EMUC-B201

USB to CANbus

User Manual

Table of Contents

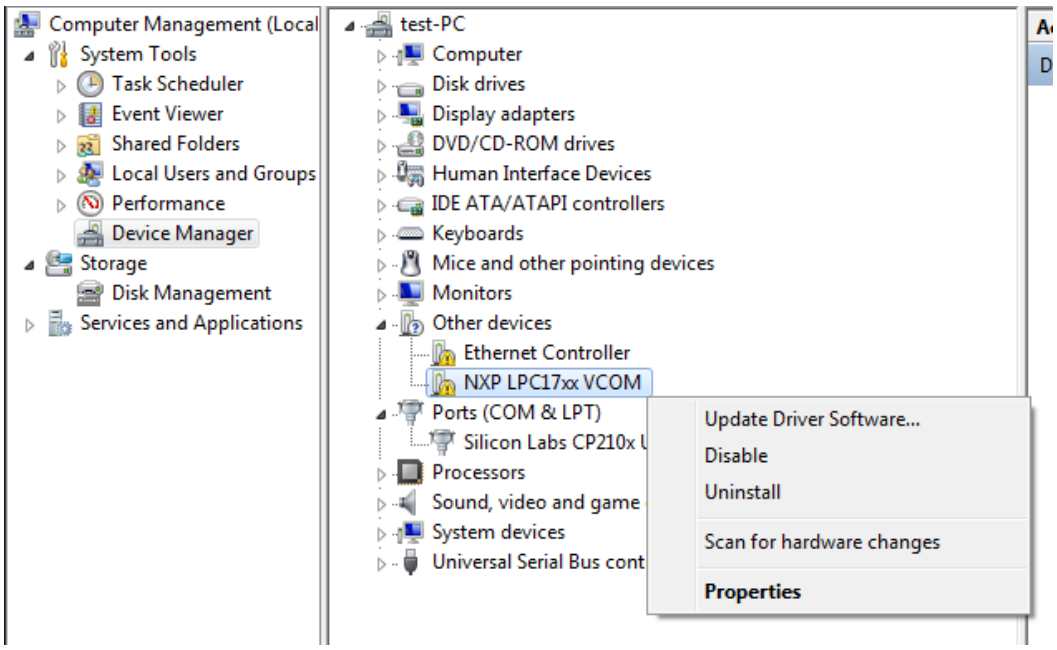
1. Windows 1
1.1. Install Driver 1
1.2. EMUC-B201 Test Utility 3
2. Linux 4
2.1. Install Driver 4
2.2. EMUC-B201 Test Utility 4
3. Software API 5
3.1. General Definition 5
3.2. CAN Channel and Baudrate Parameter 6
3.3. VER_INFO 6
3.4. DATA_INFO 6
3.5. Function Definition 7
3.6. Load Library (C/C++) 7
3.7. Function Description 8
3.7.1. EMUCShowVer 8
3.7.2. EMUCOpenDevice 9
3.7.3. EMUCCloseDevice 9
3.7.4. EMUCSetCAN 9
3.7.5. EMUCResetCAN 10
3.7.6. EMUCSend 10
3.7.7. EMUCReceive 11
Contact us 13

1. Windows

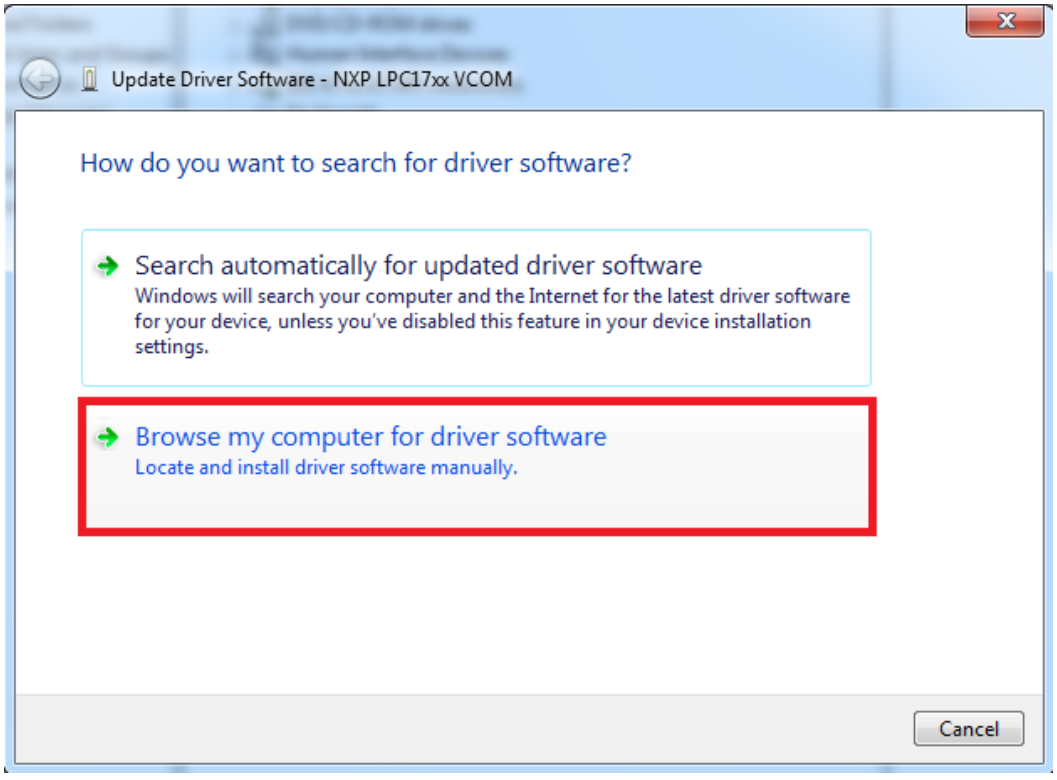
1.1. Install Driver

Install EMUC-B201 either into mPCIe slot or with USB pin header. The device named “NXP LPC 17xx VCOM” can be found in “Device Manager”.

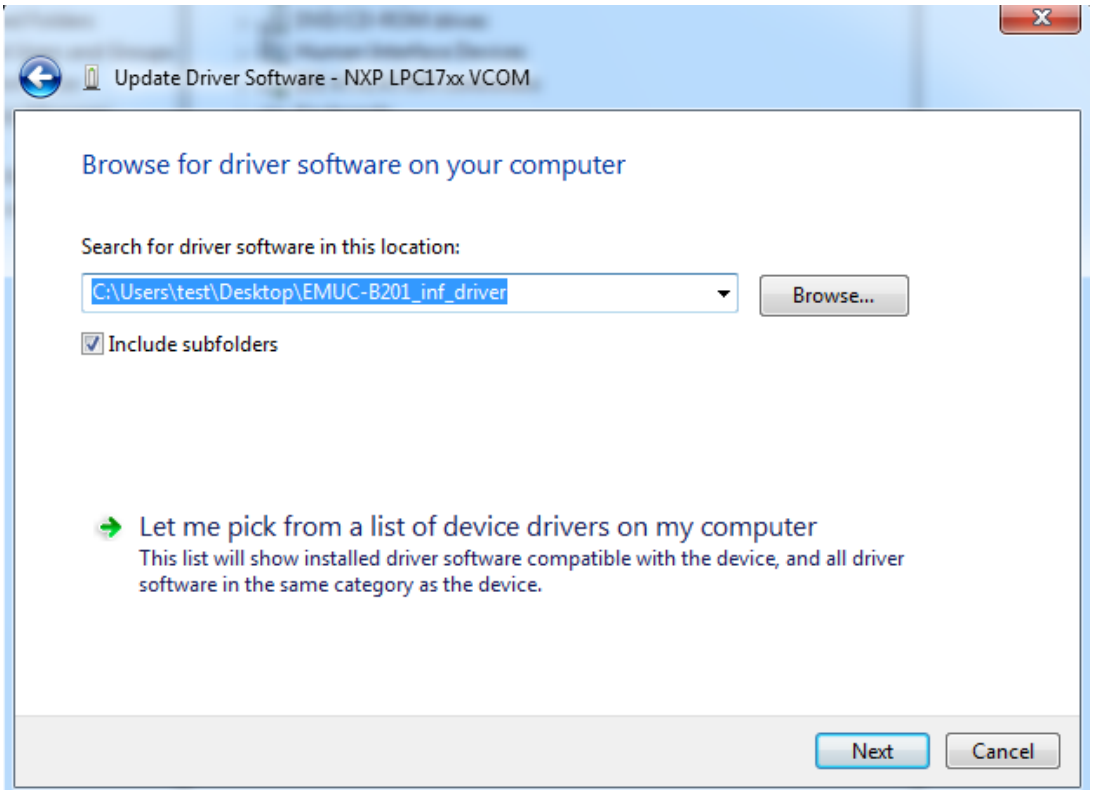
Click “Update Driver Software” to install driver.



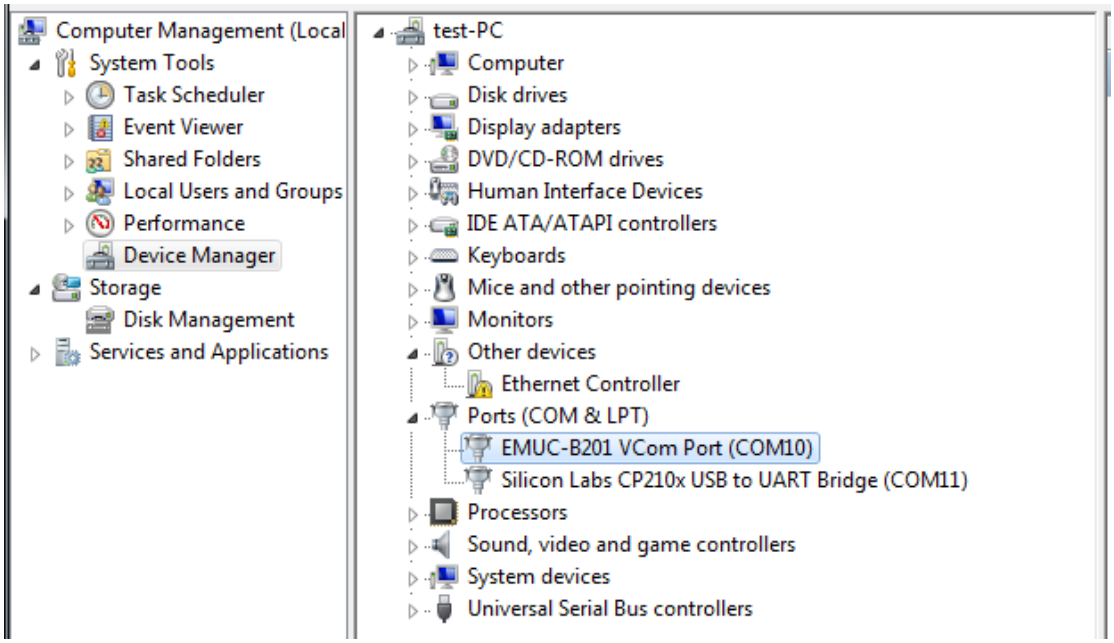
Select “Browse my computer for driver software”



Browse the path linking to EMUC-B201 driver.

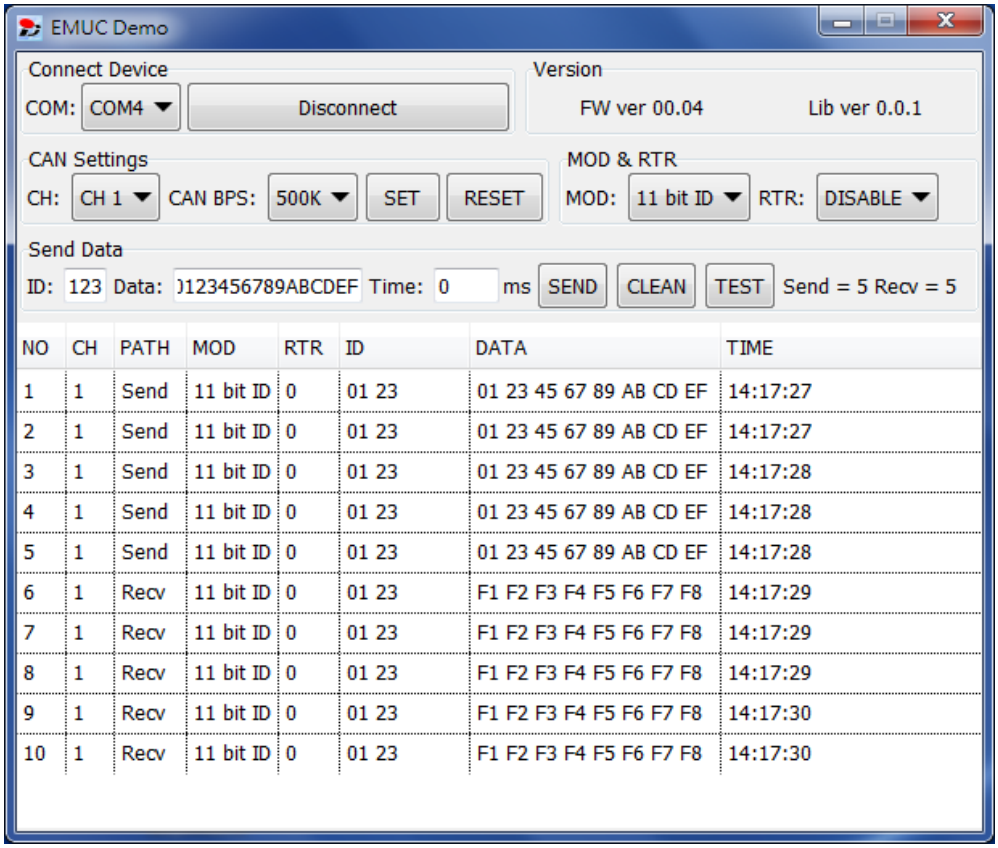


After installing the driver, device can be recognized as a COM port named “EMUC-B201 VCom Port”.



1.2. EMUC-B201 Test Utility

You can use this tool to test EMUC-B201.



NO	CH	PATH	MOD	RTR	ID	DATA	TIME
1	1	Send	11 bit ID	0	01 23	01 23 45 67 89 AB CD EF	14:17:27
2	1	Send	11 bit ID	0	01 23	01 23 45 67 89 AB CD EF	14:17:27
3	1	Send	11 bit ID	0	01 23	01 23 45 67 89 AB CD EF	14:17:28
4	1	Send	11 bit ID	0	01 23	01 23 45 67 89 AB CD EF	14:17:28
5	1	Send	11 bit ID	0	01 23	01 23 45 67 89 AB CD EF	14:17:28
6	1	Recv	11 bit ID	0	01 23	F1 F2 F3 F4 F5 F6 F7 F8	14:17:29
7	1	Recv	11 bit ID	0	01 23	F1 F2 F3 F4 F5 F6 F7 F8	14:17:29
8	1	Recv	11 bit ID	0	01 23	F1 F2 F3 F4 F5 F6 F7 F8	14:17:29
9	1	Recv	11 bit ID	0	01 23	F1 F2 F3 F4 F5 F6 F7 F8	14:17:30
10	1	Recv	11 bit ID	0	01 23	F1 F2 F3 F4 F5 F6 F7 F8	14:17:30

Connect Device

COM: Select the port which is recognized as “EMUC VCom Port” in Device Manager, then click “Connect”

CAN Setting

CH: Send frame from CAN1, CAN2 or both.

CAN BPS: Set CAN baudrate, EMUC supports 50K, 125K, 250K, 500K, 1000K, after selecting please click “SET” to take effect.

RESET: Reset CAN port to clear register.

MOD & RTR

MOD: Select 11 or 29 bit CAN ID mode.

RTR: Enable or disable Remote Transmit Request.

Send Data

ID: Input CAN ID, maximum of 11bit and 29bit is 7FF and 1FFFFFFF.

Data: Input data , max is 8 byte.

Time: Set time interval of sending frame. It will only send once when time =0. The Minimum value is 10ms. During sending, you can click “SEND” again or set time=0 to stop it.

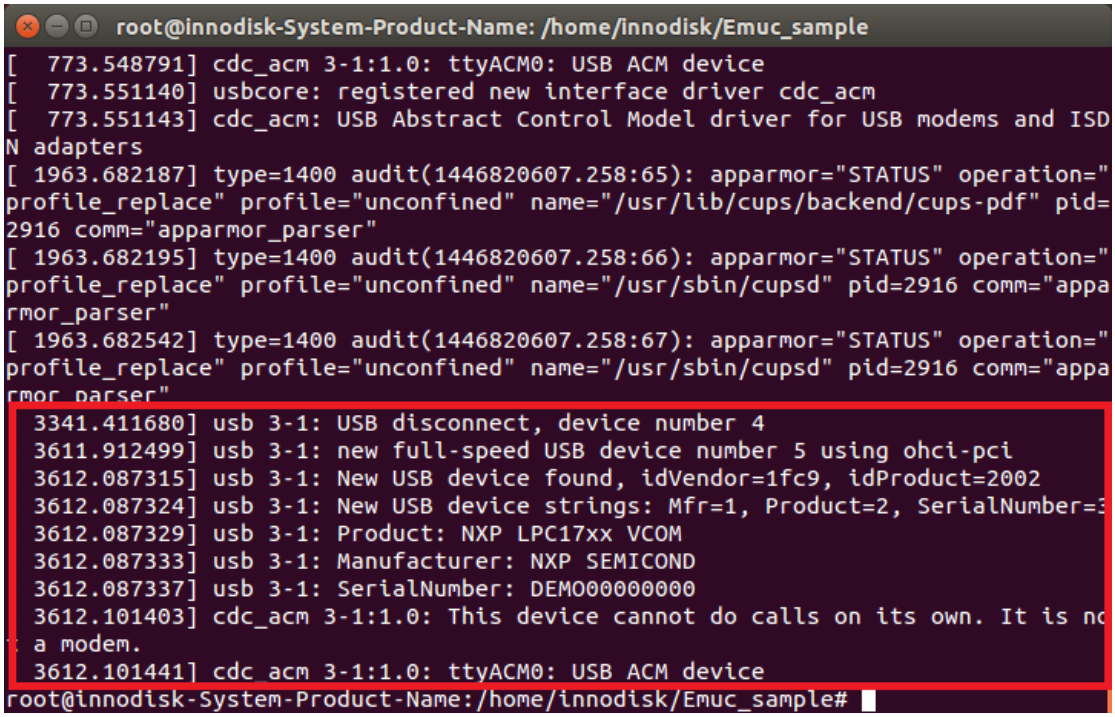
CLEAN: Clear the record list.

2. Linux

2.1. Install Driver

Install EMUC-B201 either into mPCIe slot or with USB pin header.The device will be recognized as ttyACM% (%=0,1…) by using native CDC-ACM driver.

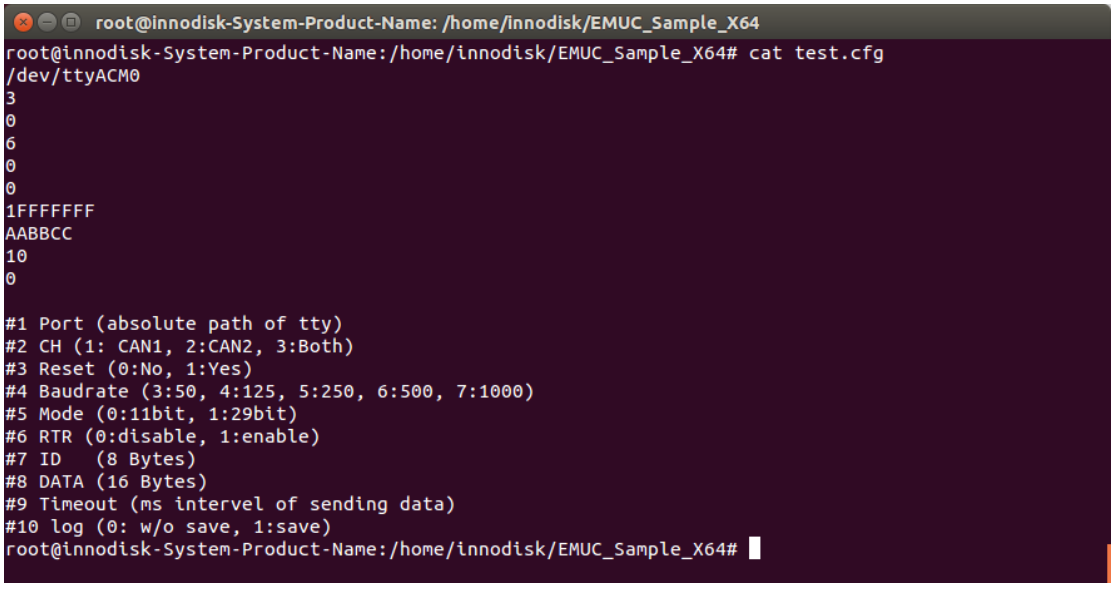
Type command “dmesg” to see below messages.



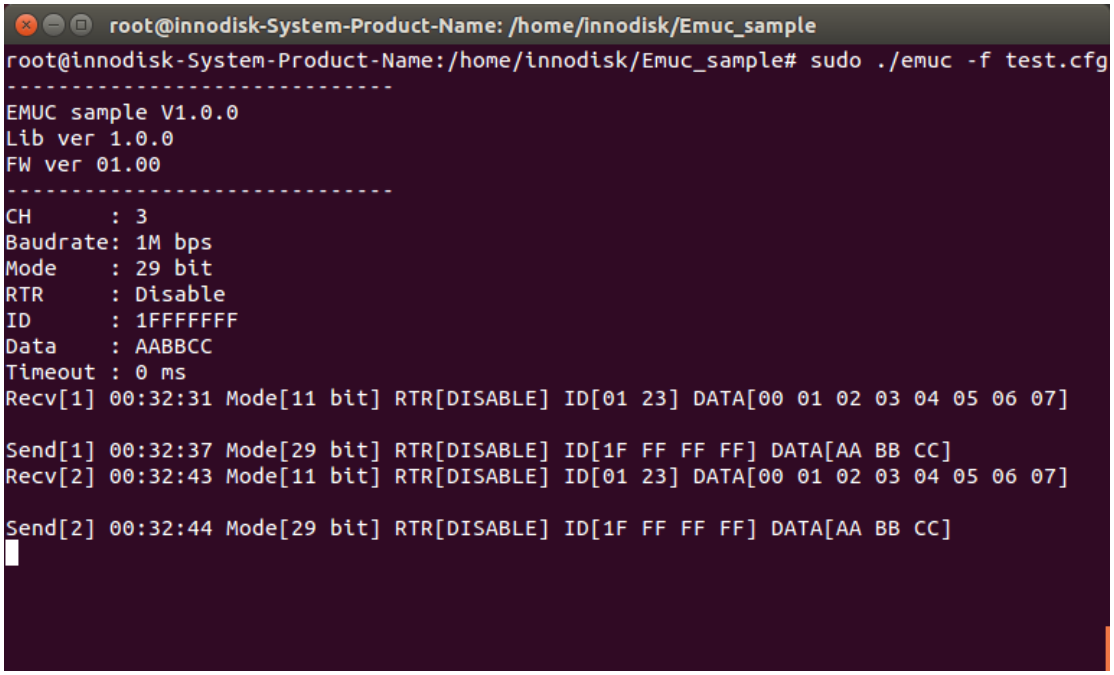
2.2. EMUC-B201 Test Utility

You can use “EMUC_Sample” to test EMUC-B201.

Before executing the command, you must check test.cfg as below.



Execute “sudo ./emuc -f test.cfg” to start thread by root privilege.
When Timeout=0ms, you can press “Enter” to send frame once.



3. Software API

EMUC API is based on a dynamic library (DLL) in Windows and static library (.a) in Linux to control EMUC-B201.
lib_emuc.h includes declaration and data structure requested for programming.

NOTE: Linux API supports the following port names.
"/dev/ttyS0","/dev/ttyS1","/dev/ttyS2","/dev/ttyS3","/dev/ttyS4","/dev/ttyS5",
"/dev/ttyS6","/dev/ttyS7","/dev/ttyS8","/dev/ttyS9","/dev/ttyS10","/dev/ttyS11",
"/dev/ttyS12","/dev/ttyS13","/dev/ttyS14","/dev/ttyS15","/dev/ttyUSB0",
"/dev/ttyUSB1","/dev/ttyUSB2","/dev/ttyUSB3","/dev/ttyUSB4","/dev/ttyUSB5",
"/dev/ttyAMA0","/dev/ttyAMA1","/dev/ttyACM0","/dev/ttyACM1",
"/dev/rfcomm0","/dev/rfcomm1","/dev/ircomm0","/dev/ircomm1",
"/dev/cuau0","/dev/cuau1","/dev/cuau2","/dev/cuau3",
"/dev/cuaU0","/dev/cuaU1","/dev/cuaU2","/dev/cuaU3".

3.1. General Definition

```
#define MAX_COM_PORT      16
#define MAX_BUF           512
```

```
#define ID_LEN              4
#define DATA_LEN            8
#define VER_LEN             16
```

3.2. CAN Channel and Baudrate Parameter

```
enum
{
    /* channel */
    EMUC_CH_1                   = 1,
    EMUC_CH_2,
    EMUC_CH_BOTH,

    /* baudrate */
    EMUC_BAUDRATE_50K       = 3,
    EMUC_BAUDRATE_125K,
    EMUC_BAUDRATE_250K,
    EMUC_BAUDRATE_500K,
    EMUC_BAUDRATE_1M,
};
```

3.3. VER_INFO

Description: Data structure is used for the API function "*EMUCShowVer".*

```
typedef struct
{
    char fw[VER_LEN];
    char api[VER_LEN];
} VER_INFO;
```

3.4. DATA_INFO

Description: Data structure is used for the API function "EMUCReceive" and "EMUCSend".

```
typedef struct
{
    int             com_port;

    int             channel;
    int             mod;
```

```
    int                 rtr;
    int                 dlc;

    unsigned char id[ID_LEN];
    unsigned char data[DATA_LEN];

} DATA_INFO;
```

3.5. Function Definition

```
typedef int    (*EMUC_OPEN_DEVICE)    (int port);
typedef void (*EMUC_CLOSE_DEVICE) (int port);
typedef int    (*EMUC_SHOW_VER)        (int port, VER_INFO *ver);
typedef int    (*EMUC_SET_CAN)         (int port, int ch, int bdrate);
typedef int    (*EMUC_RESET)           (int port);
typedef int    (*EMUC_RECEIVE)         (DATA_INFO *info);
typedef int    (*EMUC_SEND)            (DATA_INFO *info);
```

3.6. Load Library (C/C++)

```
static HMODULE                  hDLL;
static EMUC_OPEN_DEVICE    EMUCOpenDevice;
static EMUC_CLOSE_DEVICE EMUCCloseDevice;
static EMUC_SHOW_VER        EMUCShowVer;
static EMUC_SET_CAN         EMUCSetCAN;
static EMUC_RESET           EMUCReset;
static EMUC_RECEIVE         EMUCReceive;
static EMUC_SEND            EMUCSend;

static int load_dll(void)
{
    hDLL = LoadLibrary("lib_emuc.dll");

    if(!hDLL)
        return FALSE;

    EMUCOpenDevice    = (EMUC_OPEN_DEVICE)    GetProcAddress(hDLL,
"EMUCOpenDevice");
    EMUCCloseDevice = (EMUC_CLOSE_DEVICE) GetProcAddress(hDLL,
"EMUCCloseDevice");
```

```
    EMUCShowVer         = (EMUC_SHOW_VER)        GetProcAddress(hDLL,
"EMUCShowVer");
    EMUCSetCAN          = (EMUC_SET_CAN)         GetProcAddress(hDLL,
"EMUCSetCAN");
    EMUCReset           = (EMUC_RESET)           GetProcAddress(hDLL,
"EMUCReset");
    EMUCReceive         = (EMUC_RECEIVE)         GetProcAddress(hDLL,
"EMUCReceive");
    EMUCSend            = (EMUC_SEND)            GetProcAddress(hDLL,
"EMUCSend");

    return TRUE;
}
```

3.7. Function Description

This chapter describes API functions and parameters.

3.7.1. EMUCShowVer

Description: Get firmware and library version.

SYSTAX:

```
int EMUCShowVer(int port, VER_INFO *ver)
```

Parameter:

```
typedef struct
{
    char fw[VER_LEN];
    char api[VER_LEN];
} VER_INFO;
```

Member:

fw: [output] Firmware version, length 16 bytes

api: [output] API version, length 16 bytes

Return Status Code:

Value	Return Value
0	SUCCESS
-1	EXCEPETION

3.7.2. EMUCOpenDevice

Description: Open virtual COM port.

SYSTAX:

int EMUCOpenDevice(int port)

Parameter:

port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)

Return Status Code:

Value	Return Value
0	SUCCESS
1	COM_NOT_EXIST
5	COM_IN_USE

3.7.3. EMUCCloseDevice

Description: Close virtual COM port.

SYSTAX:

void EMUCCloseDevice(int port)

Parameter:

port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)

3.7.4. EMUCSetCAN

Description: Set baud rate of CAN port.

SYSTAX:

int EMUCSetCAN(int port, int ch, int bdrate)

Parameter:

port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)

ch: [input] The CAN port number. (1:CAN1, 2:CAN2, 3:Both)

CAN Port	VALUE
CAN 1	EMUC_CH1
CAN 2	EMUC_CH2
Both	EMUC_BOTH

bdrate: [input] The baud rate of the CAN port

Baud rate	VALUE
50 kbps	EMUC_BAUDRATE_50K
125 kbps	EMUC_BAUDRATE_125K
250 kbps	EMUC_BAUDRATE_250K
500 kbps	EMUC_BAUDRATE_500K
1M bps	EMUC_BAUDRATE_1M

Return Status Code:

Value	Return Value
0	SUCCESS
-1	UNSUCCESS

3.7.5. EMUCResetCAN

Description: Reset CAN port and clearing register without changing baudrate

SYSTAX:

```
int EMUCResetCAN(int por)
```

Parameter:

port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)

Return Status Code:

Value	Return Value
0	SUCCESS
-1	RESET FAIL

3.7.6. EMUCSend

Description: Send data from USB to CAN.

SYSTAX:

```
int EMUCSend (DATA_INFO *info)
```

Parameter:

```
typedef struct
{
    int com_port;
    int channel;
    int mod;
```

```
    int rtr;
    int dlc;
    unsigned char id[ID_LEN];
    unsigned char data[DATA_LEN];
} DATA_INFO;
```

Member:
com_port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)
channel: [input] The CAN port number. (1:CAN1, 2:CAN2, 3:Both)
mod: [input] ID mode (0:11 bit, 1:29 bit)
rtr: [input] Remote transmit request (0:disable, 1:enable)
dlc: [input] Data length (Range 0 ~ 8)
id[ID_LEN]: [input] CAN ID (ID_LEN = 4)
data[DATA_LEN]: Data (DATA_LEN = 8)

3.7.7. EMUCReceive

Description: Receive data from CAN to USB.

SYSTAX:

```
int EMUCReceive (DATA_INFO *info)
```

Parameter:

```
typedef struct
{
    int com_port;
    int channel;
    int mod;
    int rtr;
    int dlc;
    unsigned char id[ID_LEN];
    unsigned char data[DATA_LEN];
} DATA_INFO;
```

Member:
com_port: [input] The virtual COM port number. (1:COM1, 2:COM2, 3:COM3…)
channel: [output] The CAN port number. (1:CAN1, 2:CAN2, 3:Both)

mod: [output] ID mode (0:11 bit, 1:29 bit)
rtr: [output] Remote transmit request (0:disable, 1:enable)
dlc: [output] Data length (Range 0 ~ 8)
id[ID_LEN]: [output] The CAN ID (ID_LEN = 4)
data[DATA_LEN]: [output] Data (DATA_LEN = 8)

Return Status Code:

Value	Return Value
0	No data
>0	Get data

Contact us

Headquarters (Taiwan)

5F., No. 237, Sec. 1, Datong Rd., Xizhi Dist., New Taipei City 221, Taiwan
Tel: +886-2-77033000
Email: sales@innodisk.com

Branch Offices:

USA

usasales@innodisk.com
+1-510-770-9421

Europe

eusales@innodisk.com
+31-040-282-1818

Japan

jpsales@innodisk.com
+81-45-594-7581

China

sales_cn@innodisk.com
+86-755-21673689

www.innodisk.com

November 6, 2015