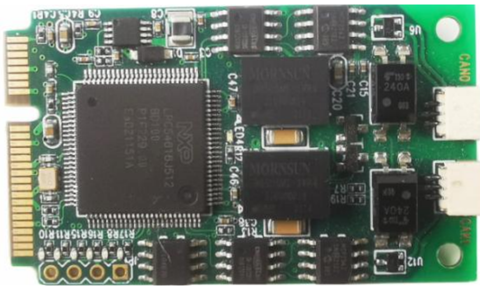
High Speed USB2.0 To CAN FD Converter
Data Field Up to 12M Max

USB To CANFD Series Products
User Mannual

		
SavvyCAN-FD-X2 USB To Dual Channel CANFD	SavvyCAN-FD-C USB To CANFD Cable	SavvyCAN-FD-mPCI USB To Dual Channel CANFD with mini pcie interface (usb2.0 bus)

Date	Revision	Description
2023/01/01	V1.0	First Released

Document Catalogue

Document Catalogue 2
1Introduction 3
    1.1 Product Series 3
    1.2 Product Features 3
2Hardware Description 3
    2.1 SavvyCAN-FD-X2 3
        Specification 4
        Pins Out Description 5
        LED Indication 6
    2.2 SAVVYCAN-FD-C 6
        Specification 7
        Pins Out Description 8
        LED Indication 9
    2.3 SavvyCAN-FD-mPCI 9
        Specification 10
        Pins Out Description 11
        LED Indication 12
        Design Reference 13
3 Connection 15
    3.1 Enable/Disable Build In 120Ω Term Resistor 15
    3.2 SavvyCAN-FD-X2/SAVVYCAN-FD-C Connection 16
    3.3 SavvyCAN-FD-mPCI Connection Figure 17
4 Software Description 17
    4.3 CAN-UTILS/C/Python For Linux 17
        4.3.1 Linux Support List 17
        4.3.2 Hardware Connection 19
        4.3.3 CAN-UTILS DEMO 20
        4.3.4 C Demo 22
        4.3.5 Python3 Demo 22
        4.3.6 Software Description 23

1Introduction

1.1 Product Series

Product Name	CAN Version	Bitrate	Isolation	Max Rate	Interface	Time Stamp
SavvyCAN-FD-X2	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	USB 2*DSUB 9PIN	1us
SavvyCAN-FD-mPCI	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	MINIPCIE (USB Bus)	1us
SavvyCAN-FD-C	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	1.5M USB Cable 1*D-Sub 9PIN	1us

The new CAN FD standard (CAN with Flexible Data rate) is primarily characterized by higher bandwidth for data transfer.

The maximum of 64 data bytes per CAN FD frame (instead of 8 so far) can be transmitted with bit rates up to 12 Mbit/s.

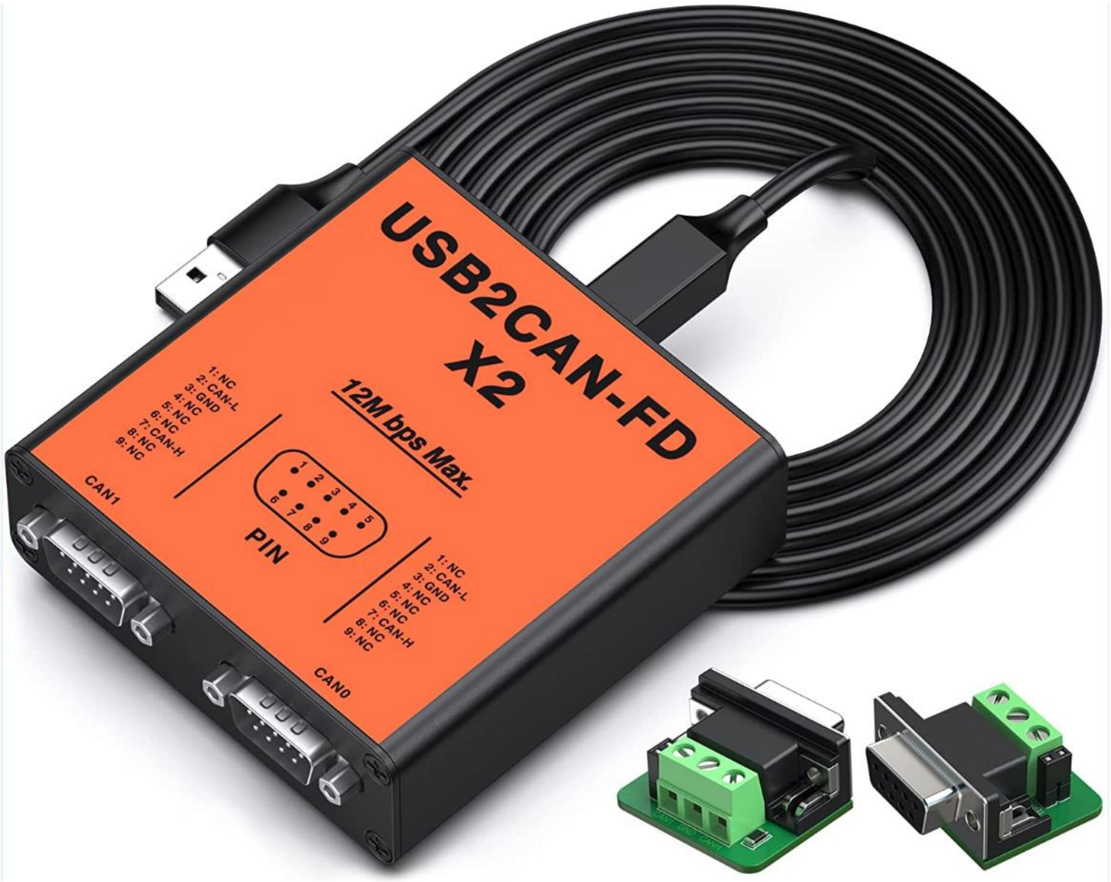
1.2 Product Features

1 True High-speed USB 2.0 Compatible with CAN specifications 2.0A/2.0B/FD;
2 CAN FD support for ISO and Non-ISO standard support software switch;
3 CAN FD bit rates data field (64 bytes max.) from 25 kbit/s up to 12Mbit/s;
4 Class CAN bit rates data field from 25 kbit/s up to 1 Mbit/s;
5 Time stamp Resolution Up to 1 μs;
6 Each CAN FD Signal &Power Separately Isolated Up to 2500 Volts against USB;
7 Bus load measurements including error frames and overload frames on physical bus;
8 Induced error generation for incoming and outgoing CAN messages;

2Hardware Description

2.1 SavvyCAN-FD-X2

The SavvyCAN-FD-X2 is a plug and play high speed USB2.0 to CANFD adapter enables the connection of dual channel CANFD networks to a computer via USB. Each CAN FD channel is separately isolated against USB with a maximum of 2500V.


USB2CAN-FD
X2
12M bps Max.
1: NC
2: CAN-L
3: GND
4: NC
5: NC
6: NC
7: CAN-H
8: NC
9: NC
CAN1
PIN
1 2 3 4 5
6 7 8 9
1: NC
2: CAN-L
3: GND
4: NC
5: NC
6: NC
7: CAN-H
8: NC
9: NC
CAN0

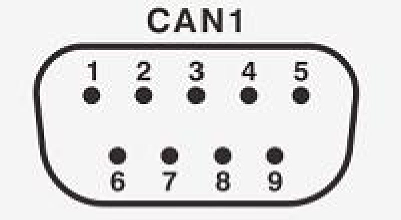
Specification

Connector	
CANFD	Dual Channel 9PIN D-SUB Connectors
USB	USB plug type A（Computer） USB plug type B（Devcie）
CAN Features	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s
USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)

Galvanic isolation	Signal &Power Separately Isolated by 2500 Volts against USB IEC 61000-4-2 Contact IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 µs
Built In 120Ω Termination Resistor	Enable/Disable Through Software
Others	
Temperature	-40°~ 85°
PCBA Size (L * W * H)	84x80x28 mm
Weight	190g

Pins Out Description

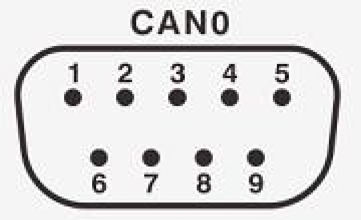
CAN1



PIN OUT Description

1	NC
2	CANL bus line (dominant low)
3	CAN_GND
4	NC
5	NC
6	NC
7	CANH bus line (dominant high)
8	NC
9	NC

CAN0（Normal）

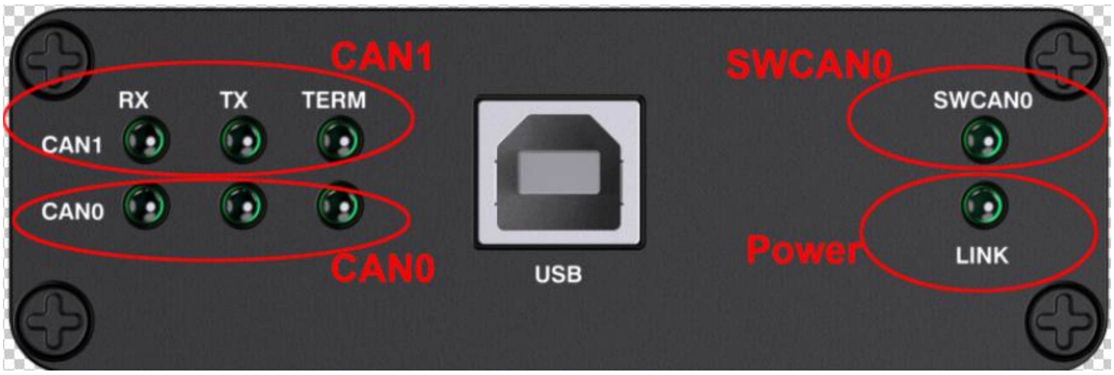


PIN OUT Description

1	NC
2	CANL bus line (dominant low)
3	CAN_GND
4	NC
5	NC
6	NC
7	CANH bus line (dominant high)
8	NC
9	NC

Note1: Not connect GND do not affect normal communication, if cable with shielding suggest connect to GND;

LED Indication



When plug SavvyCAN-FD-X2 device to Computer All lights are flashing for one second. Then TERM LED And LINK LED turns to be green.

Led Name	Description
Power LED Indicator	Power Up And Driver Installted LINK LED turns to be blinking.
CAN1 LED Indicator	TX LED Blinking, Sending Data; RX LED Blinking, Receiving Data; TERM LED Green,120Ω Activated;
CAN0 LED Indicator	TX LED Blinking, Sending Data; RX LED Blinking, Receiving Data; TERM LED Green,120Ω Activated;

2.2 SAVVYCAN-FD-C

The SAVVYCAN-FD-C is a plug and play high speed USB2.0 to CANFD converter comes with 1.2m high quality Cable enables the connection of one channel CANFD networks to a computer via USB. CANFD is isolated protection against USB 2500V.

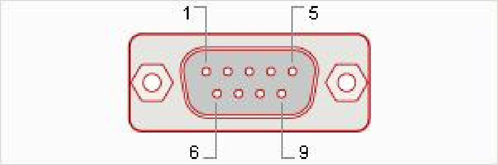
pibiger
PU2CANFD
RXD
TXD
TER
LINK

Specification

Connector	
CANFD	9PIN D-SUB Connectors
USB Cable	1.2M High Quality USB Cable with type A
CAN Features	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s
USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)
Galvanic isolation	Signal &Power Separately Isolated by 2500 Volts against USB

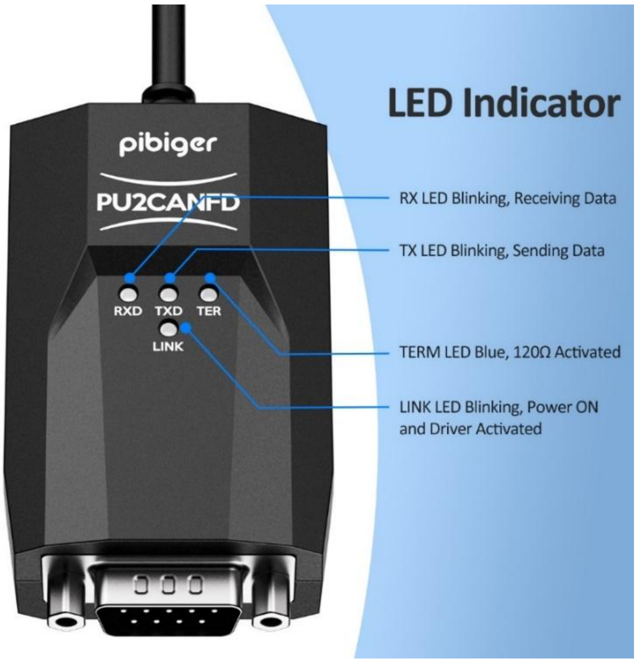
	IEC 61000-4-2 Contact IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 µs
Built In 120Ω Termination Resistor	Disable/Enable Through Software
Others	
Temperature	-40°~ 85°

Pins Out Description



1	NC
2	CAN-L
3	GND
4	NC
5	NC
6	NC
7	CAN-H
8	NC
9	NC

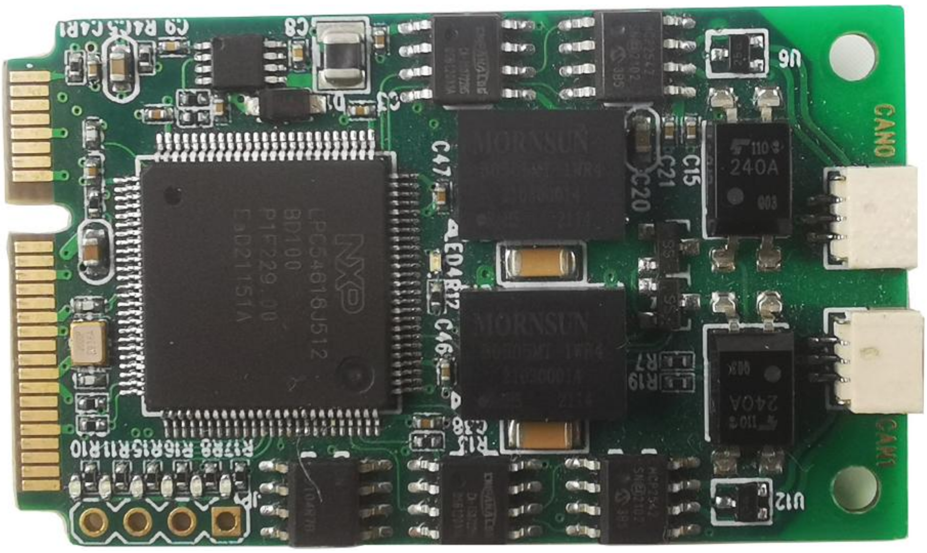
LED Indication


LED Indicator
pibiger
PU2CANFD
RXD
TXD
TER
LINK
RX LED Blinking, Receiving Data
TX LED Blinking, Sending Data
TERM LED Blue, 120Ω Activated
LINK LED Blinking, Power ON and Driver Activated

Led Name	LED Status	Description
RXD	Blinking	Receiving Data
TXD	Blinking	Sending Data
TER	Blue Led On	Build In 120Ω Term Resistor Enable;
LINK	Blinking	Power Up and Driver Installed.
	Blue Led On	Power Up Driver Not Installed.

2.3 SavvyCAN-FD-mPCI

The SavvyCAN-FD-mPCI is a plug and play high speed USB2.0 to CANFD CAN Card. CAN FD each channel is separately isolated against USB with a maximum of 2500V.

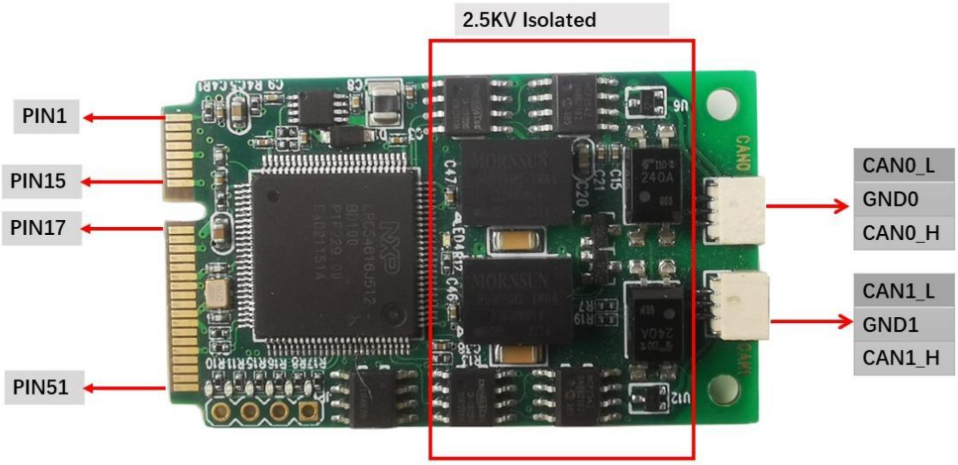


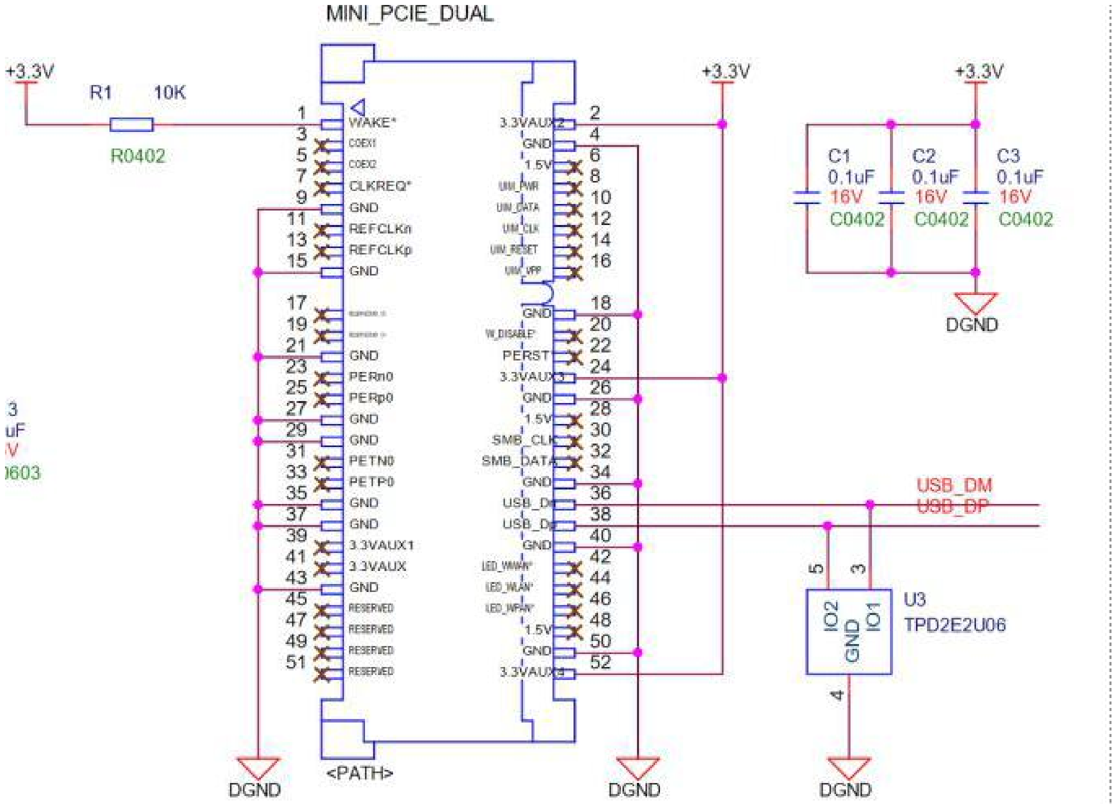
Specification

Connector	
Dual Channel CANFD	2X3PIN 1mm Connector
Mini PCIE USB	USB BUS
CAN Features	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s
USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)
Galvanic isolation	Signal &Power Separately Isolated by 2500 Volts against USB IEC 61000-4-2 Contact IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 µs
Built In 120Ω Termination Resistor	Activated/Deactivated Through Software
Others	
Temperature	-40°~ 85°

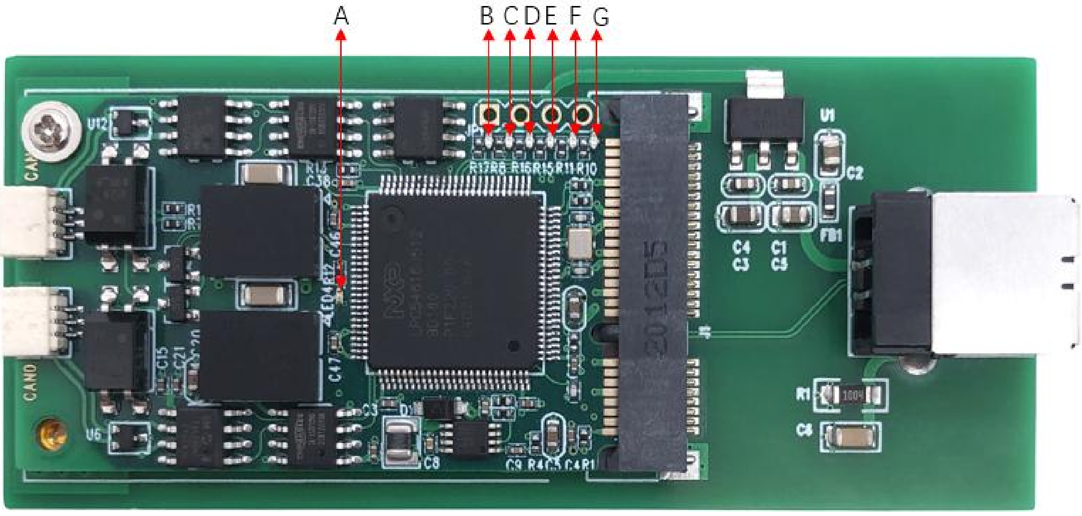
PCBA Size (L * W * H)	84x80x28 mm
Weight	191 g

Pins Out Description


2.5KV Isolated
PIN1
PIN15
PIN17
PIN51
CAN0_L
GND0
CAN0_H
CAN1_L
GND1
CAN1_H


MINI_PCIE_DUAL
+3.3V
R1 10K
R0402
1 WAKE*
3 COEX1
5 COEX2
7 CLKREQ*
9 GND
11 REFCLKn
13 REFCLKp
15 GND
17
19
21 GND
23 PERn0
25 PERp0
27 GND
29 GND
31 PETN0
33 PETP0
35 GND
37 GND
39 3.3VAUX1
41 3.3VAUX
43 GND
45 RESERVED
47 RESERVED
49 RESERVED
51 RESERVED
2 3.3VAUX2
4 GND
6 1.5V
8 UIM_PWR
10 UIM_DATA
12 UIM_CLK
14 UIM_RESET
16 UIM_VPP
18 GND
20 W_DISABLE*
22 PERST*
24 3.3VAUX3
26 GND
28 1.5V
30 SMB_CLK
32 SMB_DATA
34 GND
36 USB_D-
38 USB_D+
40 GND
42 LED_WWAN*
44 LED_WLAN*
46 LED_WPAN*
48 1.5V
50 GND
52 3.3VAUX4
<PATH>
DGND
+3.3V
+3.3V
C1 0.1uF 16V C0402
C2 0.1uF 16V C0402
C3 0.1uF 16V C0402
DGND
USB_DM
USB_DP
U3
TPD2E2U06
IO2
GND
IO1
5
3
4
DGND
DGND
3
uF
V
)603

LED Indication

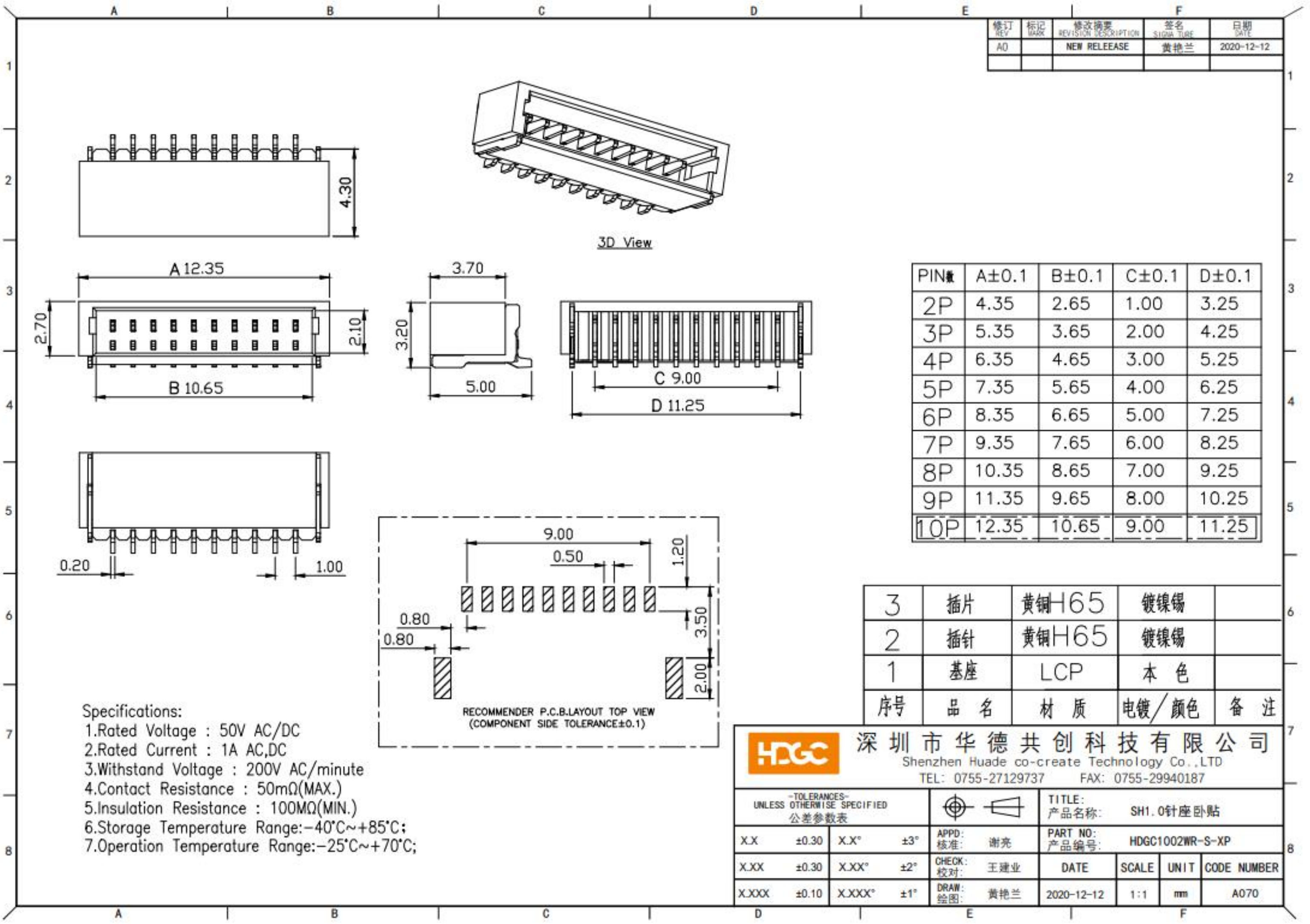


	Description	Indication
A	Power&Driver LED	Blinking, Driver install and Power Up
B	CAN0 120Ω Term	On, CAN0 Term Enable; Off, CAN0 Term Disable
C	CAN1 120Ω Term	On, CAN1 Term Enable; Off, CAN1 Term Disable
D	CAN0 Receive	Blinking， CAN0 Receiving Data;
E	CAN0 Send	Blinking， CAN0 Sending Data;
F	CAN1 Receive	Blinking， CAN1 Receiving Data;
G	CAN1 Send	Blinking， CAN0 Receiving Data;

Design Reference

3PIN Connector Board End

Vendor：HDGC1002WR-S-3P


A
B
C
D
E
F
1
2
3
4
5
6
7
8
4.30
3D View
A 12.35
2.70
2.10
B 10.65
3.70
3.20
5.00
C 9.00
D 11.25
0.20
1.00
9.00
0.50
1.20
0.80
0.80
3.50
2.00
RECOMMENDER P.C.B.LAYOUT TOP VIEW
(COMPONENT SIDE TOLERANCE±0.1)

修订 REV	标记 MARK	修改摘要 REVISION DESCRIPTION	签名 SIGNA TURE	日期 DATE
A0		NEW RELEEASE	黄艳兰	2020-12-12

PIN数	A±0.1	B±0.1	C±0.1	D±0.1
2P	4.35	2.65	1.00	3.25
3P	5.35	3.65	2.00	4.25
4P	6.35	4.65	3.00	5.25
5P	7.35	5.65	4.00	6.25
6P	8.35	6.65	5.00	7.25
7P	9.35	7.65	6.00	8.25
8P	10.35	8.65	7.00	9.25
9P	11.35	9.65	8.00	10.25
10P	12.35	10.65	9.00	11.25

序号	品 名	材 质	电镀/颜色	备 注
3	插片	黄铜H65	镀镍锡	
2	插针	黄铜H65	镀镍锡	
1	基座	LCP	本 色	

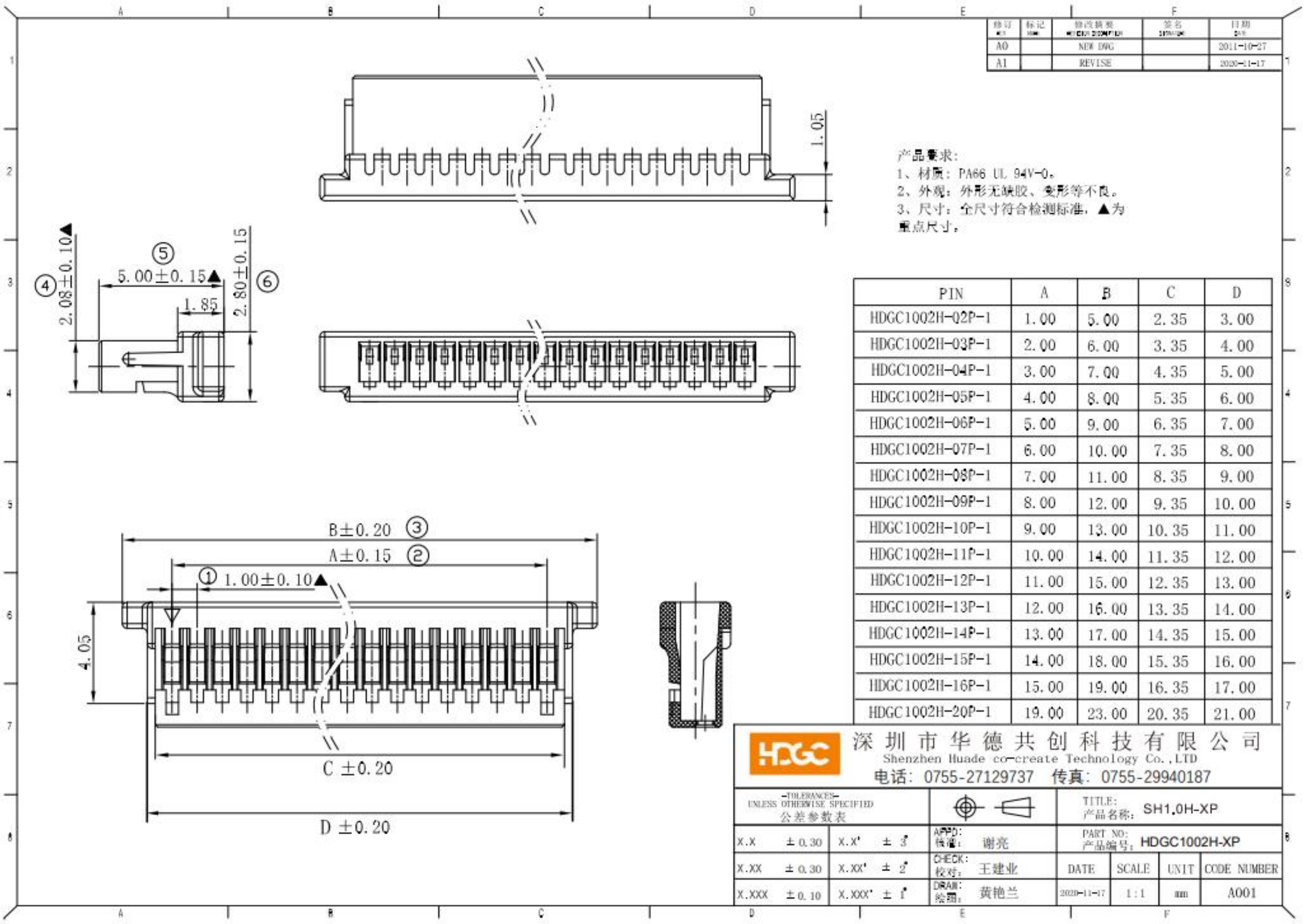
Specifications:
1.Rated Voltage : 50V AC/DC
2.Rated Current : 1A AC,DC
3.Withstand Voltage : 200V AC/minute
4.Contact Resistance : 50mΩ(MAX.)
5.Insulation Resistance : 100MΩ(MIN.)
6.Storage Temperature Range:-40℃~+85℃;
7.Operation Temperature Range:-25℃~+70℃;

深圳市华德共创科技有限公司
Shenzhen Huade co-create Technology Co.,LTD
TEL: 0755-27129737 FAX: 0755-29940187

-TOLERANCES- UNLESS OTHERWISE SPECIFIED 公差参数表					TITLE: 产品名称: SH1.0针座卧贴			
X.X	±0.30	X.X°	±3°	APPD: 核准: 谢亮	PART NO: 产品编号: HDGC1002WR-S-XP			
X.XX	±0.30	X.XX°	±2°	CHECK: 校对: 王建业	DATE	SCALE	UNIT	CODE NUMBER
X.XXX	±0.10	X.XXX°	±1°	DRAW: 绘图: 黄艳兰	2020-12-12	1:1	mm	A070

Design End

Vendor：HDGC1002H-5P



修订	标记	修改摘要	签名	日期
A0		NEW DWG		2011-10-27
A1		REVISE		2020-11-17

产品要求:
1、材质: PA66 UL 94V-0。
2、外观: 外形无缺胶、变形等不良。
3、尺寸: 全尺寸符合检测标准, ▲为重点尺寸。

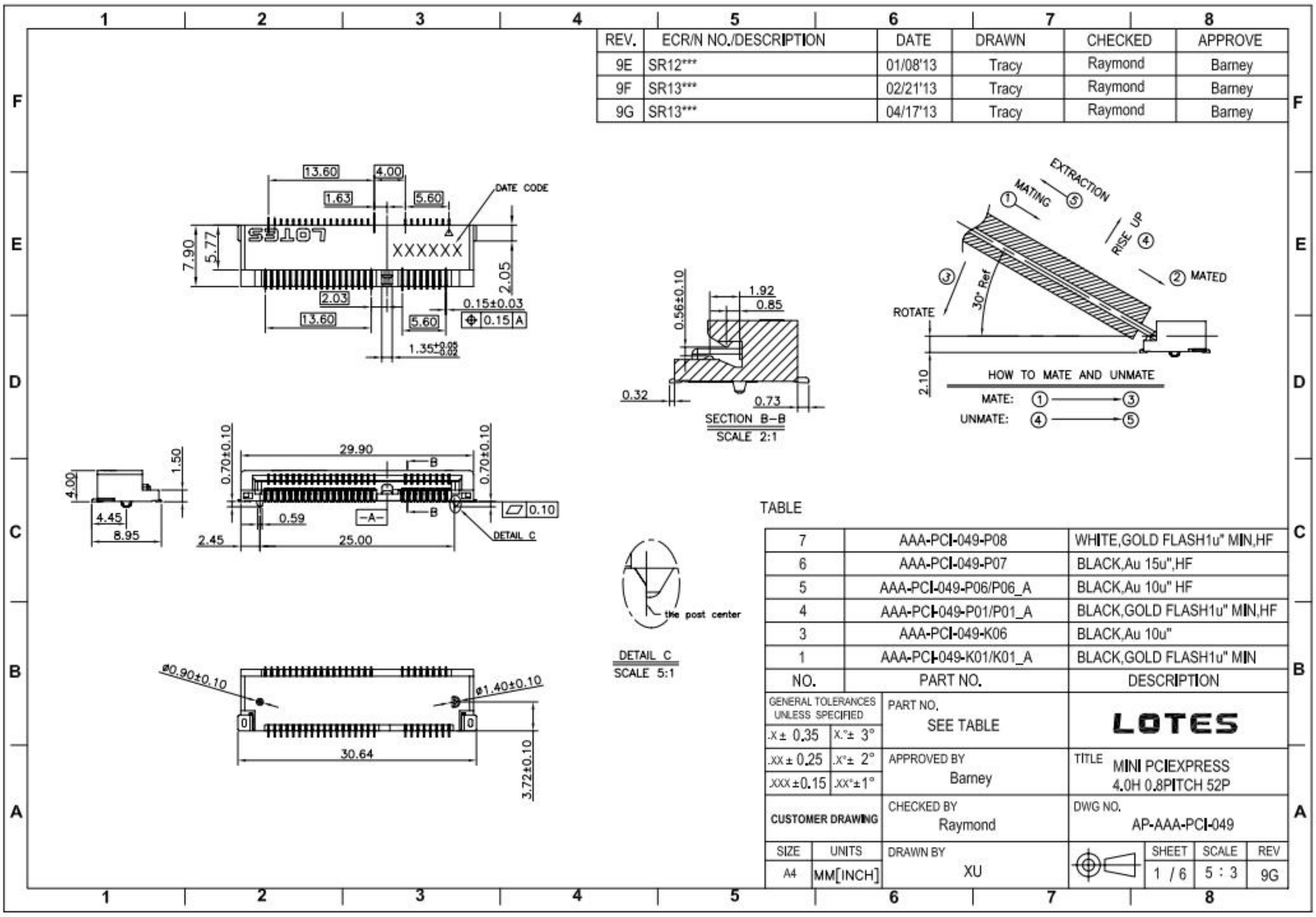
PIN	A	B	C	D
HDGC1002H-02P-1	1.00	5.00	2.35	3.00
HDGC1002H-03P-1	2.00	6.00	3.35	4.00
HDGC1002H-04P-1	3.00	7.00	4.35	5.00
HDGC1002H-05P-1	4.00	8.00	5.35	6.00
HDGC1002H-06P-1	5.00	9.00	6.35	7.00
HDGC1002H-07P-1	6.00	10.00	7.35	8.00
HDGC1002H-08P-1	7.00	11.00	8.35	9.00
HDGC1002H-09P-1	8.00	12.00	9.35	10.00
HDGC1002H-10P-1	9.00	13.00	10.35	11.00
HDGC1002H-11P-1	10.00	14.00	11.35	12.00
HDGC1002H-12P-1	11.00	15.00	12.35	13.00
HDGC1002H-13P-1	12.00	16.00	13.35	14.00
HDGC1002H-14P-1	13.00	17.00	14.35	15.00
HDGC1002H-15P-1	14.00	18.00	15.35	16.00
HDGC1002H-16P-1	15.00	19.00	16.35	17.00
HDGC1002H-20P-1	19.00	23.00	20.35	21.00

HDGC
深圳市华德共创科技有限公司
Shenzhen Huade co-create Technology Co.,LTD
电话: 0755-27129737 传真: 0755-29940187

-TOLERANCES- UNLESS OTHERWISE SPECIFIED 公差参数表			TITLE: 产品名称: SH1.0H-XP
X.X ±0.30	X.X° ±3°	APPD: 核准: 谢亮	PART NO: 产品编号: HDGC1002H-XP
X.XX ±0.30	X.XX° ±2°	CHECK: 校对: 王建业	DATE / SCALE / UNIT / CODE NUMBER
X.XXX ±0.10	X.XXX° ±1°	DRAW: 绘图: 黄艳兰	2020-11-17 / 1:1 / mm / A001

MINIPCIE Connector

Vendor：Lotes，AAA-PCI-049-K01


1
2
3
4
5
6
7
8
F
E
D
C
B
A
13.60
4.00
1.63
5.60
DATE CODE
LOTES
XXXXXX
7.90
5.77
2.05
2.03
0.15±0.03
13.60
5.60
0.15 A
1.35+0.05 -0.02
0.56±0.10
1.92
0.85
0.32
0.73
SECTION B–B
SCALE 2:1
EXTRACTION
MATING
①
⑤
RISE UP
④
② MATED
③
ROTATE
30° Ref
2.10
HOW TO MATE AND UNMATE
MATE: ① → ③
UNMATE: ④ → ⑤
1.50
4.00
4.45
8.95
0.70±0.10
29.90
B
0.70±0.10
0.59
-A-
0.10
2.45
25.00
DETAIL C
the post center
DETAIL C
SCALE 5:1
ø0.90±0.10
ø1.40±0.10
30.64
3.72±0.10
TABLE

REV.	ECR/N NO./DESCRIPTION	DATE	DRAWN	CHECKED	APPROVE
9E	SR12***	01/08'13	Tracy	Raymond	Barney
9F	SR13***	02/21'13	Tracy	Raymond	Barney
9G	SR13***	04/17'13	Tracy	Raymond	Barney

NO.	PART NO.	DESCRIPTION
7	AAA-PCI-049-P08	WHITE,GOLD FLASH1u" MIN,HF
6	AAA-PCI-049-P07	BLACK,Au 15u",HF
5	AAA-PCI-049-P06/P06_A	BLACK,Au 10u" HF
4	AAA-PCI-049-P01/P01_A	BLACK,GOLD FLASH1u" MIN,HF
3	AAA-PCI-049-K06	BLACK,Au 10u"
1	AAA-PCI-049-K01/K01_A	BLACK,GOLD FLASH1u" MIN

GENERAL TOLERANCES UNLESS SPECIFIED		PART NO. SEE TABLE	LOTES
.X± 0.35	X.°± 3°		
.XX± 0.25	.X°± 2°	APPROVED BY Barney	TITLE MINI PCIEXPRESS 4.0H 0.8PITCH 52P
.XXX±0.15	.XX°±1°		
CUSTOMER DRAWING		CHECKED BY Raymond	DWG NO. AP-AAA-PCI-049
SIZE A4	UNITS MM[INCH]	DRAWN BY XU	SHEET 1 / 6 · SCALE 5 : 3 · REV 9G

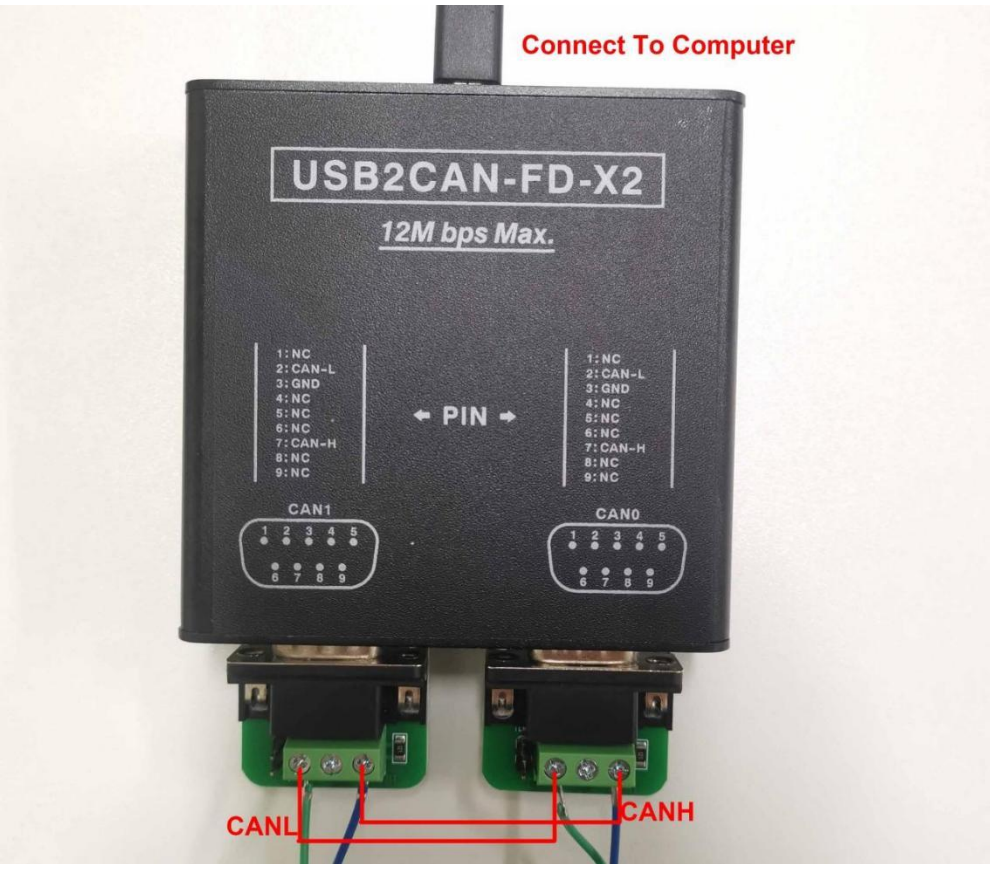
3 Connection

3.1 Enable/Disable Build In 120Ω Term Resistor

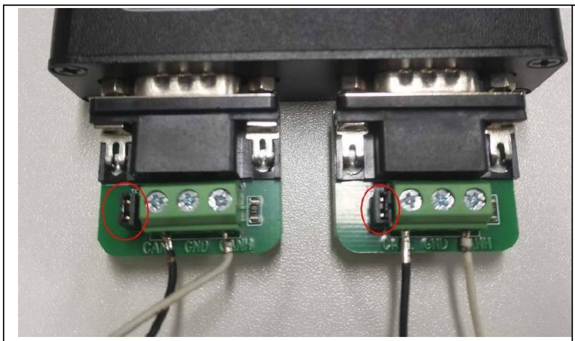
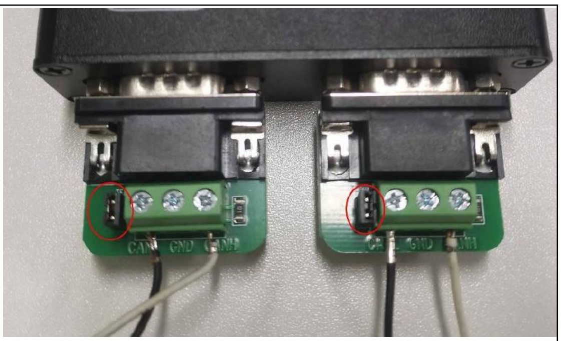
Products **comes enable build in 120Ω term resistor in default**.

If you need external connection of 120Ω termination resistor

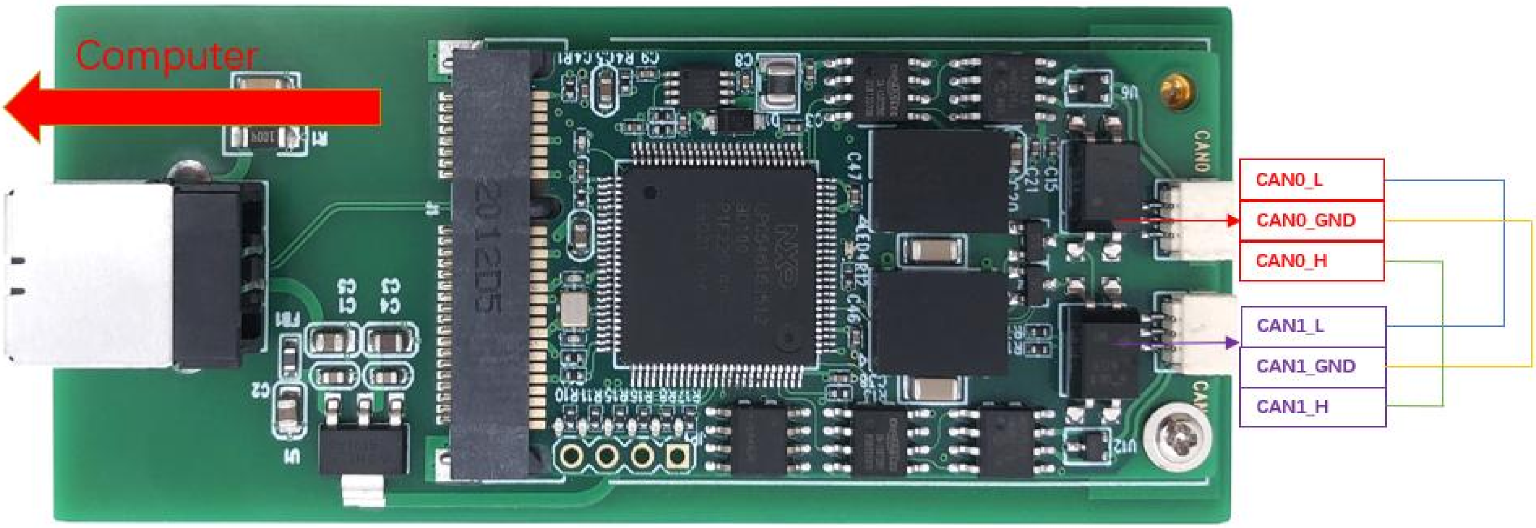
3.2 SavvyCAN-FD-X2/SAVVYCAN-FD-C Connection

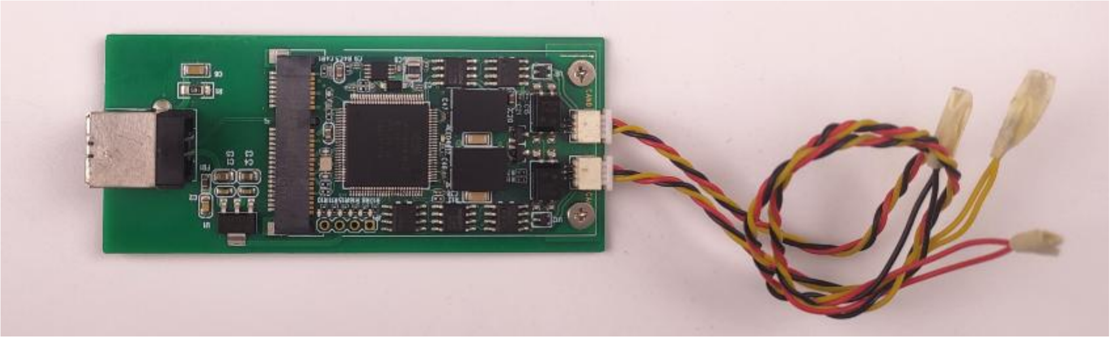

Connect To Computer
USB2CAN-FD-X2
12M bps Max.
1:NC
2:CAN-L
3:GND
4:NC
5:NC
6:NC
7:CAN-H
8:NC
9:NC
← PIN →
1:NC
2:CAN-L
3:GND
4:NC
5:NC
6:NC
7:CAN-H
8:NC
9:NC
CAN1
1 2 3 4 5
6 7 8 9
CAN0
1 2 3 4 5
6 7 8 9
CANL
CANH

Notes:

 Note: Remove jumper When build in 120Ω termination resistor Enable.	 Note: Add jumper When build in 120Ω termination resistor Disable.

3.3 SavvyCAN-FD-mPCI Connection Figure


Computer
CAN0_L
CAN0_GND
CAN0_H
CAN1_L
CAN1_GND
CAN1_H



4 Software Description

4.3 CAN-UTILS/C/Python For Linux

（USE AS SOCKET CAN）

4.3.1 Linux Support List

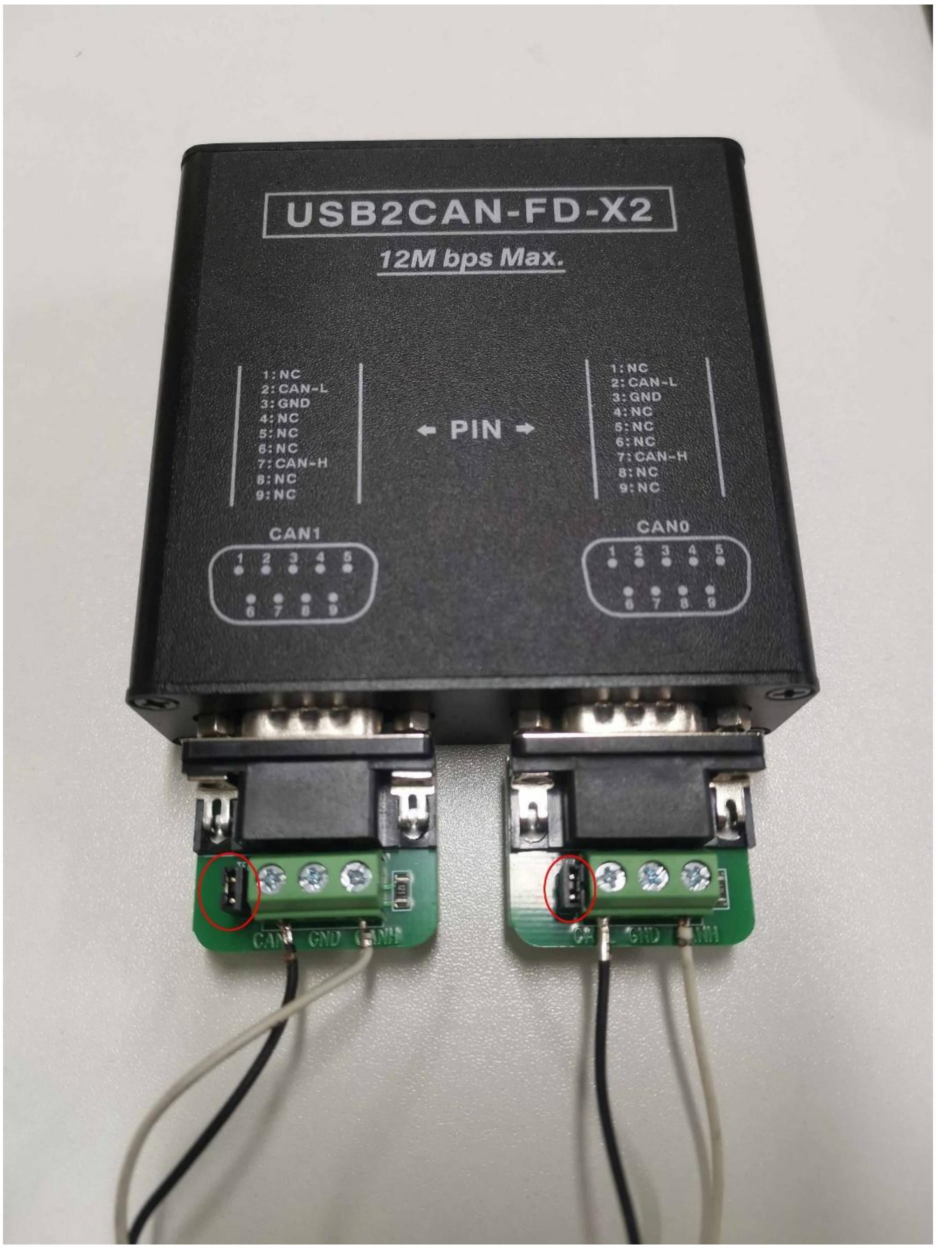
SavvyCAN-FD-X2 device can run properly without any additional driver request on all Linux system as below.

	amd64	**i386**	**arm64**	**armhf**	**ppc64el**
Ubuntu:					
Trusty 14.04 LTS	🟥	🟥	🟥	🟥	
Xenial 16.04 LTS	🟥	🟥	🟥	🟥	🟥
Bionic 18.04 LTS	🟥	🟥	🟥	🟥	🟥
Cosmic 18.10	🟥	🟥	🟥	🟥	
Disco 19.04	🟥	🟥	🟥	🟥	
Eoan 19.10	🟥	🟥	🟥	🟥	
Focal 20.04 LTS	🟥	🟥	🟥	🟥	🟥
Groovy 20.10	🟥	🟥	🟥	🟥	🟥
Hirsute 21.04	🟥	🟥	🟥	🟥	🟥
OpenSUSE Tumbleweed	see Xenial				
Debian:					
Wheezy 7.11	🟥	🟥	🟥	🟥	
Jessie 8.11	🟥	🟥	🟥	🟥	
Stretch 9.9	🟥	🟥	🟥	🟥	
Buster 10	🟥	🟥	🟥	🟥	🟥
Bullseye 11	🟥	🟥	🟥	🟥	🟥

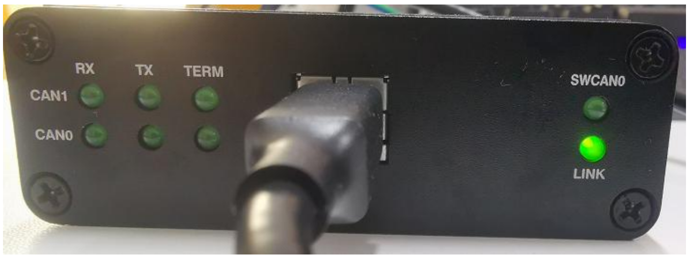
4.3.2 Hardware Connection

Connect device to your Linux computer As below picture and follow chapter 2.2 to activated 120Ω resistor by hardware, use the 2pcs db9 to termination board we provide and **put on jumper in red circle**.

CAN 0 Channel	**Connection**	**CAN 1 Channel**
CAN_L(pin 2)	--	CAN_L(pin 2)
CAN_H(pin 7)	--	CAN_H(pin 7)


USB2CAN-FD-X2
12M bps Max.
1:NC
2:CAN-L
3:GND
4:NC
5:NC
6:NC
7:CAN-H
8:NC
9:NC
← PIN →
1:NC
2:CAN-L
3:GND
4:NC
5:NC
6:NC
7:CAN-H
8:NC
9:NC
CAN1
CAN0
1 2 3 4 5
6 7 8 9
1 2 3 4 5
6 7 8 9
CANL GND CANH
CANL GND CANH

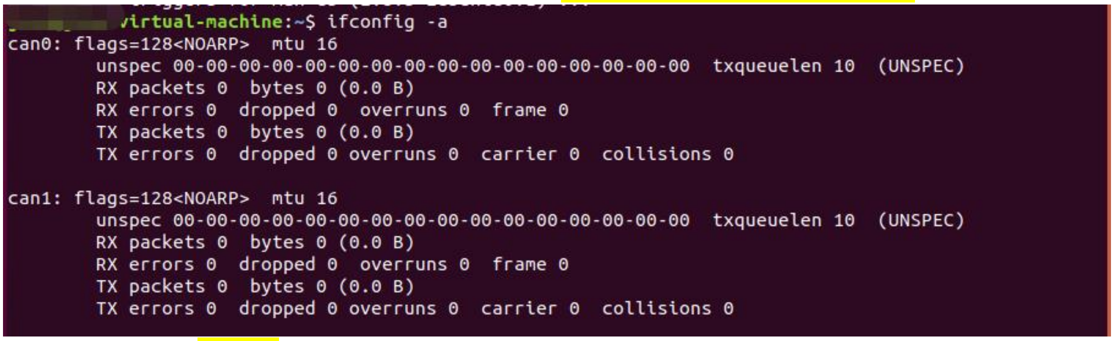
LED Indication should be as below picture:



4.3.3 CAN-UTILS DEMO

Prepare

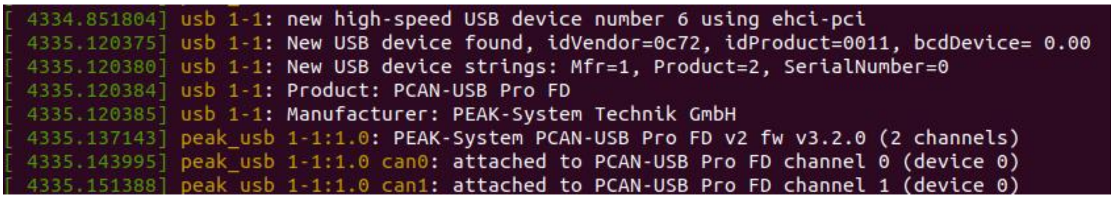
Type command ifconfig -a to check ‘can0’ and ‘can1’device is available in system, if you can not find the command ifconfig, use command sudo apt-get install net-tools



```
[redacted]virtual-machine:~$ ifconfig -a
can0: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (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

can1: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (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
```

Type command dmesg to check more information



```
[ 4334.851804] usb 1-1: new high-speed USB device number 6 using ehci-pci
[ 4335.120375] usb 1-1: New USB device found, idVendor=0c72, idProduct=0011, bcdDevice= 0.00
[ 4335.120380] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 4335.120384] usb 1-1: Product: PCAN-USB Pro FD
[ 4335.120385] usb 1-1: Manufacturer: PEAK-System Technik GmbH
[ 4335.137143] peak_usb 1-1:1.0: PEAK-System PCAN-USB Pro FD v2 fw v3.2.0 (2 channels)
[ 4335.143995] peak_usb 1-1:1.0 can0: attached to PCAN-USB Pro FD channel 0 (device 0)
[ 4335.151388] peak_usb 1-1:1.0 can1: attached to PCAN-USB Pro FD channel 1 (device 0)
```

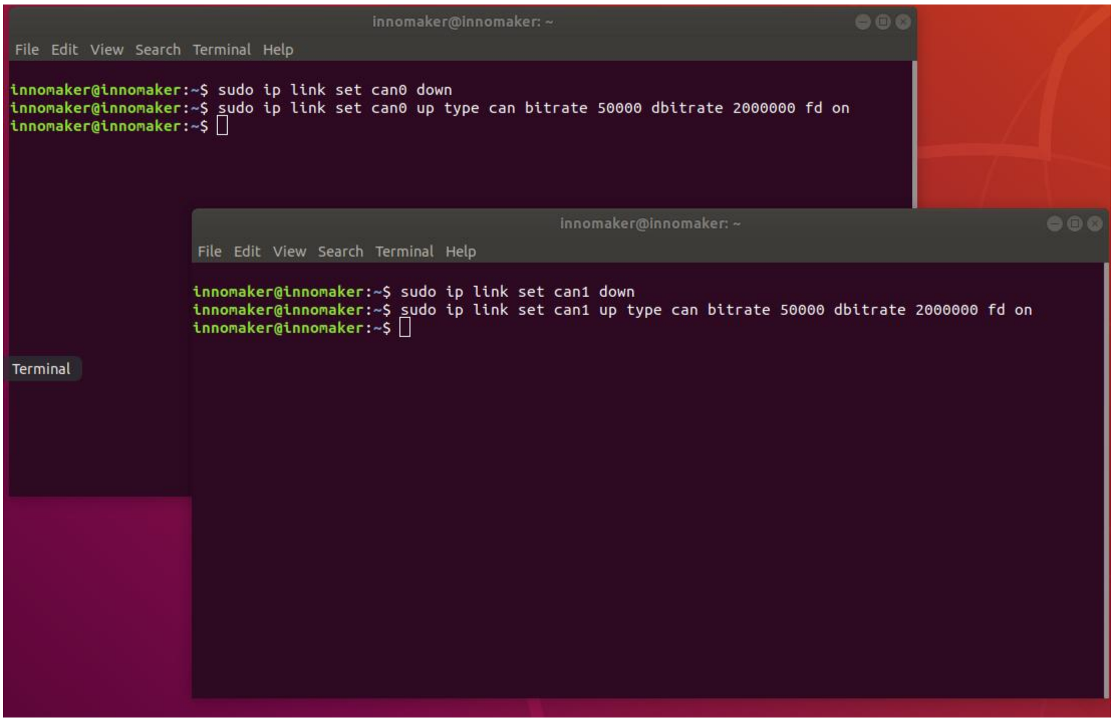
Type command sudo apt-get install can-utils to install can-utils.

Note:
This tool is a very easy way to test SavvyCAN-FD-X2 module communication. There is only a simple use instruction. For more details, please refer to can-utils user manual and source code. https://github.com/linux-can/can-utils/

Send/Receive

Initialize CAN port, Open two termination command for can0 and can1.
sudo ip link set can0 down
sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on

```
sudo ip link set can1 down
sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on
```


innomaker@innomaker: ~
File Edit View Search Terminal Help
innomaker@innomaker: ~
File Edit View Search Terminal Help
Terminal

```
innomaker@innomaker:~$ sudo ip link set can0 down
innomaker@innomaker:~$ sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on
innomaker@innomaker:~$ 
```

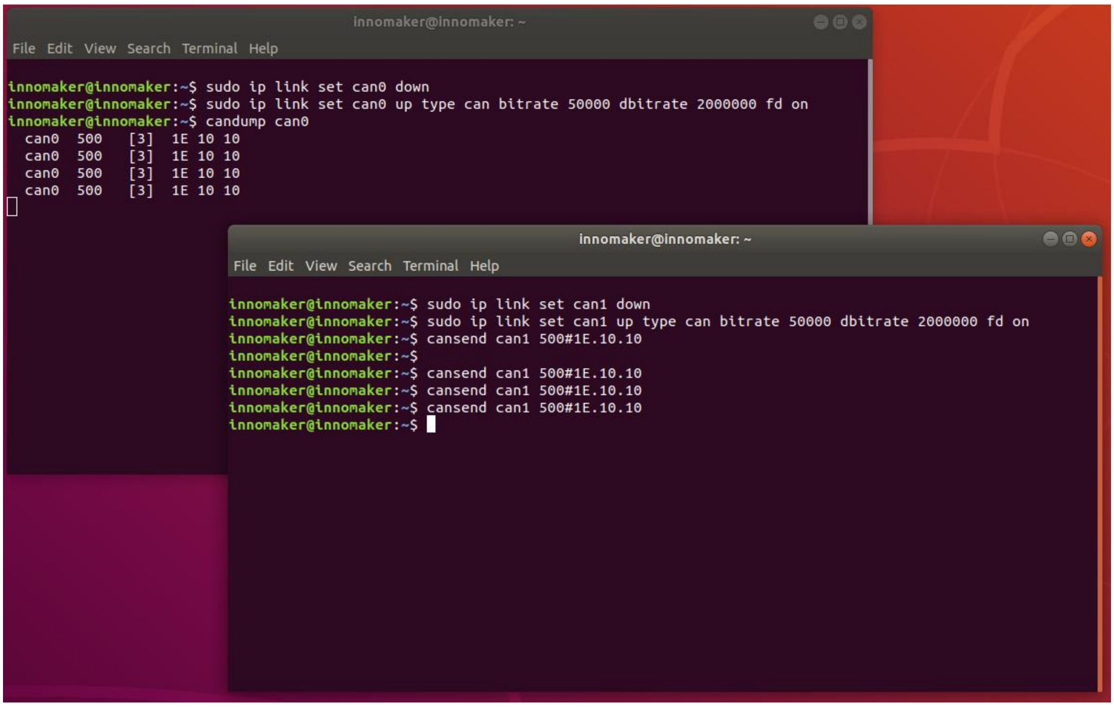
```
innomaker@innomaker:~$ sudo ip link set can1 down
innomaker@innomaker:~$ sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on
innomaker@innomaker:~$ 
```

<1>Set can0 as receiver

```
candump can0
```

<2>Set can1 as sender

```
cansend can1 500#1E.10.10
```


innomaker@innomaker: ~
File Edit View Search Terminal Help
innomaker@innomaker: ~
File Edit View Search Terminal Help

```
innomaker@innomaker:~$ sudo ip link set can0 down
innomaker@innomaker:~$ sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on
innomaker@innomaker:~$ candump can0
  can0  500   [3]  1E 10 10
  can0  500   [3]  1E 10 10
  can0  500   [3]  1E 10 10
  can0  500   [3]  1E 10 10
```

```
innomaker@innomaker:~$ sudo ip link set can1 down
innomaker@innomaker:~$ sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on
innomaker@innomaker:~$ cansend can1 500#1E.10.10
innomaker@innomaker:~$
innomaker@innomaker:~$ cansend can1 500#1E.10.10
innomaker@innomaker:~$ cansend can1 500#1E.10.10
innomaker@innomaker:~$ cansend can1 500#1E.10.10
innomaker@innomaker:~$ 
```

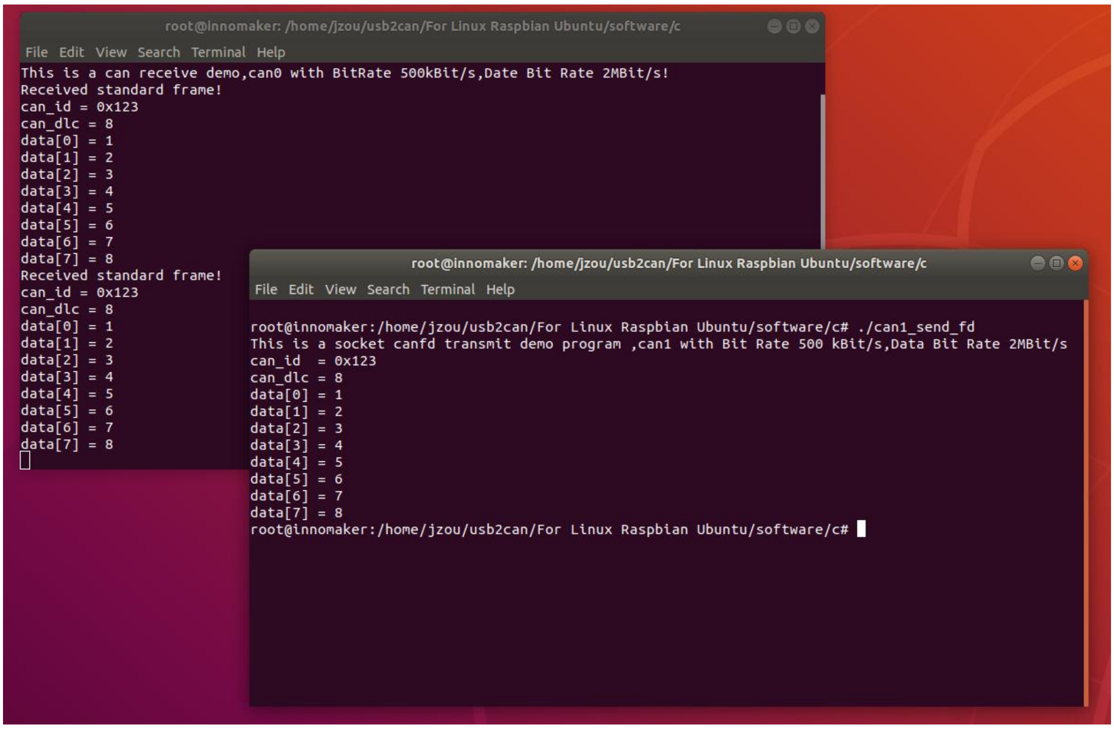
4.3.4 C Demo

<1>Send CAN0 As Receiver,

```
sudo ./can0_receive_fd
```

<2>Set CAN1 As Sender

```
sudo ./can1_send_fd
```


root@innomaker: /home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c
File Edit View Search Terminal Help

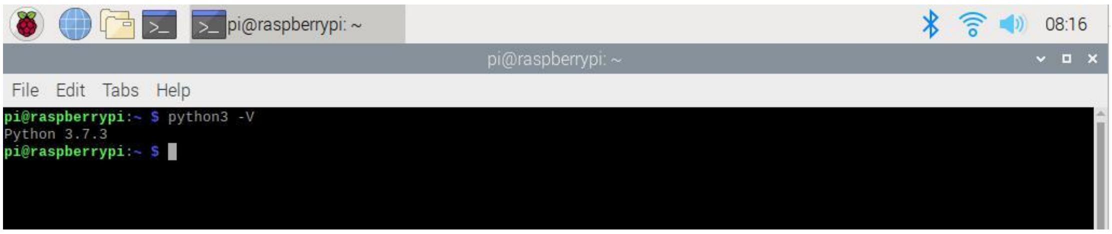
```
This is a can receive demo,can0 with BitRate 500kBit/s,Date Bit Rate 2MBit/s!
Received standard frame!
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
Received standard frame!
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
```

```
root@innomaker:/home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c# ./can1_send_fd
This is a socket canfd transmit demo program ,can1 with Bit Rate 500 kBit/s,Data Bit Rate 2MBit/s
can_id  = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
root@innomaker:/home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c#
```

4.3.5 Python3 Demo

(1) Check the Python version of your Raspbian. Python 3.7.3 default in 2019-09-26-Raspbian.img. Our Demo can run on any Python3 version.

```
python3 -V
```


pi@raspberrypi: ~
08:16
pi@raspberrypi: ~
File Edit Tabs Help

```
pi@raspberrypi:~ $ python3 -V
Python 3.7.3
pi@raspberrypi:~ $
```

(2) If you can’t find the Python3 in system. Install the Python3

```
sudo apt-get install python3-pip
```

(3) Install Python CAN library.

```
sudo pip3 install python-can
```

(4) Set CAN0 as receiver

```
sudo python3 receive.py
```

(5) Set CAN1 as sender

```
sudo python3 send.py
```

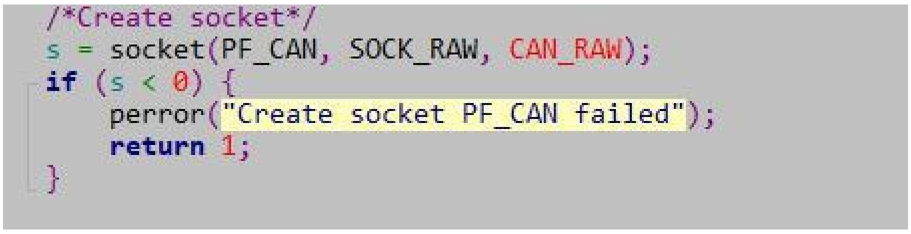
4.3.6 Software Description

Now with previous demo's code to show you how to program socket can in Raspbian with C and Python . The socket can is an implementation of CAN protocols(Controller Area Network) for Linux. CAN is a networking technology which has widespread use in automation, embedded devices, and automotive fields. While there have been other CAN implementations for Linux based on character devices, Socket CAN uses the Berkeley socket API, the Linux network stack and implements the CAN device drivers as network interfaces. The CAN socket API has been designed as similar as possible to the TCP/IP protocols to allow programmers, familiar with network programming, to easily learn how to use CAN sockets. For more Socket CAN detail please refer to below link:
https://www.kernel.org/doc/Documentation/networking/can.txt
https://elinux.org/CAN_Bus

Programming in C

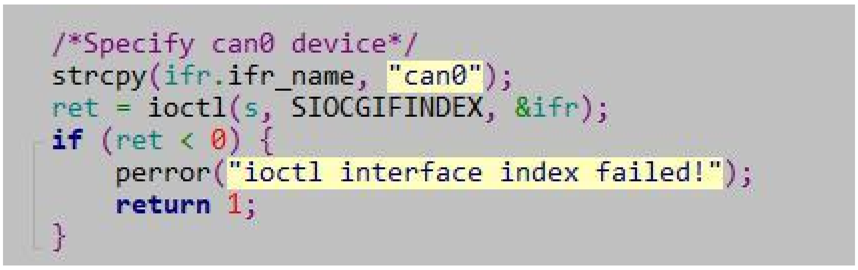
For Sender's codes

(1): Create the socket, If an error occurs then the return result is -1.



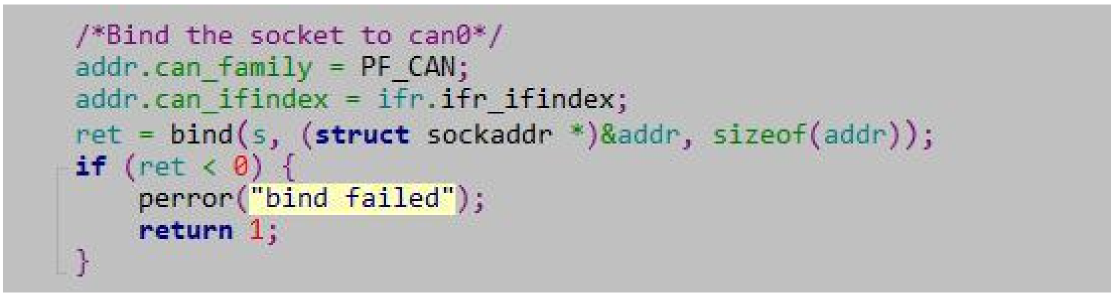
```
/*Create socket*/
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
if (s < 0) {
    perror("Create socket PF_CAN failed");
    return 1;
}
```

(2): Locate the interface to "can0" or other name you wish to use. The name will show when you execute "./ifconfig –a".



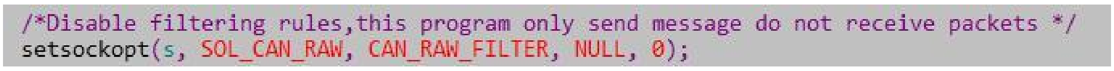
```
/*Specify can0 device*/
strcpy(ifr.ifr_name, "can0");
ret = ioctl(s, SIOCGIFINDEX, &ifr);
if (ret < 0) {
    perror("ioctl interface index failed!");
    return 1;
}
```

(3): Bind the socket to “can0”.



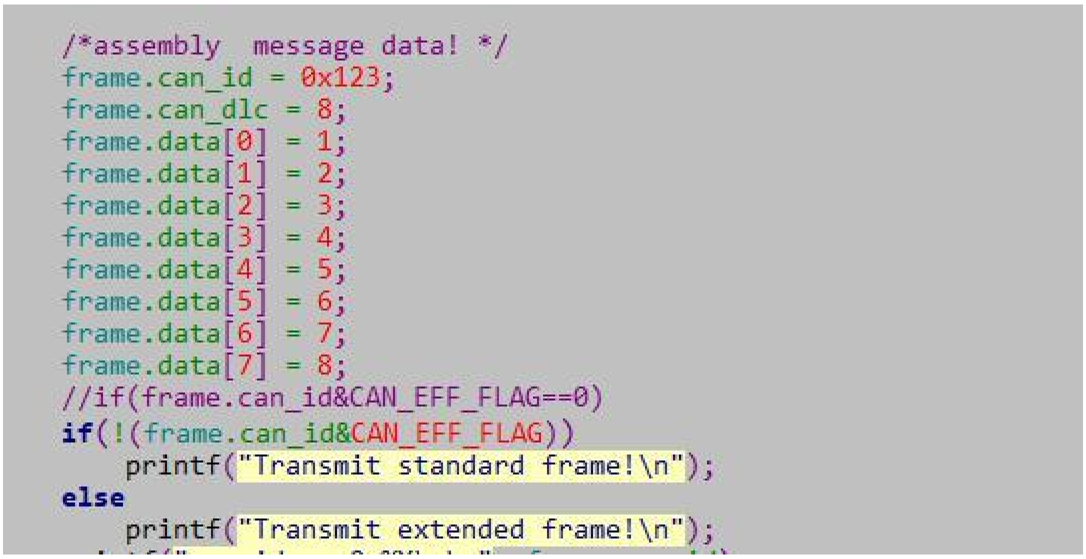
```
/*Bind the socket to can0*/
addr.can_family = PF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));
if (ret < 0) {
    perror("bind failed");
    return 1;
}
```

(4): Disable sender’s filtering rules,this program only send message do not receive packets.



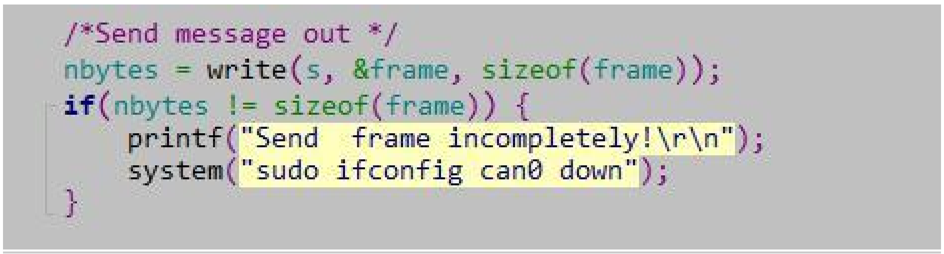
```
/*Disable filtering rules,this program only send message do not receive packets */
setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, NULL, 0);
```

(5): Assembly data to send.



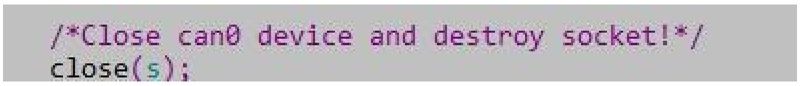
```
/*assembly  message data! */
frame.can_id = 0x123;
frame.can_dlc = 8;
frame.data[0] = 1;
frame.data[1] = 2;
frame.data[2] = 3;
frame.data[3] = 4;
frame.data[4] = 5;
frame.data[5] = 6;
frame.data[6] = 7;
frame.data[7] = 8;
//if(frame.can_id&CAN_EFF_FLAG==0)
if(!(frame.can_id&CAN_EFF_FLAG))
    printf("Transmit standard frame!\n");
else
    printf("Transmit extended frame!\n");
```

(6): Send message to the can bus.You can use the return value of write() to check whether all data has been sent successfully .



```
/*Send message out */
nbytes = write(s, &frame, sizeof(frame));
if(nbytes != sizeof(frame)) {
    printf("Send  frame incompletely!\r\n");
    system("sudo ifconfig can0 down");
}
```

(7): Close can0 device and disable socket.

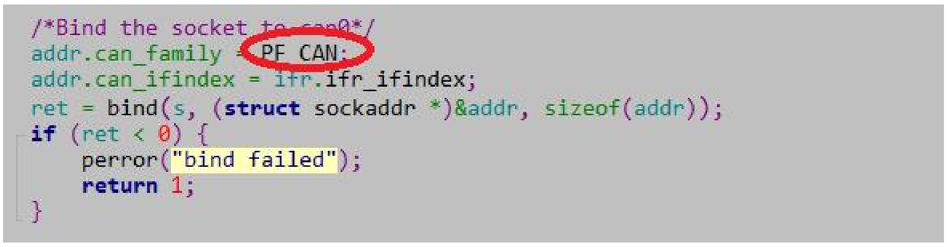


```
/*Close can0 device and destroy socket!*/
close(s);
```

For Receiver's codes

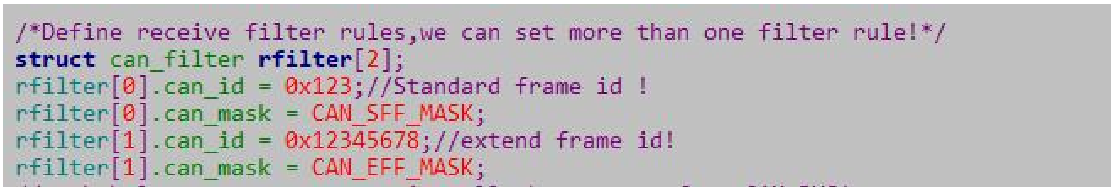
(1)step 1 and (2) is same as Sender's code.

(3):It's different from Sender's.



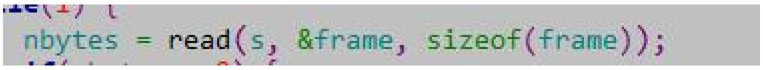
```
/*Bind the socket to can0*/
addr.can_family = PF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));
if (ret < 0) {
    perror("bind failed");
    return 1;
}
```

(4): Define receive filter rules,we can set more than one filters rule.



```
/*Define receive filter rules,we can set more than one filter rule!*/
struct can_filter rfilter[2];
rfilter[0].can_id = 0x123;//Standard frame id !
rfilter[0].can_mask = CAN_SFF_MASK;
rfilter[1].can_id = 0x12345678;//extend frame id!
rfilter[1].can_mask = CAN_EFF_MASK;
```

(5): Read data back from can bus.



```
nbytes = read(s, &frame, sizeof(frame));
```

Programming in Python

Import

import os

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux. We usually use os.system() function to execute a shell command to set CAN.

import can

The python-can library provides Controller Area Network support for Python, providing common abstractions to different hardware devices, and a suite of utilities for sending and receiving messages on a CAN bus.

For more information about python-can, please to below link:

https://python-can.readthedocs.io/en/stable/index.html

```
ifconfig
```

If you are use Ubuntu system, It may can’t use the ‘ifconfig‘ command. Please install the net tools.

```
sudo apt install net-tools
```

Simple common functions

(1) Set bitrate and start up CAN device.

```
os.system('sudo ip link set can0 type can bitrate 1000000')
os.system('sudo ifconfig can0 up')
```

(2) Bind the socket to ‘can0’.

```
can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan_ctypes')
```

(3) Assembly data to send.

```
msg = can.Message(arbitration_id=0x123, data=[0, 1, 2, 3], extended_id=False)
```

(4) Send data.

```
can0.send(msg)
```

(5) Receive data.

```
msg = can0.recv(30.0)
```

(6) Close CAN device

```
os.system('sudo ifconfig can0 down')
```

Error Frame

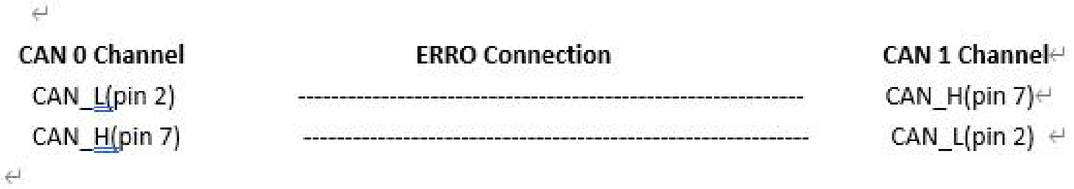
You may receive some error frame marked in red when you use the USB2CANX2-FD module. They will tell you what problem does the USB2CANX2-FD module meet on your CAN Bus.

Some people would say why didn’t they meet the error frame with other tool or USB to CAN module before. The truth is that most of the tool filter out the error frame to avoid controversy and support. They just show nothing when there are some error on the CAN Bus. We want to show the all raw data to help you to analyze your CAN BUS. Some error can be ignored, but some error maybe the hidden danger for your CAN BUS.

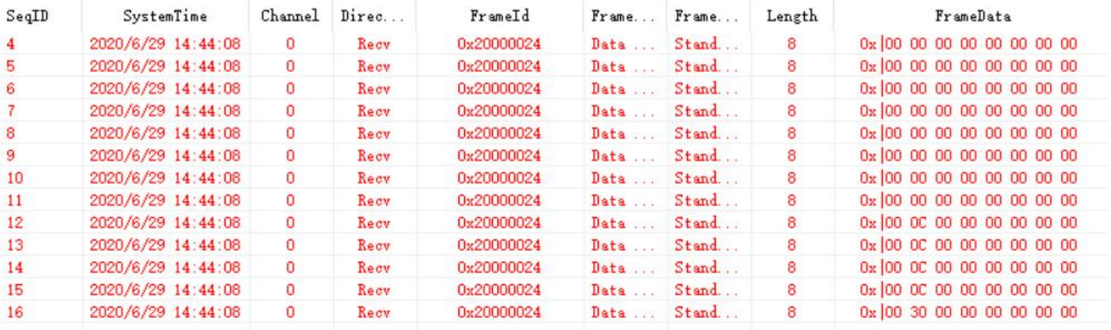
For the error frame ID description, please refer to below link:

https://github.com/linux-can/can-utils/blob/master/include/linux/can/error.h

Now we take a simple case to show you how to analyze the error frame ID. I made the incorrect connection between the USB2CAN module and the CAN Bus, to see what happens.



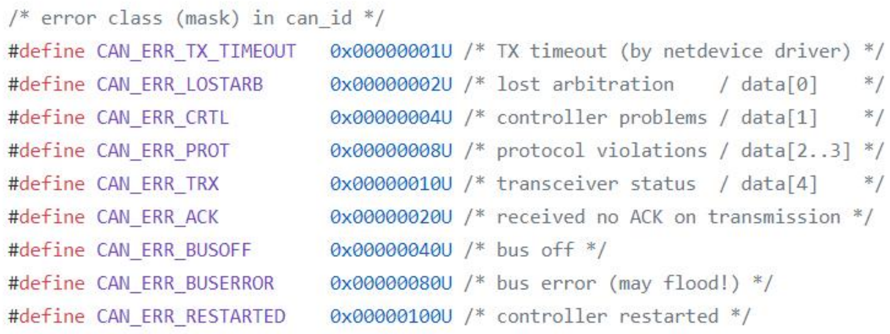
CAN 0 Channel	ERRO Connection	CAN 1 Channel
CAN_L(pin 2)	---	CAN_H(pin 7)
CAN_H(pin 7)	---	CAN_L(pin 2)



SeqID	SystemTime	Channel	Direc...	FrameId	Frame...	Frame...	Length	FrameData
4	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
5	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
6	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
7	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
8	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
9	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
10	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
11	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 00 00 00 00 00 00 00
12	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 0C 00 00 00 00 00 00
13	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 0C 00 00 00 00 00 00
14	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 0C 00 00 00 00 00 00
15	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 0C 00 00 00 00 00 00
16	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x|00 30 00 00 00 00 00 00

As Above, We received error frame Id: 0x20000024 and 2 set of 8 byte Frame Data:
data[0]=0x00, data[1]=0x0C,data[3] to data[7] are all 0x00 .
data[0]=0x00, data[1]=0x30,data[3] to data[7] are all 0x00 .

According the above error frame ID description link:



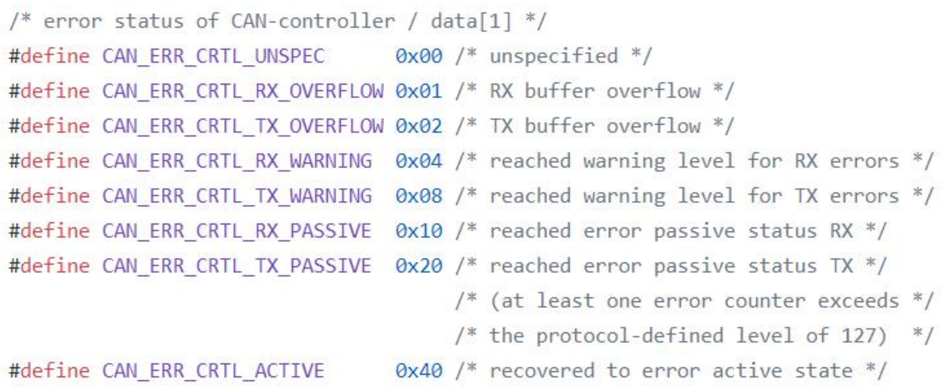
```
/* error class (mask) in can_id */
#define CAN_ERR_TX_TIMEOUT   0x00000001U /* TX timeout (by netdevice driver) */
#define CAN_ERR_LOSTARB      0x00000002U /* lost arbitration    / data[0]    */
#define CAN_ERR_CRTL         0x00000004U /* controller problems / data[1]    */
#define CAN_ERR_PROT         0x00000008U /* protocol violations / data[2..3] */
#define CAN_ERR_TRX          0x00000010U /* transceiver status  / data[4]    */
#define CAN_ERR_ACK          0x00000020U /* received no ACK on transmission */
#define CAN_ERR_BUSOFF       0x00000040U /* bus off */
#define CAN_ERR_BUSERROR     0x00000080U /* bus error (may flood!) */
#define CAN_ERR_RESTARTED    0x00000100U /* controller restarted */
```

This Error frame ID = 0x200000000 | 0x00000020|0x00000004
= 0x200000000 | CAN_ERR_ACK|CAN_ERR_CRTL

So the USB2CANX2-FD meet two problem ‘received no ACK on transmission’ and ‘controller problems’.

For problem ‘received no ACK on transmission’ may case by the not CAN-BUS or other module on the CAN BUS are only listen mode(No ACK).

For problem ‘controller problems’, refer to the data[1] description:



```
/* error status of CAN-controller / data[1] */
#define CAN_ERR_CRTL_UNSPEC      0x00 /* unspecified */
#define CAN_ERR_CRTL_RX_OVERFLOW 0x01 /* RX buffer overflow */
#define CAN_ERR_CRTL_TX_OVERFLOW 0x02 /* TX buffer overflow */
#define CAN_ERR_CRTL_RX_WARNING  0x04 /* reached warning level for RX errors */
#define CAN_ERR_CRTL_TX_WARNING  0x08 /* reached warning level for TX errors */
#define CAN_ERR_CRTL_RX_PASSIVE  0x10 /* reached error passive status RX */
#define CAN_ERR_CRTL_TX_PASSIVE  0x20 /* reached error passive status TX */
                                      /* (at least one error counter exceeds */
                                      /* the protocol-defined level of 127)  */
#define CAN_ERR_CRTL_ACTIVE      0x40 /* recovered to error active state */
```

data[1] = 0x0C = 0x04|0x08 = CAN_ERR_CRTL_RX_WARNING|CAN_ERR_CRTL_TX_WARNING
It means the USB2CAN module can’t send/receive data properly and reached warning level.

data[1] = 0x30 = 0x10|0x20 = CAN_ERR_CRTL_RX_PASSIVE | CAN_ERR_CRTL_TX_PASSIVE
It means the USB2CAN module can’t send/receive data too much, USB2CAN module into error status.

Summing up the above, the error frame tell us, USB2CAN module can’t get ACK from CAN BUS and can’t send data to the CAN Bus. So the CAN Bus may not inexistence or the connection error.