ExUC USB CANFD

Linux SocketCAN Driver

Installation Guide

Rev 1.2

Revision History

Revision	Date	Description
1.0	2024/07/05	Initial Release
1.1	2024/10/15	remove 2 API function EMUCReceiveNonblockCS (Used for C#) EMUCEnableSendQueue
1.2	2024/12/23	Add 4port version

Table of Contents

1. Introduction

The documents descript how to install and test innodisk CANFD series expansion card.

We provide CANFD API for application programming in Windows and Linux.

Supported Operation System

Windows	10, 11
Linux (cdc-acm driver)	Kernel 5.x and above
Linux (SocketCAN driver)	Kernel 5.x and above

2. Hardware Installation

2.1. EMUC

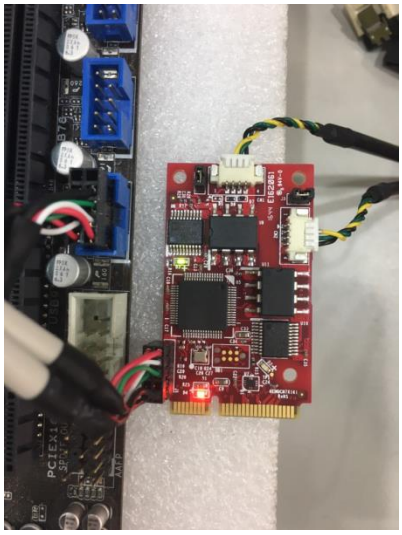
EMUC CAN Bus module uses USB 2.0 input interface, there are dual options to install the module.

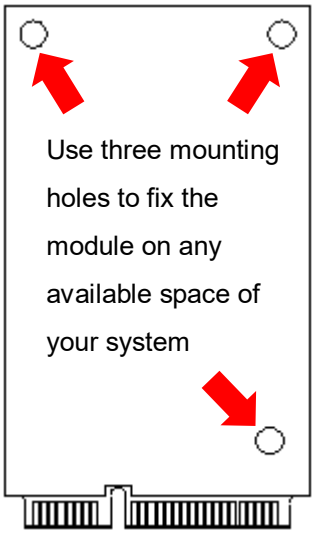
2.1.1. mPCIe Slot

Install the module to mPCIe slot which has USB 2.0 interface.

2.1.2. USB Pin Header

Don’t need to connect mPCIe golden finger, it can be connected through USB pin headers on the PCB to the motherboard. Then use three mounting holes to fix the module on any available space of your system.





NOTE: *This USB cable in the picture is not included in the package; you need to design your own USB cable.*

2.2. EGUC

EGUC CAN Bus module uses USB 2.0 input interface. Please install the card into M.2 B or M key slot which has USB 2.0 interface.

2.3. ESPC

2.3.1. ESPC only

ESPC uses PCIe Gen2 x1 interface. Please install the card into PCIe slot directly.

2.3.2. ESPC+EGUC

Install EGUC-F2S3/F4S3 on M.2 slot (USB interface) can expand 2-4 more CAN ports. Must connect a USB cable (not included) into the 9-pin USB header to provide USB 2.0 signal for the M.2 slot.

3. Linux OS

Type command “lsusb” to check USB CAN device exist.

- ExUC CAN FD dual ports

```
Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Device 004: ID 05e3:0608 Genesys Logic, Inc. Hub
Device 015: ID 413c:250e Dell Computer Corp. Dell Laser Mouse MS3220
Device 014: ID 1b1c:1b4f Corsair CORSAIR K68 RGB Mechanical Gaming Keyboard
Device 013: ID 1a40:0101 Terminus Technology Inc. Hub
Device 009: ID 196d:f004 innodisk innodisk USB Dual CANFD
Device 008: ID 0e8d:0608 MediaTek Inc. Wireless_Device
Device 006: ID 048d:5702 Integrated Technology Express, Inc. ITE Device
Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- ExUC CAN FD quad ports

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 001 Device 003: ID 196d:0201 Innodisk USB Drive 3ME
Bus 001 Device 006: ID 046d:c53f Logitech, Inc. USB Receiver
Bus 001 Device 004: ID 1b1c:1b4f Corsair CORSAIR K68 RGB Mechanical Gaming Keyboard
Bus 001 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 001 Device 008: ID 0e8d:0608 MediaTek Inc. Wireless_Device
Bus 001 Device 007: ID 048d:5702 Integrated Technology Express, Inc. ITE Device
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 002: ID 196d:f002 innodisk USB Quad CANFD
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

3.1. Driver Installation

The device will be recognized as ttyACM% (%=0, 1…) by using CDC-ACM kernel driver.

Note: *Linux kernel has native CDC-ACM kernel driver. Some Linux OS may need to add CDC-ACM configuration manually in building process. In different Linux OS may have different tty name.*

Type command "dmesg" to see messages below.
Generally the name would be ttyACM0 or ttyACM1 in Linux.

- ExUC CAN FD dual ports

```
[  939.001271] usb 1-2: new high-speed USB device number 9 using xhci_hcd
[  939.150275] usb 1-2: New USB device found, idVendor=196d, idProduct=f004, bcdDevice= 3.00
[  939.150288] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[  939.150293] usb 1-2: Product: innodisk USB Dual CANFD
[  939.150297] usb 1-2: Manufacturer: innodisk
[  939.153101] cdc_acm 1-2:1.0: ttyACM0: USB ACM device
```

- ExUC CAN FD quad ports

If the card is 4-port version, there will be 2 ttyACM ports.

```
[    4.144315] usb 3-1: New USB device found, idVendor=196d, idProduct=f002, bcdDevice= 3.00
[    4.144322] usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[    4.144327] usb 3-1: Product: USB Quad CANFD
[    4.144330] usb 3-1: Manufacturer: innodisk
```

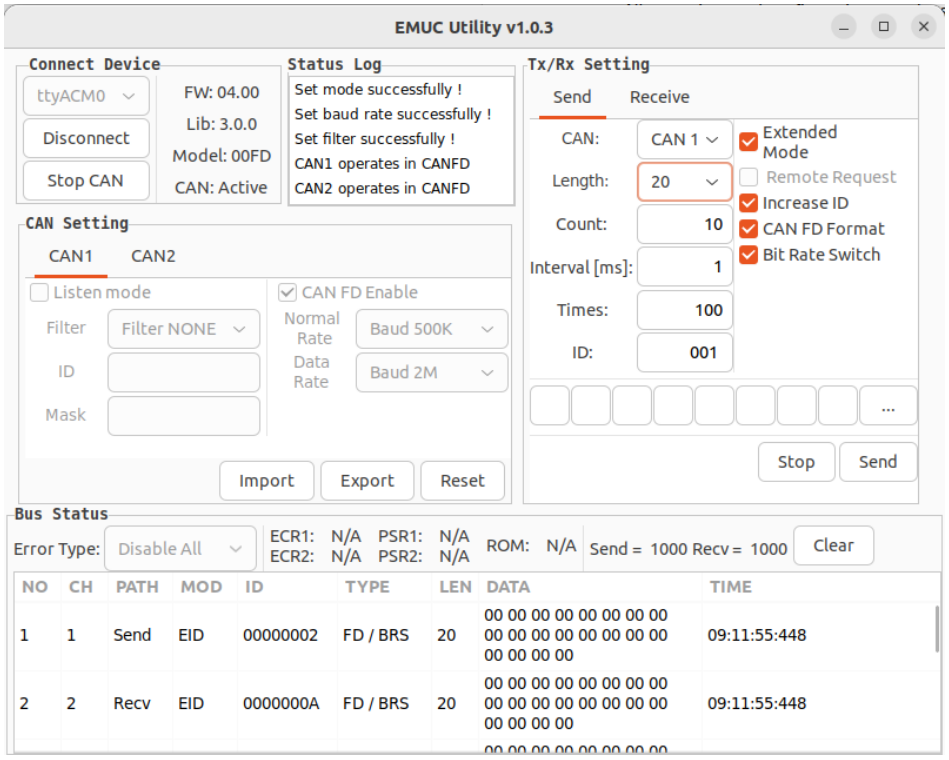
```
[    7.741564] cdc_acm 3-1:1.0: ttyACM0: USB ACM device
[    7.742189] cdc_acm 3-1:1.2: ttyACM1: USB ACM device
[    7.742231] usbcore: registered new interface driver cdc_acm
[    7.742233] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
```

3.2. CANFD Test Utility

All operations and configurations are the same as Windows version. Please refer to *3.2 CAN FD Test Utility*

Before running the utility, you need to use command "chmod +x" to give executable permission to it.

```
root@innodisk:/home/innodisk/2emuc/Utility# chmod +x emuc
root@innodisk:/home/innodisk/2emuc/Utility# ./emuc
```



3.3. SocketCAN

ExUC can support SocketCAN by additional driver and user space tool in Linux kernel. Before installing SocketCAN driver, you must confirm that the Linux Kernel include SocketCAN kernel module and recognize ExUC as ttyACM%(%=0,1,…) by using native CDC-ACM driver.

3.3.1. Build driver and user-space tool

Please copy kernel development packages into your system and type “make” command in root folder of this package.

There should be two output files:

- **emuc2socketcan.ko**: Kernel driver of ExUC SocketCAN
- **emucd_32** or **emucd_64**: User-space tool for enabling ExUC SocketCAN

```
root@inno-2034-dev:/home/jeff/Documents/CANFD/SocketCAN# make
make[1]: Entering directory '/home/jeff/Documents/CANFD/SocketCAN/driver'
make -C/lib/modules/`uname -r`/build M=/home/jeff/Documents/CANFD/SocketCAN/driver modules
make[2]: Entering directory '/usr/src/linux-headers-6.5.0-15-generic'
  CC [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/main.o
  CC [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/emuc_parse.o
  CC [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/transceive.o
  LD [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/emuc2socketcan.o
  MODPOST /home/jeff/Documents/CANFD/SocketCAN/driver/Module.symvers
  CC [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/emuc2socketcan.mod.o
  LD [M]  /home/jeff/Documents/CANFD/SocketCAN/driver/emuc2socketcan.ko
  BTF [M] /home/jeff/Documents/CANFD/SocketCAN/driver/emuc2socketcan.ko
Skipping BTF generation for /home/jeff/Documents/CANFD/SocketCAN/driver/emuc2socketcan.ko due to unavailability of vmlinux
make[2]: Leaving directory '/usr/src/linux-headers-6.5.0-15-generic'
make[1]: Leaving directory '/home/jeff/Documents/CANFD/SocketCAN/driver'
make[1]: Entering directory '/home/jeff/Documents/CANFD/SocketCAN/utility'
  Compiling 'ini.c' ...
  Compiling 'main.c' ...
  Building 'emucd_64' VER=...
make[1]: Leaving directory '/home/jeff/Documents/CANFD/SocketCAN/utility'
root@inno-2034-dev:/home/jeff/Documents/CANFD/SocketCAN#
```

You can type “emucd_64 -h” for help.

./emucd_64 -s7 /dev/ttyACM0 (500 KBPS on both channel)
./emucd_64 -s79 /dev/ttyACM0 (500 KBPS on ch1, 1000 KBPS on ch2)
./emucd_64 -s7 -d2 -f11 /dev/ttyACM0 (500 KBPS/2 MBPS with CAN FD on both channel)

```
jeff@inno-2034-dev:~/Documents/CANFD/SocketCAN$ ./emucd_64

Usage: ./emucd_64 [options] <tty> [canif-name] [canif2-name]

Options: -s <speed>[<speed>] (set CAN speed 3..7)
                4: 100  KBPS
                5: 125  KBPS
                6: 250  KBPS
                7: 500  KBPS
                8: 800  KBPS
                9: 1000 KBPS
                A: 400  KBPS
                B: CUSTOM_OPT_0
                C: CUSTOM_OPT_1
         -d <DataRate> [<DataRate>] (set DataRate 2...10)
                2: 2    MBPS
                4: 4    MBPS
                5: 5    MBPS
                6: 6    MBPS
         -e <errorType>[<errorType>] (set CANbus error type)
                0: EMUC_DIS_ALL
                1: EMUC_EE_ERR
                2: EMUC_BUS_ERR
                3: EMUC_EN_ALL
         -T <index>[<index>] (set custom timing configuration)
                0: set time_cfg0.ini to Customized_OPT_INDEX 0
                1: set time_cfg1.ini to Customized_OPT_INDEX 1
                2: set both cfg.ini to both Customized_OPT_INDEX
         -f          (enable CAN FD)
         -F          (stay in foreground; no daemonize)
         -h          (show this help page)
         -v          (show version info)
         -t          (set open tty device timeout [sec])

Examples:
./emucd_64 -v /dev/ttyACM0
./emucd_64 -s7 /dev/ttyACM0
./emucd_64 -s7 -e3 /dev/ttyACM0
./emucd_64 -s79 /dev/ttyACM0 can0 can1
./emucd_64 -s79 -t10 /dev/ttyACM0 can0 can1
./emucd_64 -sB -T /dev/ttyACM0 # set bit timing config
./emucd_64 -s7 -d2 -f11 /dev/ttyACM0
(Note: emucd_32 for 32-bit OS)
```

3.3.2. SocketCAN Driver Installation

There are shell scripts “start.sh” and “end.sh” to install the driver and enable SocketCAN interface.

start.sh

Please modify the baud rate, data rate, FDF_x(x=1,2) and tty port setting depend on the environment needs.

- ExUC CAN FD dual ports

```
dev_name=ttyACM0
baudrate=7 # 0~3: support FW version >= 03.00
           # 0: 5   KBPS, 1: 10  KBPS,  2: 20  KBPS, 3: 50  KBPS,
           # 4: 100 KBPS, 5: 125 KBPS,  6: 250 KBPS, 7: 500 KBPS,
           # 8: 800 KBPS, 9: 1 MBPS,    A: 400 KBPS
datarate=2 # 2: 2 MBPS, 4: 4 MBPS,  5: 5 MBPS, 6: 6 MBPS
FDF_1=1 # 0: DISABLE CAN1 FD mode (aka CAN2.0B only), 1:ENABLE CAN1 FD mode (downward compatibility CAN2.0B)
FDF_2=1 # 0: DISABLE CAN2 FD mode (aka CAN2.0B only), 1:ENABLE CAN2 FD mode (downward compatibility CAN2.0B)
```

- ExUC CAN FD quad ports

```
### parameter
### /dev/ttyACM0
dev_1=ttyACM0
dev_1_socket_1_name=canfd0
dev_1_socket_2_name=canfd1
dev_1_baudrate=7    # 0~3: support FW version >= 03.00
                    # 0: 5   KBPS, 1: 10  KBPS,  2: 20  KBPS, 3: 50  KBPS,
                    # 4: 100 KBPS, 5: 125 KBPS,  6: 250 KBPS, 7: 500 KBPS,
                    # 8: 800 KBPS, 9: 1 MBPS,    A: 400 KBPS, B: CUSTOM_OPT_0, C: CUSTOM_OPT_1
dev_1_datarate=4    # 2: 2 MBPS, 4: 4 MBPS,  5: 5 MBPS
dev_1_socket_1_FDF=1 # 0: DISABLE CAN1 FD mode (aka CAN2.0B only), 1:ENABLE CAN1 FD mode (downward compatibility CAN2.0B)
dev_1_socket_2_FDF=1 # 0: DISABLE CAN2 FD mode (aka CAN2.0B only), 1:ENABLE CAN2 FD mode (downward compatibility CAN2.0B)
dev_1_error_type=0 # 0: EMUC_DIS_ALL, 1: EMUC_EE_ERR, 2: EMUC_BUS_ERR, 3: EMUC_EN_ALL
### /dev/ttyACM1
dev_2=ttyACM1
dev_2_socket_1_name=canfd2
dev_2_socket_2_name=canfd3
dev_2_baudrate=7    # 0~3: support FW version >= 03.00
                    # 0: 5   KBPS, 1: 10  KBPS,  2: 20  KBPS, 3: 50  KBPS,
                    # 4: 100 KBPS, 5: 125 KBPS,  6: 250 KBPS, 7: 500 KBPS,
                    # 8: 800 KBPS, 9: 1 MBPS,    A: 400 KBPS, B: CUSTOM_OPT_0, C: CUSTOM_OPT_1
dev_2_datarate=4    # 2: 2 MBPS, 4: 4 MBPS,  5: 5 MBPS
dev_2_socket_1_FDF=1 # 0: DISABLE CAN1 FD mode (aka CAN2.0B only), 1:ENABLE CAN1 FD mode (downward compatibility CAN2.0B)
dev_2_socket_2_FDF=1 # 0: DISABLE CAN2 FD mode (aka CAN2.0B only), 1:ENABLE CAN2 FD mode (downward compatibility CAN2.0B)
dev_2_error_type=0 # 0: EMUC_DIS_ALL, 1: EMUC_EE_ERR, 2: EMUC_BUS_ERR, 3: EMUC_EN_ALL
```

end.sh

```
sudo pkill -2 emucd_64
sleep 0.2
sudo rmmod emuc2socketcan
#rm /lib/modules/$(uname -r)/kernel/drivers/net/can/emuc2socketcan.ko
```

You can start/end SocketCAN interface simply by using the scripts.

-$ chmod +x start.sh

-$ ./start.sh

You can see the CAN interface name by “ifconfig” command.

```
root@inno-2034-dev:/home/jeff/Documents/CANFD/SocketCAN# ifconfig
canfd0: flags=193<UP,RUNNING,NOARP>  mtu 72
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

canfd1: flags=193<UP,RUNNING,NOARP>  mtu 72
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device base 0x101
```

If use quad ports version, there will be the port from canfd0 to canfd3.

3.3.3. CAN-utils

After SocketCAN setup is finished, you can use open source project “can-utils” to test by “cansend” and “candump”.

(https://github.com/linux-can/can-utils).

- Install CAN-utils

  - $ apt-get install can-utils

- use can0 to send and can1 to receive. (e.g. CAN2.0B)

```
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#R1
yichen@yichen-MS-7971:~$ cansend can0 111#R2
yichen@yichen-MS-7971:~$ cansend can0 111#R3
yichen@yichen-MS-7971:~$
```

```
yichen@yichen-MS-7971:~$ candump can1
  can1  111   [8]  11 22 33 44 55 66 77 88
  can1  111   [8]  11 22 33 44 55 66 77 88
  can1  111   [8]  11 22 33 44 55 66 77 88
  can1  111   [1]  remote request
  can1  111   [2]  remote request
  can1  111   [3]  remote request
```

- use canfd0 to send and canfd1 to receive.(e.g. CAN FD)

```
jeff@inno-2034-dev:~$ cansend canfd0 123##788FFEEDDCCBBAA
jeff@inno-2034-dev:~$ cansend canfd0 123##988FFEEDDCCBBAA0099887766
jeff@inno-2034-dev:~$ cansend canfd0 123##A88FFEEDDCCBBAA009988776655443322
jeff@inno-2034-dev:~$ cansend canfd0 123##B88FFEEDDCCBBAA00998877665544332211001122
jeff@inno-2034-dev:~$ cansend canfd0 123##C88FFEEDDCCBBAA0099887766554433221100112211223344
jeff@inno-2034-dev:~$
```

```
jeff@inno-2034-dev:~/Documents/CANFD/SocketCAN$ candump canfd1 -tz
 (000.000000)  canfd1  123  [07]  88 FF EE DD CC BB AA
 (009.351488)  canfd1  123  [12]  88 FF EE DD CC BB AA 00 99 88 77 66
 (023.992083)  canfd1  123  [16]  88 FF EE DD CC BB AA 00 99 88 77 66 55 44 33 22
 (042.222035)  canfd1  123  [20]  88 FF EE DD CC BB AA 00 99 88 77 66 55 44 33 22 11 00 11 22
 (047.732989)  canfd1  123  [24]  88 FF EE DD CC BB AA 00 99 88 77 66 55 44 33 22 11 00 11 22 11 22 33 44
```

3.3.4. Boot Up Script

We provide Linux boot up script to initial SocketCAN interface automatically after system boot up.

run_emucd

Please modify the baud rate and tty port setting depend on the environment needs.

```
### parameter
socket_name_1=canfd0
socket_name_2=canfd1
dev_name=ttyACM0
baudrate=7 # 0~3: support FW version >= 03.00
           # 0: 5   KBPS, 1: 10  KBPS,  2: 20  KBPS, 3: 50  KBPS,
           # 4: 100 KBPS, 5: 125 KBPS,  6: 250 KBPS, 7: 500 KBPS,
           # 8: 800 KBPS, 9: 1 MBPS,    A: 400 KBPS
datarate=4 # 2: 2 MBPS, 4: 4 MBPS,  5: 5 MBPS, 6: 6 MBPS,
           # 8: 8 MBPS, 10: 10 MBPS
FDF_1=1 # 0: DISABLE CAN1 FD mode (aka CAN2.0B only), 1:ENABLE CAN1 FD mode (downward compatibility CAN2.0B)
FDF_2=1 # 0: DISABLE CAN2 FD mode (aka CAN2.0B only), 1:ENABLE CAN2 FD mode (downward compatibility CAN2.0B)
error_type=0 # 0: EMUC_DIS_ALL, 1: EMUC_EE_ERR, 2: EMUC_BUS_ERR, 3: EMUC_EN_ALL
### bit timing configuration
bit_timing_en=0 # 0: DISABLE bit timing configuration, 1: ENABLE bit timing configuration
# if you want to use bit timing config, please enable bit_timing_en and set the below baudrate
```

Run the following command in the “release” folder to add/remove boot up script.

- $ chmod +x add_2_boot.sh
- $ ./add_2_boot.sh

```
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EMUC_B202/Linux/SocketCAN/bootexec$ ./add_2_boot.sh
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EMUC_B202/Linux/SocketCAN/bootexec$
```

- $ chmod +x remove_boot.sh
- $ ./ remove_boot.sh

```
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EMUC_B202/Linux/SocketCAN/bootexec$ ./remove_boot.sh
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EMUC_B202/Linux/SocketCAN/bootexec$
```

3.3.5. CAN Error Frame

CAN error frame can be dumped by adding the parameter “-e” when running the emucd_32 or emucd_64 utility.

```
emucd_64 -s7 -e3 /dev/ttyACM0
```

It can be simply set the error type by editing “start.sh”.
“run_emucd” of boot up script has this parameter as well.

```
### parameter
socket_name_1=canfd0
socket_name_2=canfd1
dev_name=ttyACM0
baudrate=7 # 0~3: support FW version >= 03.00
           # 0: 5   KBPS, 1: 10  KBPS,  2: 20  KBPS, 3: 50  KBPS,
           # 4: 100 KBPS, 5: 125 KBPS,  6: 250 KBPS, 7: 500 KBPS,
           # 8: 800 KBPS, 9: 1 MBPS,   A: 400 KBPS
datarate=4 # 2: 2 MBPS, 4: 4 MBPS,  5: 5 MBPS, 6: 6 MBPS,
           # 8: 8 MBPS, 10: 10 MBPS
FDF_1=1 # 0: DISABLE CAN1 FD mode (aka CAN2.0B only), 1:ENABLE CAN1 FD mode (downward compatibility CAN2.0B)
FDF_2=1 # 0: DISABLE CAN2 FD mode (aka CAN2.0B only), 1:ENABLE CAN2 FD mode (downward compatibility CAN2.0B)
error_type=0 # 0: EMUC_DIS_ALL, 1: EMUC_EE_ERR, 2: EMUC_BUS_ERR, 3: EMUC_EN_ALL
### bit timing configuration
bit_timing_en=0 # 0: DISABLE bit timing configuration, 1: ENABLE bit timing configuration
```

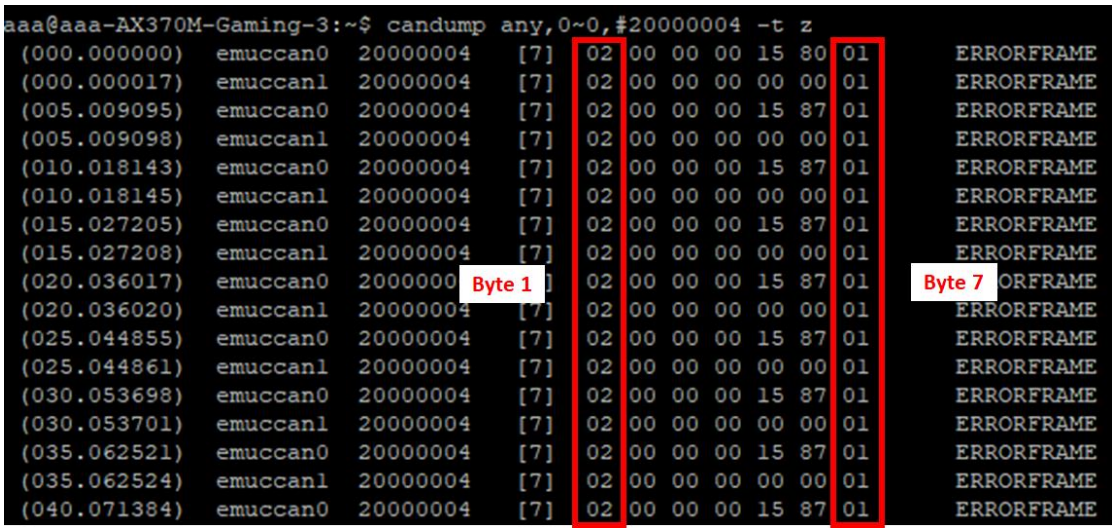
0: EMUC_DIS_ALL: disable all error frame.

1: EMUC_EE_ERR: enable EEPROM error only.

2: EMUC_BUS_ERR: enable CAN bus error only.

3: EMUC_EM_ALL: enable both EERPOM and CAN bus error.

CAN error frame can be dumped through the following command of CAN-utils.



Byte1: Error Type, 0x01=EEPROM Error, 0x02=Bus Error

Byte2~Byte7: Bus Error Register, please refer to *3.2.Register mapping table of CAN error status.*

4. Appendix

4.1. Register mapping table of CAN error status

- ECR (Error Counter Register)

bit 23-16 CEL: CAN Error Logging

bit 15 RP: Receive Error Passive (0 = REC < 128, 1 = REC >= 128)

bit 14-8 REC: Receive Error Counter (0~127)

bit 7-0 TEC: Transmit Error Counter (0~255)

- PSR (Protocol Status Register)

Bit 22-16 TDCV: Transmitter Delay Compensation Value

Bit 15 Reserved

Bit 14 PXE: Protocol Exception Event

Bit 13 RFDF: Received a CAN FD Message

Bit 12 RBRS: BRS flag of last received CAN FD Message

Bit 11 RESI: ESI flag of last received CAN FD Message

Bit 10-8 DLEC: Data Phase Last Error Code

Bit 7 BO: Bus_Off Status

Bit 6 EW: Warning Status

Bit 5 EP: Error Passive

Bit 4-3 ACT: Activity

Bit 2-0 LEC: Last Error Code

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

+31-40-3045-400

Japan

jpsales@innodisk.com

+81-3-6667-0161

China

sales_cn@innodisk.com

+86-755-21673689

www.innodisk.com

December 23, 2024