USB2-F-7x01 Programming Guide

Document Reference No.: CP_000035

Version 1.1

Issue Date: 2019-03-21

The USB2-F-7x01 (USB-to CAN) is a replacement product for the Connective Peripherals CANUSB and it provides a simple method of adapting CANbus devices to USB.

The USB2-F-7001 (USB to CAN) CAN Plus adapter, adds additional features, new advanced commands and increase performance of the previous products. The USB2-F-7101 has all the same features of the USB2-F-7001 and includes optical isolation between the CAN transceiver and the CAN controller.

This programming guide describes the Windows-based API for use with high-level languages.

Connective Peripherals Pte Ltd
178 Paya Lebar Road, #07-03 Singapore 409030
Tel.: +65 67430980 Fax: +65 68416071

E-Mail (Support): support@connectiveperipherals.com Web: www.connectiveperipherals.com/products

Table of Contents

1 Introduction

1.1 Functional Description

The USB2-F-7001 is a USB to CANbus adapter which incorporates the FTDI FT245R USB to FIFO interface in conjunction with the PIC18F2680 to provide a fast, simple way to interface CANbus devices to a host PC with a USB port. The USB2-F-7101 provides the same functionality as the USB2-F-7001 while including optical isolation between the CANbus transceiver and the CANbus controller. Throughout this document references to USB2-F-7001, USB2-F-7101 and USB2-F-7x01 are equivalent.

This document describes the API function calls which work in conjunction with the FTDI D2XX device drivers. It is assumed that the USB2-F-7001 and related device drivers have been installed at this point. For instructions on installing the adapter and drivers, please see the USB2-F-7x01 datasheet.

2 Installation

2.1 DLL Installation

2.1.1 Windows Desktop and Server

There is no specific DLL installer. The DLL, LIB and Header files are typically copied to an application project directory. Different programming languages and compiler configurations require different locations. Consult your programming environment documentation for the correct locations to place the files contained within the distribution.

Currently programming with following high-level programming languages VB6, VB6 .NET, C#, VC++ is supported by the USB2-F-7x01 DLL.

2.1.2 Mac OS X, Linux, Windows CE

This API is not available under Mac OS X, Linux or Windows CE. Access to the USB2-F-7x01 is available through the Virtual COM Port ASCII commands as described in the USB2-F-7x01 Datasheet.

2.2 High-Level language API

2.2.1 canplus_getFirstAdapter

Summary

Search the system for available USB2-F-7x01 adapters.

Definition

int canplus_getFirstAdapter (char *szAdapter, int size);

Parameters

szAdapter	pointer to the serial number of the first USB2-F-7x01 found
size	size of the buffer to receive the serial number

Remarks

The serial number is a string consisting of 8-bytes plus null termination. Allocate a 9 byte buffer to obtain the serial number.

Adapter detection may need to be done on a separate thread or process if it is likely that the USB2-F-7x01 adapters are installed and/or removed after the device driver has been loaded.

Return Codes

>0	Number of USB2-F-7x01 adapters found
<= 0	Error as noted below:

ERROR_CANPLUS_MEMORY_ERROR

If szAdapter buffer size (size variable) is less than 9B

ERROR_CANPLUS_FAIL

Unable to get information about adapters

ERROR_CANPLUS_NO_DEVICE

No devices found

2.2.2 canplus_getNextAdapter

Summary

Search the system for available USB2-F-7x01 adapters. A call to "canplus_getFirstAdapter" is required prior to accessing this call. Repeat this call for each adapter.

Definition

int canplus_getNextAdapter (char *szAdapter, int size);

Parameters

szAdapter	pointer to the serial number of the next USB2-F-7x01 found
size	size of the buffer to receive the serial number

Remarks

The serial number is a string consisting of 8-bytes plus null termination. Allocate a 9 byte buffer to obtain the serial number.

Adapter detection may need to be done on a separate thread or process if it is likely that the USB2-F-7x01 adapters are installed and/or removed after the device driver has been loaded.

Return Codes

>0 ERROR_CANPLUS_OK

<= 0 Error as noted below:

ERROR_CANPLUS_MEMORY_ERROR

If szAdapter buffer size (size variable) is less than 9 Bytes

ERROR_CANPLUS_FAIL

canplus_getFirstAdapter was not called.

ERROR_CANPLUS_NO_DEVICE

No devices found

2.2.3 canplus_Open

Summary

Open a channel to the USB2-F-7x01.

Definition

CANHANDLE canplus_Open(LPCSTR szID, LPCSTR szBitrate, LPCSTR acceptance_code, LPCSTR acceptance_mask, unsigned long flags);

Parameters

szID	USB2-F-7x01 serial number
szBitrate	10 = 10Kbps 20 = 20Kbps 50 = 50Kbps 100 = 100Kbps 250 = 250Kbps 500 = 500Kbps 800 = 800Kbps 1000 = 1000Kbps aabbcc = custom bit rate aa = BRGCON1 register of Microchip PIC 18F2680 bb = BRGCON2 register of Microchip PIC 18F2680 cc = BRGCON3 register of Microchip PIC 18F2680 The USB2-F-7x01 utilizes a 24MHz clock for the PIC 18F2680. Use this value when referring to the PIC datasheet if a custom transmission rate different from the eight pre-defined rates is required.
acceptance_code	11-bit (0x000 to 0x7FF) or 29-bit (0x00000000 to 0x1FFFFFFF) code used in filtering specific or ranges of CAN messages. Used in conjunction with the acceptance mask. Note: When referring to the PIC datasheets, Acceptance Code is synonymous with Acceptance Filter.
acceptance_mask	11-bit (0x000 to 0x7FF) or 29-bit (0x00000000 to 0x1FFFFFFF) code used in filtering specific or ranges of CAN messages. Used in conjunction with the acceptance code.
flags	CANPLUS_FLAG_TIMESTAMP – True = The timestamp function will be enabled by the USB2-F-7x01. False = Timestamps will be determined by the device driver.

Remarks

If multiple USB2-F-7x01 adapters exist in a system, user must call canplus_getFirstAdapter and canplus_getNextAdapter to get serial numbers of the connected adapters. The serial number must be passed to canplus_Open in szID to open a specific device. If szID == NULL, the first USB2-F-7x01 device will be opened.

Only one open command will be successful for a single device. At this point multiple channels to a single physical device are not supported.

Any argument to this API may be set to NULL to use default value of the field. By default a CAN channel is opened to accept all standard and extended CAN frames at 1Mbps bit rate.

Default timeouts for the underlying FTDI D2XX FT_Read and FT_Write are set to 1sec.

Return Codes

handle	Handle to the device if successful
<= 0	Error as noted below: ERROR_CANPLUS_FAIL Unable to open communication to USB2-F-7x01 device ERROR_CANPLUS_OPEN SUBSYSTEM Unable to open CAN channel ERROR_CANPLUS_COMMAND_SUBSYSTEM Failed in setting other parameters (e.g. setting timestamp, bitrate, acceptance code or acceptance mask)

2.2.4 canplus_Close

Summary

Close the CAN channel with handle h.

Definition

int canplus_Close (CANHANDLE h);

Parameters

h	the handle of the CAN channel which will be closed

Remarks

Any data present in the data buffers will be lost, even if not fully transmitted on the CAN channel. Ensure there is enough time to transmit a full frame prior to closing a channel.

Return Codes

>0	ERROR_CANPLUS_OK	
		CAN channel is closed successfully
<=0	Error as noted below:	
	ERROR_CANPLUS_FAIL	
		CAN channel could not be closed

2.2.5 canplus_Read

Summary

Read a message from the open channel with handle h. Message is contained within the message structure.

Definition

int canplus_Read(CANHANDLE h, CANMsg *msg);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01
msg	received message frame structure

Remarks

```
// Message flags
        #define CANMSG_EXTENDED 0x80 // Extended CAN id
        #define CANMSG_RTR 0x40      // Remote frame
// CAN Frame
        typedef struct {
                _u32 id;                // Message id
                _u32 timestamp;         // timestamp in milliseconds
                _u8 flags;              // [extended_id|1][RTR:1][reserver:6]
                _u8 len;                // Frame size (0.8)
                _u8 data[ 8 ];          // Databytes 0..7
        } CANMsg;
```

Return Codes

>0 ERROR_CANPLUS_OK

CAN channel is closed successfully

<=0 Error as noted below:

ERROR_CANPLUS_NO_MESSAGE

The receive buffer is empty

2.2.6 canplus_Write

Summary

Write a message to the open channel with handle h. Message is contained within the message structure.

Definition

int canplus_Write(CANHANDLE h, CANMsg *msg);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01
msg	transmitted message frame structure

Remarks

```
// Message flags
    #define CANMSG_EXTENDED 0x80 // Extended CAN id
    #define CANMSG_RTR 0x40      // Remote frame
// CAN Frame
    typedef struct {
        _u32 id;                 // Message id
        _u32 timestamp;          // timestamp in milliseconds
        _u8 flags;               // [extended_id|1][RTR:1][reserver:6]
        _u8 len;                 // Frame size (0.8)
        _u8 data[ 8 ];           // Databytes 0..7
    } CANMsg;
```

Return Codes

>0	ERROR_CANPLUS_OK Standard/Extended Frame written successfully
<=0	Error as noted below: ERROR_CANPLUS_FAIL Standard/Extended Frame write Failure

2.2.7 canplus_Status

Summary

Retrieve adaptor status for channel with handle h.

Definition

int canplus_Status(CANHANDLE h);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01

Remarks

Returns < 0 error codes if failure. Limit use of "canplus_Status" as it may degrade transmission and reception of valid frames.

Return Codes

>0 Status value with bit values as below

```
// Status bits
        #define CANSTATUS_RECEIVE_FIFO_FULL 0x01
        #define CANSTATUS_TRANSMIT_FIFO_FULL 0x02
        #define CANSTATUS_ERROR_WARNING 0x04
        #define CANSTATUS_DATA_OVERRUN 0x08
        #define CANSTATUS_ERROR_PASSIVE 0x20
        #define CANSTATUS_ARBITRATION_LOST 0x40
        #define CANSTATUS_BUS_ERROR 0x80
```

<=0 Error as noted below:

ERROR_CANPLUS_FAIL

Unable to get the status

ERROR_CANPLUS_COMMAND_SUBSYSTEM

Status command returned a failure

2.2.8 canplus_VersionInfo

Summary

Retrieve adaptor hardware, firmware and driver versions as well as the adapter serial number for channel with handle h.

Definition

int canplus_VersionInfo(CANHANDLE h, LPSTR verinfo);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01

verinfo	Format: "VHhFf-Nxxxx-nnnnn-CCCCCCCCCC"
	V, N = Constant
	H = Hardware_Major
	h = Hardware_Minor
	F = Firmware_Major
	f = Firmware_Minor
	x = CANPLUS serial #
	n = Windows device driver version
	C = Custom String, Default "EasySync Ltd"

Remarks

Version information is used for informational purposes only.

Return Codes

>0	ERROR_CANPLUS_OK	
		CAN channel is closed successfully
<=0	Error as noted below:	
	ERROR_CANPLUS_FAIL	
		CAN channel version information could not be retrieved

2.2.9 canplus_Flush

Summary

Clears the transmit buffers for channel with handle h.

Definition

int canplus_Flush(CANHANDLE h);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01

Remarks

Return Codes

>0 ERROR_CANPLUS_OK

CAN channel is closed successfully

<=0 Error as noted below:

ERROR_CANPLUS_FAIL

CAN channel could not be flushed

2.2.10 canplus_Reset

Summary

Resets the device with handle h. The canplus_Open must be used to open a new handle.

Definition

int canplus_Reset(CANHANDLE h);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01

Remarks

Return Codes

>0 ERROR_CANPLUS_OK

CAN channel is closed successfully

<=0 Error as noted below:

ERROR_CANPLUS_FAIL

CAN channel could not be reset

2.2.11 canplus_Listen

Summary

Set the USB2-F-7x01 CANPLUS device in the CAN Listen mode. Note that at least three CAN devices must be on the CAN network for this mode to work successfully.

Definition

int canplus_Listen(CANHANDLE h);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01

Remarks

Return Codes

>0 ERROR_CANPLUS_OK

CAN channel is closed successfully

<=0 Error as noted below:

ERROR_CANPLUS_FAIL

CAN channel could enter listen mode

2.2.12 canplus_SetTimeouts

Summary

Set blocking timeouts for a channel with handle h.

Definition

int canplus_SetTimeouts(CANHANDLE h, _u32 receiveTimeout, _u32 transmitTimeout);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01
receiveTimeout	number of milliseconds to release receive blocks
transmitTimeout	number of milliseconds to release transmit blocks

Remarks

Default blocking is 1sec. This command provides a mechanism to wait for different durations.

Return Codes

>0	ERROR_CANPLUS_OK CAN channel is closed successfully
<=0	Error as noted below: ERROR_CANPLUS_FAIL CAN channel timeouts could not be configured

2.2.13 canplus_setReceiveCallBack

Summary

Define a function that will receive a callback on all incoming messages. This is a blocking call and must be called on a separate thread. To deregister this callback function, the canplus_setReceiveCallback can be called with cbfn equal to NULL.

Definition

int canplus_setReceiveCallBack(CANHANDLE h, LPFNDLL_RECEIVE_CALLBACK cbfn);

Parameters

h	handle returned by canplus_Open for the desired USB2-F-7x01
cbfn	The callback function should be defined as void fn(CANMsg *pMsg);

Remarks

Note that the channel has to be open to be able to set a callback function.

Return Codes

>0	ERROR_CANPLUS_OK CAN channel is closed successfully
<=0	Error as noted below: ERROR_CANPLUS_FAIL CAN channel could enter listen mode

Sample code

```
void cbFunc(CANMsg *pMsg)
        {
                // callback function
        }
        void RegisterCB(LPFNDLL_RECEIVE_CALLBACK cbFunc)
        {
                int status;
                if (ERROR_CANPLUS_OK == (status = canplus_setReceiveCallBack(hnd, cbFunc)))
                        printf("canplus_setCallBack successful!\n");
        }
        DWORD ThreadId;
        HANDLE hThreadCB;

        if ( NULL == ( hThreadCB = CreateThread( NULL,
                        0, (LPTHREAD_START_ROUTINE) RegisterCB,
                        cbFunc, 0, &ThreadId ) ) )
                {
                                // Failure
                }
```

2.2.14 Error Return Codes

Summary

The commands listed in the above sections have the following error codes in common. Not all commands will return all codes:

Definition

```
#define ERROR_CANPLUS_OK 1
#define ERROR_CANPLUS_FAIL -1
#define ERROR_CANPLUS_OPEN_SUBSYSTEM -2
#define ERROR_CANPLUS_COMMAND_SUBSYSTEM -3
#define ERROR_CANPLUS_NOT_OPEN -4
#define ERROR_CANPLUS_TX_FIFO_FULL -5
#define ERROR_CANPLUS_INVALID_PARAM -6
#define ERROR_CANPLUS_NO_MESSAGE -7
#define ERROR_CANPLUS_MEMORY_ERROR -8
#define ERROR_CANPLUS_NO_DEVICE -9
#define ERROR_CANPLUS_TIMEOUT -10
#define ERROR_CANPLUS_INVALID_HARDWARE -11
```

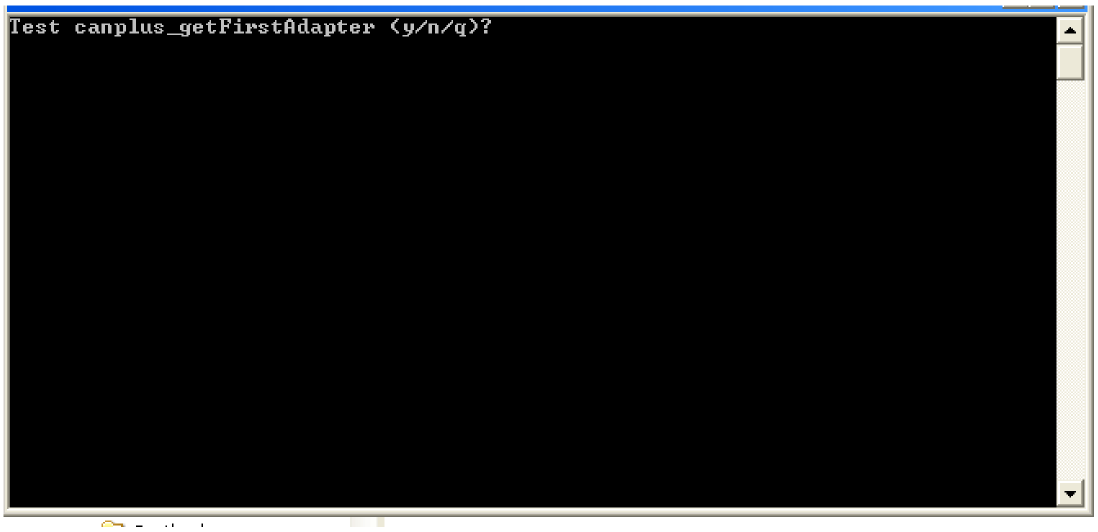
3 Example Application

The DLL Demo Usage Application is a C-language example that provides a quick test of the API commands defined in this Programming Guide. The application source code can be obtained through Connective Peripherals Support. Contact information is available at the end of this Guide.

DLL Demo usage instructions

1. Extract all the files from ZIP file to your PC. The DLL demo executable file is located in the following folder (DLLDemo.exe)
   ..\DLLDemo\release\DLLDemo.exe
2. Connect the USBCANPlus (USB2-F-7x01) on a operational CAN network
3. Run the DLLDemo.exe the following APIs under test will start running

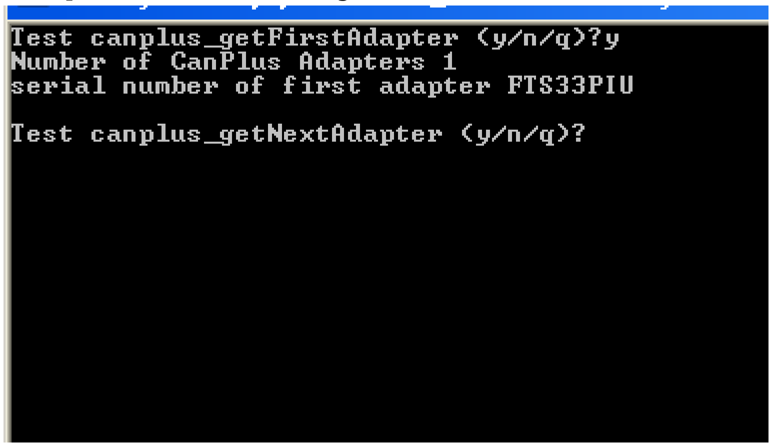
   a) *API: canplus_getFirstAdapter*
   Type in small letter " y " and <Enter>, it will execute the API



```
Test canplus_getFirstAdapter (y/n/q)?
```

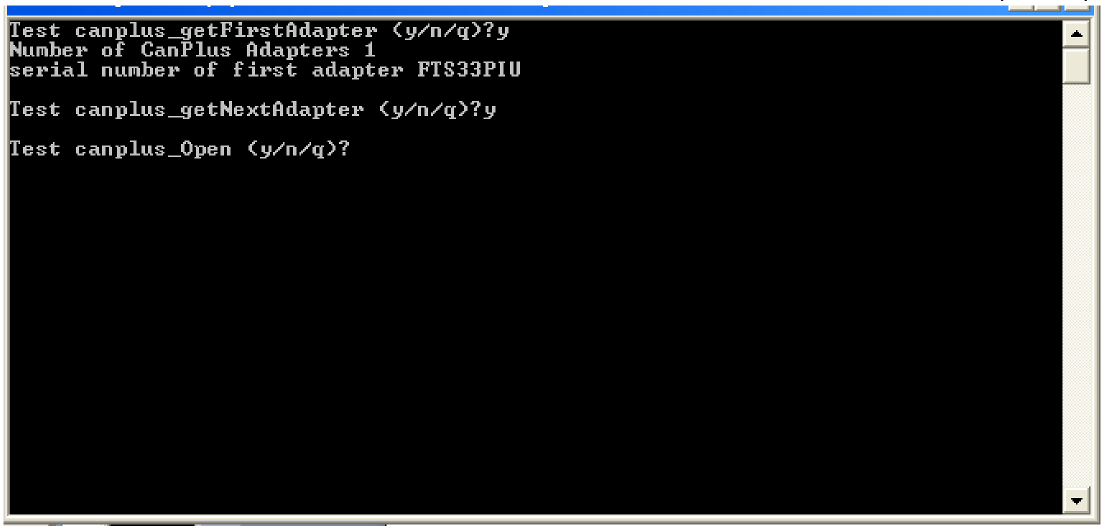
   If the API is executed successfully it will show how many adapters are connected and serial number of first adapter.

   b) *API: canplus_getNextAdapter:*
   Then it will prompt to run canplus_getNextAdapter API. Since for simple testing we have connected only one adapter this API will not return anything if you run it. So type in 'y' [small letter] and <ENTER> it will go to next API test.



```
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?
```

c) *API: canplus_Open:*

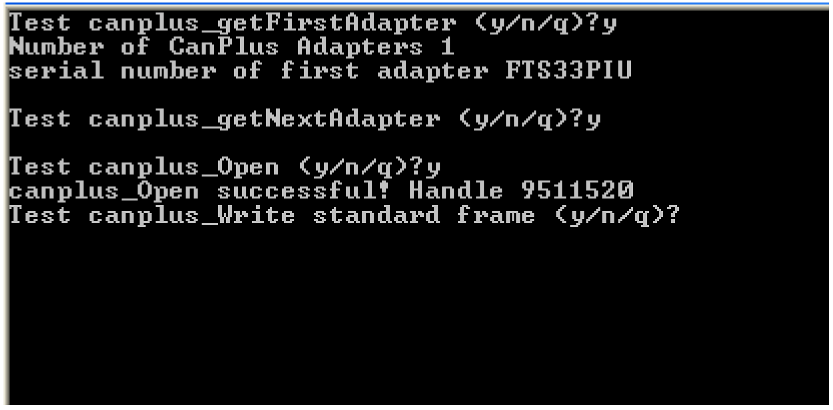


```
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?
```

Type in ‘y’ and <ENTER>, It will run the canplus_Open API and return the USB handle number operating with the device. At this point CAN Open command is executed with the Supplied CAN bit rates, time stamp flags. So the Green LED on the device will be solid “green”. In the demo code the time stamp is turned ON, and CAN bit rate is 1 mbps.

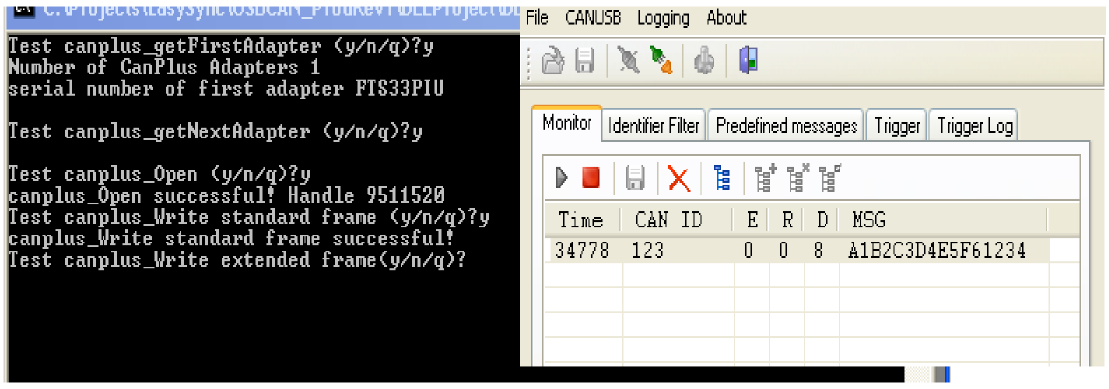
d) *API: canplus_Write:*

At this point the demo code get into testing canplus_Write API. First Standard frame will be transmitted with ID : 123, 8 bytes data, Data: A1B2C3D4E5F61234



```
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?
```

Type ‘y’ and <ENTER>



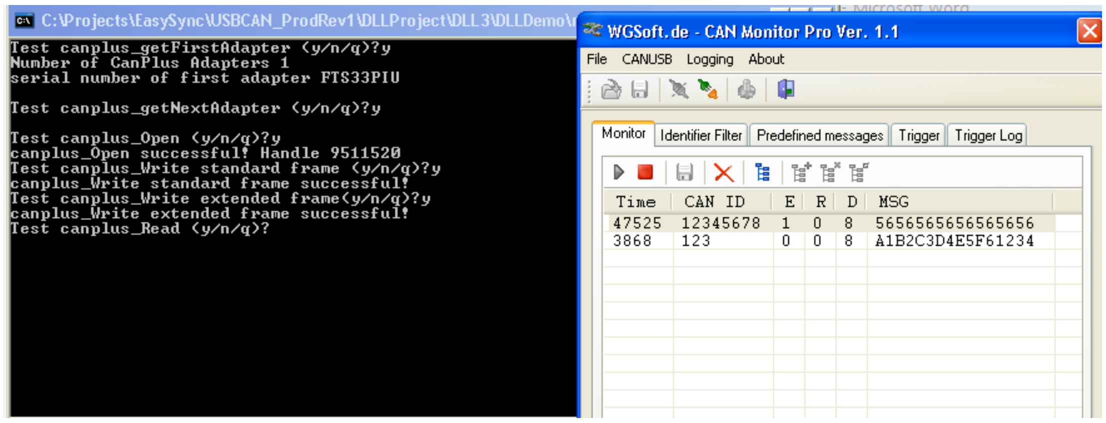
```
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?
```

Time	CAN ID	E	R	D	MSG
34778	123	0	0	8	A1B2C3D4E5F61234

Then you can monitor the standard CAN frame transmitted on CAN network.

Next demo is to use API canplus_Write for transmitting extended frame. Demo code is written with extended ID: 12345678, DataBytes: 8, Data: 5656565656565656

Type 'y' and press <ENTER>

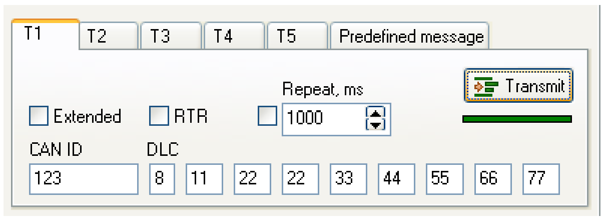


```
C:\Projects\EasySync\USBCAN_ProdRev1\DLLProject\DLL3\DLLDemo\
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?
```

WGSoft.de - CAN Monitor Pro Ver. 1.1

```
 Time    CAN ID     E  R  D  MSG
 47525   12345678   1  0  8  5656565656565656
 3868    123        0  0  8  A1B2C3D4E5F61234
```

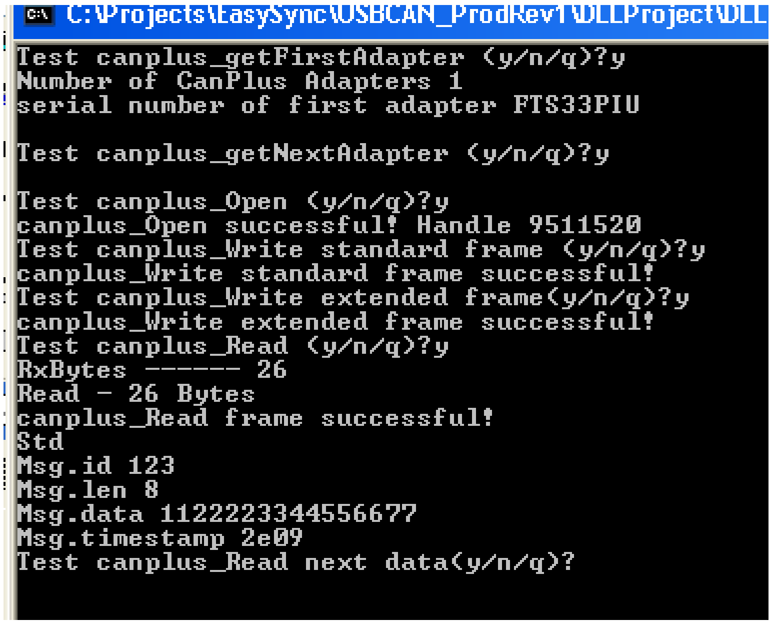
e) *API: canplus_Read:*
Next API demo is canplus_Read. You can send multiple bytes on CAN network and the API will read the received CAN messages.



☐ Extended ☐ RTR ☐ Repeat, ms 1000

CAN ID	DLC								
123	8	11	22	22	33	44	55	66	77

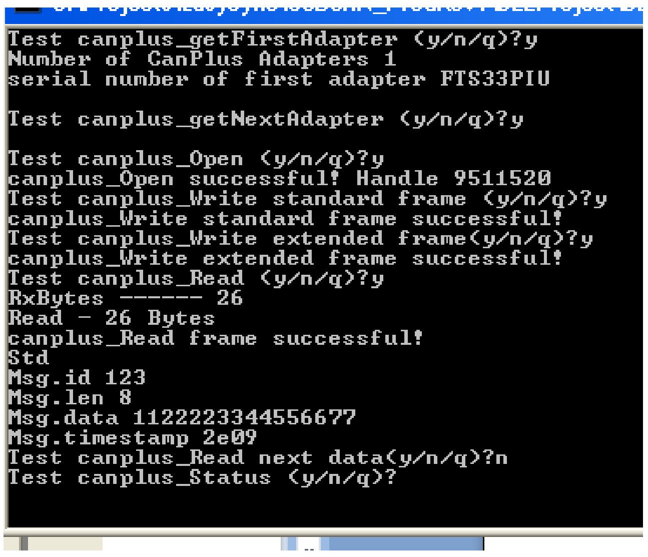
Type 'y' and <ENTER>



```
C:\Projects\EasySync\USBCAN_ProdRev1\DLLProject\DLL
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------ 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 2e09
Test canplus_Read next data(y/n/q)?
```

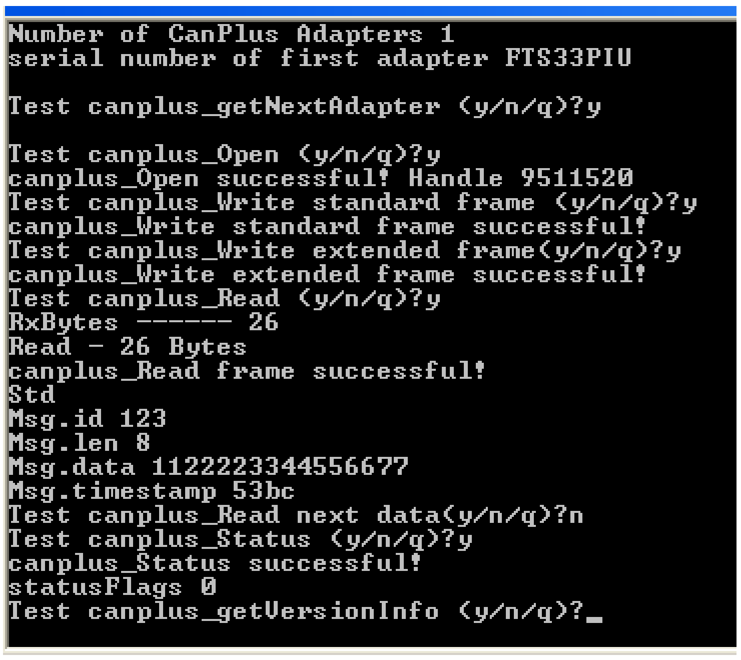
The API canplus_Read, reads the received message and we have a buffer to store 1024 standard/extended frames in DLL. This API will return the number of bytes received, with details on can message like ID, data length, DATA payload and timestamp info…

Now type '**n**' to exit reading the CAN messages.



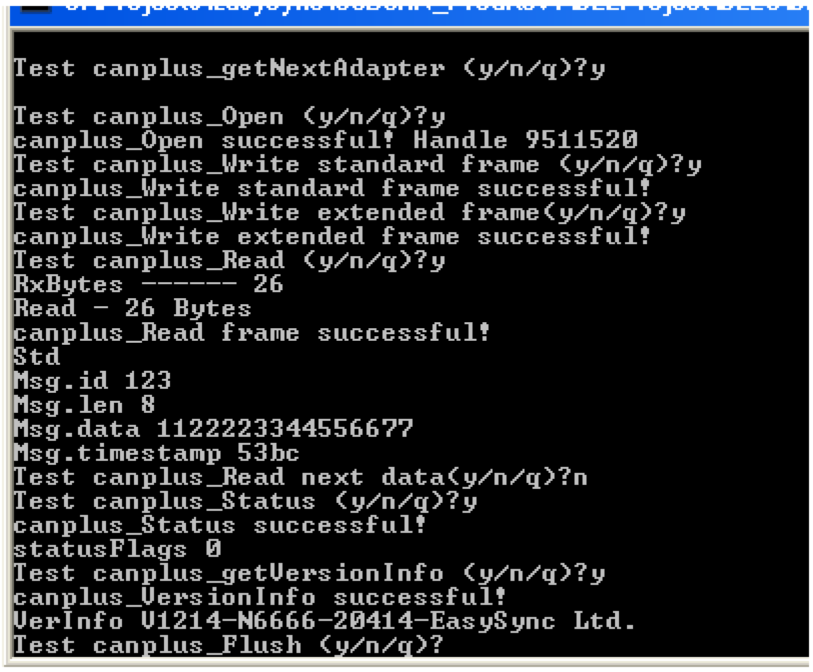
```
Test canplus_getFirstAdapter (y/n/q)?y
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------- 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 2e09
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?
```

f) *API: canplus_Status:*
Next API in the demo code is canplus_Status. It will return the status of CAN communication bus. Type 'y' and <ENTER> If there are no errors on CAN bus statusFlags value '0' will be returned



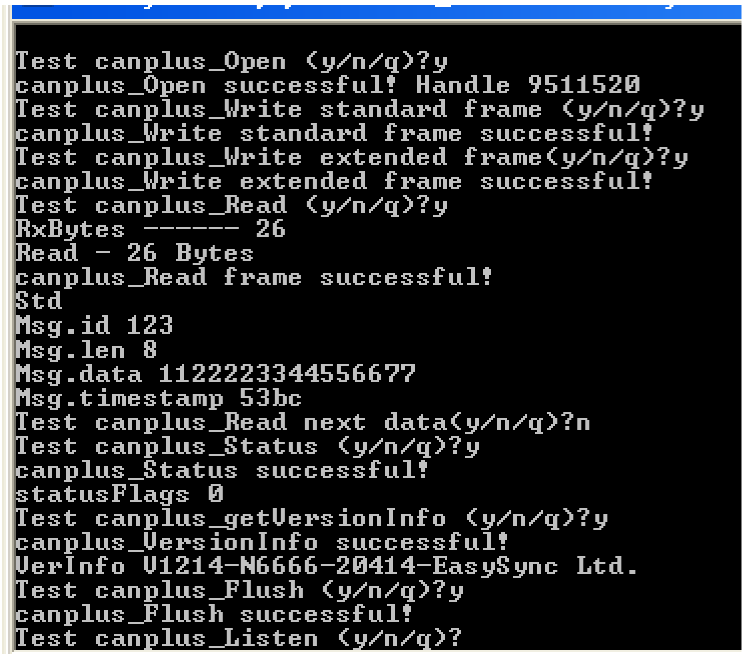
```
Number of CanPlus Adapters 1
serial number of first adapter FTS33PIU

Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------- 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?_
```

g) *API: canplus_getVersionInfo:*
Next API is canplus_getVersionInfo. This will return CAN device serial number, HW/SW version and driver version. Type 'y' and <ENTER>



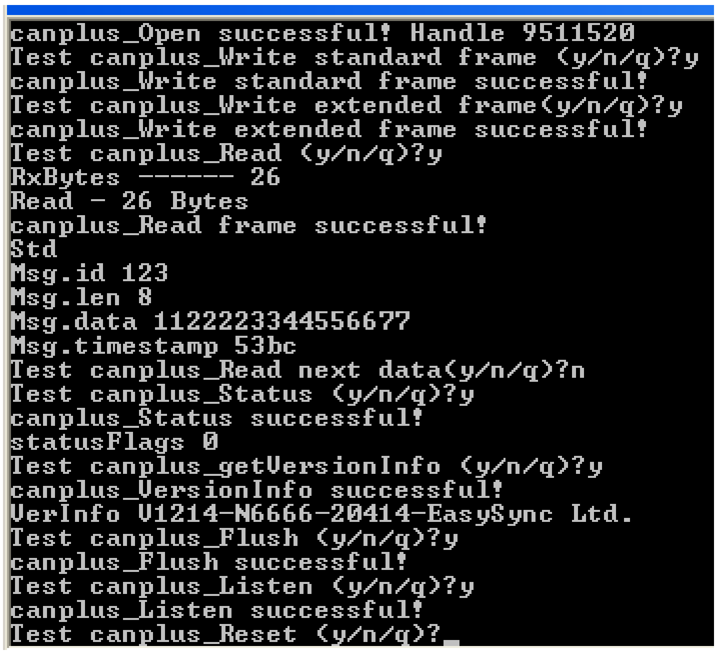
```
Test canplus_getNextAdapter (y/n/q)?y

Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------- 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?y
canplus_VersionInfo successful!
VerInfo V1214-N6666-20414-EasySync Ltd.
Test canplus_Flush (y/n/q)?
```

h) *API: canplus_Flus:*
Next API is canplus_Flush. This will flush the CAN TX and RX buffers. Type'y' and <ENTER>



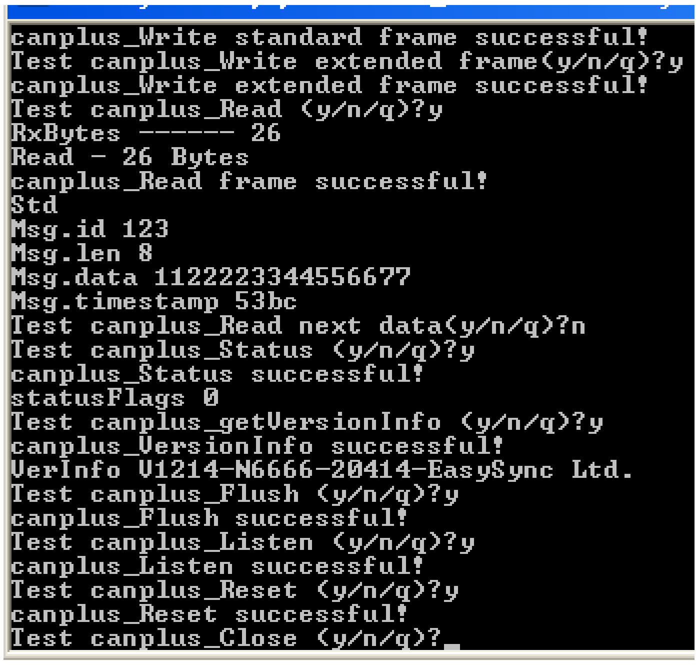
```
Test canplus_Open (y/n/q)?y
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------- 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?y
canplus_VersionInfo successful!
VerInfo V1214-N6666-20414-EasySync Ltd.
Test canplus_Flush (y/n/q)?y
canplus_Flush successful!
Test canplus_Listen (y/n/q)?
```

i) API: canplusListen:
Next API is canplus_Listen. This API will place the USB2-F-7x01 into Listen mode. The green LED will start blinking. Type 'y' and <ENTER>



```
canplus_Open successful! Handle 9511520
Test canplus_Write standard frame (y/n/q)?y
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------ 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?y
canplus_VersionInfo successful!
VerInfo V1214-N6666-20414-EasySync Ltd.
Test canplus_Flush (y/n/q)?y
canplus_Flush successful!
Test canplus_Listen (y/n/q)?y
canplus_Listen successful!
Test canplus_Reset (y/n/q)?_
```

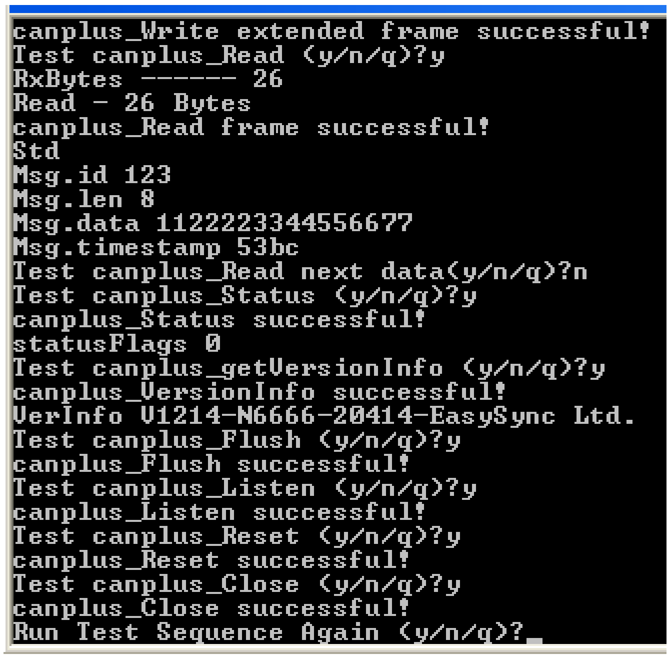
j) *API: canplus_Reset:*
Next API is canplus_Reset. This API will reset the PIC on the USB2-F-7x01. Type 'y' and <Enter>



```
canplus_Write standard frame successful!
Test canplus_Write extended frame(y/n/q)?y
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------ 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?y
canplus_VersionInfo successful!
VerInfo V1214-N6666-20414-EasySync Ltd.
Test canplus_Flush (y/n/q)?y
canplus_Flush successful!
Test canplus_Listen (y/n/q)?y
canplus_Listen successful!
Test canplus_Reset (y/n/q)?y
canplus_Reset successful!
Test canplus_Close (y/n/q)?_
```

Wait till you see the Green LED blinking status.

k) *API: canplus_Close:*
Next API is canplus_Close. This API will close the CAN channel and USB handle.



```
canplus_Write extended frame successful!
Test canplus_Read (y/n/q)?y
RxBytes ------- 26
Read - 26 Bytes
canplus_Read frame successful!
Std
Msg.id 123
Msg.len 8
Msg.data 1122223344556677
Msg.timestamp 53bc
Test canplus_Read next data(y/n/q)?n
Test canplus_Status (y/n/q)?y
canplus_Status successful!
statusFlags 0
Test canplus_getVersionInfo (y/n/q)?y
canplus_VersionInfo successful!
VerInfo V1214-N6666-20414-EasySync Ltd.
Test canplus_Flush (y/n/q)?y
canplus_Flush successful!
Test canplus_Listen (y/n/q)?y
canplus_Listen successful!
Test canplus_Reset (y/n/q)?y
canplus_Reset successful!
Test canplus_Close (y/n/q)?y
canplus_Close successful!
Run Test Sequence Again (y/n/q)?_
```

Observe the CAN Close Green LED blink status. At this point you can run the test sequence again by typing ‘y’ or exit the DLLDemo application by typing ‘n’ or ‘q’.

4 Contact Information

Global Headquarters – Singapore

Connective Peripherals Pte Ltd
178 Paya Lebar Road
#07-03
Singapore 409030

Tel: +65 67430980
Fax: +65 68416071

E-Mail (Sales)	sales@connectiveperipherals.com
E-Mail (Support)	support@connectiveperipherals.com
Web Site URL	http://www.connectiveperipherals.com
Web Shop URL	http://www.connectiveperipherals.com

Appendix A – References

Bosch CAN Specification, Version 2.0:

http://www.semiconductors.bosch.de/pdf/can2spec.pdf

CAN in Automation (CiA):

www.can-cia.org

Future Technology Devices International Ltd. (FTDI)

www.ftdichip.com

Microchip

www.microchip.com

Appendix B – Revision History

Revision	Changes	Date
1.0	Initial release	2009-02-20
1.01	Changed references from USB2-F-7001 to USB2-f-7x01 to accommodate isolated adapter	2009-02-25
1.1	Re-branding to reflect the migration of the product to Connective Peripherals name – logo change, copyright changed, contact information Changed, all internal hyperlinks changed.	2019-03-21