icsneo40 DLL API Example in LabVIEW

LabVIEW with IntrepidCS ValueCAN and neoVI Hardware series

User Guide

Version 1.0 - March 11, 2019

Intrepid Control Systems, Inc.
31601 Research Park Drive Madison Heights, MI 48071 USA
(ph) +1-586-731-7950 (fax) +1-586-731-2274
www.intrepidcs.com

automotive engineering tool alliance
www.aeta-rice.com

Version History

Version Number	Date	Description / Major Changes
1.0	March 11, 2019	Initial Release

Table of Contents

1. Introduction and Overview

1.1 Introduction:

Intrepid APIs provides a simple way to access the ValueCAN and neoVI hardware series with WIN32 development tools. This documentation describes how to use the API for custom applications in LabVIEW. User can connect to ValueCAN or neoVI Hardware series device, Transmit/Receive CAN, CAN FD and Ethernet messages and set CAN Baud rate using this LabVIEW VI Example.

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

1.2 Hardware and software requirements:

- **Software:**
  - **LabVIEW 2017 version:** This Example is compatible with LabVIEW version 2017 onwards.

  - **IntrepidCS API Installer Kit:** Intrepid API installer kit installs neoVI DLL in the respective location in the PC based on system configuration.
    Below is link for the API installer kit:
    https://cdn.intrepidcs.net/updates/files/RP1210KitInstall.zip

- **Hardware:**
  - **neoVI device:** LabVIEW Example supports below Intrepid Devices

    CAN & CAN-FD Interface:
    - ValueCAN 3 (CAN only)
    - ValueCAN 4-1(CAN only)
    - ValueCAN 4-2 (CAN & CAN FD)
    - neoVI RED (CAN only)
    - neoVI FIRE (CAN only)
    - neoVI FIRE 2 (CAN & CANFD)
    - neoVI ION (CAN & CANFD)

    Ethernet Interface:
    - RAD-Star 2 (CAN, CANFD & Ethernet)
    - RAD-Galaxy (CAN, CANFD & Ethernet)

1.3 Driver and API Support File Installation and Setup:

User needs to install drivers and support files to allow the hardware to be accessed via its API. Unzip RP1210KitInstall.zip folder downloaded from link in section 1.2. and run 'APIInstallKit.exe' with admin rights.

1.4 Hardware Hookup Diagrams:

ValueCAN series are USB powered devices. DB-9 connector on ValueCAN can be connected to CAN networks on ECU. Refer Figure-13 for connection details.

NeoVI device e.g. neoVI Hardware series needs a 12-30 V DC power supply. Normally this power is taken from vehicle Battery using OBD-II cable. You can use a custom made cable as well. Connect neoVI device to ECU or Vehicle OBD-II port. Connect USB to PC running LabVIEW. Refer Figure-14

RAD-Star 2 has a power jack to connect 12 V adapter. Custom cable can be made to connect with Broad-R network using NanoMQS connector received with RAD-Star 2. Refer Figure-15 for details.

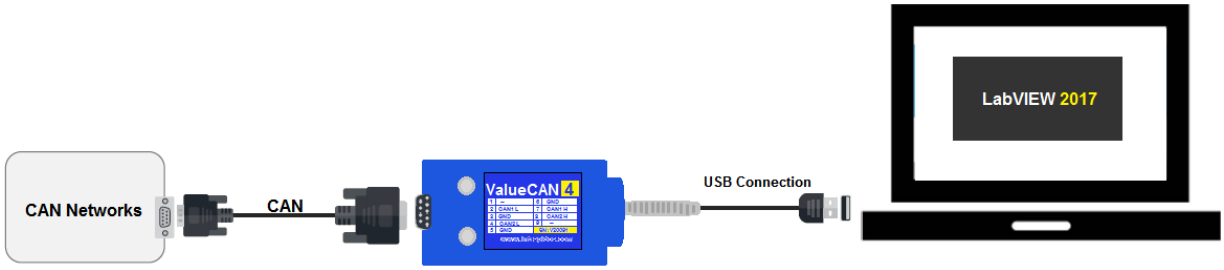

Figure-13: ValueCAN4 connection diagram

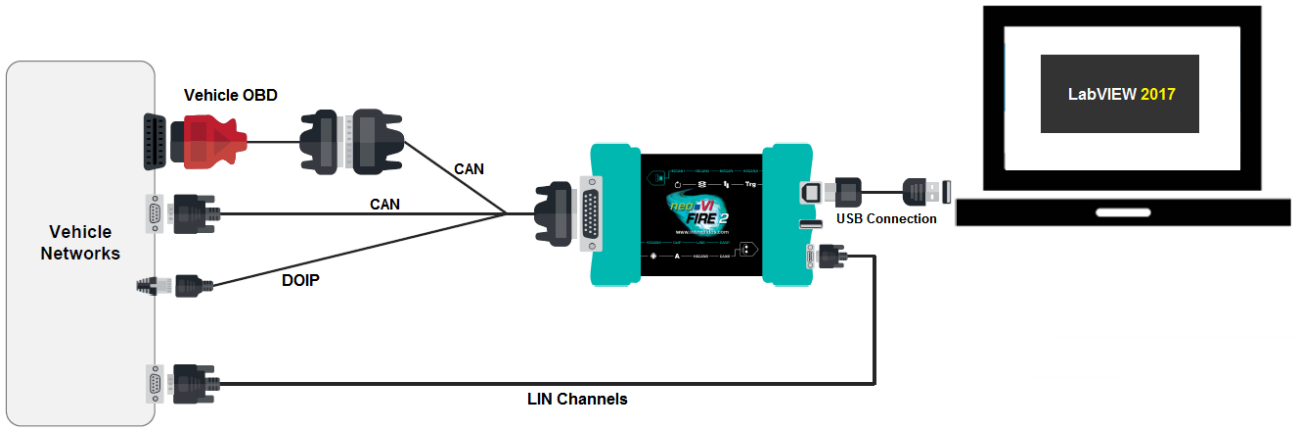

Figure-14: neoVI FIRE-2 connection diagram

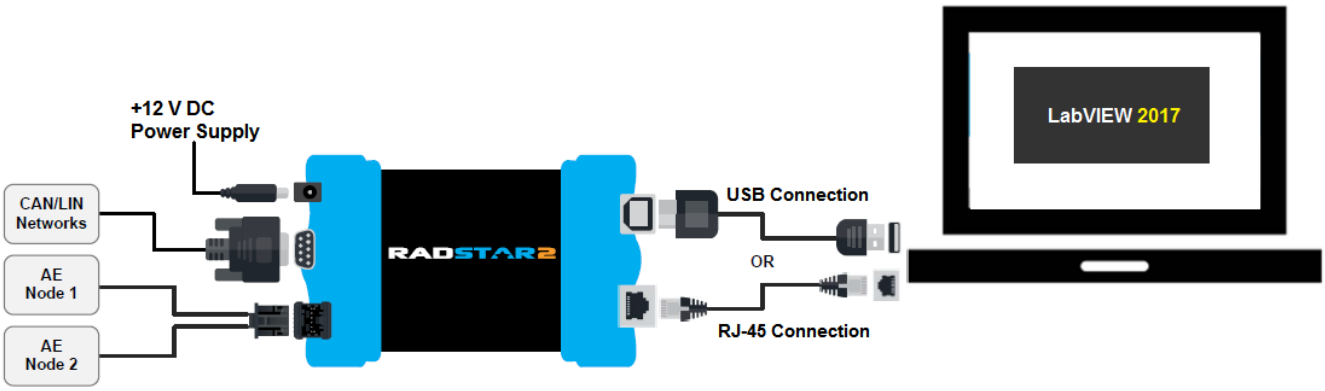

Figure-15: RAD-Star2 connection diagram

2. Overview of Main LabVIEW GUI

Main VI of LabVIEW example has four different controls designed (Figure-16). They are as below:
Connection tab (Figure-16 Section 1) has all controls to connect LabVIEW with ValueCAN/neoVI devices. User can connect or disconnect device, view device type, serial number and check icsneo40 DLL version.
Transmit Message tab (Figure-16 Section 2) shows all settings for transmitting message on CAN, CAN FD and Ethernet networks.
Receive Message tab (Figure-16 Section 3) is for receiving messages on CAN, CAN FD, Ethernet bus and view them in LabVIEW VI.
Baud Rate tab (Figure-16 Section 4) has option to select CAN network and set its baud rate.

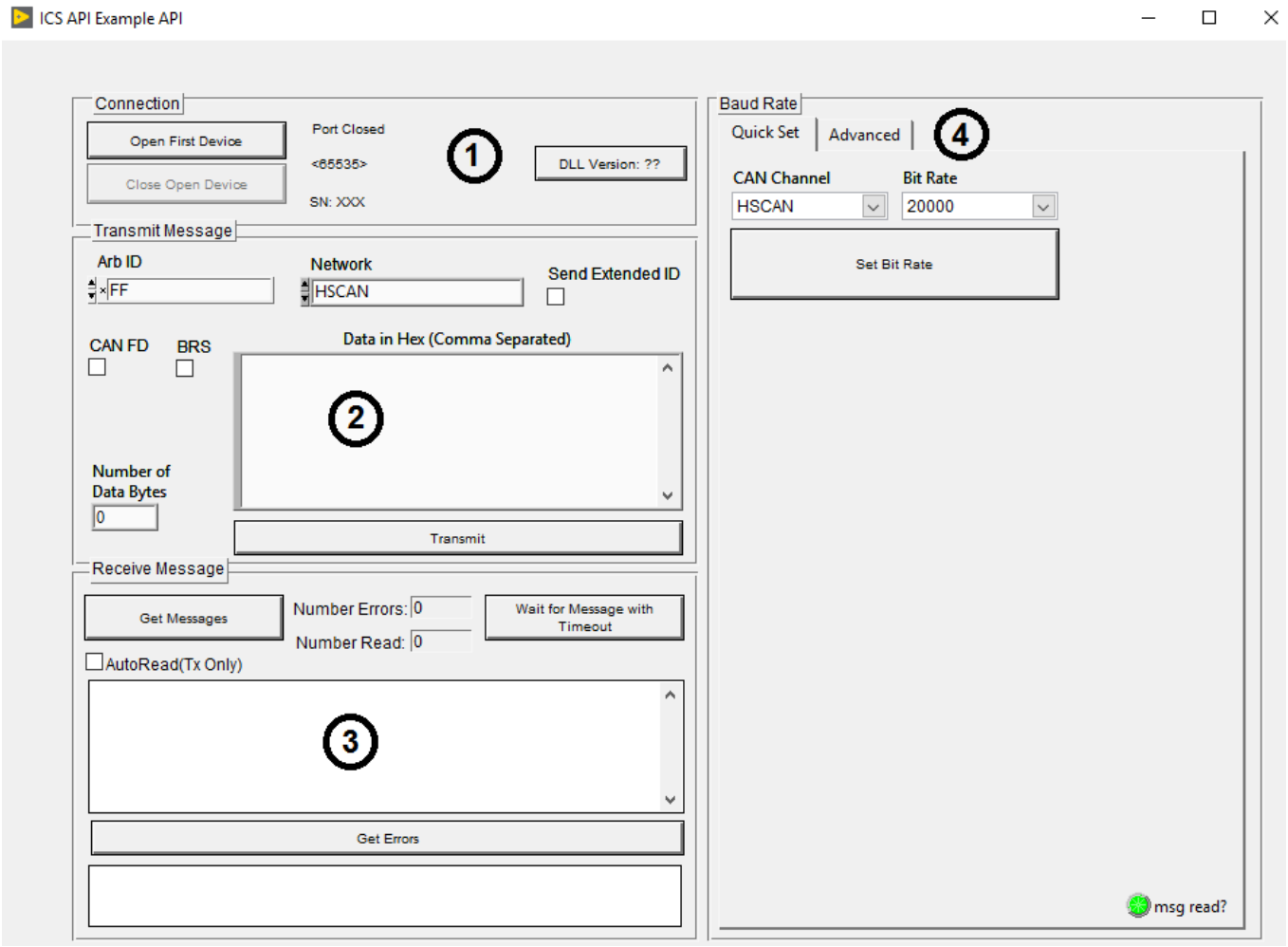

Figure-16

3. Intrepid APIs in LabVIEW

3.1 Connect device:

Main VI in LabVIEW example has a tab for connecting ValueCAN & neoVI device in LabVIEW. After clicking on 'Open First Device' button LabVIEW executes FindNeoDevices and OpenNeoDevice API from icsneo40.dll.
Connection Tab shows 'Port Open' message, device Name and device serial number after successful connection. Refer Figure-17

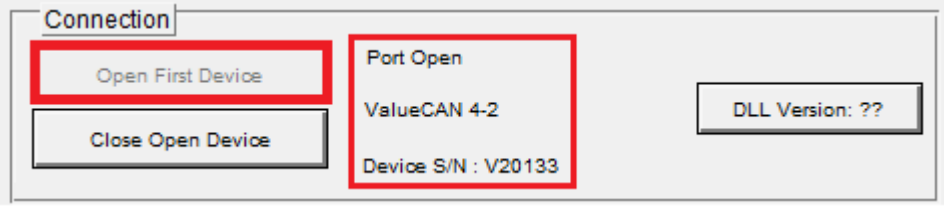

Figure-17

API details:

- **FindDevices:**

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

int _stdcall icsneoFindDevices(NeoDeviceEx* pNeoDeviceEx, int* pNumDevices , unsigned int * DeviceTypes,unsigned int numDeviceTypes, POptionsFindNeoEx* pOptionsNeoEx, unsigned int* reserved);

- **OpenNeoDevice:**

This method opens a communication link to the neoVI device.

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

3.2 Disconnect device:

'Close Open device' button disconnects neoVI device from LabVIEW. After clicking on 'Close Open Device' button LabVIEW executes 'ClosePort' API from icsneo40.dll.
Connection Tab shows 'Port Closed' message. Refer Figure-18

Figure-18

API details:

- **ClosePort:**

This method closes the communication link with the neoVI hardware.

int_stdcall icsneoClosePort(void * hObject, int * pNumberOfErrors);

3.3 Get IntrepidCS API version:

'DLL version' button gets icsneo40.dll version and shows it on main VI. After clicking the button LabVIEW executes 'GetDLLVersion' API from icsneo40.dll.
Refer Figure-19

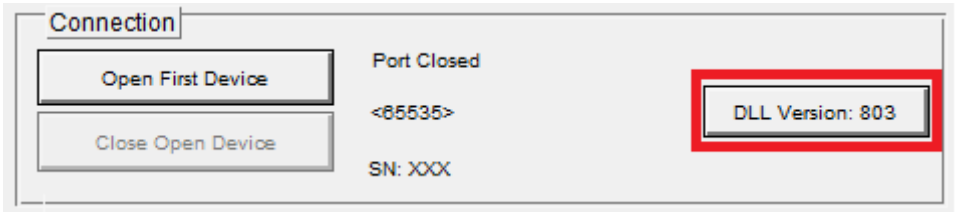

Figure-19

API details:

- **GetDLLVersion:**

This method returns the software version of DLL.

int_stdcall icsneoGetDLLVersion();

3.4 Transmit Message:

Transmit message section in main VI has multiple settings to send message on CAN, CAN FD or Ethernet network.

- Procedure to send CAN, CAN FD message:
  1. Enter arbitration ID in 'Arb ID' field. Select check box 'Send Extended ID' if Arb ID is 29 bit.
  2. Select CAN network from the 'Network' drop down list.
  3. Enter data bytes (each byte should be comma separated).
  4. Mention DLC in 'Number of Data Bytes' field.
  5. Click on 'Transmit' button. Refer Figure-20
  6. For transmitting CAN FD message enable 'CAN FD' check box and follow steps 1 to 5. Bit Rate switch setting can be enabled by enabling 'BRS' check box. Refer Figure-21

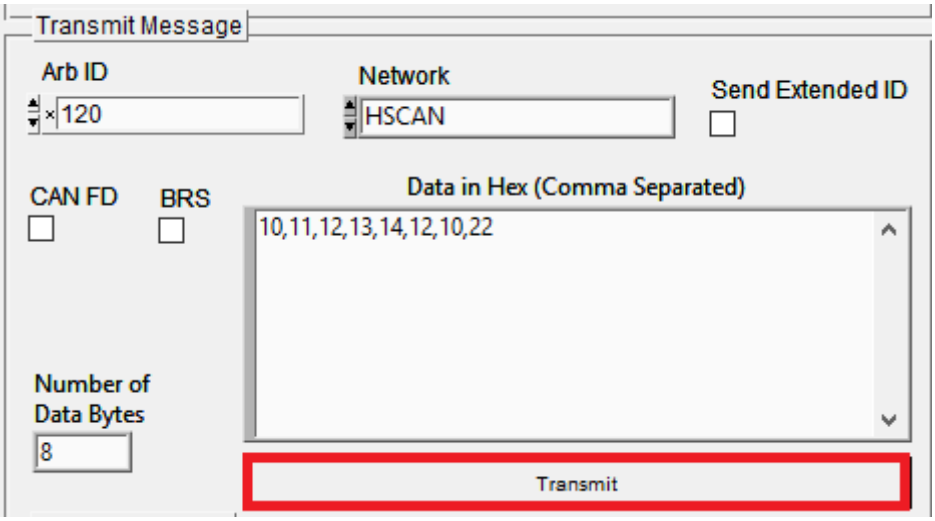

Figure-20

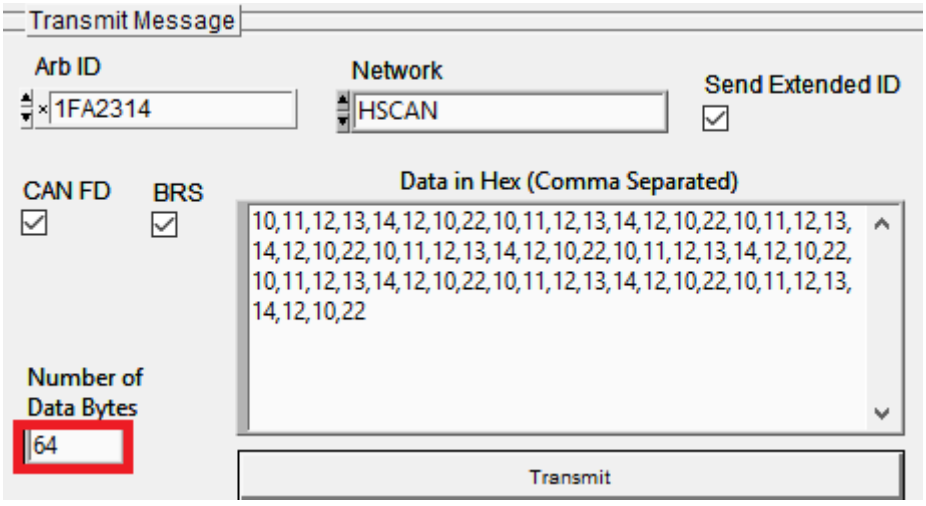

Figure-21

- Procedure to Send Ethernet message:

1. Select Broad-R network from the network drop down list.
2. Enter data bytes (each byte should be comma separated).
3. Mention DLC in 'Number of Data Bytes'.
4. Click on Transmit button. Refer Figure-22

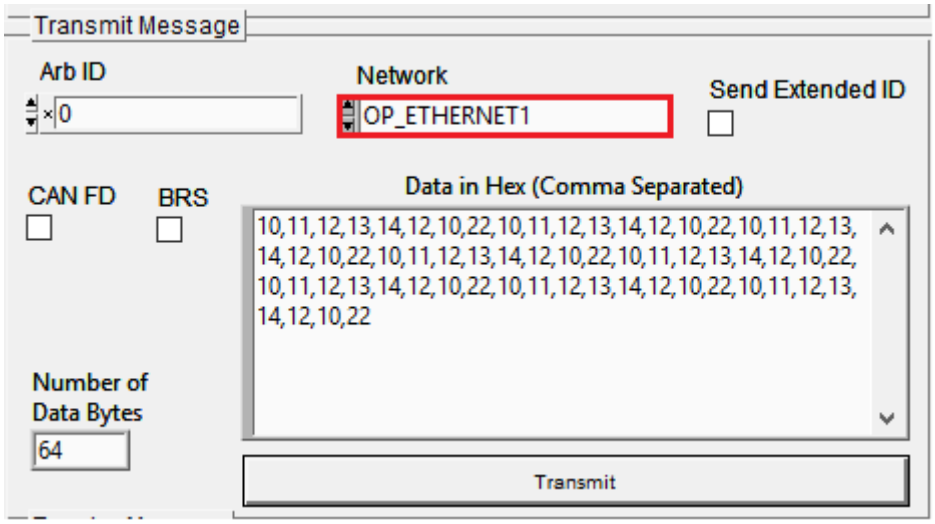

Figure-22

API details:

- **TxMessages:**

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

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

- **TxMessagesEx**:

This method transmits longer messages asynchronously to vehicle networks using the neoVI hardware. Method supports CAN FD and Ethernet networks.

int_stdcall icsneoTxMessagesEx(void
* hObject, icsSpyMessage *pMsg,int lNetworkID, int lNumMessages, int *NumTxe
d, int reserved);

3.5 Receive message:

Receive message section in Main VI reads and shows message received on CAN, CAN FD or Ethernet network. Click 'Get Messages' button to see messages in log window.
Message transmitted from LabVIEW on CAN, CAN FD or Ethernet can be viewed in Receive message window after enabling 'AutoRead' Check box (figure-23).

API details:

- **GetMessages:**

This method reads messages from the neoVI device.

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

- **GetTimeStampForMsg:**

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

int_stdcall icsneoGetTimeStampForMsg(int hObject, icsSpyMessage*pMsg, icsSpy Message *pMsg, double *pTimeStamp);

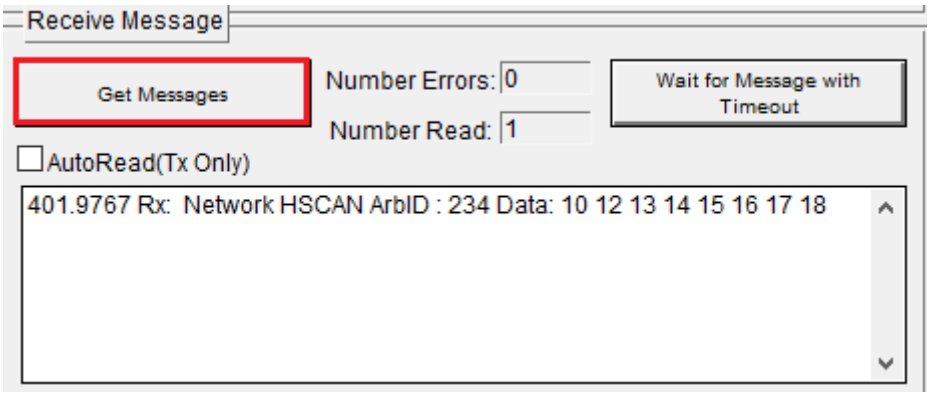

Figure-23

3.6 Set Bit Rate of CAN channels:

Baud Rate section in Main VI has a tab to set Baud Rate of multiple CAN channels supported by IntrepidCS devices.
You can set desired baud rate to specific CAN channel using CAN channel drop down list and Bit Rate drop drown list. Click button 'Set Bit Rate' to apply changes to baud rate setting. Refer Figure-24

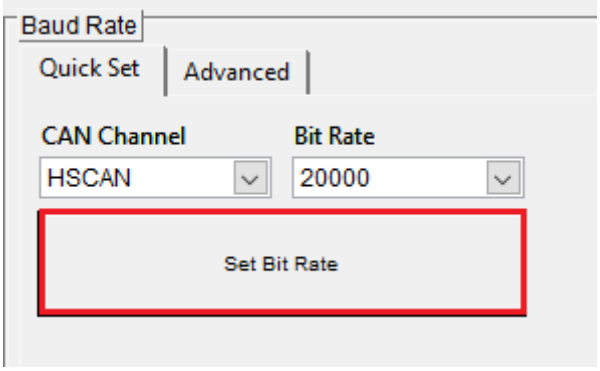

Figure-24

API details:

- **SetBitRate:**

This method sets bit rates for networks on neoVI device.

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

4. Helper DLL

LabVIEW example uses a Helper.dll from transmitting and receiving long message. This DLL is having two APIs defined in it.
'ConvertArrayToVoidPointer' API is used for transmitting CAN FD and Ethernet frames of data more than 8 bytes.
'GetLongData API' returns an array having data bytes of received message.

icsSpyMessage structure:
This structure is used to present messages both received and transmitted by ValueCAN and neoVI devices. This structure can also be represented as an array of bytes.

icsSpyMessage Strucure

```
Public Structure icsSpyMessage
    Dim StatusBitField As Int32    '4
    Dim StatusBitField2 As Int32    'new '4
    Dim TimeHardware As UInt32    ' 4
    Dim TimeHardware2 As UInt32    'new ' 4
    Dim TimeSystem As UInt32    ' 4
    Dim TimeSystem2 As UInt32
    Dim TimeStampHardwareID As Byte    'new ' 1
    Dim TimeStampSystemID As Byte
    Dim NetworkID As Byte    'new ' 1
    Dim NodeID As Byte
    Dim Protocol As Byte
    Dim MessagePieceID As Byte    ' 1
    Dim ExtraDataPtrEnabled As Byte    '1
    Dim NumberBytesHeader As Byte    ' 1
    Dim NumberBytesData As Byte    ' 1
    Dim DescriptionID As Int16    ' 2
    Dim ArbIDOrHeader As Int32    '// Holds (up to 3 byte 1850 header or 29 bit CAN header) '4
    Dim Data1 As Byte '1
    Dim Data2 As Byte '1
    Dim Data3 As Byte '1
    Dim Data4 As Byte '1
    Dim Data5 As Byte '1
    Dim Data6 As Byte '1
    Dim Data7 As Byte '1
    Dim Data8 As Byte '1
    Dim StatusBitField3 As Int32
    Dim StatusBitField4 As Int32
    Dim iExtraDataPtr As IntPtr    '4 or 8 depending on system
    Dim MiscData As Byte '1
End Structure
```

- **Transmit Log message from LabVIEW:**

For transmitting CAN FD or Ethernet frame containing over 8 bytes we need to set below values to structure elements:

ExtraDataPtrEnabled =1

iExtraDataPtr = Pointer to data bytes for CAN FD and Ethernet messages. This field contains address of Data byte array.

Helper.dll is having API 'ConvertArrayToVoidPointer' which takes icsSpyMessage structure, network ID, and array containing data bytes and transmits CAN FD or Ethernet Frame.

CovertArrayToVoidPointer API information:

long CovertArrayToVoidPointer(void * hObject, icsSpyMessage *pMsg, unsigned int lNetworkID, unsigned int lNumMessages, unsigned int *NumTxed, char p_array[1000],int size)

Parameters:

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

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

lNetworkID
[in] Specifies the network to transmit the message on. See Network ID List for a list of valid Network ID values. Network support varies by neoVI device.

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.

NumTxed
[out] Specifies the number of messages that have been transmitted.

p_Array
[in] Array containing data bytes over 8 byte size.

Size
[in] Size of p_Array. This is number of bytes transmitted in CAN FD or Ethernet frame.

- **Receive Long Message in LabVIEW:**

  GetMessage API returns pMsg structure having a pointer to data bytes for CAN FD or Ethernet message.
  GetLongData API from helper.dll takes this pointer and returns an array containing data bytes.

  GetLongData API information:

  void GetLongData(icsSpyMessage *pMsg, char * resPointer)

  Parameters:

  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.

  resPointer
  [out] Array of Data bytes

5. Annexure

5.1 Network ID list:

DEVICE	0
HSCAN	1
MSCAN	2
SW CAN	3
LSFT CAN	4
FORDSCP	5
J1708	6
JVPW	8
ISO	9
ISO2	14
ISO14230	15
LIN	16
OP_ETHERNET1	17
OP_ETHERNET2	18
OP_ETHERNET3	19
ISO3	41
HSCAN2	42
HSCAN3	44
OP_ETHERNET4	45
OP_ETHERNET5	46
ISO4	47
LIN2	48
LIN3	49
LIN4	50
MOST	51
CGI	53
HSCAN4	61
HSCAN5	62
UART	64
UART2	65
UART3	66
UART4	67
SWCAN2	68
ETHERNET_DAQ	69
TEXTAPI_TO_HOST	71
OP_ETHERNET6	73
OP_ETHERNET7	75
OP_ETHERNET8	76
OP_ETHERNET9	77
OP_ETHERNET10	78
OP_ETHERNET11	79
FLEXRAY1A	80
FLEXRAY1B	81

FLEXRAY2A	82
FLEXRAY2B	83
LIN5	84
FLEXRAY	85
OP_ETHERNET12	87
MOST25	90
MOST50	91
MOST150	92
ETHERNET	93
GMFSA	94
TCP	95
HSCAN6	96
HSCAN7	97
LIN6	98
LSFT CAN2	99

5.2 Protocol

CAN	1
GMLAN	2
ISO9141	5
KEYWORD2000	6
GM_ALDL_UART	7
CHRYSLER_CCD	8
CHRYSLER_SCI	9
FORD_UBP	10
LIN	12
J1708	13
J1939	15
FLEXRAY	16
MOST	17
GME_CIM_SCL_KLINE	19
SPI	20
I2C	21
GENERIC_UART	22
ETHERNET	29
CAN FD	30
TCP	32