

# 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++ 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. It provides one interface for both USB and RS232 operation. It's a low level DLL, meaning there is little checking on the data you pass to it. Therefore, your code must make sure it properly calls the API to avoid crashes.

For applications 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.

## Getting Started

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

## History

- \* Isochronous USB support has been dropped because it was not being used or is necessary.
- \* The Transmit ID is reduced from 16 bits to 14 bits

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, January 02, 2004*

## Basic Operation - intrepidcs API

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

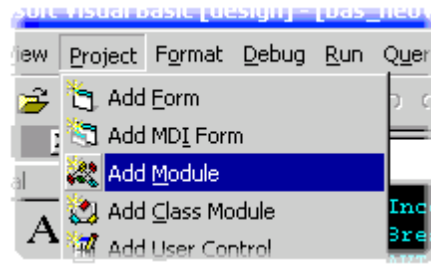
- 1) Start your application.
- 2) Open the driver and create (if necessary) the neoVI object using the [OpenPort](#) method.
- 3) Transmit messages using the [TxMessages](#) method.
- 4) Read messages on the network using the [GetMessages](#) method.
- 5) Optionally readout any errors using the [GetErrorMessages](#) 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, April 24, 2001*

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

To use the `intrepidcs` API in Visual Basic 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.**

Please see the [Visual Basic example](#) for more information on using the intrepidcs API with VB.

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

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

- 1) Copy the import library "[icsnVC40.lib](#)", header file "[icsnVC40.h](#)", and data structure file "[icsspyData.h](#)" to your project directory.
- 2) Link to the icsnVC40.lib import library via the project settings link tab (Figure 1). This dialog is accessible via the "Project" pull down menu in Visual C++.

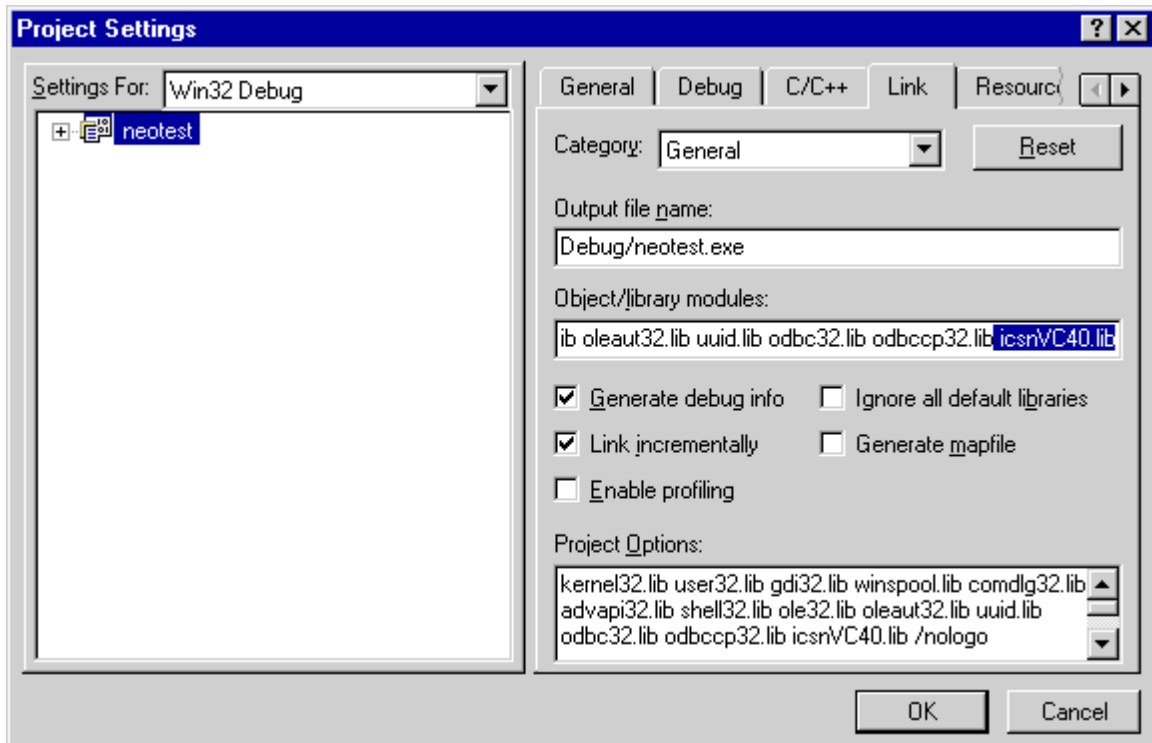


Figure 1 - Link to the "icsnVC40.lib"

- 3) Include the header icsnVC40.h in your C/C++ module (Figure 2).

```
//  
  
#include "stdafx.h"  
#include "resource.h"  
#include "icsnVC40.h"           // include the neoVI  
  
#define MAX_LOADSTRING 100  
  
// Global Variables:  
HINSTANCE hInst;                // c:  
TCHAR szTitle[MAX_LOADSTRING]; // T:  
TCHAR szWindowClass[MAX_LOADSTRING]; // T:
```

Figure 2 - The #include statement in the C/C++ module.

- 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++.

## Using the intrepidcs API in Visual C++ 2005 - intrepidcs API

### Do the following steps to use neoVI in Visual C++ 2005:

1) Copy the import library "[icsnVC40.lib](#)", header file "[icsnVC40.h](#)", and data structure file "[icsspyData.h](#)" to your project directory.

2) Link to the icsnVC40.lib import library via the project settings (Figure 1). This dialog is accessible via the "Project" pull down menu in Visual C++ 2005.

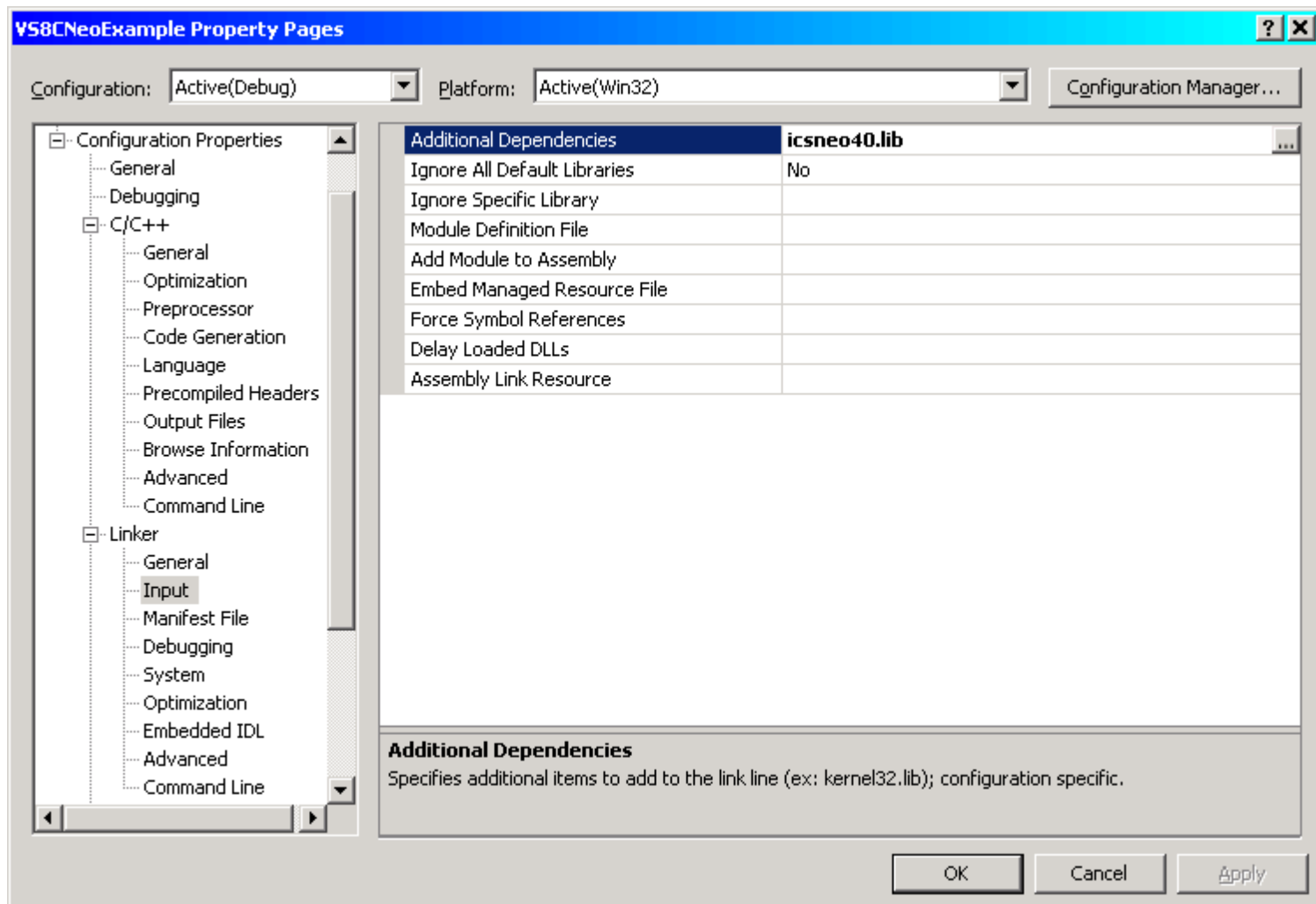


Figure 1 - Link to the "icsnVC40.lib"

3) Include the header icsnVC40.h in your C/C++ module (Figure 2).

```
#pragma once
#include "icsSpyData.h"
#include "icsnVC40.h"
#include "string.h"
#include <string>
#include <vector>

namespace VS8CNeoExample {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
```

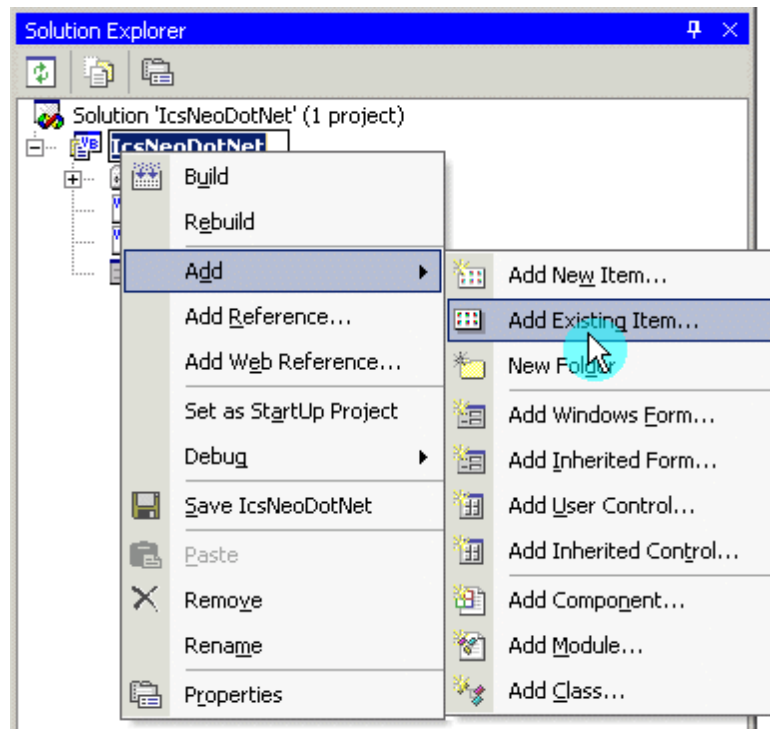
4) Finally, call the methods as defined in the [Basic Operation](#) document.

Please see the [Visual C++ 2005 example](#) for more information on using the neoVI API with Visual C++.



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

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 [Basic Operation](#) document.

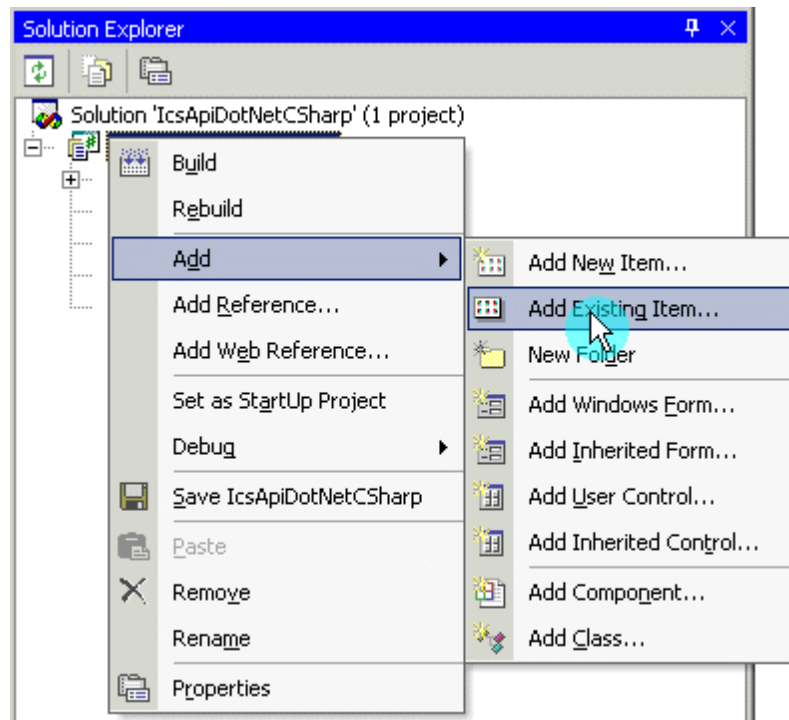


**Figure 1 - Add Module Command From the VB.NET Menu.**

Please see the [Visual Basic Dot Net 2003 example](#) for more information on using the intrepidcs API with VB Dot Net.

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

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, it will be possible call the functions as defined in the [Basic Operation](#) document.



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

Please see the [C# Dot Net 2003 example](#) for more information on using the intrepidcs API with VB Dot Net.



## Using the intrepidcs API in Borland C++ Builder - intrepidcs API

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

- 1) Copy the import library "[icsnBC40.lib](#)", header file "[icsnBC40.h](#)", and data structure file "[icsspyData.h](#)" to your project directory.
- 2) Link to icsnBC40.lib import library via the "Add to Project..." option (Figure 1). This dialog is accessible via the "Project" pull down menu in C++ Builder. When the file dialog appears, select the icsnBC40.lib.

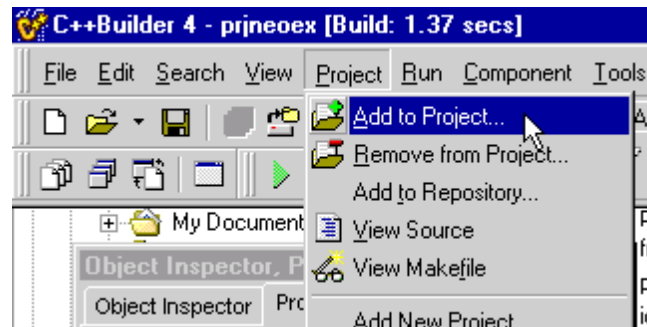


Figure 1 - Add the Link to the "icsnBC40.lib"

- 3) Your project manager will now show the import library icsnBC40.lib (Figure 2).

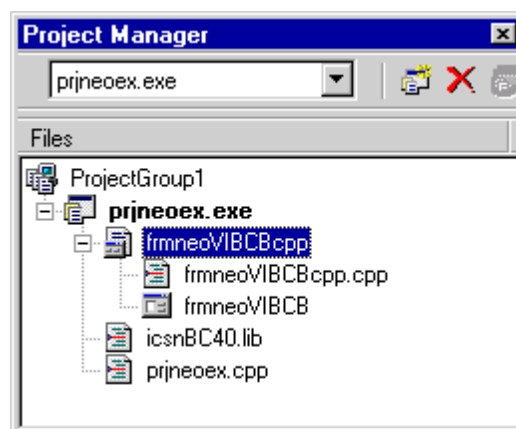
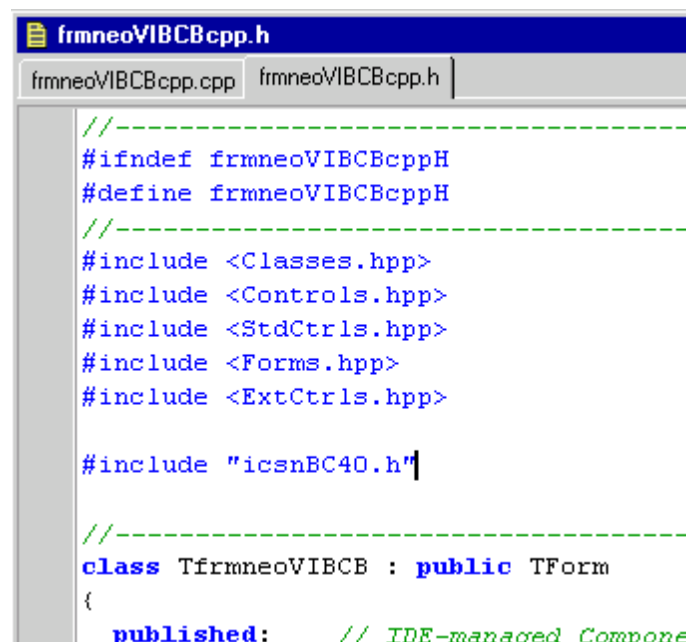


Figure 2 - The import library "icsnBC40.lib" is loaded.

- 3) Include the header icsnBC40.h in your C/C++ module (Figure 3).



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

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

Please see the [Borland C++ Builder example](#) for more information on using the neoVI API with Borland C++ Builder.

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, April 24, 2001*

## Using the intrepidcs API in Delphi - intrepidcs API

### 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.

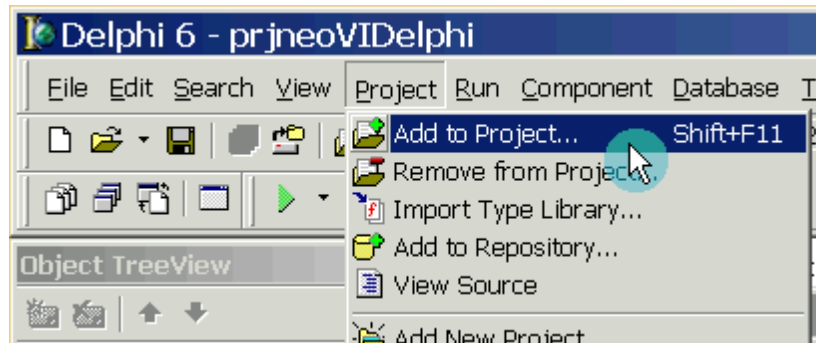


Figure 1 - Add the Link to the "icsn40.pas"

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

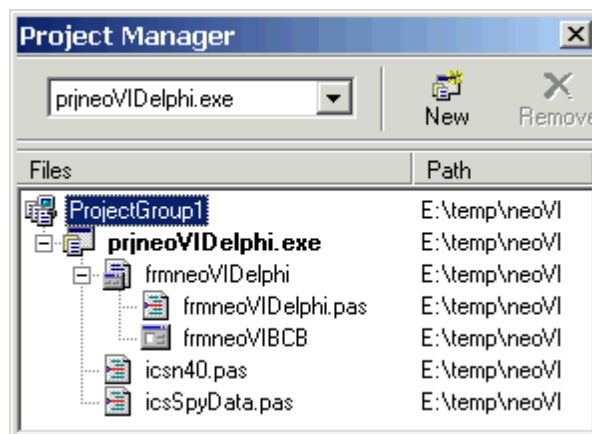


Figure 2 - The import library "icsn40.pas" is loaded.

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

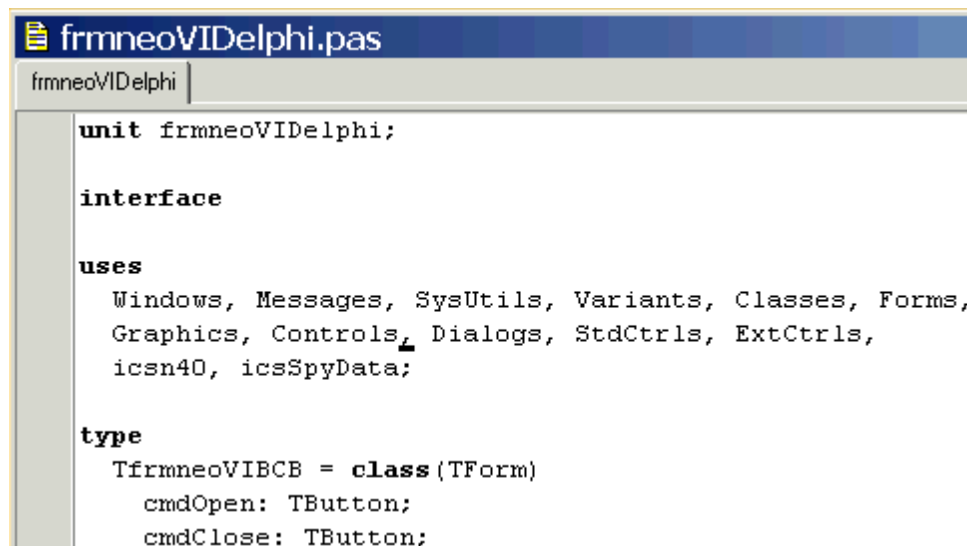


Figure 3 - Adding to Interface Uses

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

Please see the [Borland Delphi example](#) for more information on using the neoVI API with Borland Delphi.

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Friday, January 02, 2004*

# Using the API in LabVIEW - intrepidcs API

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

## Overview

The intrepidcs API packages all of the WIN32 methods into LabVIEW sub VIs. These sub VIs make it easy for you to use neoVI with National Instruments LabVIEW. These sub VIs appear in your LabVIEW functions palette like any other LabVIEW VIs.

Most of the LabVIEW Sub VI's are directly related to the intrepidcs API WIN32 equivalents. Table 1 gives a brief description of each VI. A major point about the VIs, is that instead of [message structures](#), the LabVIEW VIs use an [array of bytes to represent messages](#).



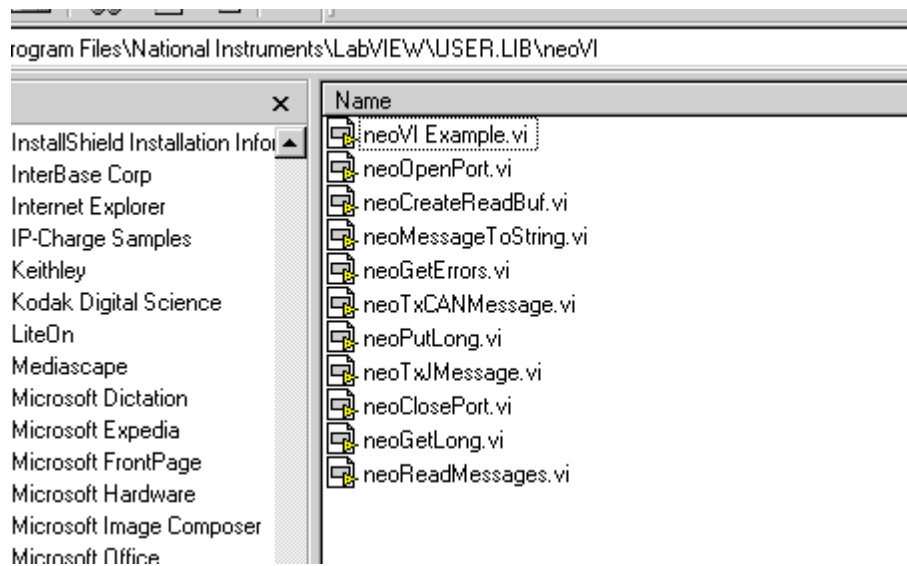
Figure 1 - neoVI LabVIEW VIs in the LabVIEW function palette.

Table 1 - neoVI VIs

VI	Description
neoOpenPort	Calls the <a href="#">OpenPort</a> method
neoCreateReadBuf	Creates a buffer of 20,000 messages to be used in <a href="#">GetMessages</a> .
neoMessageToString	Example of how to take a message and build a string representation of it.
neoGetErrors	Calls the <a href="#">GetErrorMessages</a> method.
neoTxCANMessage	Calls the <a href="#">TxMessages</a> method. The VI interface is setup to transmit a CAN type message.
neoPutLong	Used to set integer (int32) values in a message byte array.
neoTxJMessage	Calls the <a href="#">TxMessages</a> method. The VI interface is setup to transmit a non-CAN type message.
neoClosePort	Calls the <a href="#">ClosePort</a> and <a href="#">FreeObject</a> method.
neoGetLong	Used to get long values (int32) from a message byte array.
neoReadMessages	Calls the <a href="#">GetMessages</a> method.

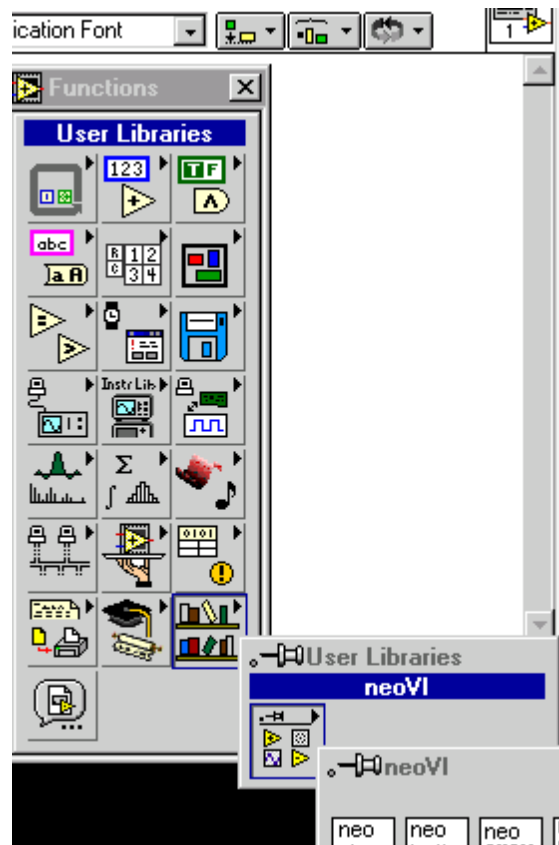
## Steps to use neoVI in LabVIEW

1) Create a sub-directory in the LabVIEW user library subdirectory (..\LabVIEW\USER.LIB\) called neoVI. Copy the neoVI VIs in [labv\\_neo.zip \(168 Kb\)](#) to the subdirectory (figure 2).



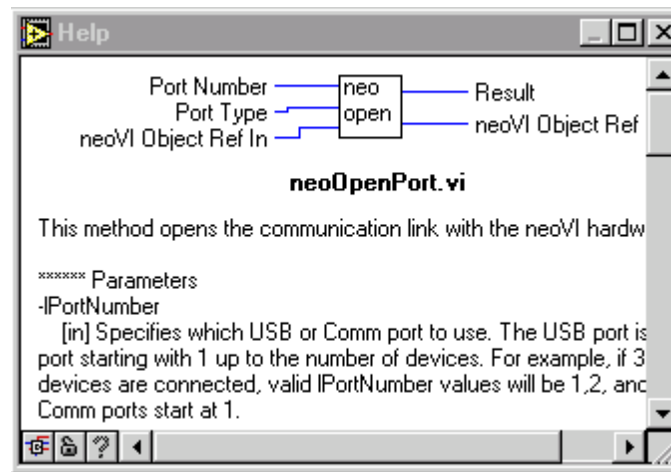
**Figure 2 - the neoVI VIs installed into a user directory**

2) Access the neoVI VIs like other LabVIEW VIs in the LabVIEW function palette. The VIs are located in the User Libraries section (Figure 3) of the functions palette.



**Figure 3 - neoVI VIs in the LabVIEW function palette.**

3) Use the standard LabVIEW help and this help file to help you develop your application (figure 4).



**Figure 4 - The LabVIEW help window displays help on neoVI VIs.**

Please see the [LabVIEW example](#) for more information on using the intrepidcs API with LabVIEW.

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

### 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).

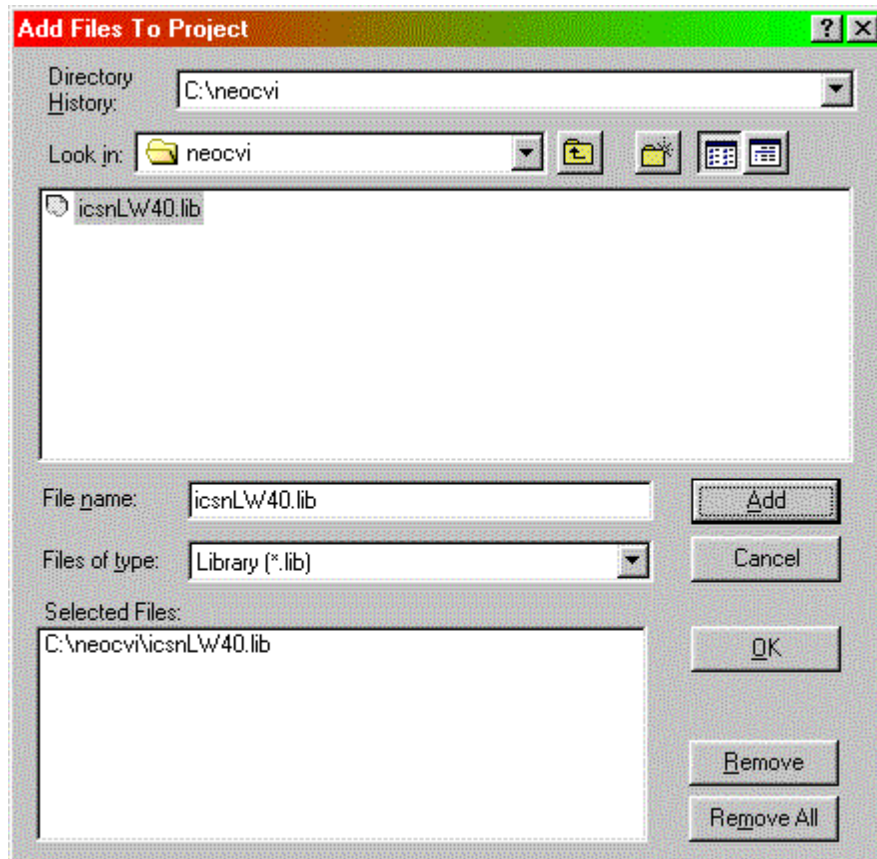


Figure 1 - Link to the "icsnLW40.lib"

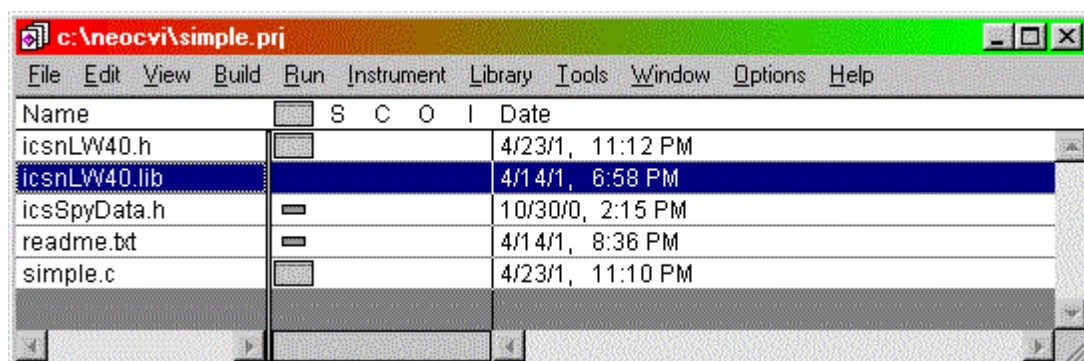
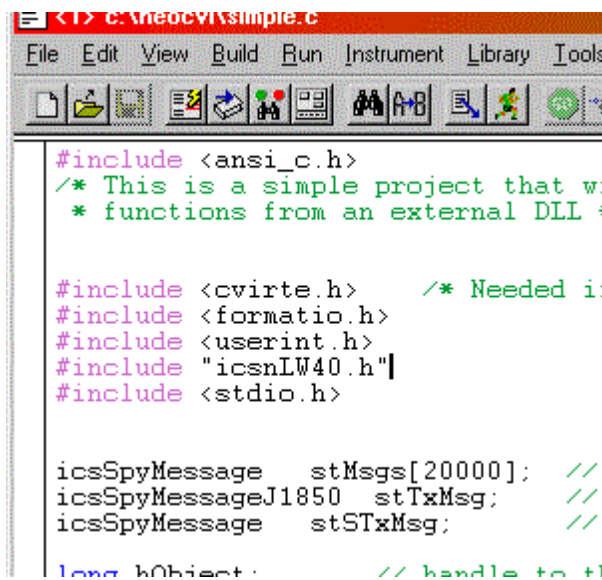


Figure 2 - The project window with the library added.

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



A screenshot of a C code editor window. The title bar shows the file path 'C:\IntrepidCVI\simple.c'. The menu bar includes 'File', 'Edit', 'View', 'Build', 'Run', 'Instrument', 'Library', and 'Tools'. The toolbar contains various icons for file operations and development. The code in the editor is as follows:

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

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

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

long hObject;                  // handle to tl
```

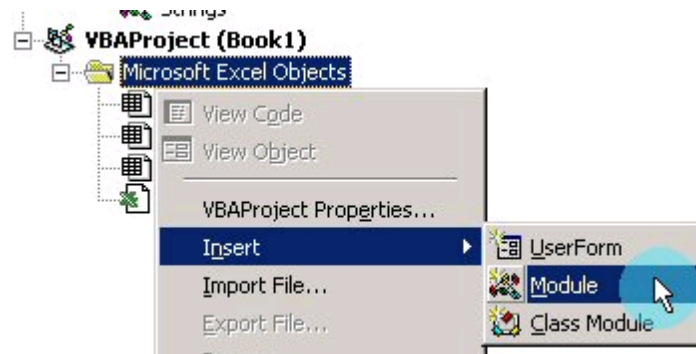
**Figure 3 - The #include statement in the C module.**

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

Please see the [LabWindows CVI example](#) for more information on using the intrepidcs API with LabWindows CVI.

## Using the intrepidcs API in Excel - intrepidcs API

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.**

Please see the [Excel VBA Example](#) for more information on using the intrepidcs API with VBA and Excel.

## WIN32 API Overview - intrepidcs API

Table 1 Below lists the intrepidcs API calls.

Name	Description
<a href="#">OpenPortEx</a>	This method opens the communication link with the neoVI hardware.
EnableNetworkCom	This method enables the communications
<a href="#">ClosePort_</a>	This method closes the communication link with the neoVI hardware.
<a href="#">FreeObject</a>	This method destroys the neoVI driver object and frees associated memory.
<a href="#">GetMessages</a>	This method reads messages from the neoVI hardware.
<a href="#">TxMessages</a>	This method transmits messages to vehicle networks using the neoVI hardware.
<a href="#">GetErrorMessages</a>	This method reads the neoVI DLL error message queue.

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, April 24, 2001*

## 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 method opens the communication link with the neoVI hardware.**

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

#### *lPortNumber*

[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 lPortNumber values will be 1, 2, and 3. Comm ports also start at 1. For TCP/IP it specifies the TCP/IP port number.

#### *lPortType*

[in] Specifies the port type of what the device is connected to. This parameter is used with lPortNumber 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.

#### *lDriverType*

[in] Specifies which neoVI driver to use. This should always be set to INTREPIDCS\_DRIVER\_STANDARD (0).

#### *lIPAddressMSB*

[in] Specifies upper two bytes of TCP/IP address if the lPortType is set to NEOVI\_COMMTYPE\_TCPIP (bytes 4 and 3 of this TCP/IP address: 1.2.3.4).

#### *lIPAddressLSBOrBaudRate*

[in] Specifies the baud rate if lPortType 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.

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 iReturnVal = 0; //iReturn value tells status of the function call
int iPMSB = 0;
int iPLSB = 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)
{
    iPLSB = 57600;
    iReturnVal = icsNeoDll.icsneoOpenPortEx(iPortNumber,
Convert.ToInt32(ePORT_TYPE.NEOVI_COMMTYPE_RS232),
        Convert.ToInt32(eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDARD),iPMSB ,iPLSB,1, ref
bNetworkIDs,    ref m_hObject);
}
else
{
    iReturnVal = icsNeoDll.icsneoOpenPortEx(iPortNumber,
Convert.ToInt32(ePORT_TYPE.NEOVI_COMMTYPE_USB_BULK ),
        Convert.ToInt32(eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDARD),iPMSB , iPLSB, 1, ref
bNetworkIDs,ref m_hObject);
}

if (iReturnVal == 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
}
```

## 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.



## 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 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 must be called before you run the [SendConfiguration](#) API. This disables all vehicle network data. This function is also useful if you application is not ready to receive large amounts of network data.

---

### 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);
```



## 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 Integer,  
ByRef pNumberOfErrors As Integer) As Integer
```

### C# Declare

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

### Parameters

#### *hObject*

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

#### *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

Each [OpenPort](#) call should match a ClosePort call.

---

### Examples

#### Visual Basic Example

```
Private m_hObject As Long    '// Declared at form level and previously open with a call  
to OpenPort  
  
Dim lResult As Long '// Used to store the return variable of the ClosePort method  
Dim lNumberOfErrors As Long '// number of errors in the drivers error queue  
  
 '// close the port associated with neoVI  
lResult = icsneoClosePort(m_hObject, lNumberOfErrors)
```

```

'// test the returned result
If CBool(lResult) Then
    MsgBox "Port Closed Successfully"
Else
    MsgBox "Problem Closing Port"
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

```

int iResult = 0; //Space to Store Result of Function Call
int iNumOfErrors = 0; //Storage for the Number of Errors

//Call for Closing the port
iResult = icsNeoDll.icsneoClosePort(m_hObject, ref iNumOfErrors);
//Check the Result to see if successful
if (iResult==0)
{
    MessageBox.Show("Problem Closing Port"); //Show error
}
else
{
    MessageBox.Show("Port Closed OK")
}
// set the portopen flag to closed
m_bPortOpen = false; //Set flag for other functions

```

## Visual Basic .NET Example

```

Dim lResult As Long
Dim iNumberOfErrors As Integer

lResult = icsneoClosePort(m_hObject, iNumberOfErrors) 'Close port command
If CBool(lResult) Then 'Check for bad close
    MsgBox("Port Closed OK!")
Else
    MsgBox("Problem Closing Port")
End If
m_bPortOpen = False

```

## 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 destroys the neoVI driver object and frees associated memory.**

### C/C++ Declare

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

### Visual Basic Declare

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

### Visual Basic .NET Declare

```
Public Declare Sub icsneoFreeObject Lib "icsneo40.dll" (ByVal hObject As Integer)
```

### C# Declare

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

## Parameters

*hObject*

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

## Return Values

None.

## Remarks

This should be called before the application exits or the handle variable will be destroyed. Failure to call this method will result in a memory leak.

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
```

*Last Updated : Monday, December 08, 2003*

## GetMessages 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 messages from the neoVI hardware.

### 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 Integer, ByRef pMsg As icsSpyMessage, _
    ByRef pNumberOfMessages As Integer, ByRef pNumberOfErrors As Integer) As Integer
```

### C# Declare

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

## Parameters

#### *hObject*

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

#### *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 should be sized to fit up to 20,000 messages.

#### *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. You can read out the errors by calling the [GetErrorMessages](#) method.

## Return Values

If the read was done successfully, the return value will be non-zero.

## 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 20000) 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 = GetMessages(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[19999]; // 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

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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, December 08, 2003*

## 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 char TimeStampHardwareID;
    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;
    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 char TimeStampHardwareID;
    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
    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 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 icsSpyMessageJ1850
{
    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 use 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 there 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> .
TimeHardware TimeHardware2	<p>This is the hardware time stamp. The <code>TimeStamp</code> is reset to zero every time the <a href="#">OpenPort</a> method is called.</p> <p>For the <b>neoVI</b> hardware, <code>TimeHardware2</code> is more significant than <code>TimeHardware</code>. The resolution of <code>TimeHardware</code> is 1.6µs and and <code>TimeHardware2</code> is 104.8576 ms. To calculate the time of the message in seconds use the following formula: "<code>TimeStamp (sec) = TimeHardware2* 0.1048576 + TimeHardware2 * 0.0000016</code>".</p> <p>For the <b>neoVI PRO or ValueCAN</b> hardware, <code>TimeHardware2</code> is more significant than <code>TimeHardware</code>. The resolution of <code>TimeHardware</code> is 1.0µs and and <code>TimeHardware2</code> is 65.536 ms. To calculate the time of the message in seconds use the following formula: "<code>TimeStamp (sec) = TimeHardware2* 0.065536 + TimeHardware2 * 0.000001</code>".</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 stored in the Header(1 to 4) array.
NumberBytesData	Holds the number of bytes in the Data(1 to 8) array or the number of bytes in a CAN remote frame (The DLC).
DescriptionID	Not Used in the neoVI API.
Header(1 To 4) or ArbIDOrHeader	Holds up to 3 byte 1850 header (bytes 1 through 3) or a 29 bit CAN header.
Data(1 To 8)	Holds the 8 data bytes in CAN messages or bytes 4 through 11 in J1850/ISO messages.
AckBytes(1 To 8)	Not Used in the neoVI API.
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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 11, 2003*



## 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](#) which 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
icsSpyStatusGPSDataValue = 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
    icsSpyStatusGPSDataValue = 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_TXMSG	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_AVSIREC_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.
SPY_STATUS_INIT_MESSAGE	icsSpyStatusInitMessage	This is set if the transmitted message should 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 04, 2003*

## 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 lNetwork As Long, ByVal lNumMessages As Long) As Long
```

### Visual Basic .NET Declare

```
Public Declare Function icsneoTxMessages Lib "icsneo40.dll" (ByVal hObject As Integer,
ByRef pMsg As icsSpyMessage, ByVal iNetwork As Integer, ByVal iNumMessages As Integer)
As Integer
```

### C# Declare

```
[DllImport("icsneo40.dll")]
public static extern int icsneoTxMessages(int hObject, ref icsSpyMessage pMsg, int
iNetwork, int iNumMessages);
```

## Parameters

#### *hObject*

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

#### *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. Can be one of the following values:  
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. NETID\_DEVICE  
transmits on to the neoVI Device Virtual Network (see users manual).

#### *lNumMessages*

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

## Return Values

If the transmit operation was started successfully, the return value will be non-zero.

## 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](#) API call. 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.ArbIDOrHeader = 0xFF;
stMsg.NumberBytesData = 4;
stMsg.Data[0] = 0x22;
stMsg.Data[1] = 0x52;
stMsg.Data[2] = 0x12;
stMsg.Data[3] = 0x28;
```

```

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

// Transmit the message on high speed can
iResult = icsneoTxMessages(hObject,&stMsg,NETID_HSCAN,1);
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.ArbIDOrHeader = Val("&H" & txtArbID.Text) 'Set ArbID in structur

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.ArbIDOrHeader = 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 function
iResult=icsNeoDll.icsneoTxMessages(m_hObject,ref stMessagesTx,1,0);

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

```

*Last Updated : Monday, December 01, 2003*

# 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
stMsgJ1850.Data(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 2000-2004 Intrepid Control Systems, Inc.**

Last Update: *Wednesday, May 28, 2003*

## 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 p_lErrorsMsg As Long, ByRef lNumberOfErrors As Long) As Long
```

### Visual Basic .NET Declare

```
Public Declare Function icsneoGetErrorMessages Lib "icsneo40.dll" (ByVal hObject As Integer, ByRef p_lErrorsMsg As Integer, ByRef iNumberOfErrors As Integer) As Integer
```

### C# Declare

```
[DllImport("icsneo40.dll")]
public static extern int icsneoGetErrorMessages(int hObject, ref int p_lErrorsMsg, ref int lNumberOfErrors);
```

### Parameters

#### *hObject*

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

#### *p\_lErrorsMsg*

[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. A separate topic describes the possible values for [error messages](#). You can get a text description of this error using the [GetErrorInfo](#) method.

#### *lNumberOfErrors*

[out] Specifies the number of errors copied into the p\_lErrorMsg buffer described above. The maximum value will be 600.

### Return Values

If the read was done successfully, the return value will be non-zero.

### 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
Dim lCount As Integer '//Counter
Dim sErrorShort As String '//String Holding Short Error Name
Dim sErrorLong As String '//String Holding Long Error Name
Dim lSeverity As Integer '//Storage for Level of error
Dim bRestart As Boolean '//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,
lSeverity, bRestart)
        lstErrorHolder.Items.Add(sErrorShort & " - " & sErrorLong & " - Errornum: "
Next lCount
    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);
```

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Monday, December 22, 2003*



## Error Messages - intrepidcs API

[Main](#)

Table 1 lists the possible error messages returned in the [GetErrorMessages](#) API call.

**Table 1 - Error Messages**

Error	Description
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 its 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_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.
NEOVI_ERROR_DLL_RS232_CREATE_FILE=33	Problem opening RS232 port. This is probably caused by 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 powered down when in RS232 mode. It will also be 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
NEOVI_ERROR_DLL_RX_MSG_CHK_SUM_ERR=41	A neoVI message packet was properly formatted b
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
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW=45	Each network has a FIFO on the Main51 controller faster than neoVI can transmit them, you will rece
NEOVI_ERROR_DLL_MAIN51_DEV_FIFO_OVERFLOW=46	This is a FIFO over error related to messages pass uController
NEOVI_ERROR_DLL_RESET_STATUS_CHANGED=47	The neoVI reset status has changed. This error w
NEOVI_ERROR_DLL_ISO_LONG_CACHE_OVERFLOW=48	
NEOVI_ERROR_DLL_ISORX_LONG_BUFFER_ALLOC=49	
NEOVI_ERROR_DLL_J1708_LONG_CACHE_OVERFLOW=50	
NEOVI_ERROR_DLL_J1708_LONG_BUFFER_ALLOC=51	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_DEVICE=52	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_HSCAN=53	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_MSCAN=54	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_SWCAN=55	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_LSFTCAN=56	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_FORDSCP=57	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_J1708=58	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_AUX=59	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_JVPW=60	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_ISO=61	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_ISOPIC=62	
NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_MAIN51=63	

NEOVI_ERROR_DLL_MAIN51_TX_FIFO_OVERFLOW_HOST=64	
NEOVI_ERROR_DLL_READ_ENTIRE_DEEPROM_ERROR=65	
NEOVI_ERROR_DLL_WRITE_ENTIRE_DEEPROM_ERROR=66	
NEOVI_ERROR_DLL_USB_PORT_ALREADY_OPEN=67	
NEOVI_ERROR_DLL_JVPW_TX_REPORT_FIFO_ERR_IN=68	
NEOVI_ERROR_DLL_ISOJ_TX_REPORT_FIFO_ERR_IN=69	
NEOVI_ERROR_DLL_JVPW_TX_REPORT_FIFO_ERR_OUT=70	
NEOVI_ERROR_DLL_ISOJ_TX_REPORT_FIFO_ERR_OUT=71	
NEOVI_ERROR_DLL_MAIN51_TX_IN_FROM_HOST_FIFO=72	
NEOVI_ERROR_DLL_MAIN51_TX_HOST_CHKSUM=73	
NEOVI_ERROR_DLL_ISOJ_TX_HOST_MISSED_BYTE=74	
NEOVI_ERROR_DLL_NEOVI_NO_RESPONSE=75	
NEOVI_ERROR_DLL_RX_SOCKET_FIFO_OVER=76	
NEOVI_ERROR_DLL_PROBLEM_STOPPING_TXSOCKET_THREAD=77	
NEOVI_ERROR_DLL_PROBLEM_STOPPING_RXSOCKET_THREAD=78	
NEOVI_ERROR_DLL_PROBLEM_STOPPING_TXSOCKET_CLIENT_THREAD=78	
NEOVI_ERROR_DLL_PROBLEM_STOPPING_RXSOCKET_CLIENT_THREAD=79	
NEOVI_ERROR_DLL_TCP_CLIENT_TX=80	
NEOVI_ERROR_DLL_TCP_CLIENT_RX=81	
NEOVI_ERROR_DLL_TCP_CLIENT_RX SOCK=82	
NEOVI_ERROR_DLL_UNABLE_CONNECT_TO_SRVR=83	
NEOVI_ERROR_DLL_UNABLE_CREATE_CLIENT_SOCK=84	
NEOVI_ERROR_DLL_UNABLE_WSASTARTUP=85	
NEOVI_ERROR_DLL_SOCK_CL_RD_BUFFER_ALLOC=86	
NEOVI_ERROR_DLL_SOCK_CL_TX_BUFFER_ALLOC=87	
NEOVI_ERROR_DLL_SOCK_SRVR_RX_BUFFER_ALLOC=88	
NEOVI_ERROR_DLL_SOCK_SRVR_TX_BUFFER_ALLOC=89	
NEOVI_ERROR_DLL_ILLEGAL_TX_NETWORK=90	
NEOVI_ERROR_DLL_MAIN51_TX_HOST_OVERRUN=91	
NEOVI_ERROR_DLL_OPEN_GET_COMM_TIMEOUT=92	
NEOVI_ERROR_DLL_OPEN_SET_COMM_TIMEOUT=93	
NEOVI_ERROR_DLL_OPEN_READ_DEVICE_TYPE=94	
NEOVI_ERROR_DLL_OPEN_READ_DEVICE_TYPE_TOUT=95	
NEOVI_ERROR_DLL_CLOSE_PURGE_COMM=96	
NEOVI_ERROR_DLL_TX_COM_FIFO_OVERFLOW=97	
NEOVI_ERROR_DLL_GET_USB_SERIAL_DEVICES=98	



## 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

```
Private 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

```
Private Declare Function icsneoGetErrorInfo Lib "icsneo40.dll" _
    (ByVal iErrorNumber As Integer, ByVal sErrorDescriptionShort As String, _
    ByVal sErrorDescriptionLong As String, ByRef iMaxLengthShort As Integer, _
    ByRef iMaxLengthLong As Integer, ByRef lErrorSeverity As Integer, _
    ByRef lRestartNeeded As Integer) As Integer
```

### C# Declare

```
[DllImport("icsneo40.dll", ExactSpelling = true, CharSet = CharSet.Ansi ,
CallingConvention = CallingConvention.Winapi)]
public static extern int icsneoGetErrorInfo(int iErrorNumber , StringBuilder
sErrorDescriptionShort, StringBuilder sErrorDescriptionLong, ref int iMaxLengthShort,
ref int iMaxLengthLong, ref int lErrorSeverity , ref int lRestartNeeded);
```

### Parameters

#### *lErrorNumber*

[in] This is the number of the error message returned from the [GetErrorMessages](#) API call. 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.

#### *lErrorSeverity*

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

#### *lRestartNeeded*

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

## Return Values

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

## Remarks

None.

**Table 1 - Descriptions of Error Severity**

Error Severity	Description
<code>const unsigned long icsspyErrCritical=0x10;</code>	A critical error which affects operation or accuracy
<code>const unsigned long icsspyErrExclamation=0x30;</code>	An important error which may be critical depending on the application.
<code>const unsigned long icsspyErrInformation=0x40;</code>	An error which probably does not need attention.
<code>const unsigned long icsspyErrQuestion=0x20;</code>	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, sErrorShort As String, _
    sErrorLong As String, lSeverity As Long, bRestart As Boolean) 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 cmdReadErrors_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 Boolean

'// 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
    lstErrors.Clear
    For lCount = 1 To lNumberOfErrors
        Call icsneoGetDLLErrorInfo(lErrors(lCount - 1), sErrorShort, sErrorLong,
lSeverity, bRestart)
        lstErrors.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

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

```

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

    Dim lErrorLongLength As Integer
    Dim lErrorShortLength As Integer
    Dim lRestart As Integer
    Dim lResult As Integer

    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 Integer '///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 lSeverity As Integer '///Storage for Level of error
    Dim bRestart As Boolean '///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
        lstErrorHandler.Items.Clear()
        For lCount = 1 To lNumberOfErrors
            Call icsneoGetDLLErrorInfo(lErrors(lCount - 1), sErrorShort, sErrorLong,
lSeverity, bRestart)
            lstErrorHandler.Items.Add(sErrorShort & " - Description" & sErrorLong & " -
Errornum: " & lErrors(lCount - 1))
        Next lCount
    End If
End Sub

```

### 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
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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 16, 2003*

## GetErrorInfoW Method - intrepidcs API

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

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

### C/C++ Declare

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

### Parameters

#### *lErrorNumber*

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

#### *szErrorDescriptionShort*

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

#### *szErrorDescriptionLong*

[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.

#### *lErrorSeverity*

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

#### *lRestartNeeded*

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

### Return Values

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

### Remarks

GetErrorInfoW functions the same way as GetErrorInfo except for GetErrorInfoW returns WCHAR pointers to support unicode.

**Table 1 - Descriptions of Error Severity**

Error Severity	Description
<code>const unsigned long icsspyErrCritical=0x10;</code>	A critical error which affects operation or accuracy
<code>const unsigned long icsspyErrExclamation=0x30;</code>	An important error which may be critical depending on the application.
<code>const unsigned long icsspyErrInformation=0x40;</code>	An error which probably does not need attention.
<code>const unsigned long icsspyErrQuestion=0x20;</code>	An error which is not understood.

### Examples

#### 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);
        icsneoGetErrorInfoW(iErrors[lCount],szDescriptionShort,szDescriptionLong,
&lMaxLengthShort,&lMaxLengthLong,&lErrorSeverity,&lRestartNeeded);

        OutputDebugString(szDescriptionShort);
        OutputDebugString(TEXT("\n"));
    }
}
else
    OutputDebugString(TEXT("No Errors to report\n"));

```

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, October 12, 2006*

## 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, ByVal p_bData As Byte, ByVal lNumBytes As Long) As Long
```

### C# Declare

```
[DllImport("icsneo40.dll")]
public static extern int icsneoSendConfiguration(int hObject, ref byte p_bData, int iNumBytes);
```

### Visual Basic .NET Declare

```
Public Declare Function icsneoSendConfiguration Lib "icsneo40.dll" _
    (ByVal hObject As Integer, ByVal p_bData As Byte, ByVal iNumBytes As Integer) As Integer
```

### Parameters

#### *hObject*

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

#### *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.

#### *lNumBytes*

[in] Specifies which neoVI driver to use. This should always be set to INTREPIDCS\_DRIVER\_STANDARD (0).

### Return Values

If the configuration has been updated successfully the return value will be 1. If the function fails the return value will be zero.

### Remarks

The SendConfiguration method will only update the configuration defined in the [Configuration Array](#) topic. It will also apply checking to the data so that neoVI is not programmed to an illegal state. For example, setting the CAN controller to an illegal operating mode. You must disable communication with the [EnableNetworkCom](#) API before this API will operate.

---

### Examples

#### Visual Basic Example

```

Private m_hObject As Long      '// Declared at form level

Dim 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

icsneoEnableNetworkCom(m_hObject,0)
lResult = icsneoSendConfiguration(m_hObject, bConfigBytes(0), lNumBytes)
icsneoEnableNetworkCom(m_hObject,1)

'// 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;

        icsneoEnableNetworkCom(m_hObject,0);
        lResult = icsneoSendConfiguration(hObject, bConfigBytes, iNumConfigBytes);
        icsneoEnableNetworkCom(m_hObject,1);

        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
Call icsneoEnableNetworkCom(m_hObject, 0)
lResult = icsneoSendConfiguration(m_hObject, bConfigBytes(0), iNumBytes)
Call icsneoEnableNetworkCom(m_hObject, 1)

'// 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
icsNeoDll.icsneoEnableNetworkCom(m_hObject,0);
lResult = icsNeoDll.icsneoSendConfiguration(m_hObject, ref bConfigBytes[0], iNumBytes);
icsNeoDll.icsneoEnableNetworkCom(m_hObject,0);

// 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, April 19, 2007*

## 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.

### C/C++ Declare

```
int _stdcall icsneoGetConfiguration(int hObject, unsigned char * pData, int *
p_lNumBytes);
```

### Visual Basic Declare

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

### Visual Basic .NET Declare

```
Public Declare Function icsneoGetConfiguration Lib "icsneo40.dll" _
    (ByVal hObject As Integer, ByRef p_bData As Byte, _
    ByRef iNumBytes As Integer) As Integer
```

### C# Declare

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

## Parameters

#### *hObject*

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

#### *p\_lNumBytes*

[in/out] This argument should be passed as the size of your array. On return it indicates how many bytes that were written. Since the API currently only supports 1024 bytes this API should always be set to 1024.

## Return Values

If the the configuration has been read successfully the return value will be 1. If the function fails the return value will be zero.

## 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 23, 2003*

Configuration Array - intrepidcs API

[Parameters](#) - [Remarks](#) - [C/C++ example](#) - [VB example](#)

The configuration array allows the software user to change neoVI options via the programming environment

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.	<div>Bitfield for General IO</div> <table><tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>X</td><td>X</td><td>Misc2 Output Default Value  0=off  1=0n</td><td>Misc1 Output Default Value  0=off  1=0n</td><td>X</td><td>X</td><td>MISC 2 Data Direction  0=Output  1=Input</td><td>MISC 1 Data Direction  0=Output  1=Input</td></tr></table>	7	6	5	4	3	2	1	0	X	X	Misc2 Output Default Value  0=off  1=0n	Misc1 Output Default Value  0=off  1=0n	X	X	MISC 2 Data Direction  0=Output  1=Input	MISC 1 Data Direction  0=Output  1=Input
7	6	5	4	3	2	1	0												
X	X	Misc2 Output Default Value  0=off  1=0n	Misc1 Output Default Value  0=off  1=0n	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_CANN_CNF1, NEO_CFG_MPIC_LSFT_CANN_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_CANN_CNF2, NEO_CFG_MPIC_LSFT_CANN_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_CANN_CNF3, NEO_CFG_MPIC_LSFT_CANN_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_CANN_MODE, NEO_CFG_MPIC_LSFT_CANN_MODE	Sets the mode of CAN controller	See Notes	<table><tr><th>Mode</th><th>Value</th></tr><tr><td>Loopback</td><td>4</td></tr><tr><td>ListenOnly</td><td>2</td></tr><tr><td>Normal</td><td>1</td></tr></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.	<table><tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>Dynamically Adjust Address  0=No 1=Yes</td><td>Bit Rate  0 = 41.6K 1=83.3Kbps</td></tr></table>	7	6	5	4	3	2	1	0	X	X	X	X	X	X	Dynamically Adjust Address  0=No 1=Yes	Bit Rate  0 = 41.6K 1=83.3Kbps
7	6	5	4	3	2	1	0												
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.	<table><tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>1</td><td>0</td><td>1</td><td>Tx Driver B Enable</td><td>Tx Driver A Enable</td><td>Rx B Enable</td><td>Rx Diff Enable</td><td>Rx A Enable</td></tr></table>	7	6	5	4	3	2	1	0	1	0	1	Tx Driver B Enable	Tx Driver A Enable	Rx B Enable	Rx Diff Enable	Rx A Enable
7	6	5	4	3	2	1	0												
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_ENTRIES	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_EADDR (ISO)																			
ISO_TX_IFS_TIME_EADDR (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_SETTING 1 (ISO)			
EE_DATA_INIT_NUM_STEPS (ISO)			
EE_DATA_INIT_STEP1_COUNT (ISO)			
EE_DATA_INIT_STEP1_IO (ISO)			
EE_DATA_INIT_STEP2_COUNT (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

XX.

---

## 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 bSCPIDIDs(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, bSCPIDs, &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);
    }

```



## 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 DLL as a 32 bit integer.

### 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 09, 2003*

## 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 with the [OpenPort](#) method.

#### *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 in the [OpenPort](#) method.

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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Tuesday, December 30, 2003*

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

### 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](#) method.

---

### 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
```



## 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_CFMMsgFilter,
                                           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, _
    pFF_CFMMsgFilter As spyFilterLong, _
    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_CFMMsgFilter 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, out spyFilterLong pFF_CFMMsgFilter, out icsSpyMessage pFlowCTxMsg, int lCFTIMEoutMs, int lFlowCBlockSize, int lUsesExtendedAddressing, int lUseHardwareIfPresent);
```

### Parameters

#### *hObject*

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

#### *lNetwork*

[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.

#### *lEnable*

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

#### *pFF\_CFMMsgFilter*

[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 IFlowCBlockSize argument.

#### *ICFTimeOutMs*

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

#### *IFlowCBlockSize*

[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).

#### *IUsesExtendedAddressing*

[in] Indicates the 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 = &HFFF

'// 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_CFMSGFilter;
icsSpyMessage FlowCTxMsg;

// reset the structures to zero

memset(&FF_CFMSGFilter,0,sizeof(FF_CFMSGFilter));
memset(&FlowCTxMsg,0,sizeof(FlowCTxMsg));

// setup the filter

FF_CFMSGFilter.Header = 0x7E8;
FF_CFMSGFilter.HeaderMask = 0xFFF;
FlowCTxMsg.StatusBitField=2;
FlowCTxMsg.NumberBytesData =8;
FlowCTxMsg.ArbIDOrHeader = 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_CFMSGFilter, &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.ArbIDOrHeader = 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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 24, 2003*

## 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 lNetwork, int lReserved1, int lClearRxStatus,
                                     int * lReserved2,
                                     int * lRxStatus)
```

### 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 lReserved2 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.

#### *lNetwork*

[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.

#### *lReserved1*

[in] Set to zero.

#### *lClearRxStatus*

[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.

#### *lRxStatus*

[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 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Wednesday, December 24, 2003*

## 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 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

#### *lDriverType*

[in] Specifies which neoVI driver to use. This should always be set to INTREPIDCS\_DRIVER\_STANDARD (0).

#### *lGetSerialNumbers*

[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.

#### *lStopAtFirst*

[in] Indicates whether the function should stop at the first device found (lStopAtFirst=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\_lDeviceTypes*

[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\_NEO4 (0), INTREPIDCS\_DEVICE\_VCAN (1), or INTREPIDCS\_DEVICE\_NEO6 (2).

#### *p\_lComPorts*

[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\_lSerialNumbers*

[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_NEO4
                    lstCOMDevices.AddItem "neoVI 4 SN: " & lSerialNumbers(lCount) & " on
```

```

COM" & lComPorts(lCount)
    Case INTREPIDCS_DEVICE_NEO6
        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]; //Array 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
```

*Last Updated : Monday, December 08, 2003*

## FindAllUSBDevices 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 number of hardware devices connected to the PC.**

### C/C++ Declare

```
int _stdcall icsneoFindAllUSBDevices(int lDriverType,
    int iGetSerialNumbers,
    int *iDevices,
    int *iSerialNumbers,
    int *iOpenedStates,
    int *iNumDevices);
```

### Visual Basic Declare

```
Public Declare Function icsneoFindAllUSBDevices Lib "icsneo40.dll" _
    (ByVal lDriverID As Long, ByVal lGetSerialNumbers As Long, ByRef p_lDevices
As Long, _
    ByRef p_lSerialNumbers As Long, ByRef p_lOpenedDevices As Long, ByRef
lNumDevices As Long) As Long
```

### Visual Basic .NET Declare

```
Public Declare Function icsneoFindAllUSBDevices Lib "icsneo40.dll" _
    (ByVal lDriverID As Integer, ByRef lGetSerialNumbers As Integer, ByRef p_lDevices As
Integer, _
    ByRef p_lSerialNumbers As Integer, ByRef p_lOpenedDevices As Integer, ByRef
lNumDevices As Integer) As Integer
```

### C# Declare

```
[DllImport("icsneo40.dll")]
public static extern int icsneoFindAllUSBDevices(int lDriverID, int lGetSerialNumbers,
ref int p_lDevices, ref int p_lSerialNumbers, ref int p_lOpenedDevices, ref int
lNumDevices);
```

### Parameters

#### *lDriverType*

[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.

#### *iDevices*

[out] Pointer to array of at least 128 elements. This array will be filled in with the valid USB device numbers found. The number of device numbers will be loaded in the iNumDevices parameter.

#### *iSerialNumbers*

[out] Pointer to array of at least 128 elements. This array will be filled in with the serial numbers of each connected device.

#### *iOpenedStates*

[out] Pointer to array of at least 128 elements. This array will be filled in with a 1 if the device is

currently opened.

#### *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

```
Dim lResult As Long 'Holds the Result of the function call
Dim iDevices(0 To 127) As Long 'Array containing the
Dim iSerialNumbers(0 To 127) As Long 'Array holding the serial numbers of attached
devices
Dim iOpenedStatus(0 To 127) As Long 'Array holding the Port status of devices
Dim iNumDevices As Long 'Tells the number of attached devices
Dim Counter As Integer

'Find all USB Device and get serial numbers
lResult = icsneoFindAllUSBDevices(INTREPIDCS_DRIVER_STANDARD, 1, _
    iDevices(0), iSerialNumbers(0), iOpenedStatus(0), iNumDevices)

If lResult = 1 Then

    For Counter = 0 To 127
        'Display the gathered information in a list box
        lstUsbDevices.AddItem ("USB #" & iDevices(Counter) & " SN-" & _
            iSerialNumbers(Counter) & " State-" & iOpenedStatus(Counter))
    Next Counter

Else
    MsgBox ("Problem Getting Device Information")
End If
```

### C/C++ Example

XX

### C# Example

```
int lResult = 0; //Storage for call result
int[] iDevices = new int[127]; //Array for Device Numbers
int[] iSerialNumbers = new int[127]; //Array for Serial numbers of devices
int iNumDevices = 0; //Storage number of devices
int[] iOpenedStatus = new int[127]; //Array holding Open Status of Devices
int Counter = 0; //Counter for Position indication

//Call for getting All of the USB device information
lResult =
icsNeoDll.icsneoFindAllUSBDevices(Convert.ToInt32(eDRIVER_TYPE.INTREPIDCS_DRIVER_STANDAR
D),1,ref iDevices[0],ref iSerialNumbers[0],ref iOpenedStatus[0],ref iNumDevices);

//check results and act accordingly
```

```

if(lResult==1)
{
    //Fill List Box with USB Device information
    for(Counter=0;Counter<127; Counter++)
    {
        lstUsbDevices.Items.Add("USB #" + Convert.ToString(iDevices[Counter]) + " SN-" +
        Convert.ToString(iSerialNumbers[Counter]) +
        " State-" + Convert.ToString(iOpenedStatus[Counter]));
    }
}
else
{
    MessageBox.Show("Could Not Find anything")
}

```

## Visual Basic .NET Example

```

Dim lResult As Integer '///Storage for call result
Dim iDevices(127) As Integer '///Array for Device Numbers
Dim iSerialNumbers(127) As Integer '///Array for Serial numbers of devices
Dim iOpenedStatus(127) As Integer '///Array holding Open Status of Devices
Dim iNumDevices As Integer '///Storage number of devices
Dim Counter As Integer '///Counter for Position indication

//Call for getting All of the USB device information
lResult = icsneoFindAllUSBDevices(INTREPIDCS_DRIVER_STANDARD, 1, iDevices(0),
iSerialNumbers(0), iOpenedStatus(0), iNumDevices)
//check results and act accordingly
If lResult = 1 Then
    //Fill List Box with USB Device information
    For Counter = 0 To 127
        lstUsbDevices.Items.Add("USB #" + Convert.ToString(iDevices(Counter)) _
        + " SN-" + Convert.ToString(iSerialNumbers(Counter)) + " State-" _
        + Convert.ToString(iOpenedStatus(Counter)))
    Next Counter
Else
    MsgBox("Could not find anything")
End If

```

## 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
    txtOverflowCount.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);
}
```

## WIN32 API Examples Overview - intrepidcs API

This section includes examples for all of the following: 1) [Visual Basic](#), 2) [Visual C++](#), 3) [Visual Basic.NET](#), 4) [C#](#) 5) [Borland C++ Builder](#), 6) [LabVIEW](#), and 7) [LabWindows CVI](#).

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Thursday, December 11, 2003*

## Visual Basic Example - intrepidcs API

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 \(10kB\)](#)

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

The screenshot shows a Visual Basic application window titled "neoVI VB Example for WIN32 DLL". The interface is divided into several sections:

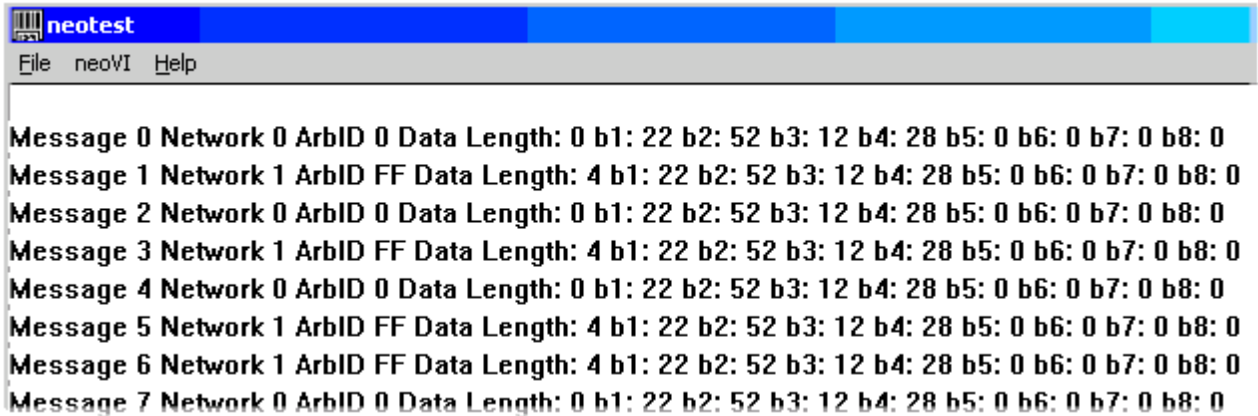
- Connection:** Contains buttons for "Open neoVI" and "Close neoVI". It includes radio buttons for "NeoVI (USB)", "NeoVI / ValueCAN (Serial)", and "TCP/IP IP Address". The "Start Stop Server" checkbox is checked, and the "Port Number" is set to 1. The "IP Address" is set to 255.255.255.255. The "Connect to first device found" radio button is selected.
- Transmit Messages:** Includes a dropdown for "Arb ID in Hex for CAN" (set to FF), a dropdown for "HSCAN", and a checkbox for "Send Extended CAN ID". Below this is a "Data In Hex" section with a dropdown (set to 0) and a row of 16 hex input fields, all containing FF. A "Transmit" button is at the bottom.
- Receive Messages:** Includes a "ReadMessages" button, an "AutoRead" checkbox, and two labels: "Label3" (blue) and "Label3" (red). Below this is a large empty text area.
- Errors:** Includes a "Read Errors" button and a large empty text area.
- Multiframe Rx:** Includes a "Network" dropdown (set to HSCAN), a "First Frame Filter (Hex)" input field (set to 777), a "Flow Control ID (Hex)" input field (set to 641), and buttons for "Read Status" and "Clear Status".
- Neo Information:** Includes a label for "ICSNeo40.dll Version" and a "Find All Attached Devices" button.
- USB Devices:** A large empty text area.
- Comm Type Devices:** A large empty text area.
- Performance:** Includes a "Setup And Enable" button, a "Disable" button, and a "Get Performan" button.

Figure 1 - The Visual Basic Example.

## 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.

The image shows a screenshot of a Windows application window titled "neotest". The window has a menu bar with "File", "neoVI", and "Help". The main content area displays a list of eight messages, each with its ID, network, arbitration ID, data length, and a list of eight data bytes (b1 through b8).

```
Message 0 Network 0 ArbID 0 Data Length: 0 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 1 Network 1 ArbID FF Data Length: 4 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 2 Network 0 ArbID 0 Data Length: 0 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 3 Network 1 ArbID FF Data Length: 4 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 4 Network 0 ArbID 0 Data Length: 0 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 5 Network 1 ArbID FF Data Length: 4 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 6 Network 1 ArbID FF Data Length: 4 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
Message 7 Network 0 ArbID 0 Data Length: 0 b1: 22 b2: 52 b3: 12 b4: 28 b5: 0 b6: 0 b7: 0 b8: 0
```

Figure 1 - The Visual C++ Example.

## Visual Basic Dot Net 2003 Example - intrepidcs API

A Visual Basic Dot Net 2003 example (Figure 1) is included to show how the API all works together. The project consists of Eight Files files. 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 8 project files are included in the following file: [VcanDotNetVB.zip \(13kB\)](#)

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

The screenshot displays the 'ICS API Dot Net VB Example for HighSpeed CAN' application window. The interface is divided into several sections:

- Top Left:** 'Connect' and 'Disconnect' buttons.
- Top Center:** 'Port Number' (text box with '1'), radio buttons for 'USB (NeoVI)' (selected), 'Serial (NeoVI, and VCAN)', and 'TCP/IP', and an 'IP Address' text box with '255.255.255.255'.
- Top Right:** 'Start Stop Server' checkbox and a text box with '4500'.
- Right Panel:** 'Neo Information' section with 'ICSNeo40.dll version' and 'Version' text boxes, a 'find All Devices' button, 'USB Devices' list box, and 'Comm Based devices' list box.
- Left Panel:** 'Arb ID' text box with '101', 'Data Bytes' section with eight '00' text boxes, and 'Number of Data Bytes' text box with '8'.
- Center:** A large empty text box for messages, with 'Transmit' button to its left, 'Get Messages' button below it, and 'Get Errors' button further down.
- Bottom Left:** 'Multiframe Rx' section with 'First Frame Filter (Hex)' text box containing '777', 'Flow Control ID (Hex)' text box containing '641', 'Setup And Enable' button, and 'Disable' button.
- Bottom Right:** 'Read Status' and 'Clear Status' buttons.

Figure 1 - The Visual Basic Dot Net 2003 Example.

## C# Dot Net 2003 Example - intrepidcs API

A C# Dot Net 2003 example (Figure 1) is included to show how the API all works together. The project consists of Ten files. 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 10 project files are included in the following file: [IcsNeoDotNetCSharp.zip \(51kB\)](#)

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

The screenshot shows a Windows application titled "ICS API Dot Net C# Example for HighSpeed CAN". The interface is divided into several sections:

- Top Left:** "Connect" and "Disconnect" buttons.
- Top Center:** "Port Number" (text box with "1"), radio buttons for "USB (NeoVI)" (selected), "Serial (NeoVI, and VCAN)", and "TCP/IP" (with "255.255.255.255" in the "IP Address" text box). A "Start Stop Server" checkbox and a "4500" text box are also present.
- Left Side:** "Arb ID" (text box with "101"), "Data Bytes" (text box with "8"), and a vertical stack of eight "00" text boxes.
- Center:** A large empty text box, a "Transmit" button, a "Get Messages" button, a text box with a scrollbar, and a "Get Errors" button.
- Bottom Left (Multiframe Rx):** "First Frame Filter (Hex)" (text box with "777"), "Flow Control ID (Hex)" (text box with "641"), "Setup And Enable" button, and "Disable" button.
- Bottom Right:** "Read Status" and "Clear Status" buttons.
- Right Side (groupBox1):** "ICSNeo40.dll version", "Version", "find All Devices", "USB Devices" (empty list box), and "Comm Baised devices" (empty list box).

Figure 1 - The C# Dot Net 2003 Example.

## Borland C++ Builder Example - intrepidcs API

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.

The screenshot displays the 'neoVI Example for Borland C++ Builder' application window. The interface is divided into several functional areas:

- Connection:** Includes buttons for 'Open neoVI' and 'Close neoVI'. Fields for 'Port Type' (set to 'First Device Found'), 'Port' (set to '1'), and 'TCP/IP Address' (set to '255.255.255.255') are present. A 'Start Stop Server' checkbox is set to 4500.
- Transmit Message:** Features an 'Arb ID (Decimal)' field set to '12', a 'Network' dropdown set to 'NETID\_HSC', and a 'Number of Data Bytes' dropdown set to '0'. Below these are fields for H1, H2, H3, B1 through B8, and a 'Count' dropdown set to '1'. A 'J1850 Header' checkbox is also visible. A 'Transmit' button is at the bottom.
- Receive Messages:** A large text area for displaying received messages.
- Errors:** Includes labels for 'lblStatus' and 'lblErrorStatus', a 'Clear' button, and a 'Read Errors' button.
- Performance:** Contains a 'Get Performance' button and fields for 'Buffer Count' (N/A), 'Buffer Max' (N/A), and 'Overflow Count' (N/A).
- GroupBox1:** Contains a 'Version' button, a 'Find All Devices' button, and two empty list boxes for 'USB Devices' and 'Comm Based Devices'.
- NeoConfig Information:** Includes a 'Get Conf' button, three 'CNF' fields (CNF 1: 1, CNF 2: B8, CNF 3: 5), a 'Send HS CAN Info' button, and a 'Read Status' button.
- Multiframe Rx:** Features 'First Frame Filter' (777) and 'Flow Control ID' (641) fields, with 'Setup and Enable' and 'Disable' buttons.

Figure 1 - The Borland C++ Builder Example.

## Borland Delphi Example - intrepidcs API

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.

The screenshot shows a Windows-style application window titled "neoVI Example for Borland Delphi". The interface is divided into several sections:

- Top Left:** "Port Type" dropdown set to "RS232", "Port" dropdown set to "1", and two buttons: "Open neoVI" and "Close neoVI".
- Top Right:** "Errors" label, "Read Errors" button, and a text area. Below this are labels "lblStatus" (blue) and "lblErrorStatus" (red).
- Middle:** "Rx Messages" label, a large empty text area, and a "Clear" button.
- Bottom:** "Network" dropdown set to "NETID\_HSCAN", "Arb ID (Decimal)" text box with "12", and checkboxes for "Xtd" and "Rtn". A "Transmit" button is to the right.
- Bottom Section:** "J1850 Header (Decimal)" dropdown set to "1", and a grid of input boxes for data bytes. The grid has columns for "Count", "H1", "H2", "H3", and then "B1" through "B8". The "Data Bytes (Decimal)" label is to the left of the "Count" box, which contains "0".

**Figure 1 - The Borland Delphi Example.**



## LabVIEW Example - intrepidcs API

A LabVIEW example (Figure 1) is included to show how the API all works together. The example VI is included with the other [neoVI VIs](#) in [labv\\_neo.zip \(168kB\)](#)

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

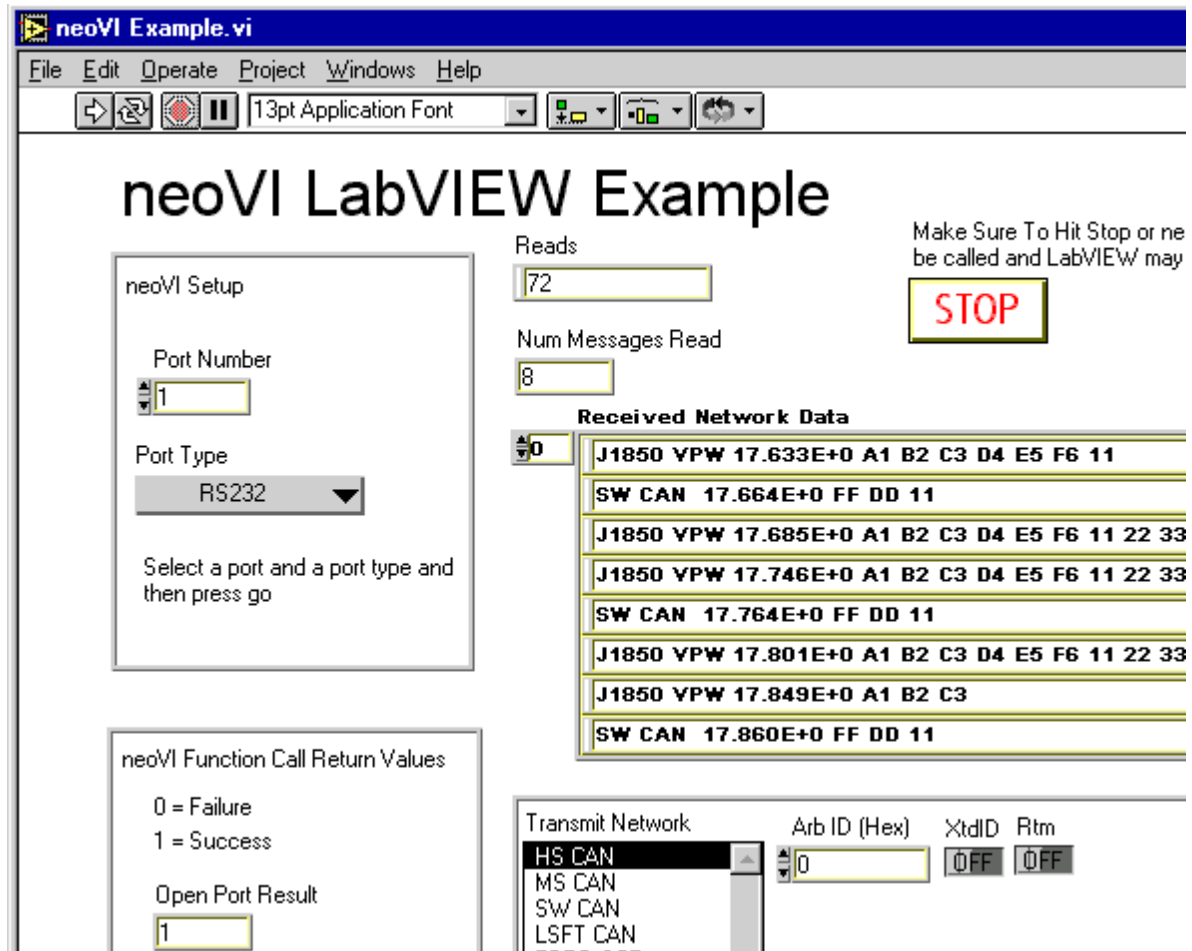


Figure 1 - The LabVIEW Example.

## LabWindows CVI Example - intrepidcs API

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.

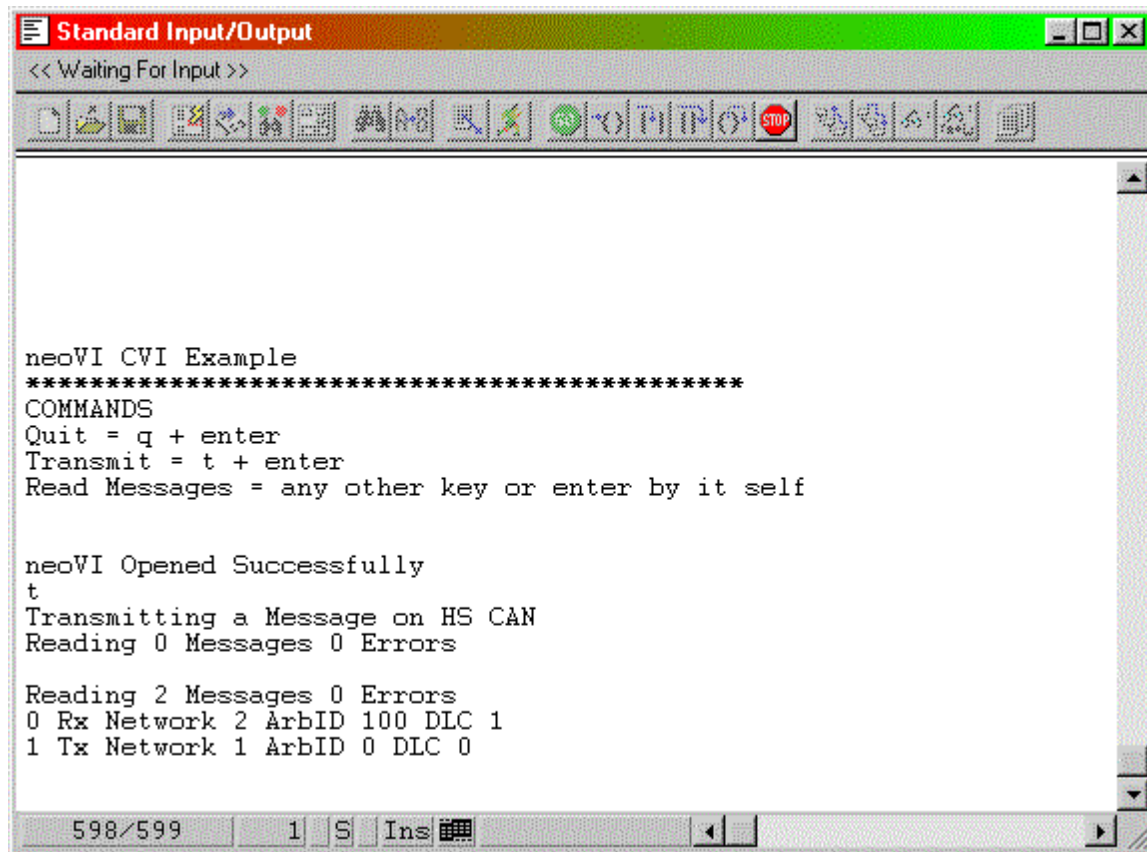


Figure 1 - The LabWindows CVI Example.

# VBA Excel - intrepidcs API

A VBA Excel example (Figure 1) is included to show how the API all works together. This project only has 1 file, NeoVIExample.XIS. Make sure macros are enabled to run this example. All the needed project files are included in the following file: [ExcelVBA.zip \(44kB\)](#)

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

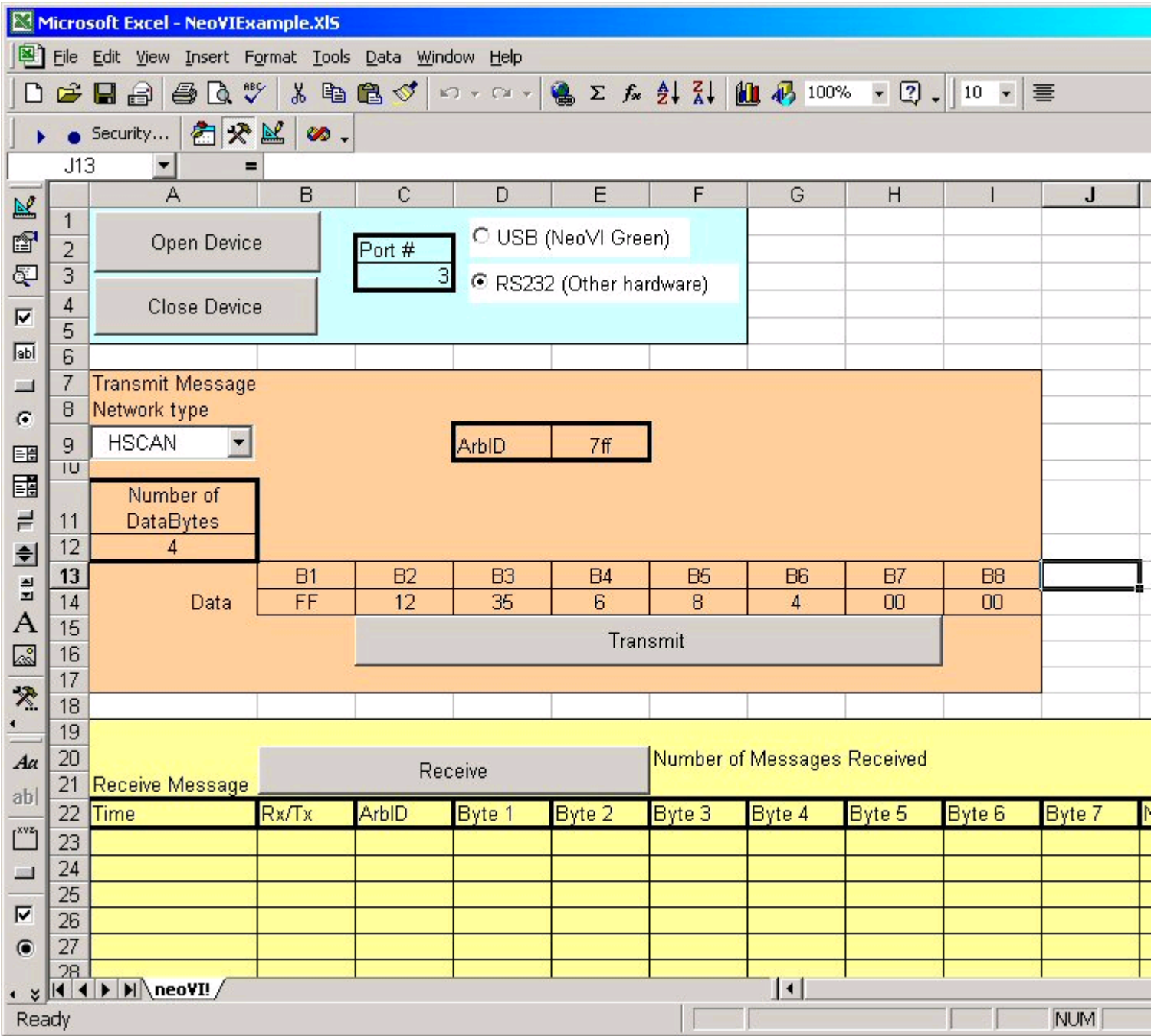


Figure 1 - The VBA Excel Example.

# 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 2000-2004 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 ValueCANor 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 2000-2004 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

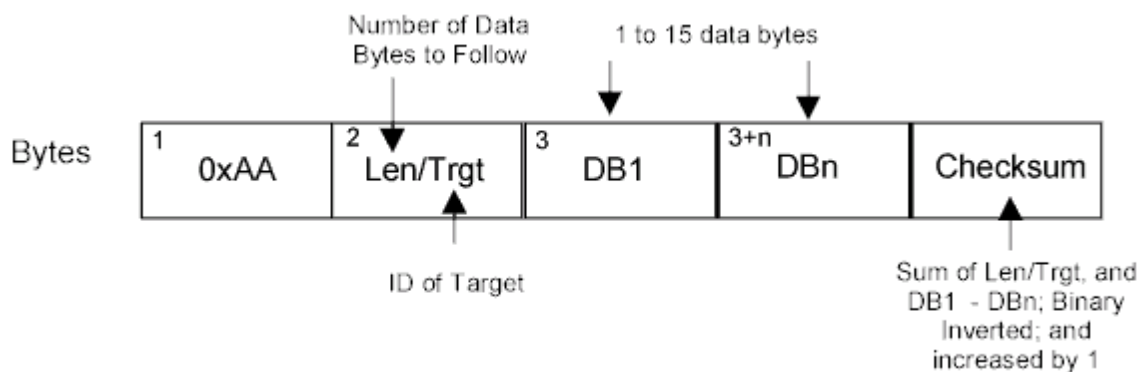
The first field of the communication packet always starts with a hexadecimal 0xAA (decimal 170).

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.**

## 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

# 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



# No Data Message - Intrepidcs API

[Main](#)

This messages 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**

# 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>

# 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	<table><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>SID Bit 2</td><td>SID Bit 1</td><td>SID Bit 0</td><td>SRR</td><td>IDE</td><td>-</td><td>EID Bit 17</td><td>EID Bit 16</td></tr></table> <p><b>SRR</b> = Set to 1 when a remote frame <b>IDE</b> = Set to 0 when this is a standard frame. <b>EID Bit 17 and 16</b> = undefined for a standard ID message</p>	b7	b6	b5	b4	b3	b2	b1	b0	SID Bit 2	SID Bit 1	SID Bit 0	SRR	IDE	-	EID Bit 17	EID Bit 16	<table><tr><td>MS Nibble</td><td>LS Nibble</td></tr><tr><td>Reserved Bits</td><td>DLC</td></tr></table> <p><b>DLC</b> = Data Length Code</p> <p>Reserved Bits = Undefined (Mask off to determine length of data)</p>	MS Nibble	LS Nibble	Reserved Bits	DLC	CAN Data bytes 0 - 8  Remote Frame Have No Data
b7	b6	b5	b4	b3	b2	b1	b0																			
SID Bit 2	SID Bit 1	SID Bit 0	SRR	IDE	-	EID Bit 17	EID Bit 16																			
MS Nibble	LS Nibble																									
Reserved Bits	DLC																									

Table 2 - Extended ID (EID) Frame Format

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTES 9-16
--------	--------	--------	--------	--------	--------	--------	--------	------------



## 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;
```

# Raw CAN Transmit Messages - Intrepidcs API

[Main](#)

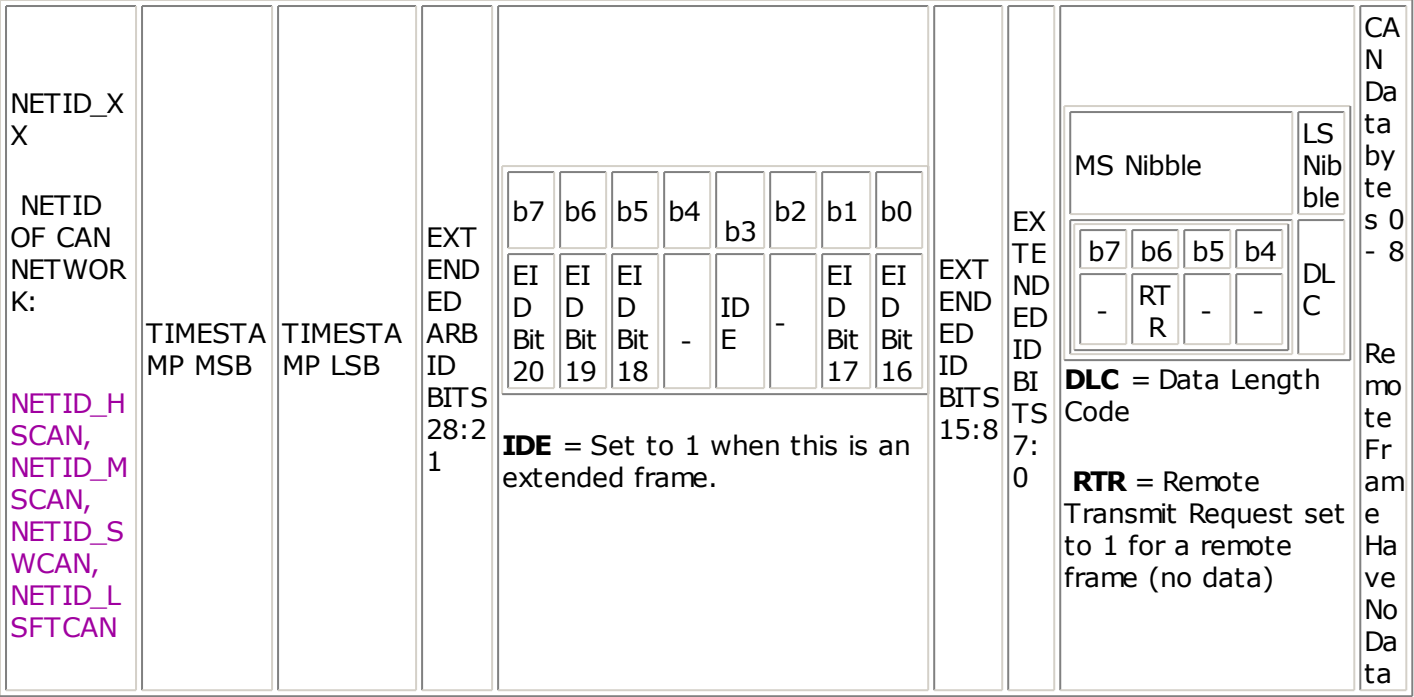
To transmit a message you must assemble a transmit message as shown in tables 1 and 2 below. This is a transmit request. When neoVI has successfully transmitted a message it will send a [transmit complete report](#). If you send messages faster than neoVI can place them on the bus you can cause a [Main51 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	<table><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>SID Bit 2</td><td>SID Bit 1</td><td>SID Bit 0</td><td>-</td><td>ID E</td><td>-</td><td>EID Bit 17</td><td>EID Bit 16</td></tr></table> <b>IDE</b> = Set to 0 when this is a standard frame. <b>EID Bit 17 and 16</b> = undefined for a standard ID message	b7	b6	b5	b4	b3	b2	b1	b0	SID Bit 2	SID Bit 1	SID Bit 0	-	ID E	-	EID Bit 17	EID Bit 16	<table><tr><td colspan="4">MS Nibble</td><td>LS Nibble</td></tr><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td rowspan="2">DLC</td></tr><tr><td>-</td><td>RTR</td><td>-</td><td>-</td></tr></table> <b>DLC</b> = Data Length Code  <b>RTR</b> = Remote Transmit Request set to 1 for a remote frame (no data)	MS Nibble				LS Nibble	b7	b6	b5	b4	DLC	-	RTR	-	-	CAN Data bytes 0 - 8  Remote Frame Have No Data
b7	b6	b5	b4	b3	b2	b1	b0																													
SID Bit 2	SID Bit 1	SID Bit 0	-	ID E	-	EID Bit 17	EID Bit 16																													
MS Nibble				LS Nibble																																
b7	b6	b5	b4	DLC																																
-	RTR	-	-																																	

Table 2 - Extended Arbitration ID (EID) Frame Format

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTES 9-16
--------	--------	--------	--------	--------	--------	--------	--------	------------



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



# 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</a> <a href="#">Error</a> <a href="#">Status</a> <a href="#">Bitfield</a> )

# 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

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
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

# 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

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTES 5-16
				ISO/LIN/KW2K Data bytes 1 - 12 *
				*
				NETID_ISO

# 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					
BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	Byte 6
NETID_ISO	TIMESTAMP MSB	TIMESTAMP LSB	See Table 2 below	00 = TRANS MIT COMPL ETE	
				ISO_LI N_SYN C_BRK_ ERR	
				ISO_LI N_SYN C_WAV _ERR	For Transmi
				ISO_LI N_MSG _ID_PR TY	Comple te:
				ISO_LI N_CHK SUM_E RR	DESCRI PTION ID MSB
				ISO_LI N_TFM AX_ERR	DE PT ID
				ISO_LI N_BIT_ ERROR	
				ISO_LI N_SYN C_LEN_ ERR	

Table 2 - Byte 4 for a Receive Frame

b7	b6	b5	b4	
0	1 Must be one for status message	0 = No Overflow  1 = Rx Overflow	Reserved	

**Lower Nibble**

Number of Bytes to follow

# 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 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

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BY 5-
NETID_ISO	TIMESTAMP MSB	TIMESTAMP LSB	See Table 2 below	Mess Byte inclu ISO/ che m if pres

Table 2 - Byte 4 for a Receive Frame

b7	b6	b5	b4	
1 = Long Message (More Bytes To Follow in consecutive messages)  0 = Single Message or End of Long Message	0  Must be zero for rx frame	0 = No Overflow  1 = Rx Overflow	Reserved	Number of Bytes of frame including checksum if present
Lower Nibble				

# Raw J1850 PWM - Intrepidcs API

[Main](#)

## Overview

Raw communications pa.

**intrepidcs API Documentation - (C) Copyright 2000-2004 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



# 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

## 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
neoVI Device Command Complete	Occurs when specific device commands are completed
SCP Transmit Complete Report	Occurs when a Ford SCP is properly transmitted by the LBCC
SCP Error Exception Report	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 2000-2004 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 2000-2005 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 2000-2005 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 2000-2005 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 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 maybe found in a programmable instrument consisting of commands and responses.

You can easily write a 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 properties have the same as there 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 two of possible responses. ok and er. ok means that the command completed successfully. "er" means there was a problem with 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	<i>methodname</i> {arguments}	ok <i>methodname</i>
Queries	<i>methodname</i> ?	ok <i>methodname</i> {propertyvalue}

### 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 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	
Vehicle Spy 3	Text API Terminal	Allows you to manual type in Text API commands and see thier responses
Vehicle Spy 3	Function Blocks	Allows you to send and recevied 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 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 Object

Command Name	Description	Example
--------------	-------------	---------

AutoDetectHardware	Sets whether Vehicle Spy will detect hardware or not.	AutoDetectHardware 1 ;sets hardware to autodetect AutoDetectHardware? ;asks for the autodetect setting
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
tx	Accesses the collection of Transmit Messages. See the topic on <a href="#">collection objects</a> .	tx.add ;adds a message at the end of the collection and returns new key
tx(index or key)	Accesses a specific transmit message by an index or a key. See the topic on transmit message objects.	tx(1).Arbid 234 ;sets the arb id in hex
gm	Accesses the gm subobject . This is discussed in a <a href="#">seperate topic</a> .	
Start	Starts Vehicle Spy	
Stop	Stops Vehicle Spy	
tx(index or key)	Transmit messages	
gp(index or key)	Graphical Panel objects	
as(index or key)	Accesses a specific application signal by an index or a key. Set the topic on <a href="#">Signal</a> objects.	
as	Accesses the collection of Application Signals. See the topic on <a href="#">collection objects</a> .	
JoystickEnabled	Indicates whether the joystick is enabled or not	
JoystickSelected	Indicates which joystick is used	
mg(index or key)	Accesses a specific <a href="#">message object</a> by an index or a key. See the topic on message objects.	mg(0).clearstats ;clears the stats of first msg
mg	Accesses the collection of Messages. See the topic on <a href="#">collection objects</a> .	
SimulationPath	Sets/Returns the path of the file used for simulation mode	
SimulationEnabled	Sets/Returns whether simulation is used or you are connecting to hardware	
ui	Controls the user interface of neoVI PRO. Described in a <a href="#">separate topic</a> .	
ao	Accesses a specific <a href="#">analog outputs</a> by an index or a key. See the topic on transmit message objects.	ao(0).value 3.42

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 disk space both available and total in kilobytes.	diskspace?
dg	Accesses a specific diagnostic jobs by an index or a key.	dg(0).start ;// starts a diag job
filedetails	Returns information size, time/date on a file in the data directory/neoVI PRO compact flash card	filedetails? test.vs3
io0value	Returns/Sets the value of the MISC 1 pin on neoVI PRO on the DB15 connector	io0value?
io1value	Returns/Sets the value of the MISC 2 pin on neoVI PRO on the DB15 connector	io1value 1
io0isoutput or io1isoutput	Returns/Sets whether MISC1 pin is an output or input.	io0isoutput 1 ;// make MISC1 and output
id	Returns the current neoVI PRO firmware ID	id? ;// get the neoVI PRO ID
ixcbusenabled	Returns/Sets whether the ixcbus is enabled	
ixcbusnetwork	Returns/Sets the network where the IXCBus protocol is used.	
loadfile	Loads a setup file from the data directory	loadfile test.vs3
status	Returns the loaded file and whether the file is running	status?
timedate	Returns/Sets the time date of the clock.	timedate?
isrunning	Returns whether if Vehicle Spy is running or not	isrunning?
copyfile	Copies a file in the data directory to another in the data directory	
deletefile	Removes a file in the data directory	
renamefile	Renames a file in the data directory	
appsave	Saves applications 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	
apprestore	Restores application signals from disk.	
gps	Accesses the GPS object	gps.latitude? gps.longitude? gps.altitude? gps.speed? gps.isvalid?

all?	returns all application signals that are not remote signals in a key=value comma seperated string. This is used to efficiently read all app signals over a slower network.	all?
allsetup?	returns all application signals that are not remote signals in a key=description comma seperated string. This is used to efficiently read all app signals descriptions over a slower network.	
gpallsetup?	returns all graphical panels in a key=description comma seperated string. This is used to efficiently read all panel descriptions over a slower network.	
gp(Key or Index).allsetup?	returns the graphical panel as an XML string.	
gp(Key or Index).all?	returns all graphical panel control values in a key=value comma seperated string. This is used to efficiently read all graphical panel data over a slower network.	gp(dial).all?

## Collection Object

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
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 exists 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 Object

Command Name	Description	Example
start	Starts the function block	fb(0).start ; starts the first function block
stop	Stops the function block	
trigger	Triggers the function block	
save	Saves the function block data	

## Signal Object

Command Name	Description	Example
value	Sets/Gets the value	

Description {Text Description}	Sets the description for a message	Set the description for number msg 4 as throttle position  mg(0).sig(0).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	

## Message Object

Command Name	Description	Example
ByteStringX	Sets the filter bytes for a network in hex	mg(0).ByteString0 110
Description {Text Description}	Sets the description for a message.	Set the description for number msg 4 as throttle position  mg(0).description Engine Data
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
<del>EnRxEvent {Return Msg, Return Value, Return Stats}</del>		<del>sg(in3433).EnRxEvent</del>
DisplayColor	Sets/Returns the color the message is displayed.	mg(0).DisplayColor 0 ;// display black
<del>EnValEvent {Return Msg, Return Value, Return Stats}</del>		
Compile	Compiles filter bytes and equations and then makes changes activate.	
ExpectedLength	Sets the expected length of a specific message	
EnableISO15765	This enables long messaging on CAN based off of ISO15765	
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	
sg(index or key)	Accesses a specific <a href="#">Signal object</a> by an index or a key. See the topic on message objects.	mg(0).clearstats ;clears the stats of first msg

sg	Accesses the collection of Signals. See the topic on <a href="#">collection objects</a> .	

## Analog Output Object

Command Name	Description	Example
value	Returns/Sets the value of the output. This is only valid if calculatefromsignal is false.	ao(0).value 3.21 ; sets output to 3.21V
calculatefromsignal	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).calculatefromsignal 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
signalkey	Returns/Sets the key of the signal attached to this analog output. This is used in conjunction with signal key.	
messagekey	Returns/Sets the key of the message attached to this analog output	

## 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

## 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	<code>ui.ledpwr 1 ;// sets the led on</code> <code>ui.ledpwr 0 ;// turns led off</code>
clear	Clears the LCD screen	<code>ui.clear ;// clears the LCD screen</code>
line	Draws a line on the LCD screen in a specified color Arguments: <b>x1,y1,x2,y2,color</b>	<code>ui.line 0,32,127,32,1 ;// draw a line in the center of the screen</code>
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.	<code>ui.print 0,28,0,2,1,Hello neoVI World</code>  <code>;// displays hello world on the screen</code>
rect	Draws a rectangle on the LCD screen with optional fill  Arguments: <b>x1,y1,x2,y2,color,fill</b>	<code>ui.rect 10,10,30,30,1,0 ;// draw a square on the screen</code>
ledex	Sets/Clears the neoVI PRO exclam (!) LED	<code>ui.ledex 1 ;// sets the led on</code> <code>ui.ledex 0 ;// turns led off</code>
leddb	Sets/Clears the neoVI PRO database LED	<code>ui.leddb 1 ;// sets the led on</code> <code>ui.leddb 0 ;// turns led off</code>
buzz	Sets/Clears the neoVI PRO buzzer	<code>ui.buzz 1 ;// turns on buzzer</code> <code>ui.buzz 0 ;// turns off buzzer</code>

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.	ui.invert 1   ; // invert on ui.invert 0   ; // invert off
dbitmap	Draws a bitmap <b>x1,y1,widthinpixels,heightinpixels,</b> <b>{csv hex bitmap}</b>	ui.dbitmap 0,0,4,8,FF,FF,FF,FF   ; // draws a block of pixels
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>	ui.circle 30,30,5,1
Pixel	Draws a pixel on the display Arguments: <b>x,y,color</b>	ui.pixel 30,30,1 ;// set pixel to blue
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	ui.operatingmode 3 ;// switch to custom mode
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 intensity 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?

## General Motors Object - Vehicle Spy 3 Text API

### Objective

General Motors specific functionality is accessed via the gm object. The syntax is as follows:

gm.sim.simulatetransmitmessages 0 ; simulate the receive messages for those in the ECU collection

### gm Object

Command Name	Description	Example
sim	Accesses the <a href="#">GMLAN simulator</a> .	gm.sim.compile ; compile the simulator

### gm.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 a 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.	gm.sim.isrunning?
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	
simulatenormal	Returns/Sets if the ECUs should simulate normal messaging	
simulatediagnostics	Returns/Sets if the ECUs should simulate diagnostics messaging	
simulatetransmitmessages	if one the tx message list is generate from the tx list of each ecu. If zero the tx message list is generated from the rx list of each ecu	
sg	a <a href="#">collection of signals</a> which will be copied to the transmit message upon compilation.	
tx	This is the <a href="#">collection of transmit messages</a> sent by the simulator. It is dynamically created by the compile method.	

replaystart	This starts the replay data file at the first value if the simulator is running.	
replaystop	This stops file replay if it is running.	
generatehvwakeup	If 1 the simulator will generate a high voltage wakeup when started	
mg	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.	
GenerateVNMF	If 1 the simulator will generate a VNMF for the single wire can ECUs involved in the simulation.	
VNMFID	This indicates the CAN ID of the VNMF used by the simulator.	

### gm.sim.ec Object

Command Name	Description	Example
ecuname	the ECU short name	
enablenetworkmanagement	if 1 the simulator uses VN states to determine which messages should be sent. Otherwise they are always sent	
key		
networkname	the network name according the GMLAN uef file "hsCAN" or "swCAN"	

### gm.sim.mg Object

Command Name	Description	Example
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.	
databytes{X}	This allows default databytes to be entered for databytes0 through databytes7.	
overrideperiod	This will override the period of the message.	
period	The overridden period. Only works if "overrideperiod" is 1	

---

Last Update 13-May-2005

Copyright 2004-2005 - Intrepid Control Systems, Inc.

## Vehicle Spy Binary File Buffer File Spec - Vehicle Spy 3

### Overview

This document describes the binary file format for Vehicle Spy 3 datafiles. This binary file format was designed to optimize file save time as opposed to file size.

### Basics

Description	Length	Notes
text identifier	6	single byte char array with text "icsbin" used to identify file type.
file version	4	unsigned int values indicate version.
length of vs3 file	4	unsigned int value indicating the length of the next section (the vs3 file). A zero indicates not vs3 file is present.
vs3 file	variable (see above)	The vs3 file used to save this binary file. This vs3 file is later used to decode the binary data into useable information.
length of text comment	4	unsigned int indicating the length of the text header comment of the saved file. SINCE THE COMMENT IS UNICODE THEREFORE THIS IS THE LENGTH IN CHARACTERS - NOT BYTES.
text comment	2 bytes per character (see above for number of characters)	Unicode text comment.
Number of bytes for stored buffer messages	4	sizeof(icsSpyMessage) * Number of messages saved to the file
Current Buffer Pointer	4	unsigned 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	unsigned 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	Struct icsSpyMessage	icsSpyMessage structures
Start Time of Collection	Struct icsspyMsgTime	This is a comparison value between the system time stamp and the neoVI timestamp.

### Structs from icsSpyData.h

```
typedef struct // matching C structure
{
    unsigned long StatusBitField; // 4
```

```

unsigned long StatusBitField2; // 4
unsigned long TimeHardware; // 4
unsigned long TimeHardware2; // 4
unsigned long TimeSystem; // 4
unsigned long TimeSystem2; // 4
unsigned char TimeStampHardwareID; // 1
unsigned char TimeStampSystemID;
unsigned char NetworkID; // 1
unsigned char NodeID;
unsigned char Protocol;
unsigned char MessagePieceID; // 1
unsigned char ExtraDataPtrEnabled; // 1
unsigned char NumberBytesHeader; // 1
unsigned char NumberBytesData; // 1
short DescriptionID; // 2
long ArbIDOrHeader; // 4
unsigned char Data[8];
unsigned char AckBytes[8];
unsigned int ExtraDataPtr;
unsigned char MiscData;
} icsSpyMessage;

```

```

typedef struct // matching C structure
{
unsigned long SystemTimeStampID; // 4
unsigned long SystemTime1; // 4
unsigned long SystemTime2; // 4
unsigned long HardwareTimeStampID; // 4
unsigned long HardwareTime1; // 4
unsigned long HardwareTime2; // 4

} icsSpyMsgTime;

```

## Example

```

// binary file load
bool cicsSpyBuffer::LoadBufferBinary(ICSCCHAR * szFilePathName, PVOID pArg, cicsString & sComment,
cicsDynamicArrayTemplate<unsigned char> & obSetupArray, icsspyMsgTime & stMsgStartTime, cicsString &
sReadErrorString)
{

HANDLE hFile;
unsigned char bTemp;
int iVersion;
unsigned long iNumBytesRead;
long lResult;
bool bInvalidFormat =true;
int iFileSize;
short iShortTemp;
int iCount;
int iBufferSizeInBytes;
int iBufferSizeInMsgs;
int iTempBuffSizeInMsgsSaved;
int iTempBuffCurrentItem;

```

```

// open the file for reading
hFile =
ICSCreateFile(szFilePathName,GENERIC_READ,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
if(hFile == INVALID_HANDLE_VALUE)
return false;

// return ReadFile(hFile, obArray, iByteCount, &iBytesRead, NULL);
// step 1: read a file header with version //////////////////////////////////////

lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 'i')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 'c')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 's')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 'b')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 'i')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&bTemp,1,&iNumBytesRead,NULL);
if(bTemp != 'n')
goto HasbInvalidFormat;
lResult = ReadFile(hFile,&iVersion,sizeof(iVersion),&iNumBytesRead,NULL);
if(iVersion != 0x101)
goto HasbInvalidFormat;
////////////////////////////////////

// if we made it this far we should be safe
bInvalidFormat = false;

HasbInvalidFormat:
if(bInvalidFormat)
{
sReadErrorString = ICSUNI("This binary file is not a valid Vehicle Spy Buffer file.");
return false;

}
////////////////////////////////////
// Start the conversion...
////////////////////////////////////

// step 2 : read the size of the setup (vs3) file //////////////////////////////////////
lResult = ReadFile(hFile,&iFileSize,sizeof(int),&iNumBytesRead,NULL);

// resize the setup array to fit the setup file
obSetupArray.Redim(iFileSize,false);

// step 3: read the setup file //////////////////////////////////////
lResult = ReadFile(hFile,obSetupArray.obItems,obSetupArray.iNumItems,&iNumBytesRead,NULL);

```

```

// step 4: read the comment as unicode //////////////////////////////////////
// first read the length in characters
lResult = ReadFile(hFile,&iFileSize,sizeof(iFileSize),&iNumBytesRead,NULL);

if (iFileSize==0)
sComment = ICSUNI("");
else
{
sComment.Redim(iFileSize+1,false);

// write the characters
for (iCount=0;iCount<iFileSize;iCount++)
{
lResult = ReadFile(hFile,&iShortTemp,sizeof(iShortTemp),&iNumBytesRead,NULL);
sComment.szOut[iCount] = (ICSCHAR) iShortTemp;

}
// null terminator
sComment.szOut[iFileSize] = 0;
}

// step 5: read the size of the binary file //////////////////////////////////
// number of messages
lResult = ReadFile(hFile,&iBufferSizeInBytes,sizeof(iBufferSizeInBytes),&iNumBytesRead,NULL);
iBufferSizeInMsgs = iBufferSizeInBytes / sizeof(icsSpyMessage);

// size of original buffer
lResult =
ReadFile(hFile,&iTempBuffSizeInMsgsSaved,sizeof(iTempBuffSizeInMsgsSaved),&iNumBytesRead,NULL);
// current buffer position
lResult = ReadFile(hFile,&iTempBuffCurrentItem,sizeof(iTempBuffCurrentItem),&iNumBytesRead,NULL);

SetBufferSize(iTempBuffSizeInMsgsSaved);
lCurrentItem = iTempBuffCurrentItem;
// lNumberOfMessagesAllTime
lResult =
ReadFile(hFile,&lNumberOfMessagesAllTime,sizeof(lNumberOfMessagesAllTime),&iNumBytesRead,NULL);

// step 6: write the array structure of messages //////////////////////////////////
lResult = ReadFile(hFile,mMessages,iBufferSizeInBytes,&iNumBytesRead,NULL);

// step 7: read the monitor start time
lResult = ReadFile(hFile,&stMsgStartTime,sizeof(stMsgStartTime),&iNumBytesRead,NULL);

// close the file handle
CloseHandle(hFile);

return true;

}

```

## Contact Information - Intrepid Control Systems, Inc.

[Main](#)



### **Intrepid Control Systems, Inc.**

45138 Cass Avenue

Utica, Michigan 48317 USA

(ph) 586.731.7950

(fax) 586.731.2274

(email) [info@intrepidcs.com](mailto:info@intrepidcs.com)

(website) [www.intrepidcs.com](http://www.intrepidcs.com)

**intrepidcs API Documentation - (C) Copyright 2000-2004 Intrepid Control Systems, Inc. [www.intrepidcs.com](http://www.intrepidcs.com)**

*Last Updated : Saturday, August 17, 2002*