

PiCAN FD DUO Isolated USER GUIDE V1.0

Product name	PiCAN FD Duo Isolated CAN-Bus Board for Raspberry Pi 4
Model number	RSP-PiCAN FD DUO ISO
Manufacturer	SK Pang Electronics Ltd

Contents

Table of Contents

1. Introduction	3
1.1. Features	3
2. Hardware Installation	3
1.2. Plug-in Screw Terminals	4
1.3. 120Ω Terminator	4
1.4. LED.....	4
3. Software Installation	4
1.5. Installing CAN Utils.....	5
1.6. Bring Up the Interface	5
4. Python Installation and Use.....	6

1. Introduction

This PiCAN FD Duo Isolated board provide two channel CAN-Bus FD capability for the Raspberry Pi. It uses the Microchip MCP2518FD CAN controller with Analog Devices ADM3055 Isolated CAN transceiver. Connection are made via 4 way plug-in screw terminal. The two channel are independently isolated.

The improved CAN FD extends the length of the data section to up to 64 bytes per frame and a data rate of up to 8 Mbps.

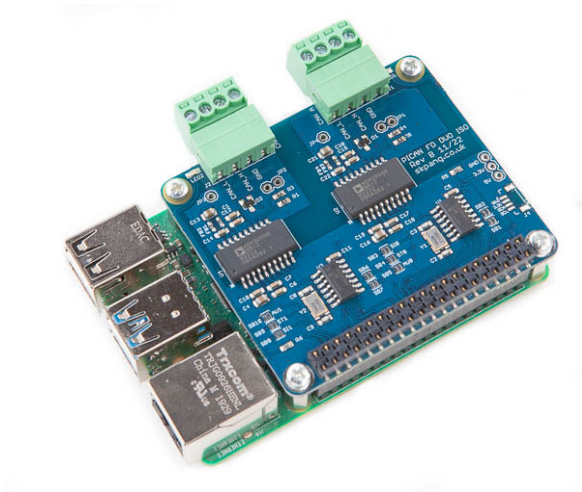
Easy to install SocketCAN driver. Programming can be done in C or Python.

1.1. Features

- Microchip MCP2518FD CAN Controller
- Analog Devices ADM3055 Isolated CAN transceiver
- Reinforced galvanic isolation
- Arbitration Bit Rate upto 1Mbps
- Data Bit Rate up to 8Mbps
- CAN FD Controller modes
- Mixed CAN2.0B and CANFD mode
- CAN2.0B mode
- Conforms to ISO11898-1:2015
- High speed SPI Interface
- CAN connection via 4way plug-in screw terminal
- 120Ω terminator ready
- LED indicator
- Four fixing holes, comply with Pi Hat standard
- SocketCAN driver, appears as can0 and can1 to application
- Interrupt RX on GPIO25 and 05

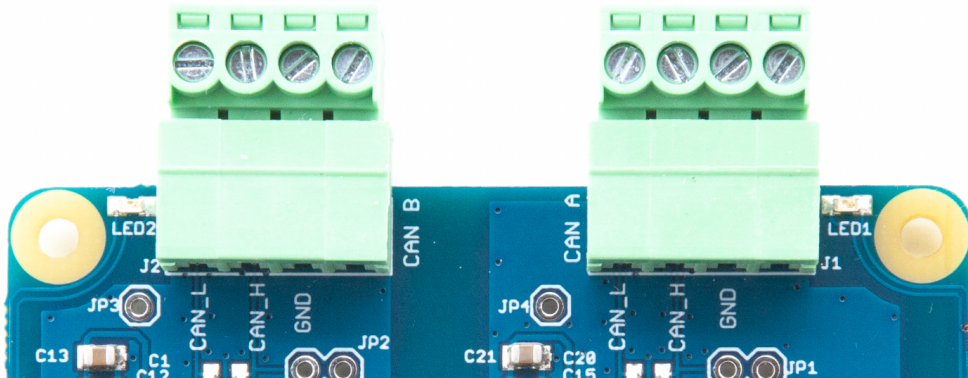
2. Hardware Installation

Before installing the board make sure the Raspberry is switched off. Carefully align the 40way connector on top of the Pi. Use spacer and screw (optional items) to secure the board.



1.2. Plug-in Screw Terminals

The CAN connections are made via the 4way plug-in screw terminals.



J1	Function	J2	Function
1	CAN_L	1	CAN_L
2	CAN_H	2	CAN_H
3	GND	3	GND
4	n/c	4	n/c

1.3. 120Ω Terminator

There is a 120Ω fitted to the board. To use the terminator solder a 2way header pin to JP1 and JP2 then insert a jumper.

1.4. LED

There is a red LED fitted to each channel. This is connected to GPIO04 and GPIO26.

3. Software Installation

It is best to start with a brand new Raspbian image. Download the latest from:

<https://www.raspberrypi.org/downloads/raspbian/>

After first time boot up, do an update and upgrade first.

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

Add the overlays by:

```
sudo nano /boot/config.txt
```

Add these lines to the end of file:

```
dtparam=spi=on
```

```
dtoverlay=mcp251xfd,spi0-0,interrupt=25
```

```
dtoverlay=mcp251xfd,spi0-1,interrupt=05
```

Reboot Pi:

```
sudo reboot
```

1.5. Installing CAN Utils

Install the CAN utils by:

```
sudo apt-get install can-utils
```

1.6. Bring Up the Interface

You can now bring the CAN interface up with CAN 2.0B at 500kbps:

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

or CAN FD at 500kbps / 2Mbps. Use copy and paste to a terminal.

```
sudo /sbin/ip link set can0 up type can bitrate 500000 dbitrate 2000000 fd on sample-point .7 dsample-point .7
```

Connect the PiCAN FD Duo ISO to your CAN network via plug-in screw terminal.

To send a CAN 2.0 message use :

```
cansend can0 7DF#0201050000000000
```

This will send a CAN ID of 7DF. Data 02 01 05 – coolant temperature request.

To send a CAN FD message with BRS use :

```
cansend can0 7df##1555555555555555
```

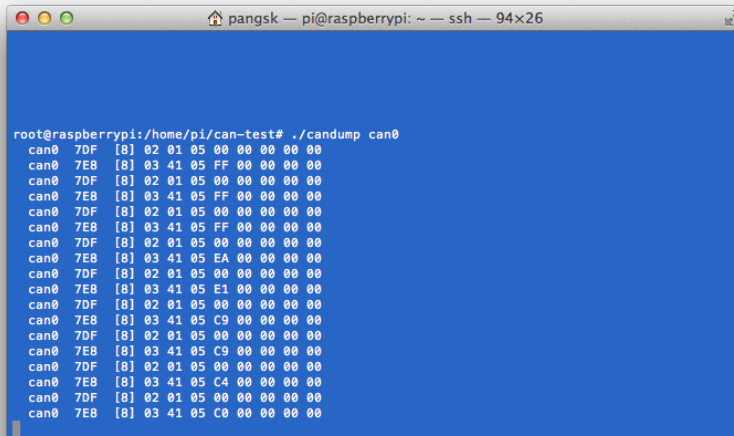
To send a CAN FD message with no BRS use :

```
cansend can0 7df##0555555555555555
```

Connect the PiCAN to a CAN-bus network and monitor traffic by using command:

```
candump can0
```

You should see something like this:



```
pangsk — pi@raspberrypi: ~ — ssh — 94x26
root@raspberrypi:/home/pi/can-test# ./candump can0
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 FF 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 FF 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 FF 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 EA 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 E1 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 C9 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 C9 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 C4 00 00 00 00
can0 7DF [0] 02 01 05 00 00 00 00 00
can0 7E8 [0] 03 41 05 C0 00 00 00 00
```

To use the other CAN channel replace can0 with can1.

4. Python Installation and Use

Ensure the driver for PiCAN FD is installed and working correctly first.

Clone the pythonCan repository by:

```
git clone https://github.com/hardbyte/python-can
```

```
cd python-can
```

```
sudo python3 setup.py install
```

Check there is no error been displayed.

Bring up the can0 interface:

```
sudo /sbin/ip link set can0 up type can bitrate 500000 dbitrate 2000000 fd on sample-point .7 dsample-point .7
```

Now start python3 and try the transmit with CAN FD and BRS set.

```
python3
```

```
import can
```

```
bus = can.interface.Bus(channel='can0', bustype='socketcan', fd = True)
```

```
msg = can.Message(arbitration_id=0x7de, extended_id=False, is_fd = True,
bitrate_switch = True, data=[0,0,0,0,0,0x1e,0x21,0xfe, 0x80, 0, 0,1,0])
```

```
bus.send(msg)
```

```

pi@raspberrypi:~/python-can $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan_native',fd = True)
>>> msg = can.Message(arbitration_id=0x7de,extended_id=False,is_fd = True, bitrate
_switch = True,data=[0,0,0,0,0,0x1e,0x21,0xfe, 0x80, 0, 0,1,0])
>>> bus.send(msg)
>>> █

```

To received messages and display on screen type in:

```
notifier = can.Notifier(bus, [can.Printer()])
```

```

pi@raspberrypi:~/python-can $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan_native',fd = True)
>>> msg = can.Message(arbitration_id=0x7de,extended_id=False,is_fd = True, bitrate_switch = True,data=[0,0,0
,0,0,0x1e,0x21,0xfe, 0x80, 0, 0,1,0])
>>> bus.send(msg)
>>> notifier = can.Notifier(bus, [can.Printer()])
>>> Timestamp: 1521407261.782672      ID: 0123      S      DLC: 5      01 22 33 44 04
Timestamp: 1521407262.494297      ID: 0123      S      DLC: 5      01 22 33 44 04
Timestamp: 1521407263.006066      ID: 0123      S      DLC: 5      01 22 33 44 04
Timestamp: 1521407263.406438      ID: 0123      S      DLC: 5      01 22 33 44 04
Timestamp: 1521407265.154456      ID: 07df      S      DLC: 8      23 41 23 41 34 23 04 00
Timestamp: 1521407265.746158      ID: 07df      S      DLC: 8      23 41 23 41 34 23 04 00
Timestamp: 1521407266.226386      ID: 07df      S      DLC: 8      23 41 23 41 34 23 04 00
Timestamp: 1521407307.873616      ID: 0123      S      F      DLC: 12     01 22 33 44 04 00 00 00 00 00 00
Timestamp: 1521407308.385764      ID: 0123      S      F      DLC: 12     01 22 33 44 04 00 00 00 00 00 00
Timestamp: 1521407308.816160      ID: 0123      S      F      DLC: 12     01 22 33 44 04 00 00 00 00 00 00
>>> █

```

Documentation for python-can can be found at :

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

More examples in github:

<https://github.com/skpang/PiCAN-FD-Python-examples>