResult = icsneoScriptStopFBlock(m_hObject, cboFBToChange.ListIndex)

If iResult = 0 Then
    lblFBStatus.Caption = "Function Block failed to Stop"
Else
    lblFBStatus.Caption = "Function Block Stopped"
End If
```

### 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");
}
```

**C# Example:**

```
Int32 iResult;

//Stop Function Block in CoreMini
iResult = icsNeoDll.icsneoScriptStopFBlock(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex));

if(iResult == 0)
{
    lblFBStatus.Text = "Function Block failed to Stop";
}
else
{
    lblFBStatus.Text = "Function Block Stopped";
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32

'//Stop Function Block in CoreMini
iResult = icsneoScriptStopFBlock(m_hObject, Convert.ToInt32(cboFBToChange.SelectedIndex))

If iResult = 0 Then
    lblFBStatus.Text = "Function Block failed to Stop"
Else
    lblFBStatus.Text = "Function Block Stopped"
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptGetScriptStatus Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptGetScriptStatus Lib "icsneo40.dll" (ByVal hObject As Long, ByRef piStatus As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptGetScriptStatus Lib "icsneo40.dll" (ByVal hObject As Int32, ByRef piStatus As Int32) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptGetScriptStatus(Int32 hObject, ref Int32 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](#).

## Examples

### Visual Basic Example

```
Dim iResult As Long
Dim iStatus As Long

'//Get CoreMini Status
iResult = icsneoScriptGetScriptStatus(m_hObject, iStatus)

If iResult = 0 Then
    lblCMStatus.Caption = "Failed to get CoreMini Status"
Else
    Select Case iStatus
        Case SCRIPT_STATUS_RUNNING
            lblCMStatus.Caption = "CoreMini Script Running"
        Case SCRIPT_STATUS_STOPPED
            lblCMStatus.Caption = "CoreMini Script Stopped"
        Case Else
            lblCMStatus.Caption = "Unhandled State"
    End Select
End If
```

### 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");
}
```

**C# Example:**

```
Int32 iResult;
Int32 iStatus=0;

//Get CoreMini Status
iResult = icsNeoDll.icsneoScriptGetScriptStatus(m_hObject, ref iStatus);

if (iResult == 0)
{
    lblCMStatus.Text = "Failed to get CoreMini Status";
}
else
{
    switch(iStatus)
    {
        case (int)ScriptStates.SCRIPT_STATUS_RUNNING:
            lblCMStatus.Text = "CoreMini Script Running";
            break;
        case (int)ScriptStates.SCRIPT_STATUS_STOPPED:
            lblCMStatus.Text = "CoreMini Script Stopped";
            break;
        default:
            lblCMStatus.Text = "Unhandled State";
            break;
    }
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32
Dim iStatus As Int32

'//Get CoreMini Status
iResult = icsneoScriptGetScriptStatus(m_hObject, iStatus)
```

```
If iResult = 0 Then
    lblCMStatus.Text = "Failed to get CoreMini Status"
Else
    Select Case iStatus
        Case SCRIPT_STATUS_RUNNING
            lblCMStatus.Text = "CoreMini Script Running"
        Case SCRIPT_STATUS_STOPPED
            lblCMStatus.Text = "CoreMini Script Stopped"
        Case Else
            lblCMStatus.Text = "Unhandled State"
    End Select
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptReadAppSignal Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptReadAppSignal Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iIndex As Long, ByRef dValue As Double) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptReadAppSignal Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iIndex As UInt32, ByRef dValue As Double) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptReadAppSignal(Int32 hObject, UInt32 iIndex, ref 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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long  
Dim ValueToSet As Double  
  
'//Read App signal  
iResult = icsneoScriptReadAppSignal(m_hObject, CLng(cboAppSig.ListIndex), ValueToSet)  
  
If iResult = 0 Then  
    txtAppSigVal.Text = "Problem!"  
Else  
    txtAppSigVal.Text = CStr(ValueToSet)  
End If
```

### 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);
}
```

**C# Example:**

```
Int32 iResult;
Double ValueToSet=0;

//Read App signal
iResult = icsNeoDll.icsneoScriptReadAppSignal(m_hObject, Convert.ToInt32(cboAppSig.SelectedIndex), ref ValueToSet);

if (iResult == 0)
{
    txtAppSigVal.Text = "Problem!";
}
else
{
    txtAppSigVal.Text = Convert.ToString(ValueToSet);
}
```

**Visual Basic .NET Example:**

```
Dim iResult As Int32
Dim ValueToSet As Double

'//Read App signal
iResult = icsneoScriptReadAppSignal(m_hObject, Convert.ToInt32(cboAppSig.SelectedIndex), ValueToSet)

If iResult = 0 Then
    txtAppSigVal.Text = "Problem!"
Else
    txtAppSigVal.Text = Convert.ToString(ValueToSet)
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

**ScriptWriteAppSignal Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [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);
```

**Visual Basic Declare**

```
Public Declare Function icsneoScriptWriteAppSignal Lib "icsneo40.dll" (ByVal hObject As Long, ByVal iIndex As Long, ByVal dValue As Double) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoScriptWriteAppSignal Lib "icsneo40.dll" (ByVal hObject As Int32, ByVal iIndex As UInt32, ByRef dValue As Double) As Int32
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern Int32 icsneoScriptWriteAppSignal (Int32 hObject, UInt32 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.

---

## Examples

### Visual Basic Example

```
Dim iResult As Long  
Dim ValueToSet As Double  
  
'//Set value to send  
ValueToSet = Val(txtAppSigVal.Text)  
  
'//Set App Signal  
iResult = icsneoScriptWriteAppSignal(m_hObject, CLng(cboAppSig.ListIndex), ValueToSet)  
  
If iResult = 0 Then  
    txtAppSigVal.Text = "Problem!"  
Else  
    txtAppSigVal.Text = "Set!"  
End If
```

### 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");
}
```

### C# Example:

```
Int32 iResult;
Double ValueToSet;

//Set value to send
ValueToSet = Convert.ToDouble(txtAppSigVal.Text);

//Set App Signal
iResult = icsNeoDll.icsneoScriptWriteAppSignal(m_hObject, Convert.ToInt32(cboAppSig.SelectedIndex), ValueToSet);

if(iResult == 0)
{
    txtAppSigVal.Text = "Problem!";
}
else
{
    txtAppSigVal.Text = "Set!";
}
```

### Visual Basic .NET Example:

```
Dim iResult As Int32
Dim ValueToSet As Double

'//Set value to send
ValueToSet = Val(txtAppSigVal.Text)

'//Set App Signal
iResult = icsneoScriptWriteAppSignal(m_hObject, Convert.ToInt32(cboAppSig.SelectedIndex), ValueToSet)

If iResult = 0 Then
    txtAppSigVal.Text = "Problem!"
Else
    txtAppSigVal.Text = "Set!"
```

End If

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

## Deprecated Functions Overview - intrepidcs API

Name	Description
<a href="#">OpenPortEx</a>	Use <a href="#">OpenNeoDevice</a> instead
<a href="#">OpenPort</a>	Use <a href="#">OpenNeoDevice</a> instead
<a href="#">EnableNetworkCom</a>	It is no longer necessary to call this before and after calling SendConfiguration
<a href="#">FindAllUSBDevices</a>	No longer supported. It is present in the API but will always return 0
<a href="#">FindAllCOMDevices</a>	Use <a href="#">FindNeoDevices</a> instead

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, September 19, 2008*

**OpenPortEx Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**This function is deprecated. Use the [OpenNeoDevice](#) method instead.**

**This function opens a connection to a neoVI device.**

**C/C++ Declare**

```
int __stdcall icsneoOpenPortEx( int lPortNumber, int lPortType, int lDriverType, int lIPAddressMSB, int  
lIPAddressLSBOrBaudRate, int bConfigRead, unsigned char * bNetworkID, int * hObject);
```

**Visual Basic Declare**

```
Public Declare Function icsneoOpenPortEx Lib "icsneo40.dll" _  
    (ByVal lPortNumber As Long, ByVal lPortType As Long, _  
     ByVal lDriverID As Long, ByVal lIPAddressMSB As Long, _  
     ByVal lIPAddressLSBOrBaudRate As Long, ByVal lForceConfigRead As Long, _  
     ByRef bNetworkID As Byte, ByRef hObject As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoOpenPortEx Lib "icsneo40.dll" (ByVal lPortNumber As Integer, ByVal lPortType As  
Integer, ByVal lDriverID As Integer, ByVal lIPAddressMSB As Integer, ByVal lIPAddressLSBOrBaudRate As Integer,  
ByVal lForceConfigRead As Integer, ByRef bNetworkID As Byte, ByRef hObject As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoOpenPortEx(int lPortNumber, int lPortType, int lDriverID, int lIPAddressMSB, int  
lIPAddressLSBOrBaudRate, int lForceConfigRead, ref byte bNetworkID, ref int hObject);
```

**Parameters****IPortNumber**

[in] Specifies which USB, TCP/IP, or Comm port to use. The USB port is a logic port starting with 1 up to the number of devices. For example, if 3 devices are connected, valid IPortNumber values will be 1,2, and 3. Comm ports also start at 1. For TCP/IP it specifies the TCP/IP port number.

**IPortType**

[in] Specifies the port type of what the device is connected to. This parameter is used with *IPortNumber* described above. This parameter can be set to NEOVI\_COMMTYPE\_RS232 (0) for a RS232 Comm port, NEOVI\_COMMTYPE\_USB\_BULK (1) for a USB, or NEOVI\_COMMTYPE\_TCPIP (3) for a TCP/IP connection. More information on the different types of hardware and the connection it uses can be found in the [OpenPortEx Hardware Type](#) help topic.

**IDriverType**

[in] Specifies which neoVI driver to use. This should always be set to INTREPIDCS\_DRIVER\_STANDARD (0).

**IPAddressMSB**

[in] Specifies upper two bytes of TCP/IP address if the IPortType is set to NEOVI\_COMMTYPE\_TCPIP (bytes 4 and 3 of this TCP/IP address: 1.2.3.4).

**IPAddressLSBOrBaudRate**

[in] Specifies the baud rate if IPortType is NEOVI\_COMMTYPE\_RS232 and the lower two bytes of the TCP/IP address (bytes 2 and 1 of this TCP/IP address: 1.2.3.4).

**bConfigRead**

[in] Specifies whether the DLL should read the configuration before enabling the device. It is recommended to set this value to 1. This parameter is ignored when the object is created because the configuration is read by default.

**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. The network IDs are specified in the following list:  
NETID\_DEVICE = 0, NETID\_HSCAN = 1, NETID\_MSCAN = 2, NETID\_SWCAN = 3, NETID\_LSFTCAN = 4, NETID\_FORDSCP = 5, NETID\_J1708 = 6, NETID\_AUX = 7, NETID\_JVPW = 8, NETID\_ISO = 9. **You may also set this parameter to NULL (zero) and the default network ID's will be used.**

The neoVI DLL will use this array when it receives a network message. For example, when the DLL receives a message from HSCAN network it will set the NetworkID parameter of the [message structure](#) with the value bNetworkID(NETID\_HSCAN).

**hObject**

[in, out] This is the handle of the neoVI driver object. If this handle is 0, the method will create a new neoVI driver object. This handle will be used in other neoVI API calls to read and transmit messages.

Every time you create a new neoVI driver object you must call [icsneoFreeObject](#) (after [ClosePort](#)). If you do not you will create a memory leak.

**Return Values**

If the port has been opened successfully and connection to neoVI is attained, the return value will be 1. If the function fails the return value will be zero. The most common reason for failure is when the neoVI is not connected to the port described in the parameters.

## Remarks

Each successful call to OpenPort should be matched with a call to [ClosePort](#). The OpenPort call will reset the timestamp clock.

---

## Examples

### Visual Basic Example

```
Dim lResult As Long '// Used to store the return value
Dim bNetworkIDs(0 To 16) As Byte '// Array of network IDs passed to the driver
Dim lCount As Long '// General Purpose Counter Variable
Dim Cntr As Long
Dim lIPMSB As Long '// MSB of IP address
Dim lIPLSB As Long '// LSB of IP address

lIPMSB = 0
lIPLSB = 0
'// Do not open if we have already done so
If m_bPortOpen Then Exit Sub

'// Initialize the network id array
For lCount = 0 To 16
    bNetworkIDs(lCount) = lCount
Next lCount

lResult = 0

lResult = icsneoOpenPortEx(Val(txtPortNumber.Text), NEOVI_COMMTYPE_USB_BULK,
                           INTREPIDCS_DRIVER_STANDARD, lIPMSB, lIPLSB, 1,      NETID_HSCAN, m_hObject)

'// test the returned result
If CBool(lResult) Then
    MsgBox("Port Opened Successfully")
    m_bPortOpen = True '// set the flag which indicates we haved opened neoVI
Else
    MsgBox("Problem Opening Port")
End If
```

## C/C++ Example

```
unsigned char bNetworkID[16];      // array of network ids
int hObject = 0;                  // holds a handle to the neoVI object
unsigned long lCount;             // counter variable
int iIPLSB;
int iIPMSB;

// initialize the networkid array
for (lCount=0;lCount<16;lCount++)
    bNetworkID[lCount] = lCount;

if (!m_bPortOpen) // only if not already opened
{
    iIPLSB = 57600;
    iIPMSB = 0;

    lResult = icsneoOpenPortEx(1
,NEOVI_COMMTYPE_RS232,INTREPIDCS_DRIVER_STANDARD,iIPMSB,iIPLSB,1,bNetworkID,&hObject);
    if (lResult == 0)
        MessageBox(hWnd,TEXT("Problem Opening Port"),TEXT("neoVI Example"),0);
    else
    {
        m_bPortOpen =true;
        MessageBox(hWnd,TEXT("Port Opened Successfully"),TEXT("neoVI Example"),0);
    }
}
```

## Visual Basic .NET Example

```
Dim lResult As Long
Dim iIPMSB As Integer
Dim iIPLSB As Integer

iIPMSB = 0
iIPLSB = 0
lResult = 0

''command To Open up the port
If optUsbDevice.Checked = False Then
    iIPLSB = 57600 ''Set Baud Rate for Serial Communication
```

```
lResult = icsneoOpenPortEx(Val(txtCommPortNum.Text), NEOVI_COMMTYPE_RS232, _  
    INTREPIDCS_DRIVER_STANDARD, iIPMSB, iiPLSB, 1, NETID_HSCAN, m_hObject)  
Else  
    lResult = icsneoOpenPortEx(Val(txtCommPortNum.Text), NEOVI_COMMTYPE_USB_BULK, _  
        INTREPIDCS_DRIVER_STANDARD, iIPMSB, iiPLSB, 1, NETID_HSCAN, m_hObject)  
End If  
  
''Check the status of the opened port  
If CBool(lResult) Then  
    MsgBox("Port Opened OK!")  
Else  
    MsgBox("Problem Opening Port")  
End If  
m_bPortOpen = True
```

## C# Example

```
byte bNetworkIDs = 1;  
int iPortNumber = 0;  
int iRetVal = 0; //iReturn value tells status of the function call  
int iIPMSB = 0;  
int iiPLSB = 0;  
  
iPortNumber = Convert.ToInt32( txtCommPortNum.Text); //Convert type of Textbox Value  
//Open the Device on the Port indicated in the Text box  
if(optUsbDevice.Checked == false)  
{  
    iiPLSB = 57600;  
    iRetVal = icsNeoDll.icsneoOpenPortEx(iPortNumber, Convert.ToInt32(ePORT_TYPE.NEOVI_COMMTYPE_RS232),  
        Convert.ToInt32(eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDARD),iIPMSB ,iiPLSB,1, ref bNetworkIDs, ref  
m_hObject);  
}  
else  
{  
    iRetVal = icsNeoDll.icsneoOpenPortEx(iPortNumber, Convert.ToInt32(ePORT_TYPE.NEOVI_COMMTYPE_USB_BULK ),  
        Convert.ToInt32(eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDARD),iIPMSB , iiPLSB, 1, ref bNetworkIDs,ref  
m_hObject);  
}  
  
if (iRetVal == 0) // test the returned result  
{
```

```
    MessageBox.Show("Problem Opening Port"); //Error, Show message
}
else
{
    MessageBox.Show("Port opened OK!");
    m_bPortOpen = true; //Set Port Opened Flag
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2008*

## OpenPortEx Hardware Type Information - intrepidcs API

**This topic discusses which port type can be used with what type of hardware.**

A required parameter for the OpenPortEx command is the Port type. Each type is and the hardware it supports is listed below.

Port type type	Applicable hardware	Notes
RS232	<ul style="list-style-type: none"><li>• neoVI Blue (USB or RS232 connection)</li><li>• neoVI Pro (USB or RS232 connection)</li><li>• neoVI Green (RS232)</li><li>• ValueCAN</li></ul>	Devices that use a USB connection listed here have a virtual COM port connection to the PC even though it is connected to the USB port on the PC.
USB	<ul style="list-style-type: none"><li>• neoVI Green</li></ul>	These devices use the USB Port type
TCP/IP	<ul style="list-style-type: none"><li>• neoVI Blue</li><li>• neoVI Pro</li><li>• neoVI Green</li><li>• ValueCAN</li></ul>	All ICS devices can be connected via a TCP/IP connection using a second PC as a gateway.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, April 19, 2007*

**FindAllCOMDevices Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

***This function is deprecated. Use [FindNeoDevices](#) instead.***

**This method returns the number of COM (both serial and USB serial) hardware devices connected to the PC.**

**C/C++ Declare**

```
int __stdcall icsneoFindAllCOMDevices(int lDriverType,  
int lGetSerialNumbers,  
int lStopAtFirst,  
int lUSBCommOnly,  
int *p_lDeviceTypes,  
int *p_lComPorts,  
int *p_lSerialNumbers,  
int *lNumDevices);
```

**Visual Basic Declare**

```
Public Declare Function icsneoFindAllCOMDevices Lib "icsneo40.dll" (ByVal lDriverID As Long,  
ByVal lGetSerialNumbers As Long, ByVal lStopAtFirst As Long,  
ByVal iUSBCommOnly As Long, ByRef p_lDeviceTypes As Long,  
ByRef p_lComPorts As Long, ByRef p_lSerialNumbers As Long, ByRef lNumDevices As Long) As  
Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoFindAllCOMDevices Lib "icsneo40.dll" _  
(ByVal lDriverID As Integer, ByVal lGetSerialNumbers As Integer, _  
ByVal lStopAtFirst As Integer, ByVal iUSBCommOnly As Integer, _  
ByRef p_lDeviceTypes As Integer, ByRef p_lComPorts As Integer, _  
ByRef p_lSerialNumbers As Integer, ByRef lNumDevices As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]  
public static extern int icsneoFindAllCOMDevices(int lDriverID, int lGetSerialNumbers, int lStopAtFirst, int  
iUSBCommOnly, ref int p_lDeviceTypes, ref int p_lComPorts, ref int p_lSerialNumber, ref int lNumDevices);
```

## Parameters

### *IDriverType*

[in] Specifies which neoVI driver to use. This should always be set to INTREPIDCS\_DRIVER\_STANDARD (0).

### *iGetSerialNumbers*

[in] Specifies whether the serial numbers should be read from the device (*iGetSerialNumbers*=1). Getting serial numbers will take longer than not doing it therefore so set this to zero if not required for the application. If a device is already opened the serial number cannot be read.

### *iStopAtFirst*

[in] Indicates whether the function should stop at the first device found (*iStopAtFirst*=1). This is useful when you only have one device connected to the PC.

### *IUSBCommOnly*

[in] Indicates to search USB serial devices only (*IUSBCommOnly*=1). Normal COM ports will not be searched. Normal COM port searches will take longer to execute.

### *p\_IDeviceTypes*

[out] Pointer to array of at least 255 elements. This array will be filled in with the type of device found. The valid device types include INTREPIDCS\_DEVICE\_NE04 (0), INTREPIDCS\_DEVICE\_VCAN (1), or INTREPIDCS\_DEVICE\_NE06 (2).

### *p\_IComPorts*

[out] Pointer to array of at least 255 elements. This array will be filled in with the com port numbers of each connected device.

### *p\_ISerialNumbers*

[out] Pointer to array of at least 255 elements. This array will be filled in with the serial number of each device if argument *iGetSerialNumbers*=1.

### *iNumDevices*

[out] Points to a value which contains the number of devices found.

---

## Return Values

If this function operates successfully the return value will be 1. If the function fails the return value will be zero.

---

## Remarks

None.

## Examples

### Visual Basic Example

**This example demonstrates loading a VB list box with all the devices connected to the PC.**

```
Dim lResult As Long
Dim lDeviceTypes(0 To 255) As Long
Dim lComPorts(0 To 255) As Long
Dim lSerialNumbers(0 To 255) As Long
Dim lNumDevices As Long
Dim lCount As Long

'// this make take a while
MousePointer = vbHourglass

'// read all com ports and get serial numbers
lResult = icsneoFindAllCOMDevices( INTREPIDCS_DRIVER_STANDARD, 1, 0, 0, _
    lDeviceTypes(0), lComPorts(0), lSerialNumbers(0), lNumDevices)

If CBool(lResult) Then
    If lNumDevices = 0 Then
        MsgBox "There are no devices connected on COM ports.", vbInformation
    Else
        '// reset the the list box
        lstCOMDevices.Clear
        For lCount = 0 To lNumDevices - 1
            '// add each item
            Select Case lDeviceTypes(lCount)
                Case INTREPIDCS_DEVICE_NE04
                    lstCOMDevices.AddItem "neoVI 4 SN: " & lSerialNumbers(lCount) & " on COM" & lComPorts(lCount)
                Case INTREPIDCS_DEVICE_NE06
                    lstCOMDevices.AddItem "neoVI PRO SN: " & lSerialNumbers(lCount) & " on COM" & lComPorts(lCount)
                Case INTREPIDCS_DEVICE_VCAN
                    lstCOMDevices.AddItem "ValueCAN SN: " & lSerialNumbers(lCount) & " on COM" & lComPorts(lCount)
            End Select
            '// store the index for the open port function
            lstCOMDevices.ItemData(lstCOMDevices.NewIndex) = lComPorts(lCount)
```

```
    Next lCount

    lstCOMDevices.ListIndex = 0
End If
Else
    MsgBox "There was a problem getting the COM devices.", vbInformation
End If

MousePointer = vbDefault
```

**C/C++ Example:****Opens first device on USB Com port either a ValueCAN or neoVI PRO**

```
int iDeviceTypes[255];
int iComPort[255];
int iSerialNum[255];
int iOpenedStates[255];
int iDeviceNumbers[255];
int iNumDevices;

if (icsneoFindAllCOMDevices(INTREPIDCS_DRIVER_STANDARD, 0,1,1,iDeviceTypes,iComPort,iSerialNum,&iNumDevices))
{

    if (iNumDevices > 0)
    {

        lResult = icsneoOpenPortEx(iComPort[0] ,NEOVI_COMMTYPE_RS232,
                                INTREPIDCS_DRIVER_STANDARD,0,57600,1,bNetworkID, &hObject);

        if (lResult == 0)
            MessageBox(hWnd,TEXT("Problem Opening Port"),TEXT("neoVI Example"),0);
        else
        {

            MessageBox(hWnd,TEXT("Port Opened Successfully"),TEXT("neoVI Example"),0);
        }
    }
else
```

```
    MessageBox(hWnd,TEXT("Problem Opening Port"),TEXT("neoVI Example"),0);  
}  
else  
  
    MessageBox(hWnd,TEXT("Problem Opening Port"),TEXT("neoVI Example"),0);
```

**C# Example:****This example demonstrates loading a list box with all the devices connected to the PC**

```
int lResult = 0; //Storage for Result of Function call  
int[] iDevices = new int[127]; //Array for the device numbers  
int[] iSerialNumbers = new int[127]; //Araay for serial numbers of attached devices  
int[] iOpenedStatus = new int[127]; //Array of the status of the driver  
int iNumDevices = 0; //Storage for the number of devices  
int[] iCommPortNumbers = new int[127]; //Array of Comm Port numbers in use  
int Counter = 0; //Counter for Counting things  
  
//Call function for Finding all Comm deivces  
lResult = icsNeoDll.icsneoFindAllCOMDevices(Convert.ToInt32 (eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDARD), 1,0,0,ref  
iDevices[0],ref iCommPortNumbers[0], ref iSerialNumbers[0],ref  
iNumDevices)  
  
//Check the status of the funciton call  
if(lResult==1)  
{  
    //Fill in list box with device findings  
    for(Counter=0;Counter<127; Counter++)  
    {  
        lstCommDevices.Items.Add("Device Type-" + Convert.ToString(iDevices[Counter]) + " SN-" +  
Convert.ToString(iSerialNumbers[Counter]) + " Port #" + Convert.ToString(iCommPortNumbers[Counter]));  
    }  
}  
else  
{  
    //display error box if could not find anything  
    MessageBox.Show("Could Not Find anything");  
}
```

**Visual Basic .NET Example:****This example demonstrates loading a list box with all the devices connected to the PC**

```
Dim lResult As Integer ''Storage for Result of Function call
Dim iDevices(127) As Integer ''Array for the device numbers
Dim iSerialNumbers(127) As Integer ''Array for Serial numbers of attached devices
Dim iOpenedStatus(127) As Integer ''Array of Status of Driver
Dim iNumDevices As Integer ''Storage for the number of devices
Dim iCommPortNumbers(127) As Integer ''Array of Comm port numbers in use
Dim Counter As Integer ''Counter for counting things

''Function call for Finding all of the Comm devices
lResult = icsneoFindAllCOMDevices(INTREPIDCS_DRIVER_STANDARD, 1, 0, 0, iDevices(0), iCommPortNumbers(0),
iSerialNumbers(0), iNumDevices)
''check the results
If lResult = 1 Then
    For Counter = 0 To 127
        ''Fill list box with device findings
        lstCommDevices.Items.Add("Device Type-" + Convert.ToString(iDevices(Counter)) + _
            " SN-" + Convert.ToString(iSerialNumbers(Counter)) + " Port #" +
            Convert.ToString(iCommPortNumbers(Counter)))
    Next Counter
Else
    ''Display error message if could not find anything
    MsgBox("Could not find anything")
End If
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Friday, September 19, 2008*

---

## **FindAllUSBDevices Method - intrepidcs API**

**This method is no longer supported and will always return 0. It remains in the API for backward compatibility.**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, September 19, 2008*

**EnableNetworkCom Method - intrepidcs API**

[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Parameters](#) - [Return Values](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

***This function is deprecated. Use the [EnableNetworkRXQueue](#) method instead.***

**This method enables or disables all vehicle network rx data.**

**C/C++ Declare**

```
int __stdcall icsneoEnableNetworkCom(int hObject, int lEnable);
```

**Visual Basic Declare**

```
Public Declare Function icsneoEnableNetworkCom Lib "icsneo40.dll" (ByVal hObject As Long, ByVal lEnable As Long) As Long
```

**Visual Basic .NET Declare**

```
Public Declare Function icsneoEnableNetworkCom Lib "icsneo40.dll" (ByVal hObject As Integer, ByVal lEnable As Integer) As Integer
```

**C# Declare**

```
[DllImport("icsneo40.dll")]
public static extern int icsneoEnableNetworkCom(int hObject, int lEnable);
```

**Parameters*****lEnable***

[in] When 1 it will enable network receive. When 0 it will disable network receive.

**Return Values**

This function returns the 1 when successful. 0 if otherwise.

**Remarks**

This function will enable and disable network traffic for all client applications connected to the neoVI. The icsneoEnableNetworkRXQueue function can be used to enable and disable the receive message queue for individual applications.

## Examples

### Visual Basic Example

```
'// disable network communications  
Call icsneoEnableNetworkCom(m_hObject, 0)
```

### C/C++ Example

```
icsneoEnableNetworkCom(m_hObject,0);
```

### Visual Basic .NET Example

```
Call icsneoEnableNetworkCom(m_hObject, 0)
```

### C# Example

```
icsNeoDll.icsneoEnableNetworkCom(m_hObject,0);
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, September 02, 2008 Monday, January 24, 2005*

## Structures, Types, and Defines Overview - intrepidcs API

Item	Description
<a href="#">MessageStructures</a>	Message structures used for reading and writing vehicle network messages
<a href="#">StatusBitfields</a>	Status bitfields in the <a href="#">Message Structures</a> that define specific attributes of the message
<a href="#">IntrepidAPIDataTypes</a>	Data types defined or re-defined by the intrepidcs API
<a href="#">SFireSettingsStructure</a>	Settings for the neoVI Fire device. Used by the <a href="#">GetFireSettings</a> and <a href="#">SetFireSettings</a> methods
<a href="#">SVCAN3SettingsStructure</a>	Settings for the ValueCAN3 device. Used by the <a href="#">GetVCAN3Settings</a> and <a href="#">SetVCAN3Settings</a> methods
<a href="#">ConfigArray</a>	Use to read and change device settings on a neoVI Blue or ValueCAN with the <a href="#">GetConfiguration</a> and <a href="#">SendConfiguration</a> methods
<a href="#">CAN_SETTINGSStructure</a>	Settings for CAN networks on neoVI Fire and ValueCAN3 devices
<a href="#">SWCAN_SETTINGSStructure</a>	Settings for SWCAN networks on neoVI Fire devices
<a href="#">LIN_SETTINGSStructure</a>	Settings for LIN networks on neoVI Fire devices
<a href="#">ISO9141_KW2000SETTINGSStructure</a>	Settings for ISO9141 and Keyword 2000 networks on neoVI Fire devices
<a href="#">ISO9141_KW2000_INIT_STEPStructure</a>	Settings for the ISO9141 and Keyword 2000 initialization step on neoVI Fire devices
<a href="#">NetworkIDList</a>	Network ID list for the neoVI API
<a href="#">NeoDevice</a>	Structure used by <a href="#">FindNeoDevices</a> and <a href="#">OpenNeoDevice</a> to locate and open neoVI devices

<a href="#">stCM_ISO157652_TxMessageStructure</a>	structure used by <a href="#">ScriptWriteISO15765_2_TxMessage</a> and <a href="#">ScriptReadISO15765_2_TxMessage</a>
<a href="#">stAPIFirmwareInfoStructure</a>	structure that contains information about neoVI device firmware versions
<a href="#">icsSpyTimeStructure</a>	contains real-time clock data. Used by the <a href="#">GetRTC</a> and <a href="#">SetRTC</a> functions

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, August 05, 2009*

**Message Structures - neoVI API**[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**These structures are used to represent messages both received and transmitted by the neoVI device. These structures can also be represented as an [array of bytes described in a separate topic](#).**

**C/C++ Declare**

```
typedef struct // matching C structure
{
    unsigned long StatusBitField;
    unsigned long StatusBitField2;
    unsigned long TimeHardware;
    unsigned long TimeHardware2;
    unsigned long TimeSystem;
    unsigned long TimeSystem2;
    unsigned charTimeStampHardwareID;
    unsigned charTimeStampSystemID;
    unsigned char NetworkID;
    unsigned char NodeID;
    unsigned char Protocol;
    unsigned char MessagePieceID;
    unsigned char ColorID;
    unsigned char NumberBytesHeader;
    unsigned char NumberBytesData;
    short DescriptionID;
    long ArbIDOrHeader;
    unsigned char Data[8];
    unsigned char AckBytes[8];
    float Value;
    unsigned char MiscData;
} icsSpyMessage;

typedef struct // matching C structure
{
    unsigned long StatusBitField;
    unsigned long StatusBitField2;
    unsigned long TimeHardware;
    unsigned long TimeHardware2;
    unsigned long TimeSystem;
    unsigned long TimeSystem2;
    unsigned charTimeStampHardwareID;
```

```
unsigned char TimeStampSystemID;
unsigned char NetworkID;
unsigned char NodeID;
unsigned char Protocol;
unsigned char MessagePieceID;
unsigned char ColorID;
unsigned char NumberBytesHeader;
unsigned char NumberBytesData;
short DescriptionID;
unsigned char Header[4];
unsigned char Data[8];
unsigned char AckBytes[8];
float Value;
unsigned char MiscData;
} icsSpyMessageJ1850;
```

## Visual Basic Declares

```
Public Type icsSpyMessage
    StatusBitField As Long
    StatusBitField2 As Long
    TimeHardware As Long
    TimeHardware2 As Long
    TimeSystem As Long
    TimeSystem2 As Long
    TimeStampHardwareID As Byte
    TimeStampSystemID As Byte
    NetworkID As Byte
    NodeID As Byte
    Protocol As Byte
    MessagePieceID As Byte
    ColorID As Byte
    NumberBytesHeader As Byte
    NumberBytesData As Byte
    DescriptionID As Integer
    ArbIDOrHeader As Long
    Data(1 To 8) As Byte
    AckBytes(1 To 8) As Byte
    Value As Single
    MiscData As Byte
End Type
```

```
Public Type icsSpyMessageJ1850
    StatusBitField As Long
    StatusBitField2 As Long
    TimeHardware As Long
    TimeHardware2 As Long
    TimeSystem As Long
    TimeSystem2 As Long
    TimeStampHardwareID As Byte
    TimeStampSystemID As Byte
    NetworkID As Byte
    NodeID As Byte
    Protocol As Byte
    MessagePieceID As Byte
    ColorID As Byte
    NumberBytesHeader As Byte
    NumberBytesData As Byte
    DescriptionID As Integer
    Header(1 To 4) As Byte
    Data(1 To 8) As Byte
    AckBytes(1 To 8) As Byte
    Value As Single
    MiscData As Byte
End Type
```

## Visual Basic .NET Declares

```
Public Structure icsSpyMessage
    Dim StatusBitField As Integer
    Dim StatusBitField2 As Integer
    Dim TimeHardware As Integer
    Dim TimeHardware2 As Integer
    Dim TimeSystem As Integer
    Dim TimeSystem2 As Integer
    Dim TimeStampHardwareID As Byte
    Dim TimeStampSystemID As Byte
    Dim NetworkID As Byte
    Dim NodeID As Byte
    Dim Protocol As Byte
    Dim MessagePieceID As Byte
    Dim ColorID As Byte
    Dim NumberBytesHeader As Byte
```

```
Dim NumberBytesData As Byte
Dim DescriptionID As Short
Dim ArBIDOrHeader As Integer
Dim Data1 As Byte
Dim Data2 As Byte
Dim Data3 As Byte
Dim Data4 As Byte
Dim Data5 As Byte
Dim Data6 As Byte
Dim Data7 As Byte
Dim Data8 As Byte
Dim AckBytes1 As Byte
Dim AckBytes2 As Byte
Dim AckBytes3 As Byte
Dim AckBytes4 As Byte
Dim AckBytes5 As Byte
Dim AckBytes6 As Byte
Dim AckBytes7 As Byte
Dim AckBytes8 As Byte
Dim Value As Single
Dim MiscData As Byte
End Structure

Public Structure icsSpyMessageJ1850
    Dim StatusBitField As Integer
    Dim StatusBitField2 As Integer
    Dim TimeHardware As Integer
    Dim TimeHardware2 As Integer
    Dim TimeSystem As Integer
    Dim TimeSystem2 As Integer
    DimTimeStampHardwareID As Byte
    DimTimeStampSystemID As Byte
    Dim NetworkID As Byte
    Dim NodeID As Byte
    Dim Protocol As Byte
    Dim MessagePieceID As Byte
    Dim ColorID As Byte
    Dim NumberBytesHeader As Byte
    Dim NumberBytesData As Byte
    Dim DescriptionID As Short
    Dim Header1 As Byte
    Dim Header2 As Byte

```

```
Dim Header3 As Byte
Dim Header4 As Byte
Dim Data1 As Byte
Dim Data2 As Byte
Dim Data3 As Byte
Dim Data4 As Byte
Dim Data5 As Byte
Dim Data6 As Byte
Dim Data7 As Byte
Dim Data8 As Byte
Dim AckBytes1 As Byte
Dim AckBytes2 As Byte
Dim AckBytes3 As Byte
Dim AckBytes4 As Byte
Dim AckBytes5 As Byte
Dim AckBytes6 As Byte
Dim AckBytes7 As Byte
Dim AckBytes8 As Byte
Dim Value As Single
Dim MiscData As Byte
End Structure
```

## C# Declares

```
[StructLayout(LayoutKind.Sequential)]
public struct icsSpyMessage
{
    public int StatusBitField;
    public int StatusBitField2;
    public int TimeHardware;
    public int TimeHardware2;
    public int TimeSystem;
    public int TimeSystem2;
    public byte TimeStampHardwareID;
    public byte TimeStampSystemID;
    public byte NetworkID;
    public byte NodeID;
    public byte Protocol;
    public byte MessagePieceID;
    public byte ColorID;
    public byte NumberBytesHeader;
```

```
    public byte NumberBytesData;
    public short DescriptionID;
    public int ArbIDOrHeader;
    public byte Data1;
    public byte Data2;
    public byte Data3;
    public byte Data4;
    public byte Data5;
    public byte Data6;
    public byte Data7;
    public byte Data8;
    public byte AckBytes1;
    public byte AckBytes2;
    public byte AckBytes3;
    public byte AckBytes4;
    public byte AckBytes5;
    public byte AckBytes6;
    public byte AckBytes7;
    public byte AckBytes8;
    public Single Value;
    public byte MiscData;
}
[StructLayout(LayoutKind.Sequential)]
public struct icssSpyMessageJ1850
{
    public int StatusBitField;
    public int StatusBitField2;
    public int TimeHardware;
    public int TimeHardware2;
    public int TimeSystem;
    public int TimeSystem2;
    public byte TimeStampHardwareID;
    public byte TimeStampSystemID;
    public byte NetworkID;
    public byte NodeID;
    public byte Protocol;
    public byte MessagePieceID;
    public byte ColorID;
    public byte NumberBytesHeader;
    public byte NumberBytesData;
    public short DescriptionID;
    public byte Header1;
```

```

public byte Header2;
public byte Header3;
public byte Header4;
public byte Data1;
public byte Data2;
public byte Data3;
public byte Data4;
public byte Data5;
public byte Data6;
public byte Data7;
public byte Data8;
public byte AckBytes1;
public byte AckBytes2;
public byte AckBytes3;
public byte AckBytes4;
public byte AckBytes5;
public byte AckBytes6;
public byte AckBytes7;
public byte AckBytes8;
public Single Value;
public byte MiscData;
}

```

## Remarks

There are two structures here. Both are equivalent. The only difference is how they represent message bytes. The icsspyMessageJ1850 provides a more convenient representation for J1850 or ISO messages with a header array holding the first three bytes of the message.

These structures can be used interchangeably in C by casting one type to the other. In Visual Basic, you can copy one structure to the other using the LSet method.

Table 1 below lists the members of the structure and specific remarks about their use.

**Table 1 - Message Structure Elements**

Item	Description
StatusBitField StatusBitField2	Bitfields which describe the message. These are described in a <a href="#">separate topic</a> .
	This is the hardware time stamp. The TimeStamp is reset to zero every time the <a href="#">OpenPort</a> method is called.

TimeHardware TimeHardware2	<p>For the <b>neoVI</b> hardware, TimeHardware2 is more significant than TimeHardware. The resolution of TimeHardware is 1.6<math>\mu</math>s and and TimeHardware2 is 104.8576 ms. To calculate the time of the message in seconds use the following formula: "Timestamp (sec) = TimeHardware2* 0.1048576 + TimeHardware * 0.0000016".</p> <p>For the <b>neoVI PRO or ValueCAN</b> hardware, TimeHardware2 is more significant than TimeHardware. The resolution of TimeHardware is 1.0<math>\mu</math>s and and TimeHardware2 is 65.536 ms. To calculate the time of the message in seconds use the following formula: "Timestamp (sec) = TimeHardware2* 0.065536 + TimeHardware * 0.000001".</p>
TimeSystem TimeSystem2	This is the system time stamp. TimeSystem is loaded with the value received from the timeGetTime call in the WIN32 multimedia API. The timeGetTime accuracy is up to 1 millisecond. See the WIN32 API documentation for more information. This timestamp is useful for time comparing with other system events or data which is not synced with the neoVI timestamp. Currently, TimeSystem2 is not used.
TimeStampHardwareID	This is an identifier of what type of hardware timestamp is used. Since neoVI's timestamp is always the same, this doesn't change.
TimeStampSystemID	This is an identifier of what type of system timestamp is used. Since WIN32 neoVI's timestamp is always the same, from the timeGetTime API, this doesn't change.
NetworkID	This is the NetworkID as assigned in the <a href="#">OpenPort</a> method. This value is used to identify which network this message was received on.
NodeID	Not Used in the neoVI API.
Protocol	This is the type of protocol which the message belongs to. Valid values are SPY_PROTOCOL_CAN, SPY_PROTOCOL_J1850VPW, and SPY_PROTOCOL_ISO9141.
MessagePieceID	Not Used in the neoVI API.
ColorID	Not Used in the neoVI API.
NumberBytesHeader	Used for J1850/ISO messages. It indicates how many bytes are

Input/Output	Description
NumberBytesData	stored in the Header(1 to 4) array.
DescriptionID	Holds the number of bytes in the Data(1 to 8) array or the number of bytes in a CAN remote frame (The DLC).
Header(1 To 4) or ArbIDOrHeader	Not Used in the neoVI API.
Data(1 To 8)	Holds up to 3 byte 1850 header (bytes 1 through 3) or a 29 bit CAN header.
AckBytes(1 To 8)	Holds the 8 data bytes in CAN messages or bytes 4 through 11 in J1850/ISO messages.
Value	Not Used in the neoVI API.
MiscData	Not Used in the neoVI API.

## Examples

### Visual Basic Examples

#### Interchangeably using the structures : Copying a icsSpyMessage to an icsSpyMessageJ1850

```
Dim stMessagesTx As icsSpyMessage
Dim stJMsg As icsSpyMessageJ1850

'// copy the J1850 message structure into the structure that will be transmitted
LSet stMessagesTx = stJMsg
```

#### Timestamps : Calculating a TimeStamp

```
Dim dTime As Double

'// Determine the timestamp
dTime = NEOVI_TIMEHARDWARE2_SCALING * stMessage.TimeHardware2 + NEOVI_TIMEHARDWARE_SCALING *
stMessage.TimeHardware
```

#### C/C++ Example

**Interchangeably using the structures : Casting a icsSpyMessage to an icsSpyMessageJ1850**

```
((icsSpyMessageJ1850 *) stMessages)[lCount].Header[0]
```

**Timestamps : Calculating a TimeStamp**

```
// Calculate the time for this message  
dTime = ((double) stMessages[lCount].TimeHardware2) * NEOVI_TIMESTAMP_2 +  
        ((double) stMessages[lCount].TimeHardware) * NEOVI_TIMESTAMP_1;
```

**C# Example****Timestamps : Calculating a TimeStamp**

```
double dTime; //Storage for message time  
  
dTime = icsNeoDll.icsneoGetTimeStamp(stMessages[lCount-1].TimeHardware, stMessages[lCount-1].TimeHardware2);
```

**Visual Basic .NET Example****Timestamps : Calculating a TimeStamp**

```
Dim dTime As Double  
  
dTime = icsneoGetTimeStamp(stMessages(lCount - 1).TimeHardware, stMessages(lCount - 1).TimeHardware2)
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, January 06, 2012*

## Using an array instead of a message structure - intrepidcs API

In some programming environments (such as [LabVIEW](#)) it maybe inconvenient for you to access the [message with a structure](#). In these cases you may use a 64 byte array in place of the structure. Table 1 below lists the locations of message items in the byte array.

With LabVIEW, items that are 4 bytes long can be read with neoGetLong.vi and set with neoPutLong.vi SubVIs.

**Table 1 - Position of Message Elements in the Byte Array**

Item	Bytes	Byte (s) Location
StatusBitField	4	0-3
StatusBitField2	4	4-7
TimeHardware	4	8-11
TimeHardware2	4	12-15
TimeSystem	4	16-19
TimeSystem2	4	20-23
TimeStampHardwareID	1	24
TimeStampSystemID	1	25
NetworkID	1	26
Protocol	1	28
NumberBytesHeader	1	31
NumberBytesData	1	32
Header(1 To 4)	Array Length 4	36-39
ArbIDOrHeader	4	36-39
Data(1 To 8)	Array Length 8	40-47



**Status Bitfields - neoVI API**[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#) - [VB.NET example](#) - [C# example](#)

**There are two status bitfields in the message structures that define specific attributes of the message. The two status bitfields are named StatusBitfield and StatusBitfield1.**

**C/C++ Declare**

```
const long SPY_STATUS_GLOBAL_ERR = 0x01;
const long SPY_STATUS_TX_MSG = 0x02;
const long SPY_STATUS_XTD_FRAME = 0x04;
const long SPY_STATUS_REMOTE_FRAME = 0x08;

const long SPY_STATUS_CRC_ERROR = 0x10;
const long SPY_STATUS_CAN_ERROR_PASSIVE = 0x20;
const long SPY_STATUS_INCOMPLETE_FRAME = 0x40;
const long SPY_STATUS_LOST_ARBITRATION = 0x80;

const long SPY_STATUS_UNDEFINED_ERROR = 0x100;
const long SPY_STATUS_CAN_BUS_OFF = 0x200;
const long SPY_STATUS_CAN_ERROR_WARNING = 0x400;
const long SPY_STATUS_BUS_SHORTED_PLUS = 0x800;

const long SPY_STATUS_BUS_SHORTED_GND = 0x1000;
const long SPY_STATUS_CHECKSUM_ERROR = 0x2000;
const long SPY_STATUS_BAD_MESSAGE_BIT_TIME_ERROR = 0x4000;
const long SPY_STATUS_IFR_DATA = 0x8000;

const long SPY_STATUS_HARDWARE_COMM_ERROR = 0x10000;
const long SPY_STATUS_EXPECTED_LEN_ERROR = 0x20000;
const long SPY_STATUS_INCOMING_NO_MATCH = 0x40000;
const long SPY_STATUS_BREAK = 0x80000;

const long SPY_STATUS_AVSI_REC_OVERFLOW = 0x100000;
const long SPY_STATUS_TEST_TRIGGER = 0x200000;
const long SPY_STATUS_AUDIO_COMMENT = 0x400000;
const long SPY_STATUS_GPS_DATA = 0x800000;

const long SPY_STATUS_ANALOG_DIGITAL_INPUT = 0x1000000;
const long SPY_STATUS_TEXT_COMMENT = 0x2000000;
const long SPY_STATUS_NETWORK_MESSAGE_TYPE = 0x4000000;
const long SPY_STATUS_VSI_TX_UNDERRUN = 0x8000000;
```

```
const long SPY_STATUS_VSI_IFR_CRC_Bit = 0x10000000;
const long SPY_STATUS_INIT_MESSAGE = 0x20000000;
const long SPY_STATUS_HIGH_SPEED_MESSAGE = 0x40000000;

// The second status bitfield
const long SPY_STATUS2_HAS_VALUE = 1;
const long SPY_STATUS2_VALUE_IS_BOOLEAN = 2;
const long SPY_STATUS2_HIGH_VOLTAGE = 4;
const long SPY_STATUS2_LONG_MESSAGE = 8;
```

## Visual Basic Declares

```
Public Enum icsSpyDataStatusBitfield
    icsSpyStatusGlobalError = 2 ^ 0
    icsSpyStatusTx = 2 ^ 1
    icsSpyStatusXtdFrame = 2 ^ 2
    icsSpyStatusRemoteFrame = 2 ^ 3
    icsSpyStatusErrCRCError = 2 ^ 4
    icsSpyStatusCANErrorPassive = 2 ^ 5
    icsSpyStatusErrIncompleteFrame = 2 ^ 6
    icsSpyStatusErrLostArbitration = 2 ^ 7
    icsSpyStatusErrUndefined = 2 ^ 8
    icsSpyStatusErrCANBusOff = 2 ^ 9
    icsSpyStatusErrCANErrorWarning = 2 ^ 10
    icsSpyStatusBusShortedPlus = 2 ^ 11
    icsSpyStatusBusShortedGnd = 2 ^ 12
    icsSpyStatusCheckSumError = 2 ^ 13
    icsSpyStatusErrBadMessageBitTimeError = 2 ^ 14
    icsSpyStatusIFRData = 2 ^ 15
    icsSpyStatusHardwareCommError = 2 ^ 16
    icsSpyStatusExpectedLengthError = 2 ^ 17
    icsSpyStatusIncomingNoMatch = 2 ^ 18
    icsSpyStatusBreak = 2 ^ 19
    icsSpyStatusAVT_VSIRecOverflow = 2 ^ 20
    icsSpyStatusTestTrigger = 2 ^ 21
    icsSpyStatusAudioCommentType = 2 ^ 22
    icsSpyStatusGPSPDataValue = 2 ^ 23
    icsSpyStatusAnalogDigitalInputValue = 2 ^ 24
```

---

```

    icsSpyStatusTextCommentType = 2 ^ 25
    icsSpyStatusNetworkMessageType = 2 ^ 26
    icsSpyStatusVSI_TxUnderRun = 2 ^ 27
    icsSpyStatusVSI_IFR_CRCBit = 2 ^ 28
    icsSpyStatusInitMessage = 2 ^ 29
    icsSpyStatusHighSpeed = 2 ^ 30
End Enum

Public Enum icsSpyDataStatusBitfield2
    icsSpyStatusHasValue = 2 ^ 0
    icsSpyStatusValueIsBoolean = 2 ^ 1
    icsSpyStatusHighVoltage = 2 ^ 2
    icsSpyStatusLongMessage = 2 ^ 3
End Enum

```

## Visual Basic .NET Declares

```

Public Enum icsSpyDataStatusBitfield
    icsSpyStatusGlobalError = 2 ^ 0
    icsSpyStatusTx = 2 ^ 1
    icsSpyStatusXtdFrame = 2 ^ 2
    icsSpyStatusRemoteFrame = 2 ^ 3
    icsSpyStatusErrCRCError = 2 ^ 4
    icsSpyStatusCANErrorPassive = 2 ^ 5
    icsSpyStatusErrIncompleteFrame = 2 ^ 6
    icsSpyStatusErrLostArbitration = 2 ^ 7
    icsSpyStatusErrUndefined = 2 ^ 8
    icsSpyStatusErrCANBusOff = 2 ^ 9
    icsSpyStatusErrCANErrorWarning = 2 ^ 10
    icsSpyStatusBusShortedPlus = 2 ^ 11
    icsSpyStatusBusShortedGnd = 2 ^ 12
    icsSpyStatusCheckSumError = 2 ^ 13
    icsSpyStatusErrBadMessageBitTimeError = 2 ^ 14
    icsSpyStatusIFRData = 2 ^ 15
    icsSpyStatusHardwareCommError = 2 ^ 16
    icsSpyStatusExpectedLengthError = 2 ^ 17
    icsSpyStatusIncomingNoMatch = 2 ^ 18
    icsSpyStatusBreak = 2 ^ 19
    icsSpyStatusAVT_VSIRecOverflow = 2 ^ 20
    icsSpyStatusTestTrigger = 2 ^ 21
    icsSpyStatusAudioCommentType = 2 ^ 22
    icsSpyStatusGPSPDataValue = 2 ^ 23

```

---

```
    icsSpyStatusAnalogDigitalInputValue = 2 ^ 24
    icsSpyStatusTextCommentType = 2 ^ 25
    icsSpyStatusNetworkMessageType = 2 ^ 26
    icsSpyStatusVSI_TxUnderRun = 2 ^ 27
    icsSpyStatusVSI_IFR_CRCBit = 2 ^ 28
    icsSpyStatusInitMessage = 2 ^ 29
    icsSpyStatusHighSpeed = 2 ^ 30
End Enum

Public Enum icsSpyDataStatusBitfield2
    icsSpyStatusHasValue = 2 ^ 0
    icsSpyStatusValueIsBoolean = 2 ^ 1
    icsSpyStatusHighVoltage = 2 ^ 2
    icsSpyStatusLongMessage = 2 ^ 3
End Enum
```

## C# Declares

```
public enum eDATA_STATUS_BITFIELD_1
{
    SPY_STATUS_GLOBAL_ERR = 0x01,
    SPY_STATUS_TX_MSG = 0x02,
    SPY_STATUS_XTD_FRAME = 0x04,
    SPY_STATUS_REMOTE_FRAME = 0x08,
    SPY_STATUS_CRC_ERROR = 0x10,
    SPY_STATUS_CAN_ERROR_PASSIVE = 0x20,
    SPY_STATUS_INCOMPLETE_FRAME = 0x40,
    SPY_STATUS_LOST_ARBITRATION = 0x80,
    SPY_STATUS_UNDEFINED_ERROR = 0x100,
    SPY_STATUS_CAN_BUS_OFF = 0x200,
    SPY_STATUS_CAN_ERROR_WARNING = 0x400,
    SPY_STATUS_BUS_SHORTED_PLUS = 0x800,
    SPY_STATUS_BUS_SHORTED_GND = 0x1000,
    SPY_STATUS_CHECKSUM_ERROR = 0x2000,
    SPY_STATUS_BAD_MESSAGE_BIT_TIME_ERROR = 0x4000,
    SPY_STATUS_IFR_DATA = 0x8000,
    SPY_STATUS_HARDWARE_COMM_ERROR = 0x10000,
    SPY_STATUS_EXPECTED_LEN_ERROR = 0x20000,
    SPY_STATUS_INCOMING_NO_MATCH = 0x40000,
    SPY_STATUS_BREAK = 0x80000,
    SPY_STATUS_AVSI_REC_OVERFLOW = 0x100000,
    SPY_STATUS_TEST_TRIGGER = 0x200000,
```

```

SPY_STATUS_AUDIO_COMMENT = 0x400000,
SPY_STATUS_GPS_DATA = 0x800000,
SPY_STATUS_ANALOG_DIGITAL_INPUT = 0x1000000,
SPY_STATUS_TEXT_COMMENT = 0x2000000,
SPY_STATUS_NETWORK_MESSAGE_TYPE = 0x4000000,
SPY_STATUS_VSI_TX_UNDERRUN = 0x8000000,
SPY_STATUS_VSI_IFR_CRC_BIT = 0x10000000,
SPY_STATUS_INIT_MESSAGE = 0x20000000,
SPY_STATUS_HIGH_SPEED_MESSAGE = 0x40000000,
}

```

```

public enum eDATA_STATUS_BITFIELD_2
{
    SPY_STATUS2_HAS_VALUE = 0,
    SPY_STATUS2_VALUE_IS_BOOLEAN = 2,
    SPY_STATUS2_HIGH_VOLTAGE = 4,
    SPY_STATUS2_LONG_MESSAGE = 8,
}

```

## Remarks

The tables below describe the bitfields.

**Table 1 - StatusBitfield Elements**

C/C++ Name	VB Name	Description
SPY_STATUS_GLOBAL_ERR	icsSpyStatusGlobalError	This is set if the message has any other error bits set.
SPY_STATUS_TX_MSG	icsSpyStatusTx	This is set if the message was transmitted by this device.
SPY_STATUS_XTD_FRAME	icsSpyStatusXtdFrame	This is set if the CAN message received or transmitted has an extended (29 bit) identifier.
SPY_STATUS_REMOTE_FRAME	icsSpyStatusRemoteFrame	This is set if the CAN message received or transmitted is a remote frame.
SPY_STATUS_CRC_ERROR	icsSpyStatusErrCRCError	This is set for J1850 VPW messages which do not have a proper CRC byte.
SPY_STATUS_CAN_ERROR_PASSIVE	icsSpyStatusCANErrorPassive	Not used in the neoVI API.
SPY_STATUS_INCOMPLETE_FRAME	icsSpyStatusErrIncompleteFrame	This is set for a J1850 VPW message which is received that ended on a non-byte boundary.

SPY_STATUS_LOST_ARBITRATION	icsSpyStatusErrLostArbitration	Not used in the neoVI API.
SPY_STATUS_UNDEFINED_ERROR	icsSpyStatusErrUndefined	This is an undefined error in the neoVI hardware
SPY_STATUS_CAN_BUS_OFF	icsSpyStatusErrCANBusOff	This is set when there is a change in error status of a MCP2510 CAN controller. The bitfield of the error status is stored in data byte 1 of the message structure. A description of this bitfield is included in this <a href="#">topic</a> .
SPY_STATUS_CAN_ERROR_WARNING	icsSpyStatusErrCANErrorWarning	Not used in the neoVI API.
SPY_STATUS_BUS_SHORTED_PLUS	icsSpyStatusBusShortedPlus	Not used in the neoVI API.
SPY_STATUS_BUS_SHORTED_GND	icsSpyStatusBusShortedGnd	Not used in the neoVI API.
SPY_STATUS_CHECKSUM_ERROR	icsSpyStatusCheckSumError	Not used in the neoVI API.
SPY_STATUS_BAD_MESSAGE_BIT_TIME_ERROR	icsSpyStatusErrBadMessageBitTimeError	This is set for J1850 VPW messages which do not meet the specified bit times for the SOF or bit signals.
SPY_STATUS_IFR_DATA	icsSpyStatusIFRData	Not used in the neoVI API.
SPY_STATUS_HARDWARE_COMM_ERROR	icsSpyStatusHardwareCommError	Not used in the neoVI API.
SPY_STATUS_EXPECTED_LEN_ERROR	icsSpyStatusExpectedLengthError	Not used in the neoVI API.
SPY_STATUS_INCOMING_NO_MATCH	icsSpyStatusIncomingNoMatch	Not used in the neoVI API.
SPY_STATUS_BREAK	icsSpyStatusBreak	This is set if the J1850 VPW break symbol has been received or is to be transmitted.
SPY_STATUS_AVSI_REC_OVERFLOW	icsSpyStatusAVT_VSIRecOverflow	Not used in the neoVI API.
SPY_STATUS_TEST_TRIGGER	icsSpyStatusTestTrigger	Not used in the neoVI API.
SPY_STATUS_AUDIO_COMMENT	icsSpyStatusAudioCommentType	Not used in the neoVI API.
SPY_STATUS_GPS_DATA	icsSpyStatusGPSDataValue	Not used in the neoVI API.
SPY_STATUS_ANALOG_DIGITAL_INPUT	icsSpyStatusAnalogDigitalInputValue	Not used in the neoVI API.
SPY_STATUS_TEXT_COMMENT	icsSpyStatusTextCommentType	Not used in the neoVI API.
SPY_STATUS_NETWORK_MESSAGE_TYPE	icsSpyStatusNetworkMessageType	This is set for all messages received from a vehicle network.
SPY_STATUS_VSI_TX_UNDERRUN	icsSpyStatusVSI_TxUnderRun	Not used in the neoVI API.
SPY_STATUS_VSI_IFR_CRC_Bit	icsSpyStatusVSI_IFR_CRCBit	Not used in the neoVI API.
		This is set if the transmitted message should

SPY_STATUS_INIT_MESSAGE	icsSpyStatusInitMessage	generate the ISO/Keyword2000 initialization waveform.
SPY_STATUS_HIGH_SPEED_MESSAGE	icsSpyStatusHighSpeed	This is set if the transmitted message is transmitted in high speed mode.

**Table 2 - StatusBitfield2 Elements**

C/C++ Name	VB Name	Description
SPY_STATUS2_HAS_VALUE	icsSpyStatusHasValue	Not used in the neoVI API.
SPY_STATUS2_VALUE_IS_BOOLEAN	icsSpyStatusValueIsBoolean	Not used in the neoVI API.
SPY_STATUS2_HIGH_VOLTAGE	icsSpyStatusHighVoltage	This is set if the transmitted message is transmitted in high voltage wakeup mode.
SPY_STATUS2_LONG_MESSAGE	icsSpyStatusLongMessage	Not used in the neoVI API.

VB Module: bas\_neoVI.bas  
 C/C++ Header: neovi.h  
 C/C++ Library File: icsneoVI.lib  
 DLL File: icsneoVI.dll  
 VB.Net Module: bas\_neoVI.vb  
 C# Class: icsNeoClass.cs

## Examples

### Visual Basic Example

#### Determining if we transmitted a message (This is indicated by icsSpyStatusTx bit set)

```
If (Msg.StatusBitField And icsSpyStatusTx) > 0 Then
  '// The message was transmitted by us
End If
```

#### Transmitting a CAN Extended ID Remote Frame

```
'// Load the message to be transmitted ArbID = FF extended remote frame with DLC=4
```

```

With stMessagesTx
    .ArbIDOrHeader = &HFF
    .NumberBytesData = 4
    .StatusBitField = icsSpyStatusXtdFrame + icsSpyStatusRemoteFrame
End With

lResult = icsneoTxMessages(m_hObject, stMessagesTx, NETID_HSCAN, 1)

```

## C/C++ Example

### Determining if we there's an error in the message (This is indicated by SPY\_STATUS\_GLOBAL\_ERR bit set)

```

if (mMsg.StatusBitField & SPY_STATUS_GLOBAL_ERR)
{
    // This message has an error in it
}

```

## Visual Basic .NET Example

### Determining if we transmitted a message (This is indicated by icsSpyStatusTx bit set)

```

If (Msg.StatusBitField And icsSpyDataStatusBitfield.icsSpyStatusTx) > 0 Then
    '// The message was transmitted by us
End If

```

## Transmitting a CAN Extended ID Remote Frame

```

'// Load the message to be transmitted ArbID = FF extended remote frame with DLC=4
With stMessagesTx
    .ArbIDOrHeader = &HFF
    .NumberBytesData = 4
    .StatusBitField = icsSpyDataStatusBitfield.icsSpyStatusXtdFrame +
icsSpyDataStatusBitfield.icsSpyStatusRemoteFrame
End With
lResult = icsneoTxMessages(m_hObject, stMessagesTx, NETID_HSCAN, 1)

```

## C# Example

**Determining if we there's an error in the message (This is indicated by SPY\_STATUS\_GLOBAL\_ERR bit set)**

```
if (mMsg.StatusBitField & SPY_STATUS_GLOBAL_ERR)
{
    // This message has an error in it
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, March 19, 2010*

## Intrepid API Data Types

Various data types defined or re-defined by the Intrepid API.

Data Type	Definition
icscm_int16	<code>short</code>
icscm_uint16	<code>unsigned short</code>
icscm_uint32	<code>unsigned int</code>
icscm_int32	<code>int</code>
icscm_uint8	<code>unsigned char</code>
icscm_int64	<code>int64</code>

---

intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)

Last Updated : Friday, March 19, 2010

## Valid parameter names for the SetDeviceParameter and GetDeviceParameter methods

The sections below list all of the valid device parameters for each supported neoVI device.

### neoVI Fire Parameters

The valid parameters for a neoVI Fire device are listed below. See [SFireSettingsStructure](#) for a listing of the valid values for each parameter.

can1 can2 can3 can4 (see [CAN Network Parameters](#))  
swcan (see [Single Wire CAN Network Parameters](#))  
lsftcan (see [CAN Network Parameters](#))  
lin1 lin2 lin3 lin4 (see [LIN Network Parameters](#))  
cgi\_baud  
cgi\_tx\_ifs\_bit\_times  
cgi\_rx\_ifs\_bit\_times  
cgi\_chksum\_enable  
network\_enables  
network\_enabled\_on\_boot  
pwm\_man\_timeout  
pwr\_man\_enable  
misc\_io\_initial\_ddr  
misc\_io\_initial\_latch  
misc\_io\_analog\_enable  
misc\_io\_report\_period  
misc\_io\_on\_report\_events  
ain\_sample\_period  
ain\_threshold  
iso15765\_separation\_time\_offset  
iso9141\_kwp\_settings (see [ISO9141 KWP Network Parameters](#))  
perf\_en  
iso\_parity  
iso\_msg\_termination  
network\_enables\_2

### valueCAN3 Parameters

can1 can2 (see [CAN Network Parameters](#))

network\_enables

network\_enabled\_on\_boot

iso15765\_separation\_time\_offset

perf\_en

misc\_io\_initial\_ddr

misc\_io\_initial\_latch

misc\_io\_report\_period

misc\_io\_on\_report\_events

## CAN Network Parameters

The valid parameters for CAN network settings on a neoVI device are listed below. Substitute the number of the CAN channel on the device for the 'x'. The current valid CAN network specifiers are can1 through can4, depending the capabilities of the device. See

[CAN\\_SETTINGSStructure](#) for a listing of valid values for each parameter.

canx/Mode

canx/SetBaudrate

canx/Baudrate

canx/NetworkType

canx/TqSeg1

canx/TqSeg2

canx/TqProp

canx/TqSync

canx/BRP

canx/auto\_baud

## Single Wire CAN Network Parameters

The valid parameters for single wire CAN network settings on a neoVI device are listed below. See [SWCAN\\_SETTINGSStructure](#) for a listing of valid values for each parameter.

swcan/Mode

swcan/SetBaudrate

swcan/Baudrate

swcan/NetworkType

swcan/TqSeg1

swcan/TqSeg2

swcan/TqProp  
swcan/TqSync  
swcan/BRP  
swcan/high\_speed\_auto\_switch  
swcan/auto\_baud

## LIN Network Parameters

The valid parameters for a LIN network on a neoVI device are listed below. Substitute the number of the LIN channel on the device for the 'x'. The current valid LIN network specifiers are lin1 through lin4. See [LIN\\_SETTINGSStructure](#) for a listing of valid values for each parameter.

linx/Baudrate  
linx/spbrg  
linx;brgh  
linx/MasterResistor  
linx/Mode

## ISO9141\_KWP Network Parameters

The valid parameters for the ISO9141\_KWP Network are listed below. For the iso9141\_kwp\_settings/init\_steps/x/ parameters, substitute the number of the desired step for the 'x'. The current valid init\_steps range is 0 through 15, for a total of 16 steps.

iso9141\_kwp\_settings/Baudrate  
iso9141\_kwp\_settings/spbrg  
iso9141\_kwp\_settings;brgh  
iso9141\_kwp\_settings/init\_steps/x/time\_500us  
iso9141\_kwp\_settings/init\_steps/x/k  
iso9141\_kwp\_settings/init\_steps/x/l  
iso9141\_kwp\_settings/init\_step\_count  
iso9141\_kwp\_settings/p2\_500us  
iso9141\_kwp\_settings/p3\_500us  
iso9141\_kwp\_settings/p4\_500us  
iso9141\_kwp\_settings/chksum\_enabled



## SFireSettings Structure

This structure defines various settings for the neoVI Fire device.

### C/C++ Declare

```
typedef VS_MODIFIER struct _SFireSettings
{
    CAN_SETTINGS can1;
    CAN_SETTINGS can2;
    CAN_SETTINGS can3;
    CAN_SETTINGS can4;
    SWCAN_SETTINGS swcan;
    CAN_SETTINGS lsftcan;
    LIN_SETTINGS lin1;
    LIN_SETTINGS lin2;
    LIN_SETTINGS lin3;
    LIN_SETTINGS lin4;
    icscm_uint16 cgi_enable_reserved;
    icscm_uint16 cgi_baud;
    icscm_uint16 cgi_tx_ifs_bit_times;
    icscm_uint16 cgi_rx_ifs_bit_times;
    icscm_uint16 cgi_chksum_enable;
    icscm_uint16 network_enables;
    icscm_uint16 network_enabled_on_boot;
    icscm_uint16 pwm_man_timeout;
    icscm_uint16 pwr_man_enable;
    icscm_uint16 misc_io_initial_ddr;
    icscm_uint16 misc_io_initial_latch;
    icscm_uint16 misc_io_analog_enable;
    icscm_uint16 misc_io_report_period;
    icscm_uint16 misc_io_on_report_events;
    icscm_uint16 ain_sample_period;
    icscm_uint16 ain_threshold;
    icscm_uint16 iso15765_separation_time_offset;
    icscm_uint16 iso9141_kwp_enable_reserved;
    ISO9141_KW2000SETTINGS iso9141_kwp_settings;
    icscm_uint16 perf_en;
    icscm_uint16 iso_parity;
    icscm_uint16 iso_msg_termination;
    icscm_uint16 iso_tester_pullup_enable;
    icscm_uint16 network_enables_2;
```

```
ISO9141_KW2000SETTINGS iso9141_kwp_settings2;
icscm_uint16 iso_parity_2;
icscm_uint16 iso_msg_termination_2;
ISO9141_KW2000SETTINGS iso9141_kwp_settings_3;
icscm_uint16 iso_parity_3;
icscm_uint16 iso_msg_termination_3;
ISO9141_KW2000SETTINGS iso9141_kwp_settings_4;
icscm_uint16 iso_parity_4;
icscm_uint16 iso_msg_termination_4;
icscm_uint16 fast_init_network_enables_1;
icscm_uint16 fast_init_network_enables_2;
UART_SETTINGS uart;
UART_SETTINGS uart2;
STextAPISettings text_api;
}SFireSettings;
```

## Visual Basic Declares

```
Public Type SFireSettings
    can1 As CAN_SETTINGS
    can2 As CAN_SETTINGS
    can3 As CAN_SETTINGS
    can4 As CAN_SETTINGS
    swcan As SWCAN_SETTINGS
    lsftcan As CAN_SETTINGS
    lin1 As LIN_SETTINGS
    lin2 As LIN_SETTINGS
    lin3 As LIN_SETTINGS
    lin4 As LIN_SETTINGS
    cgi_enable_reserved As Integer
    cgi_baud As Integer
    cgi_tx_ifs_bit_times As Integer
    cgi_rx_ifs_bit_times As Integer
    cgi_chksum_enable As Integer
    network_enables As Integer
    network_enabled_on_boot As Integer
    pwm_man_timeout As Integer
    pwr_man_enable As Integer
    misc_io_initial_ddr As Integer
    misc_io_initial_latch As Integer
```

```
misc_io_analog_enable As Integer
misc_io_report_period As Integer
misc_io_on_report_events As Integer
ain_sample_period As Integer
ain_threshold As Integer
iso15765_separation_time_offset As Integer
iso9141_kwp_enable_reserved As Integer
iso9141_kwp_settings As ISO9141_KEYWORD2000_SETTINGS
perf_en As Integer
iso_parity As Integer
iso_msg_termination As Integer
iso_tester_pullup_enable As Integer
network_enables_2 As Integer
iso9141_kwp_settings2 As ISO9141_KEYWORD2000_SETTINGS
iso_parity_2 As Integer
iso_msg_termination_2 As Integer
iso9141_kwp_settings_3 As ISO9141_KEYWORD2000_SETTINGS
iso_parity_3 As Integer
iso_msg_termination_3 As Integer
iso9141_kwp_settings_4 As ISO9141_KEYWORD2000_SETTINGS
iso_parity_4 As Integer
iso_msg_termination_4 As Integer
fast_init_network_enables_1 As Integer
fast_init_network_enables_2 As Integer
uart As UART_SETTINGS
uart2 As UART_SETTINGS
text_api As STextAPISettings
End Type
```

## Visual Basic .NET Declares

```
<StructLayout(LayoutKind.Sequential, Pack:=2)> Public Structure SFireSettings
    Dim can1 As CAN_SETTINGS
    Dim can2 As CAN_SETTINGS
    Dim can3 As CAN_SETTINGS
    Dim can4 As CAN_SETTINGS
    Dim swcan As SWCAN_SETTINGS
    Dim lsftcan As CAN_SETTINGS
    Dim lin1 As LIN_SETTINGS
    Dim lin2 As LIN_SETTINGS
    Dim lin3 As LIN_SETTINGS
```

```
Dim lin4 As LIN_SETTINGS
Dim cgi_enable_reserved As UInt16
Dim cgi_baud As UInt16
Dim cgi_tx_ifs_bit_times As UInt16
Dim cgi_rx_ifs_bit_times As UInt16
Dim cgi_chksum_enable As UInt16
Dim network_enables As UInt16
Dim network_enabled_on_boot As UInt16
Dim pwm_man_timeout As UInt16
Dim pwr_man_enable As UInt16
Dim misc_io_initial_ddr As UInt16
Dim misc_io_initial_latch As UInt16
Dim misc_io_analog_enable As UInt16
Dim misc_io_report_period As UInt16
Dim misc_io_on_report_events As UInt16
Dim ain_sample_period As UInt16
Dim ain_threshold As UInt16
Dim iso15765_separation_time_offset As UInt16
Dim iso9141_kwp_enable_reserved As UInt16
iso9141_kwp_settings As ISO9141_KEYWORD2000_SETTINGS
Dim perf_en As UInt16
Dim iso_parity As UInt16
Dim iso_msg_termination As UInt16
Dim iso_tester_pullup_enable As UInt16
Dim network_enables_2 As UInt16
Dim iso9141_kwp_settings2 As ISO9141_KEYWORD2000_SETTINGS
Dim iso_parity_2 As UInt16
Dim iso_msg_termination_2 As UInt16
Dim iso9141_kwp_settings_3 As ISO9141_KEYWORD2000_SETTINGS
Dim iso_parity_3 As UInt16
Dim iso_msg_termination_3 As UInt16
Dim iso9141_kwp_settings_4 As ISO9141_KEYWORD2000_SETTINGS
Dim iso_parity_4 As UInt16
Dim iso_msg_termination_4 As UInt16
Dim fast_init_network_enables_1 As UInt16
Dim fast_init_network_enables_2 As UInt16
Dim uart As UART_SETTINGS
Dim uart2 As UART_SETTINGS
Dim text_api As STextAPISettings
End Structure
```

**C# Declares**

```
[StructLayout(LayoutKind.Sequential,Pack=2)]
public struct SFireSettings
{
    public CAN_SETTINGS can1;
    public CAN_SETTINGS can2;
    public CAN_SETTINGS can3;
    public CAN_SETTINGS can4;
    public SWCAN_SETTINGS swcan;
    public CAN_SETTINGS lsftcan;
    public LIN_SETTINGS lin1;
    public LIN_SETTINGS lin2;
    public LIN_SETTINGS lin3;
    public LIN_SETTINGS lin4;
    public UInt16 cgi_enable_reserved;
    public UInt16 cgi_baud;
    public UInt16 cgi_tx_ifs_bit_times;
    public UInt16 cgi_rx_ifs_bit_times;
    public UInt16 cgi_chksum_enable;
    public UInt16 network_enables;
    public UInt16 network_enabled_on_boot;
    public UInt16 pwm_man_timeout;
    public UInt16 pwr_man_enable;
    public UInt16 misc_io_initial_ddr;
    public UInt16 misc_io_initial_latch;
    public UInt16 misc_io_analog_enable;
    public UInt16 misc_io_report_period;
    public UInt16 misc_io_on_report_events;
    public UInt16 ain_sample_period;
    public UInt16 ain_threshold;
    public UInt16 iso15765_separation_time_offset;
    public UInt16 iso9141_kwp_enable_reserved;
    public ISO9141_KEYWORD2000_SETTINGS iso9141_kwp_settings;
    public UInt16 perf_en;
    public UInt16 iso_parity;
    public UInt16 iso_msg_termination;
    public UInt16 iso_tester_pullup_enable;
    public UInt16 network_enables_2;
    public ISO9141_KEYWORD2000_SETTINGS iso9141_kwp_settings2;
    public UInt16 iso_parity_2;
    public UInt16 iso_msg_termination_2;
```

```

public ISO9141_KEYWORD2000_SETTINGS iso9141_kwp_settings3;
public UInt16 iso_parity_3;
public UInt16 iso_msg_termination_3;
public ISO9141_KEYWORD2000_SETTINGS iso9141_kwp_settings_4;
public UInt16 iso_parity_4;
public UInt16 iso_msg_termination_4;
public UInt16 fast_init_network_enables_1;
public UInt16 fast_init_network_enables_2;
public UART_SETTINGS uart;
public UART_SETTINGS uart2;
public STextAPISettings text_api;
}

```

## Remarks

### Structure Elements

Item	Description
CAN_SETTINGS can1	See <a href="#">CAN_SETTINGS</a> structure
CAN_SETTINGS can2	See <a href="#">CAN_SETTINGS</a> structure
CAN_SETTINGS can3	See <a href="#">CAN_SETTINGS</a> structure
CAN_SETTINGS can4	See <a href="#">CAN_SETTINGS</a> structure
SWCAN_SETTINGS swcan	See <a href="#">SWCAN_SETTINGS</a> structure
CAN_SETTINGS lsftcan	See <a href="#">CAN_SETTINGS</a> structure
LIN_SETTING lin1	See <a href="#">LIN_SETTINGS</a> structure
LIN_SETTINGlin2	See <a href="#">LIN_SETTINGS</a> structure
LIN_SETTING lin3	See <a href="#">LIN_SETTINGS</a> structure
LIN_SETTING lin4	See <a href="#">LIN_SETTINGS</a> structure
icscm_uint16 cgi_enable_reserved	<b>Deprecated.</b> Enable and disable CGI using the <a href="#">network_enables</a> element.
icscm_uint16 cgi_baud	CGI network baud rate: 625 = 625,000kb 115 = 115,200kb
icscm_uint16 cgi_tx_ifs_bit_times	CGI network - Number of bits separating transmit frames. neoVI will ensure this number of idle bit times in between successive transmitted frames. Valid values (1-65535).

	<b>Default value = 13</b>  CGI network - Number of bits separating received frames. neoVI identifies the end of a rx frame when there is no CGI bus activity for this given number of bit times. Valid values (1-65535).																
<u>icscm_uint16</u> cgi_rx_ifs_bit_times	<b>Default value = 13</b>																
<u>icscm_uint16</u> cgi_chksum_enable	If enabled neoVI will append a 16 bit checksum to all transmitted frames. 1 to enable, 0 to disable.  <b>Default value = 1</b>																
<u>icscm_uint16</u> network_enables	<p>Bitfield containing the software license enables. Depending on the hardware license purchased the customer may have to conditionally select which hardware channels to enable. For example the neoVI Red license allows the user to enable any 2 Dual Wire CAN channels and any 2 LIN channels. To enable a specific network its corresponding bit must be set (1). In order to transmit or receive on a network it must be enabled.</p> <p>Bit field values:</p> <table border="1"> <tbody> <tr><td>HSCAN1 : 0</td><td>HSCAN3 : 8</td></tr> <tr><td>MSCAN : 1</td><td>CGI : 9</td></tr> <tr><td>LIN1 : 2</td><td>J1850VPW : 10</td></tr> <tr><td>LIN2 : 3</td><td>LIN3 : 11</td></tr> <tr><td>reserved : 4</td><td>LIN4 : 12</td></tr> <tr><td>HSCAN2 : 5</td><td>J1708 : 13</td></tr> <tr><td>LSFT : 6</td><td>SWCAN2 : 14</td></tr> <tr><td>SW_CAN : 7</td><td>LSFT2 : 15</td></tr> </tbody> </table> <p>Examples:</p> <p>Enable HSCAN1 and J1850VPW : network_enables = 1025 (401 hex) (0000010000000001 binary)</p> <p>Enable all networks: network_enables = 65535 (FFFF hex) (1111111111111111 binary)</p>	HSCAN1 : 0	HSCAN3 : 8	MSCAN : 1	CGI : 9	LIN1 : 2	J1850VPW : 10	LIN2 : 3	LIN3 : 11	reserved : 4	LIN4 : 12	HSCAN2 : 5	J1708 : 13	LSFT : 6	SWCAN2 : 14	SW_CAN : 7	LSFT2 : 15
HSCAN1 : 0	HSCAN3 : 8																
MSCAN : 1	CGI : 9																
LIN1 : 2	J1850VPW : 10																
LIN2 : 3	LIN3 : 11																
reserved : 4	LIN4 : 12																
HSCAN2 : 5	J1708 : 13																
LSFT : 6	SWCAN2 : 14																
SW_CAN : 7	LSFT2 : 15																

<code>icscm_uint16</code> network_enabled_on_boot	CoreMini is running or if neoVI is online with DLL/Vehicle Spy 3. Practically this means the the CAN controllers stay in Listen Only mode until the device goes online. Once online the neoVI loads the user settings. Setting this parameter to 1 will change this behavior so that the neoVI enables its controllers immediately on boot.  <b>Default value = 0</b>
<code>icscm_uint16</code> pwm_man_timeout	Number of milliseconds of no bus activity required before neoVI enters low power mode. Note pwr_man_enable must be set for power management to be enabled.  <b>Default value = 10000</b>
<code>icscm_uint16</code> pwr_man_enable	1 = enable Power Management, 0 = disable.  <b>Default value = 0</b>
<code>icscm_uint16</code> misc_io_initial_ddr	MISC IO Initial Data Direction Register. Controls the initial data direction of the tristates on all misc digital pins. Each bit corresponds to an individual misc pin. Bit value of 0 signifies an input and bit value 1 signifies an output. Bit values corresponding to non existent pins (EX MISC7-MISC15 on FIRE) have no effect.  <b>Default value = 0</b>  Examples:  Set MISC1 to be output, all else input: misc_io_initial_ddr = 1  Set MISC1 and MISC2 to be output, all else input: misc_io_initial_ddr = 3 (11 binary)  Set all MISC pins to output: misc_io_initial_ddr = 65535 (1111111111111111 binary)
	MISC IO Initial Latch Register. Controls the initial output latch value on all misc digital pins. Each bit corresponds to an individual misc pin. Bit value of 0 signifies an low voltage (ground) and bit value 1 signifies high voltage (3.3 V). Bit values corresponding

	<p>to non existent pins (EX MISC7-MISC15 on FIRE) have no effect.</p> <p><b>Default value = 0</b></p> <p>Examples:</p> <p>Set MISC1 to be high, all else low: misc_io_initial_latch = 1</p> <p>Set MISC1and MISC2 to be high, all else low: misc_io_initial_latch = 3 (11 binary)</p> <p>Set all MISC pins to high: misc_io_initial_latch = 65535 (1111111111111111 binary)</p> <p><b>Note: In order for digital outputs to work correctly the corresponding bit in misc_io_initial_ddr must be set to output and corrsponding bit in misc_io_analog_enable must be cleared.</b></p>
<u>icscm_uint16</u> misc_io_initial_latch	<p>MISC IO Analog Enable Register. Controls the initial analog enables on all misc analog pins. Each bit corresponds to an individual misc pin that supports analog input. Bit value of 0 signifies that corresponding misc pin is digital only, and bit value 1 signifies corresponding misc pin is analog. Note that because some MISC pins are not capable of analog they are not included in the register. For example neoVI FIRE's analog pins are MISC3-MISC6, therefore bit 0 corresponds to MISC3's analog enable. Bit values corresponding to non existent pins have no effect.</p> <p><b>Default value = 0</b></p> <p>Examples:</p> <p>Set MISC3 to be analog, all else digital. (neoVI FIRE) : misc_io_analog_enable = 1</p> <p>Set MISC3 and MISC4 to be analog, all else digital. (neoVI FIRE): misc_io_analog_enable = 3 (11 binary)</p>

	<p>Set all MISC pins to high: misc_io_analog_enable = 65535 (1111111111111111 binary)</p> <p><b>Note:</b> that in order for analog inputs to work correctly the corresponding bit in misc_io_analog_enable must be set to 1.</p>																
<u>icscm_uint16</u> misc_io_report_period	<p>Period in milliseconds of device report message holding digital and analog data.</p> <p><b>Default value = 100</b></p> <p><b>Note:</b> Periodic reporting requires misc_io_on_report_events[0] to be set.</p>																
<u>icscm_uint16</u> misc_io_on_report_events	<p>Bitfield holding enables for various report triggers for the General IO report.</p> <p><b>Default value = 0</b></p> <p>Bit field values:</p> <table border="1"> <tbody> <tr><td>REPORT_ON_PERIODIC : 0</td><td>REPORT_ON_LED2 : 8</td></tr> <tr><td>REPORT_ON_MISC1 : 1</td><td>REPORT_ON_KLINE : 9</td></tr> <tr><td>REPORT_ON_MISC2 : 2</td><td>REPORT_ON_MISC3_AIN : 10</td></tr> <tr><td>REPORT_ON_MISC3 : 3</td><td>REPORT_ON_MISC4_AIN : 11</td></tr> <tr><td>REPORT_ON_MISC4 : 4</td><td>REPORT_ON_MISC5_AIN : 12</td></tr> <tr><td>REPORT_ON_MISC5 : 5</td><td>REPORT_ON_MISC6_AIN : 13</td></tr> <tr><td>REPORT_ON_MISC6 : 6</td><td></td></tr> <tr><td>REPORT_ON_LED1 : 7</td><td></td></tr> </tbody> </table>	REPORT_ON_PERIODIC : 0	REPORT_ON_LED2 : 8	REPORT_ON_MISC1 : 1	REPORT_ON_KLINE : 9	REPORT_ON_MISC2 : 2	REPORT_ON_MISC3_AIN : 10	REPORT_ON_MISC3 : 3	REPORT_ON_MISC4_AIN : 11	REPORT_ON_MISC4 : 4	REPORT_ON_MISC5_AIN : 12	REPORT_ON_MISC5 : 5	REPORT_ON_MISC6_AIN : 13	REPORT_ON_MISC6 : 6		REPORT_ON_LED1 : 7	
REPORT_ON_PERIODIC : 0	REPORT_ON_LED2 : 8																
REPORT_ON_MISC1 : 1	REPORT_ON_KLINE : 9																
REPORT_ON_MISC2 : 2	REPORT_ON_MISC3_AIN : 10																
REPORT_ON_MISC3 : 3	REPORT_ON_MISC4_AIN : 11																
REPORT_ON_MISC4 : 4	REPORT_ON_MISC5_AIN : 12																
REPORT_ON_MISC5 : 5	REPORT_ON_MISC6_AIN : 13																
REPORT_ON_MISC6 : 6																	
REPORT_ON_LED1 : 7																	
<u>icscm_uint16</u> ain_sample_period	<p>Controls how long the Analog to Digital Converter samples before performing a convert in milliseconds. If it is set to zero the hardware will perform the conversion immediately after sampling. This option defaults to 0 but is accessible so that high impedance analog sources can still be used by manually increasing the sample period.</p> <p><b>Default value = 0</b></p>																
	Percent of full voltage change required to trigger a																

	<p>REPORT_ON_MISCX_AIN event. Valid range is 0-100.</p> <p><b>Default value = 0</b></p> <p>Examples:</p> <p>Report fires every time ADC value changes: ain_threshold = 0</p> <p>Report fires every time ADC value changes by 33 mV: ain_threshold = 1</p> <p>Report fires every time ADC value changes by 66 mV: ain_threshold = 2</p> <p>Report fires every time ADC value changes by 3.3 V (Unpractical): ain_threshold = 100</p> <p><b>Note: Periodic reporting requires proper misc_io_on_report_events bit to be set.</b></p>
<u>icscm_uint16</u> ain_threshold	<p>In an ISO15765-2 Transmission, the receiver transmits a flow control message that informs the transmitter how much time there should be between individual CAN messages. This parameter allows the user to shift that spacing to make it smaller or larger. Valid range is -1563 to 1563 units where each unit represents 6.4us. Defaults to 0. If IFS plus the offset is negative than the Tx Messages will be back to back.</p> <p><b>Default value = 0</b></p> <p>Examples:</p> <p>ISO15765-2 Tx Message Inner frame spacing is exactly what is specified in flow control message: iso15765_separation_time_offset = 0</p> <p>ISO15765-2 Tx Message Inner frame spacing is what's specified in flow control message.+ 998.4 us: iso15765_separation_time_offset = 156</p> <p>ISO15765-2 Tx Message Inner frame spacing is what's</p>

	<p>specified in flow control message.- 998.4 us: iso15765_separation_time_offset = -156</p>								
<a href="#">icscm_uint16</a> network_enables_2	<p>Bitfield containing the software license enables. Depending on the hardware license purchased the customer may have to conditionally select which hardware channels to enable. For example the neoVI Red license allows the user to enable any 2 Dual Wire CAN channels and any 2 LIN channels. To enable a specific network its corresponding bit must be set (1). In order to transmit or receive on a network it must be enabled.</p> <p>Bit field values:</p> <table border="1"> <tr> <td>KLIN1 : 0</td><td>Reserved : 4</td></tr> <tr> <td>KLIN2 : 1</td><td>UART : 5</td></tr> <tr> <td>KLIN3 : 2</td><td>UART2 : +</td></tr> <tr> <td>LIN2 : 3</td><td></td></tr> </table>	KLIN1 : 0	Reserved : 4	KLIN2 : 1	UART : 5	KLIN3 : 2	UART2 : +	LIN2 : 3	
KLIN1 : 0	Reserved : 4								
KLIN2 : 1	UART : 5								
KLIN3 : 2	UART2 : +								
LIN2 : 3									
<a href="#">ISO9141_KW2000SETTINGS</a> iso9141_kwp_settings	See <a href="#">ISO9141_KW2000SETTINGS</a> structure								
<a href="#">icscm_uint16</a> iso_parity;	ISO9141 Parity setting: 0 - no parity, 1 - even, 2 - odd								
<a href="#">icscm_uint16</a> iso_msg_termination;	ISO9141 message termination setting: 0 - use inner frame time 1 - GME CIM-SCL								
<a href="#">icscm_uint16</a> iso_tester_pullup_enable;	enables the 510 ohm pullup resistor on K Line								
<a href="#">icscm_uint16</a> network_enables_2;									
<a href="#">ISO9141_KW2000SETTINGS</a> iso9141_kwp_settings2;	See <a href="#">ISO9141_KW2000SETTINGS</a> structure								
<a href="#">icscm_uint16</a> iso_parity_2;	ISO9141 Parity setting: 0 - no parity, 1 - even, 2 - odd								
<a href="#">icscm_uint16</a> iso_msg_termination_2;	ISO9141 message termination setting: 0 - use inner frame time 1 - GME CIM-SCL								
<a href="#">ISO9141_KW2000SETTINGS</a> iso9141_kwp_settings_3;	See <a href="#">ISO9141_KW2000SETTINGS</a> structure								
<a href="#">icscm_uint16</a> iso_parity_3;	ISO9141 Parity setting: 0 - no parity, 1 - even, 2 - odd								
<a href="#">icscm_uint16</a> iso_msg_termination_3;	ISO9141 message termination setting: 0 - use inner frame time 1 - GME CIM-SCL								
<a href="#">ISO9141_KW2000SETTINGS</a> iso9141_kwp_settings_4;	See <a href="#">ISO9141_KW2000SETTINGS</a> structure								
<a href="#">icscm_uint16</a> iso_parity_4;	ISO9141 Parity setting: 0 - no parity, 1 - even, 2 - odd								
	ISO9141 message termination setting:								

<code>icscm_uint16 iso_msg_termination_4;</code>	0 - use inner frame time 1 - GME CIM-SCL		
<code>icscm_uint16 fast_init_network_enables;</code>	Bitfield containing the channels to fast wakeup on. Currently only HS and MS CAN are supported <table border="1"><tr><td>HSCAN1 : 0</td><td>MSCAN : 1</td></tr></table>	HSCAN1 : 0	MSCAN : 1
HSCAN1 : 0	MSCAN : 1		
<code>UART_SETTINGS</code> UART;	Power management must be enabled for this feature to work. See <code>UART_SETTINGS</code> structure		
<code>STextAPISettings</code> Text_API;	See <code>STextAPISettings</code> structure		

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, July 08, 2011*

## SVCAN3Settings Structure

This structure defines various settings for the ValueCAN3 device.

### C/C++ Declare

```
typedef VS_MODIFIER struct _SVCAN3Settings
{
    CAN_SETTINGS can1;
    CAN_SETTINGS can2;
    icscm_uint16 network_enables;
    icscm_uint16 network_enabled_on_boot;
    icscm_int16 iso15765_separation_time_offset;
    icscm_uint16 perf_en;
    icscm_uint16 misc_io_initial_ddr;
    icscm_uint16 misc_io_initial_latch;
    icscm_uint16 misc_io_report_period;
    icscm_uint16 misc_io_on_report_events;
} SVCAN3Settings;
```

### Visual Basic Declares

```
Public Type SVCAN3Settings
    Can1 As CAN_SETTINGS
    Can2 As CAN_SETTINGS
    Network_enables As Integer
    Network_enabled_on_boot As Integer
    Iso15765_separation_time_offset As Integer
    Perf_en As Integer
    Misc_io_initial_ddr As Integer
    Misc_io_initial_latch As Integer
    Misc_io_report_period As Integer
    Misc_io_on_report_events As Integer
End Structure
```

### Visual Basic .NET Declare

```
Public Structure SVCAN3Settings
    Dim Can1 As CAN_SETTINGS
    Dim Can2 As CAN_SETTINGS
    Dim Network_enables As Int16
```

```

Dim Network_enabled_on_boot As Int16
Dim Iso15765_separation_time_offset As Int16
Dim Perf_en As Int16
Dim Misc_io_initial_ddr As Int16
Dim Misc_io_initial_latch As Int16
Dim Misc_io_report_period As Int16
Dim Misc_io_on_report_events As Int16
End Structure

```

**C# Declares**

```

[StructLayout(LayoutKind.Sequential)]
public struct SVCAN3Settings
{
    public CAN_SETTINGS can1;
    public CAN_SETTINGS can2;
    public UInt16 network_enables;
    public UInt16 network_enabled_on_boot;
    public UInt16 iso15765_separation_time_offset;
    public UInt16 perf_en;
    public UInt16 misc_io_initial_ddr;
    public UInt16 misc_io_initial_latch;
    public UInt16 misc_io_report_period;
    public UInt16 misc_io_on_report_events;
}

```

**Remarks****Structure Elements**

Item	Description
CAN_SETTINGS can1	See <a href="#">CAN_SETTINGS</a> structure
CAN_SETTINGS can2	See <a href="#">CAN_SETTINGS</a> structure
icscm_uint16 network_enables	Bitfield containing the software license enables. Depending on the hardware license purchased the customer may have to conditionally select which hardware channels to enable. For example the neoVI Red license allows the user to enable any 2 Dual Wire CAN channels and any 2 LIN channels. To enable a specific network its corresponding bit must be set (1). In order to transmit or receive on a network it must be enabled.

	<p>Bit field values:</p> <p>HSCAN1 : 0 MSCAN : 1</p>
icscm_uint16 network_enabled_on_boot	<p>Normally neoVI only initiates its comm channels when CoreMini is running or if neoVI is online with DLL/Vehicle Spy 3. Practically this means the the CAN controllers stay in Listen Only mode until the device goes online. Once online the neoVI loads the user settings. Setting this parameter to 1 will change this behavior so that the neoVI enables its controllers immediately on boot.</p> <p><b>Default value = 0.</b></p>
icscm_int16 iso15765_separation_time_offset	<p>In an ISO15765-2 Transmission, the receiver transmits a flow control message that informs that transmitter how much time there should be between individual CAN messages. This parameter allows the user to shift that spacing to make it smaller or larger. Valid range is -1563 to 1563 units where each unit represents 6.4us. Defaults to 0. If IFS plus the offset is negative than the Tx Messages will be back to back.</p> <p><b>Default value = 0.</b></p> <p>Examples:</p> <p>ISO15765-2 Tx Message Inner frame spacing is exactly what is specified in flow control message: iso15765_separation_time_offset = 0</p> <p>ISO15765-2 Tx Message Inner frame spacing is what's specified in flow control message.+ 998.4 us: iso15765_separation_time_offset = 156</p> <p>ISO15765-2 Tx Message Inner frame spacing is what's specified in flow control message.- 998.4 us: iso15765_separation_time_offset = -156</p>
icscm_uint16 perf_en	MISC IO Initial Data Direction Register. Controls the initial data direction of the tristates on all misc digital pins. Each bit corresponds to an individual misc pin. Bit value of 0 signifies an input and bit value 1 signifies an output. Bit values corresponding to non existent pins (EX MISC7-MISC15 on FIRE) have no effect.

<pre>icscm_uint16 misc_io_initial_ddr</pre>	<p><b>Default value = 0.</b></p> <p>Examples:</p> <p>Set MISC1 to be output, all else input: misc_io_initial_ddr = 1</p> <p>Set MISC1and MISC2 to be output, all else input: misc_io_initial_ddr = 3 (11 binary)</p> <p>Set all MISC pins to output: misc_io_initial_ddr = 65535 (1111111111111111 binary)</p>
<pre>icscm_uint16 misc_io_initial_latch</pre>	<p>MISC IO Initial Latch Register. Controls the initial output latch value on all misc digital pins. Each bit corresponds to an individual misc pin. Bit value of 0 signifies an low voltage (ground) and bit value 1 signifies high voltage (3.3 V). Bit values corresponding to non existent pins (EX MISC7-MISC15 on FIRE) have no effect.</p> <p><b>Default value = 0.</b></p> <p>Examples:</p> <p>Set MISC1 to be high, all else low: misc_io_initial_latch = 1</p> <p>Set MISC1and MISC2 to be high, all else low: misc_io_initial_latch = 3 (11 binary)</p> <p>Set all MISC pins to high: misc_io_initial_latch = 65535 (1111111111111111 binary)</p> <p><b>Note:</b> In order for digital outputs to work correctly the corresponding bit in misc_io_initial_ddr must be set to output and corresponding bit in misc_io_analog_enable must be cleared.</p>
<pre>icscm_uint16 misc_io_report_period</pre>	<p>Period in milliseconds of device report message holding digital and analog data.</p> <p><b>Note:</b> Periodic reporting requires misc_io_on_report_events[0] to be set.</p>

icscm_uint16 misc_io_on_report_events	Bitfield holding enables for various report triggers for the General IO report.  <b>Default value = 0.</b>  Bit field values: <table border="1"><tr><td>REPORT_ON_PERIODIC : 0</td><td>REPORT_ON_LED2 : 8</td></tr><tr><td>REPORT_ON_MISC1 : 1</td><td>REPORT_ON_KLINE : 9</td></tr><tr><td>REPORT_ON_MISC2 : 2</td><td>REPORT_ON_MISC3_AIN : 10</td></tr><tr><td>REPORT_ON_MISC3 : 3</td><td>REPORT_ON_MISC4_AIN : 11</td></tr><tr><td>REPORT_ON_MISC4 : 4</td><td>REPORT_ON_MISC5_AIN : 12</td></tr><tr><td>REPORT_ON_MISC5 : 5</td><td>REPORT_ON_MISC6_AIN : 13</td></tr><tr><td>REPORT_ON_MISC6 : 6</td><td></td></tr><tr><td>REPORT_ON_LED1 : 7</td><td></td></tr></table>	REPORT_ON_PERIODIC : 0	REPORT_ON_LED2 : 8	REPORT_ON_MISC1 : 1	REPORT_ON_KLINE : 9	REPORT_ON_MISC2 : 2	REPORT_ON_MISC3_AIN : 10	REPORT_ON_MISC3 : 3	REPORT_ON_MISC4_AIN : 11	REPORT_ON_MISC4 : 4	REPORT_ON_MISC5_AIN : 12	REPORT_ON_MISC5 : 5	REPORT_ON_MISC6_AIN : 13	REPORT_ON_MISC6 : 6		REPORT_ON_LED1 : 7	
REPORT_ON_PERIODIC : 0	REPORT_ON_LED2 : 8																
REPORT_ON_MISC1 : 1	REPORT_ON_KLINE : 9																
REPORT_ON_MISC2 : 2	REPORT_ON_MISC3_AIN : 10																
REPORT_ON_MISC3 : 3	REPORT_ON_MISC4_AIN : 11																
REPORT_ON_MISC4 : 4	REPORT_ON_MISC5_AIN : 12																
REPORT_ON_MISC5 : 5	REPORT_ON_MISC6_AIN : 13																
REPORT_ON_MISC6 : 6																	
REPORT_ON_LED1 : 7																	

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, March 25, 2010*

**Configuration Array - intrepidcs API**[Parameters](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#)**Use to read and change device settings on a neoVI Blue or ValueCAN with the [GetConfiguration](#) and [SendConfiguration](#).****Parameters**

Name	Description	Valid Range	Notes															
NEO_CFG_MPIC_GENIO_SETUP	Controls the IO mode and initial value of pins MISC1 and MISC2	See Notes.	Bitfield for General IO															
			7	6	5	4	3	2	1	0								
			X	X	Misc2 Output Default Value 0=off 1=on	Misc1 Output Default Value 0=off 1=on	X	X	MISC 2 Data Direction 0=Output 1=Input	MISC 1 Data Direction 0=Output 1=Input								
NEO_CFG_MPIC_HS_CAN_CNF1, NEO_CFG_MPIC_MS_CAN_CNF1, NEO_CFG_MPIC_SW_CAN_CNF1, NEO_CFG_MPIC_LSFT_CAN_CNF1	CAN Controller Configuration Register 1	n/a	None.															
NEO_CFG_MPIC_HS_CAN_CNF2, NEO_CFG_MPIC_MS_CAN_CNF2, NEO_CFG_MPIC_SW_CAN_CNF2, NEO_CFG_MPIC_LSFT_CAN_CNF2	CAN Controller Configuration Register 2	n/a	None.															
NEO_CFG_MPIC_HS_CAN_CNF3, NEO_CFG_MPIC_MS_CAN_CNF3, NEO_CFG_MPIC_SW_CAN_CNF3, NEO_CFG_MPIC_LSFT_CAN_CNF3	CAN Controller Configuration Register 3	n/a	None.															
NEO_CFG_MPIC_HS_CAN_MODE, NEO_CFG_MPIC_MS_CAN_MODE, NEO_CFG_MPIC_SW_CAN_MODE, NEO_CFG_MPIC_LSFT_CAN_MODE	Sets the mode of CAN controller	See Notes	<table border="1"> <thead> <tr> <th>Mode</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Loopback</td><td>4</td></tr> <tr> <td>ListenOnly</td><td>2</td></tr> <tr> <td>Normal</td><td>1</td></tr> </tbody> </table>		Mode	Value	Loopback	4	ListenOnly	2	Normal	1						
Mode	Value																	
Loopback	4																	
ListenOnly	2																	
Normal	1																	

NEO_CFG_LBCC_SETUP_BITFIELD	Sets options for LBCC network	See Notes.	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
			X	X	X	X	X	X	Dynamically Adjust Address 0=No 1=Yes	Bit Rate 0 = 41.6K 1=83.3Kbps
NEO_CFG_LBCC_NDRC	LBCC Network Driver and Receiver Control Byte	See Notes.	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
			1	0	1		Tx Driver B Enable	Tx Driver A Enable	Rx B Enable	Rx Diff Enable Rx A Enable
NEO_CFG_LBCC_NODE_ADDRESS	Node Address of the LBCC Ford SCP Controller	0-254	None.							
NEO_CFG_LBCC_FREAD_D1	Functional Address of Function Read Data 1	0-254 : 255 = Disabled	None.							
NEO_CFG_LBCC_FREAD_D2	Functional Address of Function Read Data 1	0-254 : 255 = Disabled	None.							
NEO_CFG_LBCC_NUM LU1 ENTERIES	Number of entries in lookup table 1	0-31	Number of valid entries in table 1.							
NEO_CFG_LBCC_NUM LU1 START [1-31]	First lookup entry	n/a	Second lookup entry is NEO_CFG_LBCC_NUM LU1 START + 1, Third is NEO_CFG_LBCC_NUM LU1 START + 2							
NEO_CFG_LED_SETTING										
NEO_CFG_J1708_RXIFS	Inter-frame separation in Ms for J1708 Reception									

ISO_RX_IFS_TIME_EEADDR (ISO)			
ISO_TX_IFS_TIME_EEADDR (ISO)			
ISO_TX_INTERBYTE_TIME_EEADDR (ISO)			
EE_DATA_NUMBYTES_IN_TRIG (ISO)			
EE_DATA_TRIG_ENABLE_1 (ISO)			
EE_DATA_TRIG_ENABLE_2 (ISO)			
EE_DATA_TRIG_1_NUM_BYTES (ISO)			
EE_DATA_INIT_SETTING1 (ISO)			
EE_DATA_INIT_NUM_STEPS (ISO)			
EE_DATA_INIT_STEP1_CNT (ISO)			
EE_DATA_INIT_STEP1_IO (ISO)			
EE_DATA_INIT_STEP2_CNT (ISO)			
EE_DATA_INIT_STEP2_IO (ISO)			
EE_DATA_DEVIO_TYPE			
EE_DATA_DEVIO_INTERVAL			
EE_DATA_ADCON1			
EE_DATA_DIN_CHANGE_MASK			
EE_DATA_MISC_TRIS			
EE_DATA_MISC_INIT			
EE_DATA_LED_DURATION_TYPE			
EE_DATA_LED_TYPE			
EE_DATA_ISO_MODE			
EE_DATA_LIN_TMAX_8			
EE_DATA_LIN_TMAX_4_DIFF			
EE_DATA_LIN_TMAX_2_DIFF			
EE_DATA_RS232_SPBRG			
EE_DATA_RS232_SYNC			

**Remarks**

## Examples

### Visual Basic Example

```
Private m_hObject As Long      '// Declared at form level

Dim lResult As Long           '// Used to store the return value
Dim bNetworkIDs(0 To 16) As Byte // Array of network IDs passed to the driver
Dim bSCPIDs(0 To 255) As Byte  // Array of SCP functional IDs passed to the driver
Dim lCount As Long            '// General Purpose Counter Variable

// Initialize the network id array
For lCount = 0 To 16
    bNetworkIDs(lCount) = lCount
Next lCount

// open the first neoVI on USB
lResult = icsneoOpenPort(1, NEOVI_COMMTYPE_USB_BULK, INTREPIDCS_DRIVER_STANDARD, _
bNetworkIDs(0), bSCPIDs(0), m_hObject)

// Test the returned result
If CBool(lResult) Then
    MsgBox "Port Opened Successfully"
Else
    MsgBox "Problem Opening Port"
End If
```

### C/C++ Example

```
unsigned char bNetworkID[16];      // array of network ids
unsigned char bSCPIDs[255];        // array of SCP functional ids
int hObject = 0;                  // holds a handle to the neoVI object
unsigned long lCount;             // counter variable

// initialize the networkid array
for (lCount=0;lCount<16;lCount++)
    bNetworkID[lCount] = lCount;

// open the first neoVI on USB
```

```
if (!m_bPortOpen) // only if not already opened
{
    lResult = icsneoOpenPort(1 ,NEOVI_COMMTYPE_USB_BULK, INTREPIDCS_DRIVER_STANDARD,bNetworkID, bSCPIIDs, &hObject);
    if (lResult == 0)
        MessageBox(hWnd,TEXT("Problem Opening Port"),TEXT("neoVI Example"),0);
    else
        MessageBox(hWnd,TEXT("Port Opened Successfully"),TEXT("neoVI Example"),0);
}
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, September 18, 2008*

## CAN\_SETTINGS Structure

This structure defines settings for CAN networks on neoVI Fire and ValueCAN3 devices.

### C/C++ Declare

```
typedef VS_MODIFIER struct
{
    icscm_uint8 Mode;
    icscm_uint8 SetBaudrate;
    icscm_uint8 Baudrate;
    icscm_uint8 NetworkType;
    icscm_uint8 TqSeg1;
    icscm_uint8 TqSeg2;
    icscm_uint8 TqProp;
    icscm_uint8 TqSync;
    icscm_uint16 BRP;
    icscm_uint16 auto_baud;
} CAN_SETTINGS;
```

### Visual Basic Declares

```
Public Type CAN_SETTINGS
    Mode As Byte
    SetBaudrate As Byte
    Baudrate As Byte
    NetworkType As Byte
    TqSeg1 As Byte
    TqSeg2 As Byte
    TqProp As Byte
    TqSync As Byte
    BRP As Integer
    auto_baud As Integer
End Type
```

### Visual Basic .NET Declares

```
<StructLayout(LayoutKind.Sequential, Pack:=2)> Public Structure CAN_SETTINGS
    Dim Mode As Byte
    Dim SetBaudrate As Byte
    Dim Baudrate As Byte
    Dim NetworkType As Byte
```

```

Dim TqSeg1 As Byte
Dim TqSeg2 As Byte
Dim TqProp As Byte
Dim TqSync As Byte
Dim BRP As Int16
Dim auto_baud As Int16
End Structure

```

## C# Declares

```

[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct CAN_SETTINGS
{
    public byte Mode;
    public byte SetBaudrate;
    public byte Baudrate;
    public byte NetworkType;
    public byte TqSeg1;
    public byte TqSeg2;
    public byte TqProp;
    public byte TqSync;
    public UInt16 BRP;
    public UInt16 auto_baud;
}

```

## Remarks

### Structure Elements

Item	Description								
<code>icscm_uint8 Mode</code>	<p>CAN controller mode when the neoVI device goes online or runs a CoreMini script.</p> <p><b>Default value = 0</b></p> <table border="1"> <tr> <td>NORMAL</td><td>0</td></tr> <tr> <td>DISABLED</td><td>1</td></tr> <tr> <td>LISTEN ONLY</td><td>3</td></tr> <tr> <td>LISTEN ALL</td><td>7</td></tr> </table>	NORMAL	0	DISABLED	1	LISTEN ONLY	3	LISTEN ALL	7
NORMAL	0								
DISABLED	1								
LISTEN ONLY	3								
LISTEN ALL	7								

<p><u>scm_uint8</u> SetBaudrate</p>	<p>The bit rate of a CAN channel can be selected one of two ways. It can either be selected from a list of common bit rates (SetBaudrate=1) or the user can specify the CAN timing parameters (SetBaudrate=0)</p> <p><b>Default value = 0</b></p> <table border="1"> <tr> <td>AUTO (Select from bitrate list using Baudrate parameter)</td><td>0</td></tr> <tr> <td>USE_TQ (Use time quanta parameters)</td><td>1</td></tr> </table>	AUTO (Select from bitrate list using Baudrate parameter)	0	USE_TQ (Use time quanta parameters)	1																		
AUTO (Select from bitrate list using Baudrate parameter)	0																						
USE_TQ (Use time quanta parameters)	1																						
<p><u>icscm_uint8</u> Baudrate</p>	<p>The bit rate of a CAN channel can be selected from a list of common bit rates Write the correct enumeration for the desired bit rate and ensure that SetBaudrate is 1(auto)</p> <p><b>Default value = 8</b></p> <p><b>Note:</b> This parameter is only applicable if SetBaudrate = 0</p> <table border="1"> <tr><td>20000</td><td>0</td></tr> <tr><td>33333</td><td>1</td></tr> <tr><td>50000</td><td>2</td></tr> <tr><td>62500</td><td>3</td></tr> <tr><td>83333</td><td>4</td></tr> <tr><td>100000</td><td>5</td></tr> <tr><td>125000</td><td>6</td></tr> <tr><td>250000</td><td>7</td></tr> <tr><td>500000</td><td>8</td></tr> <tr><td>800000</td><td>9</td></tr> <tr><td>1000000</td><td>10</td></tr> </table>	20000	0	33333	1	50000	2	62500	3	83333	4	100000	5	125000	6	250000	7	500000	8	800000	9	1000000	10
20000	0																						
33333	1																						
50000	2																						
62500	3																						
83333	4																						
100000	5																						
125000	6																						
250000	7																						
500000	8																						
800000	9																						
1000000	10																						
	Currently Not used. Will be supoprted in neoVI Yellow to																						

<u>icscm_uint8</u> NetworkType	software select which CAN transceiver to use (DW vs SW vs LSFT).
<u>icscm_uint8</u> TqSeg1	Phase 1 segment
<u>icscm_uint8</u> TqSeg2	Phase 2 segment
<u>icscm_uint8</u> TqProp	Propagation delay
<u>icscm_uint8</u> TqSync	Syncro jump width
<u>icscm_uint16</u> BRP	
<u>icscm_uint16</u> auto_baud	Enables the auto bitrate feature. 1 = enable, 0 = disable.  <b>Default value = 0</b>

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, March 05, 2014*

## SWCAN\_SETTINGS Structure

This structure defines settings for SWCAN networks on neoVI Fire devices.

### C/C++ Declare

```
typedef VS_MODIFIER struct
{
    icscm_uint8 Mode;
    icscm_uint8 SetBaudrate;
    icscm_uint8 Baudrate;
    icscm_uint8 NetworkType;
    icscm_uint8 TqSeg1;
    icscm_uint8 TqSeg2;
    icscm_uint8 TqProp;
    icscm_uint8 TqSync;
    icscm_uint16 BRP;
    icscm_uint16 high_speed_auto_switch;
    icscm_uint16 auto_baud;
} SWCAN_SETTINGS;
```

### Visual Basic Declares

```
Public Type SWCAN_SETTINGS
    Mode As Byte
    SetBaudrate As Byte
    Baudrate As Byte
    NetworkType As Byte
    TqSeg1 As Byte
    TqSeg2 As Byte
    TqProp As Byte
    TqSync As Byte
    BRP As Integer
    high_speed_auto_switch As Integer
    auto_baud As Integer
End Structure
```

### Visual Basic .NET Declare

```
<StructLayout(LayoutKind.Sequential, Pack:=2)> Public Structure SWCAN_SETTINGS
    Dim Mode As Byte
    Dim SetBaudrate As Byte
```

```

Dim Baudrate As Byte
Dim NetworkType As Byte
Dim TqSeg1 As Byte
Dim TqSeg2 As Byte
Dim TqProp As Byte
Dim TqSync As Byte
Dim BRP As Int16
Dim high_speed_auto_switch As Int16
Dim auto_baud As Int16
End Structure

```

## C# Declares

```

[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct SWCAN_SETTINGS
{
    public byte Mode;
    public byte SetBaudrate;
    public byte Baudrate;
    public byte NetworkType;
    public byte TqSeg1;
    public byte TqSeg2;
    public byte TqProp;
    public byte TqSync;
    public UInt16 BRP;
    public UInt16 high_speed_auto_switch;
    public UInt16 auto_baud;
}

```

## Remarks

### Structure Elements

Item	Description				
<u>icscm_uint8</u> Mode	<p>CAN controller mode when the neoVI device goes online or runs a CoreMini script.</p> <p><b>Default value = 0</b></p> <table border="1"> <tr> <td>NORMAL</td><td>0</td></tr> <tr> <td>DISABLED</td><td>1</td></tr> </table>	NORMAL	0	DISABLED	1
NORMAL	0				
DISABLED	1				

	<table border="1"> <tr><td>LISTEN ONLY</td><td>3</td></tr> <tr><td>LISTEN ALL</td><td>7</td></tr> </table>	LISTEN ONLY	3	LISTEN ALL	7																		
LISTEN ONLY	3																						
LISTEN ALL	7																						
<u>icscm_uint8</u> SetBaudrate	<p>The bit rate of a CAN channel can be selected one of two ways. It can either be selected from a list of common bit rates (SetBaudrate=1) or the user can specify the CAN timing parameters (SetBaudrate=0)</p> <p><b>Default value = 0</b></p> <table border="1"> <tr><td>AUTO (Select from bitrate list using Baudrate parameter)</td><td>0</td></tr> <tr><td>USE_TQ (Use time quanta parameters)</td><td>1</td></tr> </table>	AUTO (Select from bitrate list using Baudrate parameter)	0	USE_TQ (Use time quanta parameters)	1																		
AUTO (Select from bitrate list using Baudrate parameter)	0																						
USE_TQ (Use time quanta parameters)	1																						
<u>icscm_uint8</u> Baudrate	<p>The bit rate of a CAN channel can be selected from a list of common bit rates Write the correct enumeration for the desired bit rate and ensure that SetBaudrate is 1 (auto)</p> <p><b>Default value = 8</b></p> <p><b>Note:</b> This parameter is only applicable if SetBaudrate = 1</p> <table border="1"> <tr><td>20000</td><td>0</td></tr> <tr><td>33333</td><td>1</td></tr> <tr><td>50000</td><td>2</td></tr> <tr><td>62500</td><td>3</td></tr> <tr><td>83333</td><td>4</td></tr> <tr><td>100000</td><td>5</td></tr> <tr><td>125000</td><td>6</td></tr> <tr><td>250000</td><td>7</td></tr> <tr><td>500000</td><td>8</td></tr> <tr><td>800000</td><td>9</td></tr> <tr><td>1000000</td><td>10</td></tr> </table>	20000	0	33333	1	50000	2	62500	3	83333	4	100000	5	125000	6	250000	7	500000	8	800000	9	1000000	10
20000	0																						
33333	1																						
50000	2																						
62500	3																						
83333	4																						
100000	5																						
125000	6																						
250000	7																						
500000	8																						
800000	9																						
1000000	10																						
	Currently Not used. Will be supoprted in neoVI Yellow to																						

<code>icscm_uint8</code> NetworkType	software select which CAN transceiver to use (DW vs SW vs LSFT).	
<code>icscm_uint8</code> TqSeg1	Phase 1 segment	
<code>icscm_uint8</code> TqSeg2	Phase 2 segment	
<code>icscm_uint8</code> TqProp	Propagation delay	
<code>icscm_uint8</code> TqSync	Syncro jump width	
<code>icscm_uint16</code> BRP		
<code>icscm_uint16</code> high_speed_auto_switch	DISABLED	0
	NO_RESISTOR	1
	WITH_RESISTOR	2
<code>icscm_uint16</code> auto_baud	Enables the auto bitrate feature. 1 = enable, 0 = disable.  <b>Default value = 0</b>	

## LIN\_SETTINGS Structure

This structure defines settings for LIN networks on neoVI Fire devices.

### C/C++ Declare

```
typedef VS_MODIFIER struct _LIN_SETTINGS
{
    icscm_uint32 Baudrate;
    icscm_uint16 spbrg;
    icscm_uint16 brgh;
    icscm_uint8 MasterResistor;
    icscm_uint8 Mode;
} LIN_SETTINGS;
```

### Visual Basic Declares

```
Public Type LIN_SETTINGS
    Baudrate As Long
    spbrg As Integer
    brgh As Integer
    MasterResistor As Byte
    Mode As Byte
End Type
```

### Visual Basic .NET Declares

```
<StructLayout(LayoutKind.Sequential, Pack:=2)>Public Structure LIN_SETTINGS
    Dim Baudrate As Int32
    Dim spbrg As Int16
    Dim brgh As Int16
    Dim MasterResistor As Byte
    Dim Mode As Byte
End Structure
```

### C# Declares

```
[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct LIN_SETTINGS
```

```
{
    public UInt32 Baudrate;
    public UInt16 spbreg;
    public UInt16 brgh;
    public byte MasterResistor;
    public byte Mode;
}
```

**Remarks****Structure Elements**

Item	Description
<u>icscm_uint32</u> Baudrate	Baudrate to be set
<u>icscm_uint16</u> spbreg	Setting for the baudrate. It is calculated using the equation below. spbreg = ((40000000/Baudrate)/16)-1
<u>icscm_uint16</u> brgh	brgh should always be set to 0
<u>icscm_uint8</u> MasterResistor	Enables the Master Resistor on the hardware. 0 = Disabled 1 = Enabled
<u>icscm_uint8</u> Mode	Sets the mode for the device. 0=Sleep 1=Slow 2=Normal 3=Fast

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, November 29, 2013*

## **ISO9141\_KEYWORD2000\_SETTINGS Structure**

**This structure defines settings for ISO9141 and Keyword 2000 networks on neoVI Fire devices.**

### **C/C++ Declare**

```
typedef VS_MODIFIER struct
{
    icscm_uint32 Baudrate;
    icscm_uint16 spbrg;
    icscm_uint16 brgh;
    ISO9141_KW2000_INIT_STEP init_steps[16]; //See the ISO9141_KW2000_INIT_STEP structure
    icscm_uint8 init_step_count;
    icscm_uint16 p2_500us;
    icscm_uint16 p3_500us;
    icscm_uint16 p4_500us;
    icscm_uint16 cksum_enabled;
} ISO9141_KEYWORD2000_SETTINGS;
```

### **Visual Basic Declares**

```
<StructLayout(LayoutKind.Sequential, Pack:=2)>Public Type ISO9141_KEYWORD2000_SETTINGS
    Baudrate As Int32
    spbrg As Int16
    brgh As Int16
    Init_Step(0 to 15) As ISO9141_KEYWORD2000_INIT_STEP
    init_step_count As Byte
    p2_500us As Int16
    p3_500us As Int16
    p4_500us As Int16
    cksum_enabled As Int16
End Type
```

### **Visual Basic .NET Declares**

```
Public Structure ISO9141_KEYWORD2000_SETTINGS
    Dim Baudrate As Int32
    Dim spbrg As Int16
    Dim brgh As Int16
    Dim Init_Step_0 As ISO9141_KEYWORD2000_INIT_STEP
    Dim Init_Step_1 As ISO9141_KEYWORD2000_INIT_STEP
```

---

```

Dim Init_Step_2 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_3 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_4 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_5 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_6 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_7 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_8 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_9 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_10 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_11 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_12 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_13 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_14 As ISO9141_KEYWORD2000__INIT_STEP
Dim Init_Step_15 As ISO9141_KEYWORD2000__INIT_STEP
Dim init_step_count As Byte
Dim p2_500us As Int16
Dim p3_500us As Int16
Dim p4_500us As Int16
Dim um_enabled As Int16
End Structure

```

## C# Declares

```

[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct ISO9141_KEYWORD2000_SETTINGS
{
    public UInt32 Baudrate;
    public UInt16 spbrg;
    public UInt16 brgh;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_0;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_1;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_2;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_3;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_4;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_5;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_6;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_7;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_8;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_9;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_10;
    public ISO9141_KEYWORD2000__INIT_STEP Init_Step_11;
}

```

---

```

public ISO9141_KEYWORD2000_INIT_STEP Init_Step_12;
public ISO9141_KEYWORD2000_INIT_STEP Init_Step_13;
public ISO9141_KEYWORD2000_INIT_STEP Init_Step_14;
public ISO9141_KEYWORD2000_INIT_STEP Init_Step_15;
public byte init_step_count;
public UInt16 p2_500us;
public UInt16 p3_500us;
public UInt16 p4_500us;
public UInt16 checksum_enabled;
}

```

**Remarks****Structure Elements**

Item	Description
<a href="#">icscm_uint32</a> Baudrate	
<a href="#">icscm_uint16</a> spbrg	
<a href="#">icscm_uint16</a> brgh	
<a href="#">ISO9141_KW2000_INIT_STEP</a> init_steps[16]	
<a href="#">icscm_uint8</a> init_step_count	
<a href="#">icscm_uint16</a> p2_500us	
<a href="#">icscm_uint16</a> p3_500us	
<a href="#">icscm_uint16</a> p4_500us	
<a href="#">icscm_uint16</a> checksum_enabled	

## STextAPISettings Structure

**This structure defines settings for Text API communication for neoVI Fire devices.**

### C/C++ Declare

```
typedef struct _STextAPISettings
{
    unsigned int can1_tx_id;
    unsigned int can1_rx_id;
    unsigned int can1_options// Set to 1 for Extended, 0 for standard
    unsigned int can2_tx_id;
    unsigned int can2_rx_id;
    unsigned int can2_options // Set to 1 for Extended, 0 for standard
    unsigned int network_enables;
    unsigned int can3_tx_id3;
    unsigned int can3_rx_id3;
    unsigned int can3_options // Set to 1 for Extended, 0 for standard
    unsigned int can4_tx_id4;
    unsigned int can4_rx_id4;
    unsigned int can4_options // Set to 1 for Extended, 0 for standard
    int Reserved0;
    int Reserved1;
    int Reserved2;
    int Reserved3;
    int Reserved4;
} STextAPISettings;
```

### Visual Basic Declares

```
Public Type STextAPISettings
    can1_tx_id As Long
    can1_rx_id As Long
    can1_options As Long'// Set to 1 for Extended, 0 for standard
    can2_tx_id As Long
    can2_rx_id As Long
    can2_options As Long'// Set to 1 for Extended, 0 for standard
    network_enables As Long
    can3_tx_id3 As Long
    can3_rx_id3 As Long
    can3_options As Long'// Set to 1 for Extended, 0 for standard
```

```

can4_tx_id4 As Long
can4_rx_id4 As Long
can4_options As Long'// Set to 1 for Extended, 0 for standard
Reserved0 As Long
Reserved1 As Long
Reserved2 As Long
Reserved3 As Long
Reserved4 As Long
End Type

```

## Visual Basic .NET Declares

```

<StructLayout(LayoutKind.Sequential, Pack:=2)> Public Structure STextAPISettings
    Dim can1_tx_id As UInt32
    Dim can1_rx_id As UInt32
    Dim can1_options As UInt32 '// Set to 1 for Extended, 0 for standard
    Dim can2_tx_id As UInt32
    Dim can2_rx_id As UInt32
    Dim can2_options As UInt32 '// Set to 1 for Extended, 0 for standard
    Dim network_enables As UInt32
    Dim can3_tx_id3 As UInt32
    Dim can3_rx_id3 As UInt32
    Dim can3_options As UInt32 '// Set to 1 for Extended, 0 for standard
    Dim can4_tx_id4 As UInt32
    Dim can4_rx_id4 As UInt32
    Dim can4_options As UInt32 '// Set to 1 for Extended, 0 for standard
    Dim Reserved0 As Int32
    Dim Reserved1 As Int32
    Dim Reserved2 As Int32
    Dim Reserved3 As Int32
    Dim Reserved4 As Int32
End Structure

```

## C# Declares

```

[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct STextAPISettings
{
    public UInt32 can1_tx_id;

```

```

public UInt32 can1_rx_id;
public UInt32 can1_options; // Set to 1 for Extended, 0 for standard
public UInt32 can2_tx_id;
public UInt32 can2_rx_id;
public UInt32 can2_options; // Set to 1 for Extended, 0 for standard
public UInt32 network_enables;
public UInt32 can3_tx_id3;
public UInt32 can3_rx_id3;
public UInt32 can3_options; // Set to 1 for Extended, 0 for standard
public UInt32 can4_tx_id4;
public UInt32 can4_rx_id4;
public UInt32 can4_options; // Set to 1 for Extended, 0 for standard
public Int32 Reserved0;
public Int32 Reserved1;
public Int32 Reserved2;
public Int32 Reserved3;
public Int32 Reserved4;
}

```

## Remarks

### Structure Elements

Item	Description							
<u>icscm_uint32</u> can_tx_id	Sets or Reads the Arbitration ID for Sending Text API commands							
<u>icscm_uint32</u> can_rx_id	Sets or Reads the Arbitration ID for Receiving API commands							
<u>icscm_uint32</u> can_options	Sets the length of the Arbitration ID's. Set to 1 for Extended and 0 for Standard							
<u>icscm_uint32</u> network_Enables	Bitfield telling which netowrk to support Text API. One one can be enabled at a time. <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>HS CAN : 1</td> <td>MS CAN : 2</td> </tr> <tr> <td>HS CAN 2 : 32</td> <td>HS CAN 3 : 256</td> </tr> <tr> <td>RS232/UART2 : 4194304</td> <td>UART1 : 2097152</td> </tr> </table>		HS CAN : 1	MS CAN : 2	HS CAN 2 : 32	HS CAN 3 : 256	RS232/UART2 : 4194304	UART1 : 2097152
HS CAN : 1	MS CAN : 2							
HS CAN 2 : 32	HS CAN 3 : 256							
RS232/UART2 : 4194304	UART1 : 2097152							
<u>icscm_uint32</u> Reserved	Not used							

*Last Updated : Friday, July 08, 2011*

## UART\_SETTINGS Structure

**This structure defines settings for UART access on neoVI Fire devices.**

### C/C++ Declare

```
typedef struct _UART_SETTINGS{
    unsigned short Baudrate;
    unsigned short spbreg;
    unsigned short brgh;
    unsigned short parity;
    unsigned short stop_bits;
    unsigned char flow_control; // 0- off, 1 - Simple CTS RTS,
    unsigned char reserved_1;
    unsigned int bOptions; //AND to combind these values invert_tx = 1 invert_rx = 2 half_duplex = 4
} UART_SETTINGS;
```

### Visual Basic Declares

```
Public Type UART_SETTINGS
    Baudrate As Integer
    spbreg As Integer
    brgh As Integer
    parity As Integer
    stop_bits As Integer
    flow_control As Byte '// 0- off, 1 - Simple CTS RTS,
    reserved_1 As Byte
    bOptions As Long '//AND to combind these values invert_tx = 1 invert_rx = 2 half_duplex = 4
End Structure
```

### Visual Basic .NET Declares

```
<StructLayout(LayoutKind.Sequential, Pack:=2)> Public Structure UART_SETTINGS
    Dim Baudrate As UInt16
    Dim spbreg As UInt16
    Dim brgh As UInt16
    Dim parity As UInt16
    Dim stop_bits As UInt16
    Dim flow_control As Byte '// 0- off, 1 - Simple CTS RTS,
    Dim reserved_1 As Byte
    Dim bOptions As UInt32 '//AND to combind these values invert_tx = 1 invert_rx = 2 half_duplex = 4
```

[End Structure](#)**C# Declares**

```
[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct UART_SETTINGS
{
    public UInt16 Baudrate;
    public UInt16 spbrg;
    public UInt16 brgh;
    public UInt16 parity;
    public UInt16 stop_bits;
    public byte flow_control; // 0- off, 1 - Simple CTS RTS,
    public byte reserved_1;
    public UInt32 bOptions; //AND to combind these values invert_tx = 1 invert_rx = 2 half_duplex = 4
}
```

**Remarks****Structure Elements**

Item	Description	
<a href="#">icscm_uint16</a> Baudrate	Holds the baud rate for the UART Connection. An example value could be 10417 or 9600	
<a href="#">icscm_uint16</a> spbrg		
<a href="#">icscm_uint16</a> brgh		
<a href="#">icscm_uint16</a> parity	Sets the Parity type. Valid values are below	
	None	0
	Even	1
	Odd	2
<a href="#">icscm_uint16</a> stop_bits	Sets the number of stop bits to use. Valid values are below.	
	One Stop Bit	1
	Two Stop Bits	2
<a href="#">icscm_uint8</a> flow_control	Set to 0 for no flow control and 1 for simple CTS RTS	
<a href="#">icscm_uint8</a> reserved		

Bitfield containing UART Options	
<code>icscm_uint32</code> bOptions	Invert Tx
	1
	Invert Rx
	2
	Half Duplex
	4

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Wednesday, July 06, 2011

## ISO9141\_KEYWORD2000\_\_INIT\_STEP Structure

This structure defines settings for the ISO9141 and Keyword 2000 initialization step on neoVI Fire devices.

### C/C++ Declare

```
typedef VS_MODIFIER struct
{
    icscm_uint16 time_500us;
    icscm_uint16 k;
    icscm_uint16 l;
} ISO9141_KEYWORD2000__INIT_STEP;
```

### Visual Basic Declares

```
Public Type ISO9141_KEYWORD2000__INIT_STEP
    time_500us As Integer
    k As Integer
    l As Integer
End Type
```

### Visual Basic .NET Declares

```
<StructLayout(LayoutKind.Sequential, Pack:=2)>Public Structure ISO9141_KEYWORD2000__INIT_STEP
    Dim time_500us As Int16
    Dim k As Int16
    Dim l As Int16
End Structure
```

### C# Declares

```
[StructLayout(LayoutKind.Sequential, Pack=2)]
public struct ISO9141_KEYWORD2000__INIT_STEP
{
    public UInt16 time_500us;
    public UInt16 k;
    public UInt16 l;
}
```

**Remarks****Structure Elements**

Item	Description
<a href="#"><u>icscm_uint16</u></a> time_500us	
<a href="#"><u>icscm_uint16</u></a> k	
<a href="#"><u>icscm_uint16</u></a> l	

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, July 08, 2011*

## stCM\_ISO157652\_TxMessage Structure

This structure is used by [ScriptWriteISO15765\\_2\\_TxMessage](#) and [ScriptReadISO15765\\_2\\_TxMessage](#).

### C/C++ Declare

```
typedef struct _stCM_ISO157652_TxMessage
{
    icscm_uint16 vs_netid; /* not supported*/
    icscm_uint16 padding; /* not supported*/
    icscm_uint32 id;

    //flow control filter
    icscm_uint32 fc_id;
    icscm_uint32 fc_id_mask;
    icscm_uint16 fc_ext_address_enable; /* not supported*/
    icscm_uint16 fc_ext_address; /*not supported */

    //flow control timeouts
    icscm_uint16 fs_timeout; /* not supported*/
    icscm_uint16 fs_wait; /* not supported */
    icscm_uint8 data[4*1024];
    icscm_uint32 num_bytes;

    /* Option bits */
    union
    {
        struct
        {
            unsigned id_29_bit_enable:1;
            unsigned fc_id_29_bit_enable:1;
        };
        icscm_uint16 flags;
    };
} stCM_ISO157652_TxMessage;
```

### Visual Basic Declares

**C# Declares****Remarks****Structure Elements**

Item	Description

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

---

*Last Updated : Friday, September 19, 2008*

**Network ID List for the neoVI API**

Network ID	Value
NETID_DEVICE	0
NETID_HSCAN	1
NETID_MSCAN	2
NETID_SWCAN	3
NETID_LSFTCAN	4
NETID_FORDSCP	5
NETID_J1708	6
NETID_JVPW	8
NETID_ISO	9
NETID_ISO2	14
NETID_LIN	16
NETID_ISO3	41
NETID_HSCAN2	42
NETID_HSCAN3	44
NETID_ISO4	47
NETID_LIN2	48
NETID_LIN3	49
NETID_LIN4	50
NETID_CGI	53

## NeoDevice Structure

A structure used by [FindNeoDevices](#) and [OpenNeoDevice](#) to locate and open neoVI devices.

### C/C++ Declare

```
typedef struct
{
    unsigned long DeviceType;
    int Handle;
    int NumberOfClients;
    int SerialNumber;
    int MaxAllowedClients;
} NeoDevice;
```

### Visual Basic Declares

```
'//Structure for neoVI device types
Public Type NeoDevice
    Dim DeviceType_As Long
    Dim Handle_As Long
    Dim NumberOfClients_As Long
    Dim SerialNumber_As Long
    Dim MaxAllowedClients_As Long
End Type
```

### Visual Basic .NET Declare

```
'//Structure for neoVI device types
Public Structure NeoDevice
    Dim DeviceType_As Int32
    Dim Handle_As Int32
    Dim NumberOfClients_As Int32
    Dim SerialNumber_As Int32
    Dim MaxAllowedClients_As Int32
End Structure
```

### C# Declares

```
[StructLayout(LayoutKind.Sequential)]
```

```
public struct NeoDevice
{
    public Int32 DeviceType;
    public Int32 Handle;
    public Int32 NumberOfClients;
    public Int32 SerialNumber;
    public Int32 MaxAllowedClients;
}
```

## Remarks

Instances of this structure are initialized and set by calling [FindNeoDevices](#). Then the structure is used by [OpenNeoDevice](#) to make a physical connection to a neoVI device.

Item	Description
unsigned long DeviceType	A bit-wise field that indicates the type of neoVI device that the structure represents. The currently supported types are :  NEODEVICE_BLUE = 1 NEODEVICE_DW_VCAN = 4 NEODEVICE_FIRE = 8 NEODEVICE_VCAN3 = 16 NEODEVICE_YELLOW = 32 NEODEVICE_RED = 64 NEODEVICE_ECU = 128 NEODEVICE_IEVB = 256 NEODEVICE_PENDANT = 512 NEODEVICE_ALL = &HFFFFFFF
int Handle	The device handle used by the API for opening a neoVI device
int NumberOfClients	Reserved for future use
int SerialNumber	Serial number of the neoVI device
int MaxAllowClients	Reserved for future use



**stAPIFirmwareInfo Structure**[C/C++ declare](#) - [VB declare](#) - [VB.NET declare](#) - [C# declare](#) - [Remarks](#)**This structure continues information regarding the firmware version of a neoVI device.****C/C++ Declare**

```
struct stAPIFirmwareInfo
{
    int iType;
    // Date and Time (2nd generation neoVI only. See 2nd Generation neoVI Devices)
    int iMainFirmDateDay;
    int iMainFirmDateMonth;
    int iMainFirmDateYear;
    int iMainFirmDateHour;
    int iMainFirmDateMin;
    int iMainFirmDateSecond;
    int iMainFirmChkSum;

    // Version data (3rd generation neoVI only. See 3rd Generation neoVI Devices)
    unsigned char iAppMajor;
    unsigned char iAppMinor;
    unsigned char iManufactureDay;
    unsigned char iManufactureMonth;
    unsigned short iManufactureYear;
    unsigned char iBoardRevMajor;
    unsigned char iBoardRevMinor;
    unsigned char iBootLoaderVersionMajor;
    unsigned char iBootLoaderVersionMinor;
};

};
```

**Visual Basic Declares**

```
Public Type stAPIFirmwareInfo
    iType As Int32
    '// Date and Time (2nd generation neoVI only. See 2nd Generation neoVI Devices)
    iMainFirmDateDay As Long
    iMainFirmDateMonth As Long
    iMainFirmDateYear As Long
    iMainFirmDateHour As Long
```

```
iMainFirmDateMin As Long  
iMainFirmDateSecond As Long  
iMainFirmChkSum As Long  
  
'// Version data (3rd generation neoVI only. See 3rd Generation neoVI Devices)  
iAppMajor As Byte  
iAppMinor As Byte  
iManufactureDay As Byte  
iManufactureMonth As Byte  
iManufactureYear As Integer  
iBoardRevMajor As Byte  
iBoardRevMinor As Byte  
iBootLoaderVersionMajor As Byte  
iBootLoaderVersionMinor As Byte  
End Type
```

## Visual Basic .NET Declares

```
Public Structure stAPIFirmwareInfo  
    Dim iType As Int32  
    '// Date and Time (2nd generation neoVI only. See 2nd Generation neoVI Devices)  
    Dim iMainFirmDateDay As Int32  
    Dim iMainFirmDateMonth As Int32  
    Dim iMainFirmDateYear As Int32  
    Dim iMainFirmDateHour As Int32  
    Dim iMainFirmDateMin As Int32  
    Dim iMainFirmDateSecond As Int32  
    Dim iMainFirmChkSum As Int32  
  
    '// Version data (3rd generation neoVI only. See 3rd Generation neoVI Devices)  
    Dim iAppMajor As Byte  
    Dim iAppMinor As Byte  
    Dim iManufactureDay As Byte  
    Dim iManufactureMonth As Byte  
    Dim iManufactureYear As Int16  
    Dim iBoardRevMajor As Byte  
    Dim iBoardRevMinor As Byte  
    Dim iBootLoaderVersionMajor As Byte  
    Dim iBootLoaderVersionMinor As Byte  
End Structure
```

## C# Declares

```
[StructLayout(LayoutKind.Sequential)]
public struct stAPIFirmwareInfo
{
    public int iType;
    // Date and Time (2nd generation neoVI only. See 2nd Generation neoVI Devices)
    public int iMainFirmDateDay;
    public int iMainFirmDateMonth;
    public int iMainFirmDateYear;
    public int iMainFirmDateHour;
    public int iMainFirmDateMin;
    public int iMainFirmDateSecond;
    public int iMainFirmChkSum;

    // Version data (3rd generation neoVI only. See 3rd Generation neoVI Devices)
    public unsigned char iAppMajor;
    public unsigned char iAppMinor;
    public unsigned char iManufactureDay;
    public unsigned char iManufactureMonth;
    public unsigned short iManufactureYear;
    public unsigned char iBoardRevMajor;
    public unsigned char iBoardRevMinor;
    public unsigned char iBootLoaderVersionMajor;
    public unsigned char iBootLoaderVersionMinor;
}
```

## Remarks

### Structure Elements

Item	Description
int iType	Indicates the generation of hardware: 2 = 2nd generation. See <a href="#">2nd Generation neoVI Devices</a> 3 = 3rd generation. See <a href="#">3rd Generation neoVI Devices</a>
int iMainFirmDateDay	1 - 31 firmware day
int iMainFirmDateMonth	1 - 12 firmware month
int iMainFirmDateYear	4 digit year (i.e. 2008) firmware year

<code>int iMainFirmDateHour</code>	0 - 23 firmware hour
<code>int iMainFirmDateMin</code>	0 - 59 firmware minutes
<code>int iMainFirmDateSecond</code>	0 - 59 firmware seconds
<code>int iMainFirmChkSum</code>	Firmware checksum
<code>unsigned char iAppMajor</code>	Application major version (3rd generation neoVI only)
<code>unsigned char iAppMinor</code>	Application minor version (3rd generation neoVI only)
<code>unsigned char iManufactureDay</code>	1 - 31 Manufacture day (3rd generation neoVI only)
<code>unsigned char iManufactureMonth</code>	1 - 12 Manufacture month (3rd generation neoVI only)
<code>unsigned short iManufactureYear</code>	4 digit year (i.e. 2008) manufacture year (3rd generation neoVI only)
<code>unsigned char iBoardRevMajor</code>	Board revision major (3rd generation neoVI only)
<code>unsigned char iBoardRevMinor</code>	Board revision minor (3rd generation neoVI only)
<code>unsigned char iBootLoaderVersionMajor</code>	Bootloader version major (3rd generation neoVI only)
<code>unsigned char iBootLoaderVersionMinor</code>	Bootloader version minor (3rd generation neoVI only)

---

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, March 29, 2010*

## intrepidCS Development FAQ's

This section covers various issues that will be encountered when writing applications using the intrepidcs API. Click on the question for more information.

- [How do I detect and handle disconnects?](#)
- [How do I set parameters on a neoVI device?](#)
- [How do I open more than one channel on a single piece of hardware?](#)
- [How do I communicate on LIN?](#)
- [How do I send a Extended Frame or a High Voltage Wakeup or ISO9141/KW2K Init?](#)
- [How do I use CoreMini Scripts in an application?](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, January 12, 2011*

## How do I detect and handle disconnects?

If the neoVI is removed from the computer or stops responding, most API calls will generate a disconnect error. If you receive a return value from an API call that indicates an error occurred, you must call the [GetLastAPIError](#) method. If the error is NEOVI\_ERROR\_DLL\_NEOVI\_NO\_RESPONSE then the neoVI has either been unplugged or is not responding.

To recover from this condition follow these steps:

- Plug the neoVI back in. Or in the case of the neoVI not responding, remove and then plug back in.
- Call the [ClosePort](#) method.
- Call the [OpenNeoDevice](#) method using the same parameters are before.

The following API functions will generate the NEOVI\_ERROR\_DLL\_NEOVI\_NO\_RESPONSE error:

<a href="#">GetMessages</a>
<a href="#">TxMessages</a>
<a href="#">WaitForRxMessagesWithTimeOut</a>
<a href="#">GetTimeStampForMsg</a>
<a href="#">EnableNetworkCom</a>
<a href="#">GetConfiguration</a>
<a href="#">SendConfiguration</a>
<a href="#">GetFireSettings</a>
<a href="#">SetFireSettings</a>
<a href="#">GetVCAN3Settings</a>
<a href="#">SetVCAN3Settings</a>
<a href="#">SetBitRate</a>
<a href="#">GetDeviceParameters</a>
<a href="#">SetDeviceParameters</a>
<a href="#">ScriptLoad</a>
<a href="#">ScriptStart</a>
<a href="#">ScriptStop</a>
<a href="#">ScriptClear</a>
<a href="#">ScriptStartFBlock</a>
<a href="#">ScriptStopFBlock</a>

<a href="#">ScriptGetFBlockStatus</a>
<a href="#">ScriptGetScriptStatus</a>
<a href="#">ScriptReadAppSignal</a>
<a href="#">ScriptWriteAppSignal</a>
<a href="#">ScriptReadRxMessage</a>
<a href="#">ScriptReadTxMessage</a>
<a href="#">ScriptWriteRxMessage</a>
<a href="#">ScriptWriteTxMessage</a>
<a href="#">ScriptReadISO15765_2_TxMessage</a>
<a href="#">ScriptWriteISO15765_2_TxMessage</a>

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, November 18, 2008*

## How do I set parameters on a neoVI device?

### **3G Devices**

- You can use a data structure to get and set all the parameters all at once for a specific device type. These functions are:

<b>ValueCAN3</b> 	<a href="#">GetVCAN3Settings</a>	<a href="#">SetVCAN3Settings</a>	see <a href="#">SVCAN3Settings</a> data structure
<b>neoVI Fire</b> 	<a href="#">GetFireSettings</a>	<a href="#">SetFireSettings</a>	see <a href="#">SFireSettings</a> data structure

- To set individual parameters you can use the [GetDeviceParameters](#) and [SetDeviceParameters](#) methods. These methods let you build a string of specific parameters to get or set without using a device specific structure. You can use the same parameter strings without regard to the type of device you are connected to. For instance, the string for setting the baudrate on the first CAN channel on a ValueCAN3 and a neoVI Fire are the same.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, November 19, 2008*

**How do I open more than one channel on a single piece of hardware? - intrepidcs API**

When you call [OpenNeoDevice](#) or [OpenPortEx](#) to connect to the device, all the enabled channels on the device are active. Nothing special needs to be done "open" another channel. Calling [GetMessages](#) will get messages on all enabled channels. You can tell which network each message is from by looking at the [NetworkID in the message structure](#) for that message. When transmitting a message using [TxMessage](#), the third parameter is the [Network ID](#) to send the message out on.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, December 24, 2010*

## How do I communicate on LIN - intrepidcs API

The very first step to working with LIN is to make sure it is enabled on your hardware device. This is done in neoVI Explorer or neoVI 3G Explorer. Please see the documentation for the device to be used on how to do this.

There are a few ways to use LIN, so it is important to know how you plan on working with LIN. Is the application going to act as a Master, Slave, or just Spy on the bus. If you plan to just spy on the bus, there isn't any special setup.

The first step to acting as a Master is to enable the Master resistor in the hardware. This is done in neoVI 3G Explorer under the desired LIN channel. This is not needed when operating as a Slave. Below is the setup for a simple Master request with no slave data. Note that the J1850 Spy Message structure is used and the LIN protected ID is in the first data byte.

### Master frame no slave Data

```
//Use J1850 Message structure
//This breaks the ARBID into 4 bytes
icssSpyMessageJ1850 stJMsg;
long lNetworkID;

//Set the Network ID
lNetworkID = NETID_FIRE_LIN1; //16 for LIN1
//Set the LIN ID, use the Protected ID
stJMsg.Header[0] = 0xf0;
//Set the Number of Databytes to 0
stJMsg.NumberBytesData = 0;

//Set the Init Message flag. This tells the
//Hardware that this is a Master frame
stJMsg.StatusBitField=SPY_STATUS_INIT_MESSAGE;
stJMsg.StatusBitField2=0;
//Set the number of Header bytes to 1
stJMsg.NumberBytesHeader = 1;

//Send the message
lResult = icsneoTxMessages(m_hObject, (icssSpyMessage * )&stJMsg, lNetworkID, 1);
```

### Master frame with Slave data

```
//Use J1850 Message structure
```

```
//This breaks the ARBID into 4 bytes
icsSpyMessageJ1850 stJMsg;
long lNetworkID;

//Set the Network ID
lNetworkID = NETID_FIRE_LIN1; //16 for LIN1
//Set the LIN ID, use the Protected ID
stJMsg.Header[0] = 0xf0;

//Set the Number of Databytes to 3
//The first 2 data bytes go in the Header section.
stJMsg.NumberBytesData = 3;

stJMsg.Header[1] = 0x11;
stJMsg.Header[2] = 0x22;
// for all the data bytes
stJMsg.Data[0] = 0x33;
stJMsg.Data[1] = 0x44;
stJMsg.Data[2] = 0x55;
stJMsg.Data[3] = 0x66;
stJMsg.Data[4] = 0x77;
stJMsg.Data[5] = 0x88;
//Checksum goes in the last byte
//This needs to be calculated to what your LIN setup uses
stJMsg.Data[6] = 0x99;

//Set the Init Message flag. This tells the
//Hardware that this is a Master frame
stJMsg.StatusBitField=SPY_STATUS_INIT_MESSAGE;
stJMsg.StatusBitField2=0;
//Set the number of Header bytes to 1
stJMsg.NumberBytesHeader = 1;

//Send the message
lResult = icsneoTxMessages(m_hObject, (icsSpyMessage * )&stJMsg, lNetworkID, 1);
```

Sending a Slave message is just like sending a Master frame with Slave data. The difference is that the SPY\_STATUS\_INIT\_MESSAGE bit is not set in the StatusBitField. This bit tells the hardware if the message is a Master frame or a slave frame. When you send a Slave message, it is sent and saved in the hardware and is sent when a Master request is received. Once the request is received, then the Slave data goes out appended to the Master Frame. The same slave data will go out with every Master request, until a new Slave message with updated data is sent from the application to the hardware using the TxMessage function.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, December 24, 2010*

**How do I send a Extended Frame or a High Voltage Wakeup or ISO9141/KW2K Init? - intrepidcs API**

Regardless if you are trying to send an Extended Frame, High Voltage Wakeup, or ISO9141/KW2K init, the key is the [Status Bitfield](#). The [Status Bitfield](#) holds extra information for the message. Each bit of this parameter is a toggle for a different parameter. To send a High Voltage Wakeup, you would set the SPY\_STATUS2\_HIGH\_VOLTAGE bit for your message. For an extended address you would set the SPY\_STATUS\_XTD\_FRAME for the message you want to have an extended address. SPY\_STATUS\_INIT\_MESSAGE is set to send an initialization waveform with a ISO9141 or Keyword 2000 message.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, December 24, 2010*

## Application Note: CoreMini Script Interface Example

### Overview

CoreMini can be a big help when writing an application. CoreMini can get better timing than a PC can give and will let you configure more complicated tasks in the hardware instead of in your code. This example uses J1979 and requests RPM over HS CAN. The user interface shows the returned value. In this example TxMessage and GetMessages are never called. The RPM value is requested from the CoreMini Script inside the hardware. The application created will put a CoreMini inside the hardware, start the script, and read the RPM Value from the script.

This example does require some knowledge of Vehicle Spy 3 and the example is written in VB.NET.

### Parts of API used

To connect to the hardware

[FindNeoDevices](#)

[OpenNeoDevice](#)

[ClosePort](#)

To load and Run the CoreMini Script

[ScriptLoad](#)

[ScriptStart](#)

[ScriptStartFBlock](#)

[ScriptStop](#)

[ScriptClear](#)

To read the data back

[ScriptReadAppSignal](#)

Vehicle Spy 3 Features used

Message Editor

Application Signals

Script Function Block

### Hardware needed

---

neoVI **FIRE**, neoVI **RED**, neoVI **Yellow**, or ValueCAN

### Files for this example

[VBCoreMiniAppExample.zip](#)

#### Important Note:

When working with CoreMini it is important to make sure that the CoreMini Script, icsneo40.dll, and firmware all match in version. This is simpler than it sounds. The firmware in a neoVI or ValueCAN will automatically update to match software. When the CoreMini Script is created, make sure to use the same version of the Vehicle Spy 3 as the version of icsneo40.dll you intended to use with your hardware. The version numbers for these 2 components will match.

[Previous](#) | [Next](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.** [www.intrepidcs.com](http://www.intrepidcs.com)

Last Update: Wednesday, January 12, 2011

## Application Note: CoreMini Script Interface Example - Part 1: Create the CoreMini Script

**1. Open Vehicle Spy:** The first step, as with any Vehicle Spy project, is to open Vehicle Spy and Log in. In the [project zip file](#), a completed VS3 file can be found. If you are new to Vehicle Spy, this could help. This example will run through the steps to build this example file.

**2. Build Messages:** The next step is to build the Request and Response message. This is done in the Messages Editor found under Spy Networks. Add a Receive and a Transmit message. Use the following parameters for the two messages.

Tx Request:

ArbID = 7E0 Data = 02 01 0C 55 55 55 55 55

Rx Response

ArbID = 7E8 Data = XX 41 0C

RPM signal is in Bytes 4 and 5. The scaling is {Raw Value}\*0.25+0

**3. Application Signal:** Now an application signal is needed. Application signals work like variables in C or VB. Open up "Scripting and Automation" and select "Application signals". Create a new one and call it "EngRPM"

**4. Function block Script:** Function block scripts works like code. Open up "Scripting and Automation" and select "Function blocks". Figure one shows an example script. Set the Start type to Manual. For more help on building scripts, please see the help for Vehicle Spy 3.

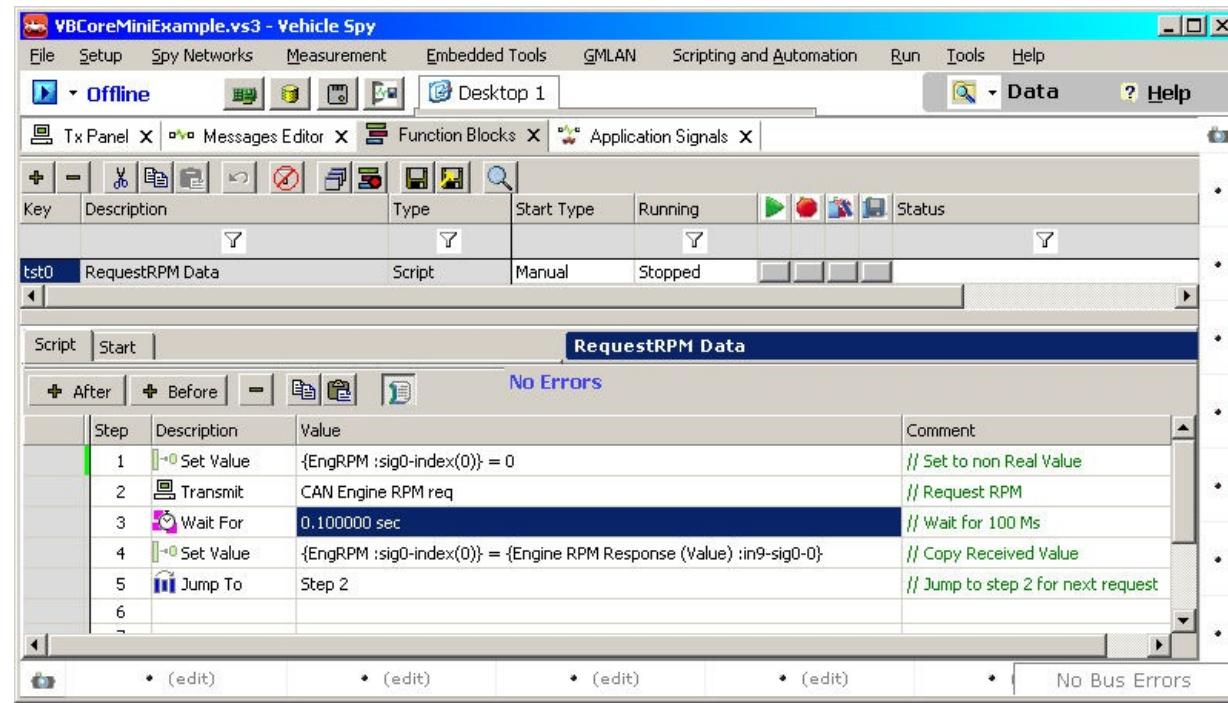


Figure 1: Function block Script

**4. Create CoreMini Script:** To make the file that will be loaded into the hardware, click on Tools, Utilities, then CoreMini Console. When this opens, it will compile and generate a "cmvspy.vs3cmb" file. This file is saved in your Data Directory folder (click on Data in the upper right of Vehicle Spy to take you right there). The CoreMini console can be closed.

[Previous](#) | [Next](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Update: Wednesday, January 12, 2011

### Application Note: CoreMini Script Interface Example - Part 2: Putting cmvspy.vs3cmb to use

**1. Connect to the hardware:** In application that is being created, the fist thing to do is to find and connect to the hardware. This is done the same was as with any project using the DLL: [FindNeoDevices](#) to find the hardware, [OpenNeoDevice](#) to connect to it, and when done [ClosePort](#) to disconnect.

**2. What to add to your code:** Now to load the CoreMini Script. The [ScriptLoad](#) command is used to load a CoreMini script. The cmvspy.vs3cmb can be read into an array and loaded using the [ScriptLoad](#) command. If you are writing your application in C or C++, Vehicle Spy generates 2 header files, cmvspy.h and cmvspy.vs3cmb.h. The 2 header files define the cmvspy.vs3cmb array and defines other parameters for the script. In the world of .NET, **IO.FileStream** works well. This is how the file is loaded in the VB.NET example.

Next, [ScriptStart](#) is called to start the script that was loaded. [ScriptStartFBlock](#) is then called to start the function block. This will request RPM and copy it to an application signal.

When finished, [ScriptStop](#) and [ScriptClear](#) can be called to remove the script from the hardware and clear the script from the hardware.

Below are example function calls. For more help, please see the help topic for that command or the included Example.

```
'-----Load Script
iResult = icsneoScriptLoad(m_hObject, CoreMiniData(0), iLength, SCRIPT_LOCATION_FLASH_MEM)
'-----Start the Script
iResult = icsneoScriptStart(m_hObject, SCRIPT_LOCATION_FLASH_MEM)
'-----Start Function block
iResult = icsneoScriptStartFBlock(m_hObject, Convert.ToInt32(0))

'-----Read Data From the script
iResult = icsneoScriptReadAppSignal(m_hObject, Convert.ToInt32(0), ValueToSet)

'-----Stop Function block
iResult = icsneoScriptStopFBlock(m_hObject, Convert.ToInt32(0))
'-----Clear the script
iResult = icsneoScriptClear(m_hObject, SCRIPT_LOCATION_FLASH_MEM)
```

**3. Read the data:** [ScriptReadAppSignal](#) can be called to request the value of the application signal. With the data we can do whatever is needed with it.

[Previous](#) | [Next](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.** [www.intrepidcs.com](http://www.intrepidcs.com)

Last Update: Wednesday, January 12, 2011

**Application Note: CoreMini Script Interface Example - Part 3: Conclusion**

**Conclusion:** CoreMini scripts are fairly easy to setup. Once loaded in the hardware, complicated tasks in code can be done by calling a function or two. Some uses could be, sending tester present, sending normal mode messages at given rates, request diagnostic information, send/receive ISO15765-2 frames, or automate messaging.

[Previous](#) | [Next](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.** [www.intrepidcs.com](http://www.intrepidcs.com)

Last Update: Wednesday, January 12, 2011

## Raw Communications - Intrepidcs API

[Main](#)

### Overview

Normally a custom application will want to access the neoVI via the [neoVI DLL](#). This DLL was written only for Microsoft WIN32 applications. Since not all neoVI applications are for Windows we offer a RAW communications API.

### Basics

The Raw Communications API let you send raw byte RS232 or USB commands to send and receive data from neoVI. The bytes sent and received from neoVI are packaged into a [neoVI communications packet](#).

neoVI Green powers up in RS232 mode. If you want to switch to USB mode (or back to RS232 from USB) you must send an [interface switch command](#). neoVI blue/neoVI PRO will select USB whenever it detects power at the USB port. ValueCAN is always USB.

When you open communications to neoVI, you will see a message every 104.768 ms for neoVI Green (65.536 ms for ValueCAN and neoVI Blue). This message is the [device status message](#). You can use this message to assist time-stamping and to detect if neoVI was disconnected or reset.

### Vehicle Network Communications

Each network has a specific message format with in the [neoVI Raw Communications](#) packet format. There are separate formats for the [CAN networks](#), [J1850VPW/ISO/J1708](#), and the [J1850 PWM networks](#).

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Friday, April 8, 2005*

## **Handshaking, Starting, and Resetting the Value CAN/neoVI Blue device - intrepidcs API Main**

### **Handshaking with ValueCAN**

To enable proper communications with the ValueCAN/neoVI Blue device, Hardware handshaking is required. This type of handshaking uses the RTS and CTS pins to allow a high speed communications between the ValueCAN device and the computer. Hardware handshaking should be enabled after starting or restarting communications with the device.

### **Starting ValueCAN/neoVI BLUE**

The procedure for starting a Value CAN or neoVI Blue is shown below in a step by step fashion.

1. Open the Comm Port that the device is associated with.
2. Clear the DTR Pin on the Comm Port.
3. Set the RTS Pin on the Comm Port.
4. Set the DTR Pin on the Comm Port
5. Wait 250ms for device to initialize.

The ValueCAN or neoVI Blue should now be ready for communications. You must send an enable start communications message to start receiving network data.

### **Restarting ValueCAN/neoVI BLUE**

If the ValueCAN or neoVI Blue Device needs to be restarted, it can be done through code.

1. Clear the DTR Pin on the Comm Port.
2. Set the RTS Pin on the Comm Port.
3. Set the DTR Pin on the Comm Port
4. Wait 250ms for device to initialize.

The ValueCAN or neoVI Blue should now be ready for communications. You must send an enable start communications message to start receiving network data.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 24, 2003*

## Raw Communications Packet - Intrepidcs API

[Main](#)

### Overview

Raw communications in the intrepidcs API centers around the Raw Communication Packet (figure 1). This packet is between 4 and 18 bytes in length and has specific fields for identification and protection.

### The Communication Packet

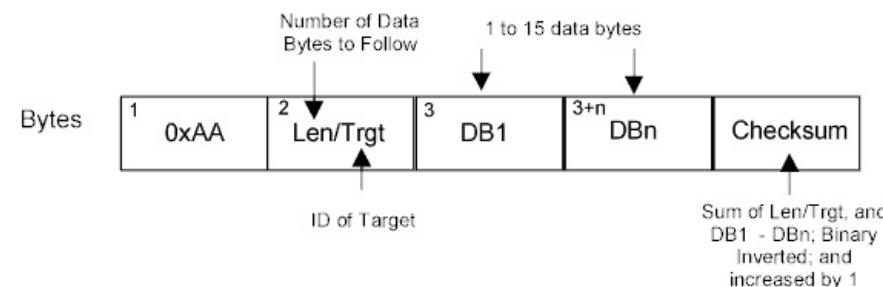
The first field of the communication packet always starts with a hexadecimal 0xAA (decimal 170).

The next byte is the Target ID/Length Byte. In this byte, the upper nibble (the length) denotes the number of data bytes to follow not including the checksum byte \*. The lower nibble of the Target ID/Length Byte includes the Target ID. The Target ID is the location for the packet on the device. For example, with the neoVI device, to send a vehicle network message on high speed CAN you would set the target id to NETID\_HSCAN.

Following the Target ID/Length byte is from 1 to 15 data bytes.

Finally, the final byte is the checksum byte. The checksum byte is a two complement of the sum of the data bytes not including the start byte or the checksum.

\* A Length of zero is not permitted but may be used in the future as an escape character



**Figure 1 - The Communications Packet consists of a start identifier, a length/target byte, 1 to 15 data bytes, and a checksum.**

Last Update: *Monday, June 17, 2002*

## Raw Device Status Message - Intrepidcs API

[Main](#) - [Compatibility](#) - [Example](#)

Raw Device Status message sends out this report every timestamp clock rollover. This is used for larger than 16 bit time-stamping and to detect a neoVI disconnection or reset.

The sequence of the message stream allows this message to act as a most significant roll over for time-stamping. Each rollover for a neoVI device indicates a ( NEOVI\_TIMESTAMP\_2 = 0.1048576 ) count. Each rollover for a neoVI Blue or ValueCAN device is a ( NEOVIPRO\_VCAN\_TIMESTAMP\_2 = 0.065536 ) count.

You can detect a disconnection by watch the SYNC\_COUNT. This is a modulus 255 counter that is incremented every time the message is sent. Your application can monitor this for consecutive numbers for connection detection.

The RESET\_STATUS should not change. If there was a device reset or a watchdog reset these bits will change.

The message format is shown below in Table 1. The 0xAA and the check sum of the [Raw Communications Packet](#) are left out of the table.

**Table 1 - Message Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4
NETID_DEVICE	0x01	SYNC_COUNT	RESET_STATUS

## Compatibility

Hardware	Present	Notes
neoVI	Yes	Message period is 0.1048576 seconds
ValueCAN	Yes	Message period is 0.065536 seconds
neoVI PRO/neoVI Blue	Yes	Message period is 0.065536 seconds

## Example

Message : "0xAA - 0x30 - 0x01 - 0xEE - 0x0C - 0xD5"

0xAA = Start Byte

0x30 = Length / Target Byte : Length is 3 Bytes To Follow and Target is 0 which is the Device

0x01 = Identifies the status message within the device network

0xEE = Current Sync Count

0x0C = Reset Status

0xD5 = Checksum

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, May 22, 2003*

**Interface Switch Command - Intrepidcs API**[Main](#)

When the neoVI Green device powers up it enters RS232 mode for neoVI to host communications. Host to neoVI communications are always active on both RS232 and USB. This allows either port usable for sending a command to change between RS232 and USB for neoVI to host communications. This command is called the interface switch command.

neoVI Blue will always automatically select USB when the USB is plugged in. Therefore, this command is not required

**Table 1 - neoVI Interface Switch Command**

BYTE 1	BYTE 2	BYTE 3
NETID_MAIN51	0x02	COMM MODE: USB=0x01 RS232=0x00

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**No Data Message - Intrepidcs API**[Main](#)

This message is sent back when a host requests USB data and there is no data available. This message prevents the USB IN transaction from blocking when there is no data. This message is used for neoVI Green only.

**Table 1 - neoVI NO DATA message**

BYTE 1	BYTE 2
NETID_MAIN51 0x0B	MAIN_51_BULKIN_NODATA 0x04

**Example**

This message is a constant four byte message **0xAA - 0x1B - 0x04 - 0xE1**

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Monday, October 07, 2002*

**Raw CAN Overview - Intrepidcs API**[Main](#)

To use the Raw interface for CAN you must be aware of the neoVI device commands which affect CAN. Table 1 lists those commands. Please see the command summary for a concise list of commands.

**Table 1 - CAN related Commands**

Command
<a href="#">CAN Rx Packet</a>
<a href="#">CAN Tx Packet</a>
<a href="#">CAN Rx Buffer Overflow</a>
<a href="#">CAN Error State Message</a>
<a href="#">CAN Tx Complete</a>

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**Raw CAN Receive Messages - Intrepidcs API**[Main](#)**Overview**

When neoVI receives a CAN message it assembles the message in a byte sequence. These formats are depicted in tables 1 and 2 below.

**Table 1 - Standard ID (SID) Frame Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTES 7-14
NETID_XX NETID OF CAN NETWORK:  NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN	TIMESTAMP MSB	TIMESTAMP LSB	STANDARD ID BITS 10:3	b7 b6 b5 b4 b3 b2 b1 b0 SID SID SID SRR IDE - EID EID Bit2 Bit1 Bit0 Bit0 SRR IDE - EID Bit17 Bit16	MS Nibble Reserved Bits  LS Nibble DLC	CAN Data bytes 0 - 8  Remote Frame Have No Data

**SRR** = Set to 1 when a remote frame  
**IDE** = Set to 0 when this is a standard frame.  
**EID Bit 17 and 16** = undefined for a standard ID message

**DLC** = Data Length Code  
 Reserved Bits = Undefined (Mask off to determine length of data)

**Table 2 - Extended ID (EID) Frame Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9-16
NETID_XX NETID OF CAN NETWORK:  NETID_HSCAN,	TIMESTAMP MSB	TIMESTAMP LSB	EXTENDED ID BITS 28:21	b7 b6 b5 b4 b3 b2 b1 b0 EID EID EID SRR IDE - EID EID Bit20 Bit19 Bit18 SRR IDE - EID EID Bit17 Bit16	EXTENDED ID BITS 15:8	EXTENDED ID BITS 7:0	MS Nibble b7 b6 b5 b4 - ERTR Rb1 Rb1 LS Nibble DLC	CAN Data bytes 0 - 8  Remote Frame Have No Data

**SRR** = Undefined for extended frame  
**IDE** = Set to 1 when this is an extended frame.  
**DLC** = Data Length Code

NETID_MSCAN,			<b>EID Bit 17 and 16</b> = undefined for a standard ID message		<b>ERTR</b> = Set to 1 when a remote frame	<b>Rb1</b> and <b>Rb0</b> = undefined	Have No Data
--------------	--	--	--	--	--	---------------------------------------	--------------

**C/C++ Examples:****Getting Standard Frame Arb ID**

```
uiArbID = (((unsigned int) bPacket[3]) << 3) + ((bPacket[4] & 0xE0) >> 5);
```

**Getting a Extended Frame Arb ID**

```
uiArbID = ( ((unsigned int) bPacket[3]) << 21) +
    (((unsigned int)bPacket[4]) & 0xE0) << 13) +
    (((unsigned int)bPacket[4]) & 0x03) << 16) +
    ((unsigned int)bPacket[5]) << 8) + bPacket[6];
```

**Determining if the message is an Extended or Standard ID**

```
// is this frame a standard or extended frame (Xtd indicated by bit 3 =1)
if (bPacket[4] & 0x8)
    { // Extended}
else
    { // Standard}
```

**Determining if the Standard Message is a Remote Frame**

```
// is this frame a remote one (indicated by bit 4)
if (bPacket[4] & 0x10)
    { // Standard ID Remote Frame}
else
    { // Regular Standard Frame }
```

**Determining if the Extended Message is a Remote Frame**

```
// is this frame a remote one (indicated by bit 6)
if (bPacket[7] & 0x40)
    { // Standard ID Remote Frame}
else
```

```
{ // Regular Standard Frame }
```

### Determining Number of Data Bytes for a Standard Frame

```
// Mask off upper nibble  
NumberBytesData = bPacket[5] & 0x0F;
```

### Determining Number of Data Bytes for an Extended Frame

```
// Mask off upper nibble  
NumberBytesData = bPacket[7] & 0x0F;
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Sunday, October 06, 2002*

**Raw CAN Transmit Messages - Intrepidcs API**[Main](#)

To transmit a message you must assemble a transmit message as shown in tables 1 and 2 below. This is a transmit request. When neoVI has successfully transmitted a message it will send a [transmit complete report](#). If you send messages faster than neoVI can place them on the bus you can cause a [Main51 Transmit FIFO](#) overflow to occur.

**Table 1 - Transmit Standard Arbitration ID (SID) Frame Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTES 7-14
NETID_XX NETID OF CAN NETWORK:  NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN	DESCRIPTION ID MSB	DESCRIPTION ID LSB	STANDARD ARB ID BITS 10:3	b7 b6 b5 b4 b3 b2 b1 b0 SID SID SID - IDE EID EID Bit2 Bit1 Bit0 - - Bit17 Bit16	MS Nibble LS Nibble b7 b6 b5 b4 - RTR - - DLC	CAN Data bytes 0 - 8 Remote Frame Have No Data

**IDE** = Set to 0 when this is a standard frame.  
**EID Bit 17 and 16** = undefined for a standard ID message

**DLC** = Data Length Code  
**RTR** = Remote Transmit Request set to 1 for a remote frame (no data)

**Table 2 - Extended Arbitration ID (EID) Frame Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9-16
NETID_XX NETID OF CAN NETWORK:  NETID_HSCAN,	TIMESTAMP MSB	TIMESTAMP LSB	EXTENDED ARB ID BITS 28:21	b7 b6 b5 b4 b3 b2 b1 b0 EID EID EID - IDE EID EID Bit20 Bit19 Bit18 - - Bit17 Bit16	EXTENDED ID BITS 15:8	EXTENDED ID BITS 7:0	MS Nibble LS Nibble b7 b6 b5 b4 - RTR - - DLC	CAN Data bytes 0 - 8 Remote Frame Have No Data

**IDE** = Set to 1 when this is an

**DLC** = Data Length Code

NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN			20:21	extended frame.			<b>RTR</b> = Remote Transmit Request set to 1 for a remote frame (no data)	None No Data
---	--	--	-------	-----------------	--	--	---	--------------------

**C/C++ Examples:****Changing the Arb ID into Bytes for a Standard Frame**

```
bData[3] = uiArbID >> 3; // standard id high ( shift left 3 bits )
bData[4] = (uiArbID & 7) << 5; // mask off upper five bits
```

**Changing the Arb ID into Bytes for an Extended Frame**

```
// 4 header bytes with the standard ID first
bData[3] = uiArbID >> 21;
bData[4] = (((uiArbID & 0x001C0000) >> 13) & 0xFF) + (((uiArbID & 0x00030000) >> 16) & 0xFF) |
8;
bData[5] = (uiArbID >> 8) & 0xFF; // extended id high
bData[6] = (uiArbID & 0xFF); // extended id id least significant bits
```

**Raw CAN Rx Buffer Overflow - Intrepidcs API**[Main](#)

Each CAN Controller in neoVI has an Rx FIFO buffer to read CAN messages. If rx buffer is full when a new CAN message arrives a CAN Rx Buffer overflow occurs. This normally will not happen but could result if the neoVI microcontroller does not reads the buffer fast enough.

**Table 1 - CAN Rx Buffer Overflow Message Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
NETID_DEVICE	0x02	NETID_XX NETID_OF CAN NETWORK:  NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN	TIMESTAMP MSB	TIMESTAMP LSB

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**Raw CAN Error State Changed - Intrepidcs API**[Main](#)

If any CAN controller in neoVI indicates an error occurs neoVI will send the Error State Changed Message. The error state message will be sent to the PC anytime any of the error bits change.

**Table 1 - CAN Error State Changed Message Format**

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
NETID_DEVICE	0x04	NETID_XX NETID_OF CAN NETWORK:  NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN	TIMESTAMP MSB	TIMESTAMP LSB	( <a href="#">MCP2510 Error Status Bitfield</a> )

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**Raw CAN Tx Complete - Intrepidcs API**[Main](#)

When neoVI detects a message has been transmitted successfully it will send a CAN Tx Complete report. This report includes the time the message was transmitted and the 14 bit description id.

**Table 1 - CAN TX Complete Message Format**

<b>BYTE 1</b>	<b>BYTE 2</b>	<b>BYTE 3</b>	<b>BYTE 4</b>	<b>BYTE 5</b>	<b>BYTE 6</b>	<b>BYTE 7</b>
NETID_DEVICE	0x03	NETID_XX NETID OF CAN NETWORK: NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN	TIMESTAMP MSB	TIMESTAMP LSB	DESCRIPTION ID MSB  Upper two bits are forced to zero	DESCRIPTION ID LSB

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

 Last Update: *Monday, October 07, 2002*

**Raw ISO/KW2K/LIN Transmit Packet - Intrepidcs API**[Main](#)**Overview**

To transmit a message you must assemble a transmit message as shown in tables 1 below. This is a transmit request. When neoVI has successfully transmitted a message it will send a transmit complete report. If you send messages faster than neoVI can place them on the bus you can cause a [Main51 Transmit FIFO](#) overflow to occur.

**LIN Bus**

You do not need to include the synchronization Break or the synchronization waveform - the hardware appends this for you. Also, to send a master message you must set the Init Flag (B7 of byte 2).

**Table 1 - ISO/KW2K/LIN transmit Frame Format**

<b>BYTE 1</b>	<b>BYTE 2</b>	<b>BYTE 3</b>	<b>BYTE 4</b>	<b>BYTES 5-16</b>																				
NETID_ISO	<table border="1"> <tr> <td colspan="2"><b>MS nibble</b></td> <td colspan="2"><b>LS Nibble</b></td> </tr> <tr> <td colspan="2"><b>Bitfield</b></td> <td colspan="2"></td> </tr> <tr> <td>b7</td><td>b6</td><td>b5</td><td>b4</td> </tr> <tr> <td>Init</td><td>Long</td><td>0</td><td>0</td> </tr> <tr> <td>Flag</td><td>Msg</td><td></td><td></td> </tr> </table>	<b>MS nibble</b>		<b>LS Nibble</b>		<b>Bitfield</b>				b7	b6	b5	b4	Init	Long	0	0	Flag	Msg			Length from length/target byte - 1	DESCRIPTION ID MSB	DESCRIPTION ID LSB ISO/LIN/KW2K Data bytes 1 - 12 *
<b>MS nibble</b>		<b>LS Nibble</b>																						
<b>Bitfield</b>																								
b7	b6	b5	b4																					
Init	Long	0	0																					
Flag	Msg																							

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, May 22, 2003*

**Raw ISO/KW2K/LIN Tx Complete or Error Status - Intrepidcs API**[Main](#)**Overview**

Besides received messages, the ISO/KW2K/LIN channel will forward additional event information to the host. These events includes transmit completion and errors. Table 1 and Table 2 below details the received frame format.

There are many errors possible on the LIN bus. Please see the neoVI online help for descriptions of the possible errors listed in byte 5 in Table 1 below.

**Table 1 - LIN Error Receive Frame Format**

<b>BYTE 1</b>	<b>BYTE 2</b>	<b>BYTE 3</b>	<b>BYTE 4</b>	<b>BYTE 5</b>	<b>Byte 6</b>	<b>Byte 7</b>
				00 = TRANSMIT COMPLETE		
NETID_ISO	TIMESTAMP MSB	TIMESTAMP LSB	See Table	ISO_LIN_SYNC_BRK_ERR ISO_LIN_SYNC_WAV_ERR ISO_LIN_MSG_ID_PRTY ISO_LIN_CHKSUM_ERR ISO_LIN_TFMAX_ERR ISO_LIN_BIT_ERROR ISO_LIN_SYNC_LEN_ERR	For Transmit Complete:	For Transmit Complete:  DESCRIPTION ID MSB DESCRIPTION ID LSB

**Table 2 - Byte 4 for a Receive Frame**

<b>b7</b>	<b>b6</b>	<b>b5</b>	<b>b4</b>		<b>Lower Nibble</b>
0	1 Must be one for status message	0 = No Overflow 1 = Rx Overflow	Reserved		Number of Bytes to follow

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, May 22, 2003*

**Raw ISO/KW2K/LIN Receive Packet - Intrepidcs API**[Main](#)**Overview**

The ISO/KW2K/LIN channel will forward all received messages to the host. Table 1 and Table 2 below details the received frame format.

When receiving a long message ( $> 12$  bytes) the channel will set the long message bit. This bit means that the next received frame is part of the current message. The end of the message is indicated by the last consecutive frame with the long message bit clear.

The channel maintains a two message receive buffer. If the device does not empty this buffer before it can be read a overflow will result. If the overflow bit is set then one or more messages were lost.

**Table 1 - ISO/KW2K/LIN Receive Frame Format**

<b>BYTE 1</b>	<b>BYTE 2</b>	<b>BYTE 3</b>	<b>BYTE 4</b>	<b>BYTES 5-16</b>
NETID_ISO	TIMESTAMP MSB	TIMESTAMP LSB	See Table 2 below	Message Bytes including ISO/LIN checksum if present

**Table 2 - Byte 4 for a Receive Frame**

<b>b7</b>	<b>b6</b>	<b>b5</b>	<b>b4</b>	
1 = Long Message (More Bytes To Follow in consecutive messages)  0 = Single	0  Must be zero for rx frame	0 = No Overflow  1 = Rx Overflow	Reserved	<b>Lower Nibble</b>  Number of Bytes of frame including checksum if present

Message or End of Long Message						
-----------------------------------	--	--	--	--	--	--

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, May 22, 2003*

---

## Raw J1850 PWM - Intrepidcs API

[Main](#)

### Overview

Raw communications pa.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Monday, June 17, 2002*

**Raw neoVI MainPic Buffer Overflow - Intrepidcs API**  
[Main](#)

There is a FIFO in the neoVI which handles communications from the MainPic to Main51 microcontrollers. If this FIFO overflows the MainPic Buffer overflow message will be generated.

**Table 1 - neoVI MainPic Buffer Overflow Message Format**

BYTE 1	BYTE 2
NETID_DEVICE	0x05

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**Main 51 Tx FIFO Overflow - Intrepidcs API**[Main](#)

Each network in neoVI has a software transmit FIFO to manage the speed difference between the time a command arrives on USB or RS232 and when it can be processed. If the command cannot be processed or transmitted as quickly as new commands arrive a Tx FIFO message will be generated. The format of the message is shown in table 1 below.

**Table 1 - Main51 Tx FIFO Overflow Message Format**

BYTE 1	BYTE 2	BYTE 3
NETID_MAIN51	0x03	NETID OF NETWORK:  NETID_DEVICE, NETID_HSCAN, NETID_MSCAN, NETID_SWCAN, NETID_LSFTCAN, NETID_FORDSCP, NETID_J1708, NETID_AUX, NETID_JVPW, NETID_ISO, NETID_ISOPIC

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**Last Update: *Monday, October 07, 2002*

**Raw Command Summary - Intrepidcs API**[Main](#)**Received Commands: Device**

Description	Summary
<a href="#">Device Status</a>	Regular Interval Output from the device which contains reset and time stamp information
<a href="#">CAN Rx Buffer Overflow</a>	Occurs when a CAN controller receive buffer is not serviced fast enough for consecutive CAN messages
<a href="#">CAN Tx Complete Msg</a>	Occurs when a CAN message setup to transmit is sent successfully
<a href="#">CAN Error Status Changed</a>	Occurs when the error state changes in any of the CAN controllers
<a href="#">neoVI MainPic Buffer Overflow</a>	Occurs when the FIFO communications from the MainPic to the main51 overflows
<a href="#">neoVI Device Command Complete</a>	Occurs when specific device commands are completed
<a href="#">SCP Transmit Complete Report</a>	Occurs when a Ford SCP is properly transmitted by the LBCC
<a href="#">SCP Error Exception Report</a>	Occurs when a LBCC generates an exception

**Received Commands: Main 51**

Description	Summary
Main 51 Rx Buffer Overflow	Occurs when the FIFO communications from the main51 to the PC overflows
Start Command	Used in RS232 communications to detect if a neoVI is connected to the serial port
<a href="#">Main 51 Tx FIFO Overflow</a>	Occurs when a transmit FIFO in the Main51 overflows
Read EEPROM Report	Returns the EEPROM information that was requested with the read EEPROM command
Write EEPROM Done	
XX	
USB No Data Response	This is returned from a USB bulk read if the read FIFO is empty
Main 51 Device TX FIFO Error	

**Received Commands: CAN Networks**

Description	Summary
<a href="#">CAN Received Message</a>	This occurs when a valid message is received from the CAN network

**Transmitted Commands: CAN Networks**

Description	Summary
<a href="#">CAN Transmit Message</a>	This command is sent to transmit a message on a CAN network

**Received Commands: Ford SCP/J1850 PWM**

Description	Summary
J1850 PWM Rx Msg	

**Received Commands: J1850 VPW**

Description	Summary
J1850 VPW Rx Msg	

**Received Commands: ISO/Keyword/LIN**

Description	Summary
<a href="#">ISO Rx Msg</a>	This occurs when a valid message occurs on the ISO/KW2K/LIN Bus
<a href="#">ISO Tx Complete or Error Status</a>	This occurs when a transmit message is sent or a error occurs on the ISO/KW2K/LIN bus

**Transmitted Commands: ISO/Keyword/LIN**

Description	Summary
<a href="#">ISO Tx Msg</a>	This command is sent to transmit a message on the ISO/KW2K/LIN Bus

**Received Commands: J1708**

Description	Summary
J1708 Rx Msg	

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, May 22, 2003*

## **neoVI Green USB interface - intrepidcs API**

### Main

#### **Overview**

This document explains the raw USB interface for neoVI Green. This document contains all the information that is necessary to write USB drivers for neoVI Green on any operating system. neoVI Green is compatible with both USB 1.1 and USB 2.

#### **USB Interface**

The USB interface uses bulk transfers. Basically, the driver should poll the IN endpoint for data after setting alternate setting 1. To send data you will send OUT data to the out endpoint. This basic interface works just like a serial port.

- \* neoVI has the default control end point
- \* neoVI has one bulk in and one bulk out endpoint on alternate setting one. Alternate setting 0 has no endpoints.
- \* neoVI has code in device (does not need firmware loading over USB)
- \* neoVI is self powered

#### **Bandwidth requirements**

The driver should insure performance of neoVI by including a USB Bulk request in the kernel mode of the driver. User mode bulk requests occurs in a user mode thread which can be stopped for dozens of milliseconds. This dozens of milliseconds is long enough to cause the 2K buffer in neoVI to overflow at high bandwidth CAN bus loads. The transmit portion is not speed critical and can therefore stay in user mode.

Assuming worst case 100 bytes per millisecond (all eight buses at 100%) that would give 2k fifo about 20 milliseconds to fill. Therefore, if you can guarantee that the FIFO is emptied every 5 milliseconds you should have a workable design.

#### **Use of driver**

The follow lists an example usage on WIN32.

#### **OPENING THE DRIVER**

- 1) The application calls open method in the DLL
- 2) The open method calls CreateFile using (neovi-XX). XX is the index of the USB device. The driver uses a mutex to protect against two programs opening the same XX id.
- 3) SetAlternate setting is called to chose the setting with bulk endpoints (see code at end of email)
- 4) The user mode thread is started which polls the bulk in endpoint
- 5) A user mode thread is started which monitors a fifo and send data out to the bulk out end point when data is present

#### **USER MODE RX THREAD**

- 1) Code starts a loop
- 2) The loop monitors a stop flag protected by a critical section
- 3) If not stop it calls device IOCONTROL IOCTL\_EZUSB\_BULK\_READ to get data with a buffer size of 2K
- 4) If neoVI doesn't have any data it will respond with a no data message (four byte sequence)
- 5) If neoVI has data it will fill the buffer
- 6) the thread sleeps for 4 ms sleep(4) and repeats loop if stop flag is not active

**USER MODE TX THREAD**

- 1) Code starts a loop
- 2) the code enters a critical section protecting the tx out buffer and get any txs data if present. it also reads a stop flag
- 3) if there is no data the thread sleeps
- 4) the code sends data using the send data method (code also below)
- 5) the code repeats if stop flag is not active

**CLOSING THE DRIVER**

- 1) both tx and rx threads are stopped using flags (they are forcefully stopped if flag method fails)
- 2) SetAlternateSetting is set to 0
- 3) CloseHandle is used to close the device
- 4) the mutex is used to free that port number

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, April 4, 2005*

## Unix-like Operating Systems - Support

Support for ValueCAN and neoVI Blue has been added to the ftdi\_sio kernel module in Linux 2.4 and 2.6. Simply load this module and connect either device via USB.

Support for ValueCAN has also been added to the FTDI driver in FreeBSD. neoVI Blue has not been tested at this time, but ought to work.

The current examples available are:

- [can\\_sniff](#), a command-line tool for monitoring HSCAN traffic.  
can\_sniff includes a library, libintrepidcs, for interfacing with the neoVI and ValueCAN.  
UNDOCUMENTED FEATURE: can\_sniff supports transmitting messages using this commandline option: -0 "CAN ID: d1 [d2 ... d8]". For example, to send the message "11 22 33 44" with ID 230, you would run: can\_sniff -0 "230: 11 22 33 44"
- [can\\_bitrate](#), a command-line tool for setting the CNF registers on the neoVI and ValueCAN.
- [IThinkICAN](#), the GUI tool for receiving and sending CAN messages on all of HSCAN, SWCAN, MSCAN, and LSFTCAN.  
IThinkICAN requires GTK+-2.4 or later.

These archives are in Bzip2 format - WinRAR should be able to extract them.

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, July 14 2005*

## J2534 Support - intrepidcs API

The icsneo40.dll implements the required imports as specified in J2534. The DLL will automatically register itself in the windows registry when loaded. This can be done by running neoVI Explorer, ValueCAN explorer, or Vehicle Spy once.

The DLL also supports the manufacturer specific IDS listed below. Also, it supports the single wire can SWCAN ids as specified in J2534-2 (SWCAN\_CAN and SWCAN\_ISO15765).

```
// manufacturer specific protocol ids
#define J2534_PROTOCOLID_ICS_NETID_HSCAN 0x10001
#define J2534_PROTOCOLID_ICS_NETID_MSCAN 0x10002
#define J2534_PROTOCOLID_ICS_NETID_SWCAN 0x10003
#define J2534_PROTOCOLID_ICS_NETID_LSFTCAN 0x10004
#define J2534_PROTOCOLID_ICS_NETID_FORDSCP 0x10005
#define J2534_PROTOCOLID_ICS_NETID_J1708 0x10006
#define J2534_PROTOCOLID_ICS_NETID_AUX 0x10007
#define J2534_PROTOCOLID_ICS_NETID_JVPW 0x10008
#define J2534_PROTOCOLID_ICS_NETID_ISO 0x10009
```

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, April 20, 2005*

## Vehicle Spy Text API

### Objective

The Text API purpose is to provide a simple text based command set for Vehicle Spy 3 to allow third party applications to take advantage of the power of Vehicle Spy without rewriting much code.

The Text API command set is text based so it can be easily used over many interfaces such as RS232, USB, Ethernet, or Wireless. It is also independent of the operating system or development environment of the host. The command set is similar to what may be found in a programmable instrument consisting of commands and responses.

You can easily write an object or function wrapper around the Text API. This will allow a more convenient use in professional languages such as C#, Java, Visual Basic, LabVIEW, or C/C++. The Text API could also exist as a simple macro language itself.

### Tips for Learning

The best way to learn the Text API is to experiment. The Text API terminal in Vehicle Spy 3 is a great tool for this purpose. Also, because most of the commands are based on the XML tags, you can just open the vs3 file in a text editor to see what properties can be manipulated via this extremely flexible API.

### Summary of Rules

- \* All commands start with the default root object.
- \* Commands are separated by a <CR>,<LF> or <CR><LF> combination.
- \* Commands are case insensitive. Arguments can be case sensitive if they are text based such as Description properties.
- \* All objects and properties have the same text as their XML element names. Properties can be changed by calling out the property and supplying an argument. Property values can be returned by using a question mark ? to query the property.

### The following rules are not yet supported

- \* There is special encoding to support non ASCII, extended characters, the % escape character, and the ; comment character. Where % will be followed with the Text number in four digit hex form. For example, %000D would equal CR, %000A would be <LF>. The four digit Text allows Unicode support for text arguments such as descriptions.
- \* The comment character is the semicolon. After the semicolon all characters are ignored until the next command.
- \* The Text API is available as UNICODE via the Vehicle Spy 3 DLL.

### Command and Queries

The basic communication consists of commands and queries. Commands set a property or execute a method. Queries request a property.

Commands and Queries will return one of two of possible responses: ok or er. "ok" means that the command completed successfully. "er"

means there was a problem with the command. The ok will have text following the command that indicates what command completed and any return values.

Events can be either asked for with a method or can appear asynchronously in the receive stream. You also must specify, or "register", which events you wish to receive.

## Syntax of Commands, Queries, and Events

Type	Syntax of Command	Syntax of Successful Response
Command	<code>methodname {arguments}</code>	ok <code>methodname</code>
Queries	<code>methodname?</code>	ok <code>methodname {propertyvalue}</code>

### Examples:

A successful start command

**Host:** Start  
**Vehicle Spy 3:** ok start

A unsuccessful root command

**Host:** Startasdf  
**Vehicle Spy 3:** er command not found:startasdf

A successful query

**Host:** AutoDetectHardware?  
**Vehicle Spy 3:** ok autodetecthardware 1

### example: LoadFile Method

**This command starts with root object and ends with a carriage return.**

**Host:** loadfile text.vs3<CR>  
**Vehicle Spy 3:** ok loadfile

### neoVI PRO Text API

The neoVI PRO supports two APIs, first is the Text API and the second is the neoVI RAW API. The Text API is the default API on the USB, COM, and Ethernet (via TCP) ports; therefore, all of the Text API commands here work with the Vehicle Spy 3 code running in the neoVI

PRO.

## Using the Text API

The following table indicates how you can interact with the Text API.

Application	Source	Comment
Vehicle Spy 3	Text API Terminal	Allows you to manually type in Text API commands and see their responses.
Vehicle Spy 3	Function Blocks	Allows you to send and receive Text API commands.
Vehicle Spy 3	Java	The Java Environment interacts with Vehicle Spy via the Text API.
Vehicle Spy 3	Via COM or TCP port	Vehicle Spy 3 can act as a COM or TCP server. Setup via Tools/Options.
neoVI PRO	neoVI PRO setup	The neoVI PRO setup allows you to send commands to neoVI PRO on the control panel.
neoVI PRO	via USB, COM and TCP ports	
neoVI PRO	Java or Function Block scripts	
DLL	The TextAPI method of the icsneo40.dll	Not yet supported.

---

## Root Objects

Command Name	Description	Example
<b>Root Objects for Application Signals</b>		
all?	returns all application signals that are not remote signals in a key=value comma separated string. This is used to efficiently read all app signals over a	all?

	slower network.	
allsetup?	returns all application signals that are not remote signals in a key=description comma separated string. This is used to efficiently read all app signals descriptions over a slower network.	
apprestore	Restores application signals from disk.	
appsave	Saves application signals to disk. Application Signals must be enabled for saving. Signals are saved in a file in the same path as the vs3 file.  {vs3 file name}.appini	
as	Accesses the collection of Application Signals. See the topic on <a href="#">Collection Objects</a> .	
as(index or key)	Accesses a specific application signal by an index or a key. Set the topic on <a href="#">Signal</a> objects.	
<b>Root Objects for Diagnostic Jobs</b>		
dg	Accesses a specific diagnostic jobs by an index or a key.	dg(0).start ;starts a diag job.
<b>Root Objects for Files and Paths</b>		
copyfile	Copies a file in the data directory to another in the data directory. Source and target are separated by a comma. Make sure the file is accessible and not in use.	copyfile MyTestfile.vs3,CopyOfMyTest.vs3
deletefile	Removes a file in the data directory. Make sure the file is accessible and not in use.	deletefile FileToDelete.vs3
dir	Returns the files in the root of the compact flash card or the Vehicle Spy data directory. A filter spec determines which files to return.	dir? *.*
diskspace	returns the amount of diskspace both available and total in kilobytes.	diskspace?
filedetails	Returns information size, time/date on a file in the data directory/neoVI PRO compact flash card.	filedetails? test.vs3
loadfile	Loads a setup file from the data directory. This command should not be called from the same instance of Vehicle spy.	loadfile test.vs3

renamefile	Renames a file in the data directory. The input and output files names are separated by a comma. Make sure the file is accessible and not in use.	renamefile FileName.vs3,NewNameForFile.vs3
status	Returns the loaded file and whether the file is running.	status?
<b>Root Objects for Function Blocks</b>		
fb	Accesses the collection of Function Blocks. See the topic on <a href="#">Collection Objects</a> .	fb.count? ;asks for the number of function blocks
fb(index or key)	Accesses a specific function block by an index or a key. Set the topic on <a href="#">function block</a> objects.	fb(1).start ;starts the first function block.  fb(tst2).stop ;stops the function block with key tst2.
<b>Root Objects for GPS or Joystick</b>		
gps	Accesses the GPS object.	gps.latitude? gps.longitude? gps.altitude? gps.speed? gps.isvalid?
JoystickEnabled	Indicates whether the joystick is enabled or not.	
JoystickSelected	Indicates which joystick is used.	
<b>Root Objects for Graphical Panels or User Interface</b>		
gp(index or key)	Graphical Panel objects.	
gp(index or key).all?	returns all graphical panel control values in a key=value comma separated string. This is used to efficiently read all graphical panel data over a slower network.	gp(dia1).all?
gp(index or key).allsetup?	returns the graphical panel as an XML string.	
gpallsetup?	returns all graphical panels in a key=description comma separated string. This is used to efficiently read all panel descriptions over a slower network.	
ui	Controls the user interface of neoVI PRO. Described in a <a href="#">separate topic</a> .	
<b>Root Objects for Hardware</b>		
ao	Accesses a specific <a href="#">analog outputs</a> by an index or a	ao(0).value 3.12

<code>ao</code>	key. See the topic on transmit message objects.	<code>ao(u).value 5.42</code>
<code>id</code>	Returns the current neoVI PRO firmware ID.	<code>id? ;get the neoVI PRO ID.</code>
<code>io0isoutput or io1isoutput</code>	Returns/Sets whether MISC1 pin is an output or input.	<code>io0isoutput 1 ;make MISC1 and output</code>
<code>io0value</code>	Returns/Sets the value of the MISC 1 pin on neoVI PRO on the DB15 connector.	<code>io0value?</code>
<code>io1value</code>	Returns/Sets the value of the MISC 2 pin on neoVI PRO on the DB15 connector.	<code>io1value 1</code>
<code>ixcbusenabled</code>	Returns/Sets whether the ixbus is enabled.	
<code>ixcbusnetwork</code>	Returns/Sets the network where the IXCBus protocol is used.	

**Root Objects for J1939**

<code>j1939dm1srcall</code>	Returns a comma separated list of SRC addresses that have active DTC's	<code>j1939dm1srcall?</code>
<code>J1939dm1src(address)</code>	<a href="#">J1939 Object</a> ; also returns list of DTCs for specified address	<code>j1939dm1src(0)?</code>

**Root Objects for Messages**

<code>mg</code>	Accesses the collection of Messages. See the topic on <a href="#">Collection Objects</a> .	
<code>mg(index or key)</code>	Accesses a specific <a href="#">message object</a> by an index or a key. See the topic on message objects.	<code>mg(0).clearstats ;clears the stats of first msg</code>
<code>tx</code>	Accesses the collection of Transmit Messages. See the topic on <a href="#">Collection Objects</a> .	<code>tx.add ;adds a message at the end of the collection and returns new key</code>
<code>tx(index or key)</code>	Accesses a specific transmit message by an index or a key. See the topic on transmit message objects.	<code>tx(1).Arbid 234 ;sets the arb id in hex</code>

**Root Objects for Simulation**

<code>ecu</code>	Accesses the ecu subobject. This is discussed in a <a href="#">separate topic</a> .	
<code>SimulationEnabled</code>	Sets/Returns whether simulation is used or you are connecting to hardware.	
<code>SimulationPath</code>	Sets/Returns the path of the file used for simulation mode.	

**Root Objects for Vehicle Spy**

AutoDetectHardware	Sets whether Vehicle Spy will detect hardware or not.	AutoDetectHardware 1 ;sets hardware to autodetect. AutoDetectHardware? ;asks for the autodetect setting.
isrunning	Returns whether if Vehicle Spy is running or not.	isrunning?
Start	Starts Vehicle Spy.	
Stop	Stops Vehicle Spy.	
timedate	Returns/Sets the time date of the clock.	timedate?

## Analog Output Objects

Command Name	Description	Example
calculatelfromsignal	Returns/Sets whether analog outputs are automatically calculated using busdecoder mode or can be set via the Text API. The power default is automatically calculated.	ao(0).calculatelfromsignal 0 ;allow manual control
EnableCalibratedValues	Returns/Sets whether analog outputs are scaled according to the calibration scaling. Disable this feature to perform calibration.	ao(0).EnableCalibratedValues 0 ;disable for calibration
messagekey	Returns/Sets the key of the message attached to this analog output.	
signalkey	Returns/Sets the key of the signal attached to this analog output. This is used in conjunction with signal key.	
value	Returns/Sets the value of the output. This is only valid if calculatelfromsignal is false.	ao(0).value 3.21 ;sets output to 3.21V

## Collection Objects

Command Name	Description	Example
Additem	Adds an item to the end of the collection.	fb.additem ;adds an item to the end of the collection.
	Indicates how many items in the collection.	

Count	Indicates how many items in the collection. Read only.	tx.count? ;how many tx messages are defined?
DeleteAllObjects	Removes all items from the collection.	as.deleteallobjects ;remove all of the app signals.
KeyExists	Determines if the specified key exists in the collection.	tx.keyexists out0 ;does this key exist in the collection?
ReturnIndexFromKey	Finds the key in the collection and returns the index. If the key is not found it returns -1.	tx.returnindexfromkey out0

## Function Block Objects

Command Name	Description	Example
save	Saves the function block data.	
start	Starts the function block.	fb(0).start ;starts the first function block.
stop	Stops the function block.	
trigger	Triggers the function block.	

## J1939 Objects

Command Name	Description	Example
dtc(code).spn	Returns the SPN number for requested DTC (specified by code)	j1939dm1src(0).dtc(0).spn?
dtc(code).FMI	Returns the FMI number for requested DTC (specified by code)	j1939dm1src(0).dtc(0).fmi?
dtc(code).oc	Returns the OC number for requested DTC (specified by code)	j1939dm1src(0).dtc(0).co?

## Message Objects

Command Name	Description	Example

ByteStringX	Sets the filter bytes for a network in hex.	mg(0).ByteString0 110
CFTimeOutsMs	Specifies the Consecutive Frame timeout in milliseconds.	
ClearStats	Clears all statistical information associated with the message. Does not clear signal stats.	mg(0).ClearStats
Compile	Compiles filter bytes and equations and then makes changes activate.	
Description {Text Description}	Sets the description for a message.	Set the description for message number 4 as Engine Data:  mg(3).description Engine Data<CR>
DisplayColor	Sets/Returns the color the message is displayed with.	mg(0).DisplayColor 0 ;display black
EnableISO15765	This enables long messaging on CAN based off of ISO15765.	
EnRxEvent {Return Msg, ReturnValue, Return Stats}		.sg(in3433).EnRxEvent
ExpectedLength	Sets the expected length of a specific message.	
FlowCArbID	Sets/Returns CAN id used for the flow control frame in iso15765 messaging.	
FlowCBlockSize	Sets/Returns the block sized used in long can messaging.	
FlowCSTmin	Sets/Returns the flow control STMin.	
refresh	Updates message table. This is useful when making changes to messages via text API	mg.refresh
sg	Accesses the collection of Signals. See the topic on <a href="#">collection objects</a> .	
sg(index or key)	Accesses a specific <a href="#">Signal object</a> by an index or a key. See the topic on message objects.	sg(0).value? ;Asks for the Value of the first signal

## Signal Objects

Command Name	Description	Example
--------------	-------------	---------

<b>value</b>	Sets/Gets the value.	
Description {Text Description}	Sets the description for a signal.	Set the description for signal number 4 as throttle position:  mg(0).sig(3).description Throttle Position<CR>
DisplayColor	Sets>Returns the color the message is displayed.	mg(0).DisplayColor 0 ;display black
Equation	Gets/Sets the equation.	
Format	Get/Sets the equation format.	
refresh	Updates signal information. This is useful when making changes to signals via text API.	

## ASCII Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

**intrepidcs API Documentation - (C) Copyright 1997-2011 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

Last Updated : Monday, April 28, 2014



## neoVI PRO UI - Vehicle Spy 3 Text API

### Objective

The neoVI PRO user interface is accessed using the UI object. The syntax is as follows:

```
ui.clear ; clears the screen
```

### neoVI PRO Display

The neoVI PRO has a 128 wide by 64 pixel high monochrome display. UI commands support x coordinates between 0 and 127 and y coordinates between 0 and 63. The UI commands supports two colors: blue (1) and white (0). Colors can be inverted using the invert command (useful in different lighting environments).

### UI Object

Command Name	Description	Example
ledpwr	Sets/Clears the neoVI PRO power LED	ui.ledpwr 1 ;// sets the led on ui.ledpwr 0 ;// turns led off
clear	Clears the LCD screen	ui.clear ;// clears the LCD screen
line	Draws a line on the LCD screen in a specified color Arguments: <b>x1,y1,x2,y2,color</b>	ui.line 0,32,127,32,1 ;// draw a line in the center of the screen
print	Prints Text on the LCD screen in a specified color, size and horizontal alignment. Arguments: <b>x1,y1,fontsize,alignment,color,{text}</b>  <b>Font Size:</b> 0) normal letters (5x8), 1) small letters (Xx5), 2) large (10x16)  <b>Alignment:</b> 0) no alignment 1) left, 2) center, 3) right. x is ignored for alignments 1 through 3.	ui.print 0,28,0,2,1>Hello neoVI World ;// displays hello world on the screen
rect	Draws a rectangle on the LCD screen with optional fill  Arguments: <b>x1,y1,x2,y2,color,fill</b>	ui.rect 10,10,30,30,1,0 ;// draw a square on the screen

ledex	Sets/Clears the neoVI PRO exclaim (!) LED	ui.ledex 1 ;// sets the led on ui.ledex 0 ;// turns led off
leddb	Sets/Clears the neoVI PRO database LED	ui.leddb 1 ;// sets the led on ui.leddb 0 ;// turns led off
buzz	Sets/Clears the neoVI PRO buzzer	ui.buzz 1 ;// turns on buzzer ui.buzz 0 ;// turns off buzzer
keys	Returns the key pad state in a bitfield:  1) Up 2) Down 4) Left 8) Right 16) check 32) X 64) O 128) Square 256) Star	ui.keys? ok keys 1 ;// the up buttons is currently pressed
keycheck	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	ui.keycheck? ok keycheck 0
keyleft	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
keyright	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
keyup	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
backlight	Sets/Clears the neoVI PRO backlight	ui.backlight 1 ;// turns on backlight ui.backlight 0 ;// turns off backlight

invert	Allows you to invert the colors on the display.	<code>ui.invert 1 // invert on ui.invert 0 // invert off</code>
dbitmap	Draws a bitmap <b>x1,y1,widthinpixels,heightinpixels,{csv hex bitmap}</b>	<code>ui.dbitmap 0,0,4,8,FF,FF,FF,FF ;// draws a block of pixels</code>
keydown	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
keyo	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
keystar	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
keybox	Returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
Circle	Draws a circle on the display  Arguments: <b>x,y,radius, color</b>	<code>ui.circle 30,30,5,1</code>
Pixel	Draws a pixel on the display  Arguments: <b>x,y,color</b>	<code>ui.pixel 30,30,1 ;// set pixel to blue</code>
operatingmode	Gets/Sets the operating mode of the display and neoVI PRO:  0) Bus Decoder Normal 1) Bus Decoder J1979 2) Vehicle Spy Mini Mode 3) Custom 4) Diag Tool 5) Test and Debug	<code>ui.operatingmode 3 ;// switch to custom mode</code>

keyx	returns and clears the keypress latch  This will return 1 if the enter key has been pressed since last time the key was checked.	
setpendant	This enables the neoVI PRO pendant	
pred	Sets the intesity of the RED led on the neoVI PRO pendant	
pblue	Sets the intensity of the Blue LED on the neoVI PRO pendant	
pgreen	Sets the intensity of the Green LED on the neoVI PRO pendant	
button	Reads the button on the pendant	ui.button?

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2008*

## ECU Object - Vehicle Spy Text API

### Objective

ECU specific functionality is accessed via the ecu object. The syntax is as follows:

`ecu.sim.simulatetransmitmessages 0 ;simulate the receive messages for those in the ECU collection`

### ecu Object

Command Name	Description	Example
sim	Accesses the <a href="#">ECU simulator</a> .	<code>ecu.sim.compile ;compile the simulator</code>

### ecu.sim Object

Command Name	Description	Example
compile	Compiles the changes made to the ECUs collection. It will regenerate the tx message collection. and copy the signals from the signal collect.	
defaultsignaltype	This is an enumerated constant indicated for signals not accounted for in the signals collection or the replay data file.	
ec	Accesses the collection of <a href="#">Simulator ECUs</a> for simulation.	
isrunning	Indicates if the simulator is running or not.	<code>ecu.sim.isrunning?</code>
manualstart	Set>Returns if the simulator starts when Vehicle Spy starts.	
replaydatafile	Set>Returns if the name of the vehicle spy log file used for replay.	
repeatreplay	Set>Returns if the replay should automatically restart when the file completes.	
start	Starts the simulator.	
stop	Stops the simulator.	
simulatetnormal	Returns/Sets if the ECUs should simulate normal	

<code>simulaternormal</code>	messaging.	
<code>simulatediagnostics</code>	Returns/Sets if the ECUs should simulate diagnostics messaging.	
<code>simulatetransmitmessages</code>	1 = tx message list is generated from the tx list of each ECU.  0 = tx message list is generated from the rx list of each ECU.	
<code>sg</code>	A <a href="#">collection of signals</a> which will be copied to the transmit message upon compilation.	
<code>tx</code>	This is the <a href="#">collection of transmit messages</a> sent by the simulator. It is dynamically created by the compile method.	
<code>replaystart</code>	This starts the replay data file at the first value if the simulator is running.	
<code>replaystop</code>	This stops file replay if it is running.	
<code>generatehv wakeup</code>	If 1 the simulator will generate a single wire CAN high voltage wakeup when started.	
<code>mg</code>	Accesses the collection of <a href="#">Messages</a> for simulation. Messages are only present if they apply some custom properties such as default message bytes or period.	
<code>GenerateVNMF</code>	If 1 the simulator will generate a VNMF for the single wire CAN ECUs involved in the simulation.	
<code>VNMFID</code>	This indicates the CAN ID of the VNMF used by the simulator.	

## ecu.sim.ec Object

Command Name	Description	Example
<code>ecuname</code>	The ECU short name.	
<code>enablenetworkmanagement</code>	If 1 the simulator uses VN (virtual network) states to determine which messages should be sent. Otherwise, they are always sent.	
<code>key</code>		
<code>networkname</code>	The network name according to the uef file "hsCAN" or "swCAN".	

**ecu.sim.mg Object**

Command Name	Description	Example
databytes{X}	This allows default databytes to be entered for databytes0 through databytes7.	
description	The description of the message. This includes the ECU name followed by a backslash. "SDM\Airbag Indications"	
disablemessage	This disables this message if 1.	
overrideperiod	This will override the period of the message.	
period	The modified message period if "overrideperiod" is 1.	

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)***Last Updated : Tuesday, December 16, 2008*

## Labview Text API Interface to Vehicle Spy 3

The links below contain example Labview VI's to interface Labview to Vehicle Spy using Vehicle Spy's [Text API](#). This interface allows for a transfer of data between the 2 applications.

[Labview Text API Interface to Vehicle Spy 3 Example VI's](#)

The link below requires internet access, but also contains Labview Runtimes.

[Labview Text API Interface to Vehicle Spy 3 Example with Interface Installer](#)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc.**

Last Update: *Thursday, October 08, 2009*

## Vehicle Spy VSB file Spec

### File Specification

Description	Length in Bytes	Notes
Text Identifier	6	single byte char array with text "icsbin" used to identify file type.
File Version	4	single byte char array indicate version.
Version Section	Variable (see below)	This data structure is determinant the File Version

### Version Specification

#### Version 0x101

This format is no longer created. [Click](#) to see Network ID's

Description	Length in Bytes	Notes
Length of Vs3 File	4	int value indicating the length of the next section (the vs3 file). A zero indicates no vs3 file is present.
Vs3 file	variable (see above)	vs3 file used to save this binary file. This vs3 file is later used to decode the binary data into usable information.
Length of text comment	4	int indicating the length of the text header comment of the saved file. SINCE THE COMMENT IS UNICODE THEREFORE HIS IS THE LENGTH IN CHARACTERS - NOT BYTES.
Text comment	2 bytes per character (see above for number of characters)	Unicode text comment.
Size of Buffer of Messages	4	sizeof(VSBSpyMessage) multiplied by Number of messages saved to the file
Current Buffer Pointer	4	int value which is the pointer to the most recent buffer item +1. (only needed if Number of All time messages > Original buffer size)
Original Buffer Size	4	int value which is the size of the buffer memory originally allocated by this buffer
Number of All time messages	4	this indicates how many messages were received by this buffer (this number will indicate overflows)
Buffer of Messages	variable (see Size of Buffer of Messages)	VSBSpyMessage structures
Start Time of Collection	struct icsspyMsgTime	this is a comparison value between the system time stamp and the neoVI time stamp.

#### Version 0x102

This format is created by the extractor. [Click](#) to see Network ID's

Description	Length in Bytes	Notes
Length of EDP Section	4	int value indicating the length of the next section (the EDP). A zero indicates no EDP section file is present.
EDP Section	variable (see above)	used to save the extra data bytes for networks such as Ethernet, Flexray, and CANFD
Buffer of Messages	variable (see Size of Buffer of Messages)	VSBSpyMessage structures
Start Time of Collection	struct icsspyMsgTime	this is a comparison value between the system time stamp and the neoVI time stamp.

#### Version 0x103

This format is created using VSpy. [Click](#) to see Network ID's

Description	Length in Bytes	Notes

Length of EDP Section	4	int value indicating the length of the next section (the EDP). A zero indicates no EDP section is present.
EDP Section	variable (see above)	used to save the extra data bytes for networks such as Ethernet, Flexray, and CANFD
Length of text comment	4	int indicating the length of the text header comment of the saved file. SINCE THE COMMENT IS UNICODE THEREFORE HIS IS THE LENGTH IN CHARACTERS - NOT BYTES.
Text comment	2 bytes per character (see above for number of characters)	Unicode text comment.
Size of Buffer of Messages	4	sizeof(VSBSPyMessage) multiplied by Number of messages saved to the file
Current Buffer Pointer	4	int value which is the pointer to the most recent buffer item +1. (only needed if Number of All time messages > Original buffer size)
Original Buffer Size	4	int value which is the size of the buffer memory originally allocated by this buffer
Number of All time messages	4	this indicates how many messages were received by this buffer (this number will indicate overflows)
Buffer of Messages	variable (see Size of Buffer of Messages)	VSBSPyMessage structures
Start Time of Collection	struct icsspyMsgTime	this is a comparison value between the system time stamp and the neoVI time stamp.

#### EDP section

Used by version 0x102 and 0x103. The ExtraDataPtr -1 of each message is a index to the EDP in EDP section. [Click](#) to see Network ID's

Description	Length in Bytes	Notes
Text identifier	12	single byte char array with text "EDP_SECTION" used to identify file type. please note the null terminator at the end
EDP Size	4	int indicating the length of the EDP
EDP	variable (see above)	used to save the extra data bytes for networks such as Ethernet, Flexray, and CANFD
.....	variable (see above)	EDP Size and EDP continue to repeat until Length of EDP Section is met.

#### Version 0x104

This format is created by the Loggers and VSpy Stream to Disk.

Description	Length in Bytes	Notes
Buffer of Messages	VSBSPyMessage structures	VSBSPyMessage structures
EDP	variable (see previous message's ExtraDataPtr)	used to save the extra data bytes for networks such as Ethernet, Flexray, and CANFD
.....	variable	Buffer of Message and EDP continue to repeat until Start Time of Collection. If you are reading the VSB file, simple continue to read until Buffer of Message size read not equal sizeof(VSBSPyMessage)
Start Time of Collection	struct icsspyMsgTime	this is a comparison value between the system time stamp and the neoVI time stamp.

#### NetWork ID

##### version 0x101

Description	Value

HSCAN	0
MSCAN	1
SWCAN	2
LSFTCAN	5
DEVICE	8
HSCAN2	9
HSCAN3	10
LIN2	11
LIN3	12
LIN4	13

version 0x102 , 0x103, 0x104

Description	Value
DEVICE	0
HSCAN	1
MSCAN	2
SWCAN	3
LSFTCAN	4
J1708	6
JVPW	8
ISO	9
ISO2	14
LIN	16
ISO3	41
HSCAN2	42
HSCAN3	44
ISO4	47
LIN2	48
LIN3	49
LIN4	50
LIN5	84
MOST	51
CGI	53
HSCAN4	61
HSCAN5	62
SWCAN2	68
ETHERNET_DAQ	69
FLEXRAY1A	80
FLEXRAY1B	81
FLEXRAY2A	82
FLEXRAY2B	83
FLEXRAY	85
MOST25	90
MOST50	91
ETHERNET	93
GMFSA	94
TCP	95
HSCAN6	96
HSCAN7	97
LIN6	98
LSFTCAN2	99
OP_ETHERNET1	17
OP_ETHERNET2	18

OP_ETHERNET3	19
OP_ETHERNET4	45
OP_ETHERNET5	46
OP_ETHERNET6	73
OP_ETHERNET7	75
OP_ETHERNET8	76
OP_ETHERNET9	77
OP_ETHERNET10	78
OP_ETHERNET11	79
OP_ETHERNET12	87

**Example**

Below is a C++ example of how to open a .vsb file (v102 & v103 & 104) and display the network messages (Ethernet, CANFD, and CAN) with some other useful information. You can view the full [C++ code](#) or Download the full project from the links provided.

**Reading File Specification**

```
bool Messages::Read(char * sFileName)
{
    unsigned long numEDP = 0;

    m_pFile = fopen(sFileName, "rb");
    if (m_pFile)
    {
        fSead( 0, SEEK_END);
        int length = ftell(m_pFile); //identify length of File
        fSead( 0, SEEK_SET);

        char id[6];
        if (!fRead((char *)id, 6) || memcmp(id, "icsbin", 6) != 0) //check for icsbin in the first 6 bytes to identify if it a VSB file
        {
            printf("Invalid VSB file\n");
            return false;
        }
        unsigned int fileversion;

        if (!fRead((char *)&fileversion, 4)){ //VSB has 4 versions 101 is no longer used
            printf("Read Error\n");
            return false;
        }
        else if (fileversion == 0x102)//this is created by the extractor
        {
            int EDPSize = ReadEDPSelction(); //the extra data section contains extra data which is different for each message type
            if (EDPSize == 0)
                fSead( VSB_HEADERSIZE, SEEK_SET);
            unsigned long numofmessages = ((length - (EDPSize + sizeofEDPSIZE)) - VSB_HEADERSIZE) / sizeof(icsSpyMessage); //calculate number of messages in file; the plus 4
            m_nMessages = numofmessages; // set number of messages in file
            if (numofmessages){
                if(!ReadMessage(numofmessages, fileversion)){ //read messages
                    printf("file was not read correctly \n");
                    return false;
                }
            }
        }
        else if (fileversion == 0x103)//this is created by VSPY
        {
            ReadEDPSelction(); //read EDP
            unsigned long numofmessages = Read103Header(); //read 103 header
            m_nMessages = numofmessages; //set number of messages in file
            if (numofmessages){
                if(!ReadMessage(numofmessages, fileversion)){ //read messages
                    printf("file was not read correctly \n");
                    return false;
                }
            }
        }
    }
}
```

```

    }
    else if (fileversion == 0x104){//this is created by various hardware and VSPY for logging
        if(!Read104()){ // 104 has a very different structure compared to the other file types
            printf("file was not read correctly \n");
            return false;
        }
    }
    else{
        printf("%s is an unsupported VSB version!\n", sFileName);
        return false;
    }
    fclose(m_pFile);
}
else
{
    printf("Could not open %s to read!\n", sFileName);
    return false;
}

return true;
}

```

**Reading EDP Section**

```

int Messages::ReadEDPSelction()
{
    unsigned long edpsize;
    bool bFixTimeStamp = false;

    if (fRead((char *)&edpsize, SIZEOFEDPSIZE) && edpsize > 0)//read size of edp section
    {
        unsigned long tempEDPSize = edpsize;
        static const char* EDP_SECTION = "EDP_SECTION";
        char obItem [EDPSECTIONSTRINGSIZE];

        // check if there's an EDP_SECTION in here
        if (fRead(obItem, EDPSECTIONSTRINGSIZE) && tempEDPSize >= EDPSECTIONSTRINGSIZE && strcmp(obItem, EDP_SECTION) == 0)//verify edp section
        {
            // the following lines of code is a bad idea for large files. but since it is an simple example I wanted to simplify the code normality you would want to use some sort of file stream.

            tempEDPSize -= EDPSECTIONSTRINGSIZE;
            m_pEDPSection = new char[tempEDPSize];
        }

        if (!fRead(m_pEDPSection, tempEDPSize)){ //load edp section into memory
            printf("invalid file read\n");
            return 0;
        }

        char * edpPtr = m_pEDPSection;
        while (tempEDPSize > 0)//parse edp section
        {
            int edpLen = *(int*)edpPtr;
            m_edps.push_back(std::pair<char*, int>(edpPtr + sizeof(int), edpLen)); //each node in this vector pertains to a different message (this will make more sense when you look at Read Message )
            tempEDPSize -= sizeof(int);
            edpPtr += edpLen + sizeof(edpLen);
            tempEDPSize -= edpLen;
        }
        return edpsize;
    }
    return 0;
}

```

**Reading 0x103 File Info**

```

unsigned long Messages::Read103Header()
{
    unsigned long theNumOfMsgs;
    size_t msgslength, currbuffptr, numalltime, commentlength, origbuffsize;

```

```

fRead((char *)&commentlength, 4); //check if there is a comment
if (commentlength > 0)
    fSeek( (long)commentlength * 2, SEEK_CUR); // skip comment (comment written in wchar_t)
fRead((char *)&msgslength, 4); //size of message section
fRead((char *)&origbufsize, 4); //original size of buffer
fRead((char *)&currbufptr, 4); // current buffer position
fRead((char *)&nunalltime, 4); // number of message all time

theNumOfMsgs = msgslength / sizeof(VSBSPyMessage); // number of messages

return theNumOfMsgs;
}

```

### Reading 0x102 and 0x103 Messages

```

bool Messages::ReadMessage(unsigned long theNumOfMsgs, unsigned int fileversion)
{
    VSBSPyMessage * pMessageRead = new VSBSPyMessage();
    m_pMessages = new icsSpyMessage[theNumOfMsgs];
    bool firstMsg = true;
    SpyMsgTime timeStampRead;
    SpyMsgTime FirstMsgTime;

    for(unsigned int i = 0; i < theNumOfMsgs; i++) { //read one message at a time
        if(!fRead((char *)pMessageRead, sizeof(VSBSPyMessage))) { //read message
            delete pMessageRead;
            delete m_pMessages;
            m_pMessages = NULL;
            theNumofMsgs = 0;
            return false;
        }

        if (firstMsg) { //record the start time of the first message
            FirstMsgTime.HardwareTimeStampID = m_pMessages[0].TimeStampHardwareID;
            FirstMsgTime.HardwareTime1 = m_pMessages[0].TimeHardware;
            FirstMsgTime.HardwareTime2 = m_pMessages[0].TimeHardware2;
            FirstMsgTime.SystemTimeStampID = m_pMessages[0].TimeStampSystemID;
            FirstMsgTime.SystemTime1 = m_pMessages[0].TimeSystem;
            FirstMsgTime.SystemTime2 = m_pMessages[0].TimeSystem2;
            firstMsg = false;
        }

        VSBMessageToNormal(&m_pMessages[i], pMessageRead); // convert structure

        bool clear = true;
        if (pMessageRead->iExtraDataPtrEnabled && pMessageRead->iExtraDataPtr) // check for edp section
        {
            if (pMessageRead->iExtraDataPtr >= 1 && pMessageRead->iExtraDataPtr <= m_edps.size())
            {
                int iEDPIndex = pMessageRead->iExtraDataPtr - 1; // identify index in the m_edps vector we made in the ReadEDPSelction function
                int payloadLength = m_edps[iEDPIndex].second;
                m_pMessages[i].pExtraDataPtr = new pExtraDataPtrHandler(); //using pExtraDataPtrHandler to simply make to identify length of edp section
                ((pExtraDataPtrHandler *) (m_pMessages[i].pExtraDataPtr))>pExtraDataPtr = m_edps[iEDPIndex].first;
                ((pExtraDataPtrHandler *) (m_pMessages[i].pExtraDataPtr))>length = payloadLength;
                clear = false;
            }
        }
        if (clear)
        {
            m_pMessages[i].ColorID = 0; //reset values
            m_pMessages[i].pExtraDataPtr = 0; //reset values
        }
        if (fileversion == 0x102) //all 102s implicitly have bit 7 high. since we're converting to 103 we need to force it
            m_pMessages[i].TimeStampHardwareID |= 0x80;
    }

    if (!fRead((char *)&timeStampRead, sizeof(timeStampRead))) //check if start logging time was recorded
        timeStampRead = FirstMsgTime; // if start logging time was not record simply set it to first message time
    m_startTimeStamp = timeStampRead;
    delete pMessageRead;
    return true;
}

```

```
}
```

### Reading 0x104

```
bool Messages::Read104() // 104 don't have a edp section they instead put the extra data after that message
{
    unsigned long EDPLength = 0, Edpsize = 0, numOfMessages = 0;

    bool firstMsg = true;
    SpyMsgTime timeStampRead;
    SpyMsgTime FirstMsgTime;

    VSBSPyMessage *pMessage = new VSBSPyMessage();

    while (fRead((char *)pMessage, sizeof(VSBSPyMessage))) // read messages to identify number of message and edp size
    {
        numOfMessages++;
        if (pMessage->ExtraDataPtrEnabled && pMessage->iExtraDataPtr > 0) // as you can see we are using extraDataptr to identify the size of the edp section for that message
        {
            Edpsize += pMessage->iExtraDataPtr;
            // The EDP is the length of the extra data following the message, skip it
            fSead((unsigned int)pMessage->iExtraDataPtr, SEEK_CUR);
        }
    }

    fSead(VSB_HEADERSIZE, SEEK_SET); //go back to the start
    //from here on out it is very similar to 103 and 102 but the location of the edp section is different

    m_pMessages = new icsSpyMessage[numOfMessages];

    for (unsigned int i = 0; i < numOfMessages; i++)
    {
        if (!fRead((char *)pMessage, sizeof(VSBSPyMessage)))
            break; //error

        if (firstMsg) {
            FirstMsgTime.HardwareTimeStampID = pMessage[0].TimeStampHardwareID;
            FirstMsgTime.HardwareTime1 = pMessage[0].TimeHardware;
            FirstMsgTime.HardwareTime2 = pMessage[0].TimeHardware2;
            FirstMsgTime.SystemTimeStampID = pMessage[0].TimeStampSystemID;
            FirstMsgTime.SystemTime1 = pMessage[0].TimeSystem;
            FirstMsgTime.SystemTime2 = pMessage[0].TimeSystem2;
            firstMsg = false;
        }

        bool clear = true;

        VSBMessageToNormal(&m_pMessages[i], pMessage);

        if (pMessage->ExtraDataPtrEnabled && pMessage->iExtraDataPtr >= 1 ) {
            EDPLength = pMessage->iExtraDataPtr;
            char * pEDP = new char[EDPLength];
            if(!fRead(pEDP, EDPLength))
                return false; //error
            m_pMessages[i].pExtraDataPtr = new pExtraDataPtrHandler();
            ((pExtraDataPtrHandler *)(m_pMessages[i].pExtraDataPtr))->pExtraDataPtr = (void *)pEDP;
            ((pExtraDataPtrHandler *) (m_pMessages[i].pExtraDataPtr))->length = EDPLength;
            clear = false;
        }

        if (clear) {
            EDPLength = 0;
            m_pMessages[i].ColorID = 0;
            m_pMessages[i].pExtraDataPtr = 0;
        }
    }

    if (!fRead((char *)&timeStampRead, sizeof(timeStampRead)))
        timeStampRead = FirstMsgTime;
    m_startTimeStamp = timeStampRead;
}
```

```
m_nMessages = numOfMessages;  
return true;  
}
```

---

**Contact Information - Intrepid Control Systems, Inc.**

[Main](#)



**Intrepid Control Systems, Inc.**

5700 18 Mile Road

Sterling Heights, Michigan 48314 USA

(ph) 586.731.7950

(fax) 586.731.2274

(email) [moreinfo@intrepidcs.com](mailto:moreinfo@intrepidcs.com)

(website) [www.intrepidcs.com](http://www.intrepidcs.com)

**intrepidcs API Documentation - (C) Copyright 2000-2016 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, January 09, 2009*