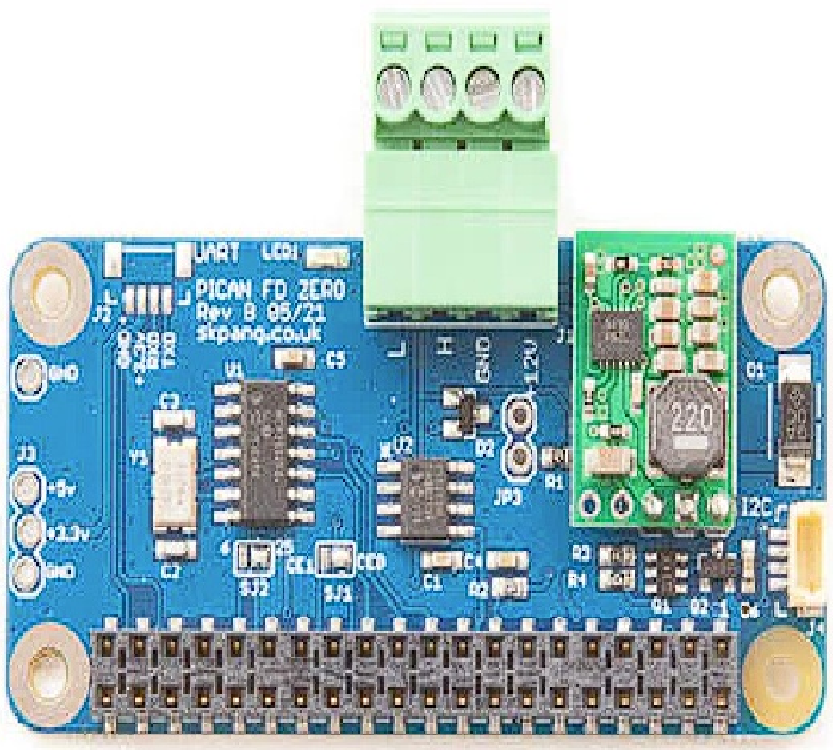
SK Pang electronics PiCAN FD Zero Raspberry Pi Zero User Guide

Contents

SK Pang electronics

SK Pang electronics PiCAN FD Zero Raspberry Pi Zero



Introduction

This PiCAN FD Zero board provide CAN-Bus FD capability for the Raspberry Pi Zero. It uses the Microchip MCP2518FD CAN controller with MCP2562FD CAN transceiver. Connection are made via 4way plug in terminal. CAN_H, CAN_L and +12v supply for the board and the Pi Zero. On board is a 1A SMPS which supplies power to the PiCAN FD and Pi Zero board.
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.

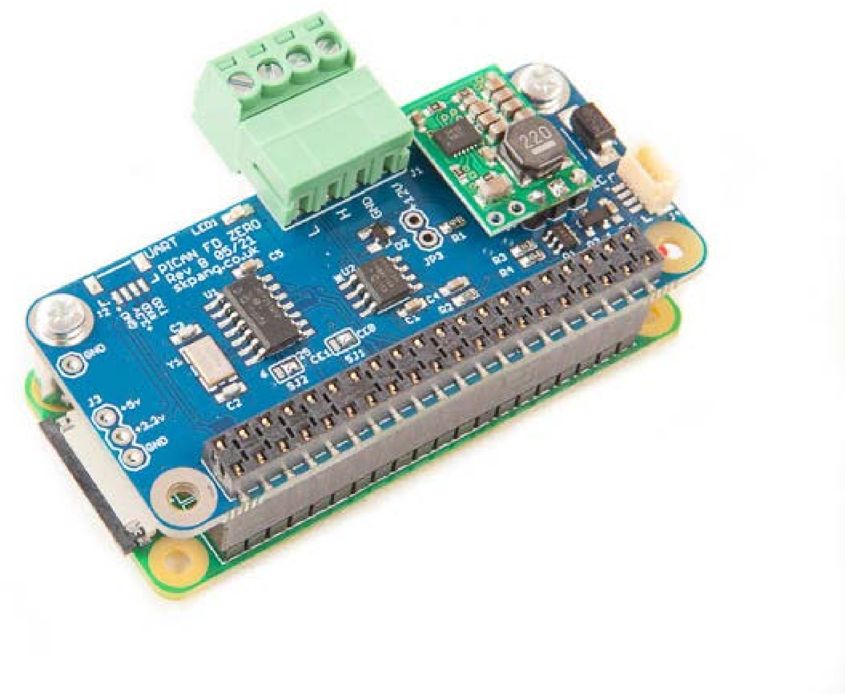
Features

- 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
- 120Ω terminator ready
- 4 way plug-in terminal for CAN and power
- 120Ω terminator ready
- LED indicator (GPIO 22)

- SocketCAN driver, appears as can0 to application
- Interrupt RX on GPIO25 or GPIO6
- Qwiic (I2C) connector for extra sensors
- 1A SMPS 6v to 20v input range

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.



Screw Terminals

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

120W Terminator

There is a 120W fitted to the board. To use the terminator solder a 2way header pin to JP3 then insert a jumper.

LED

There is a red LED fitted to the board. This is connected to GPIO22.

SMPS (Switch Mode Power Supply)

The 5v 1A SMPS module that can power the Pi and the board. It has an input voltage range of 6v to 20v.

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
```
Reboot Pi:
```
sudo reboot
```

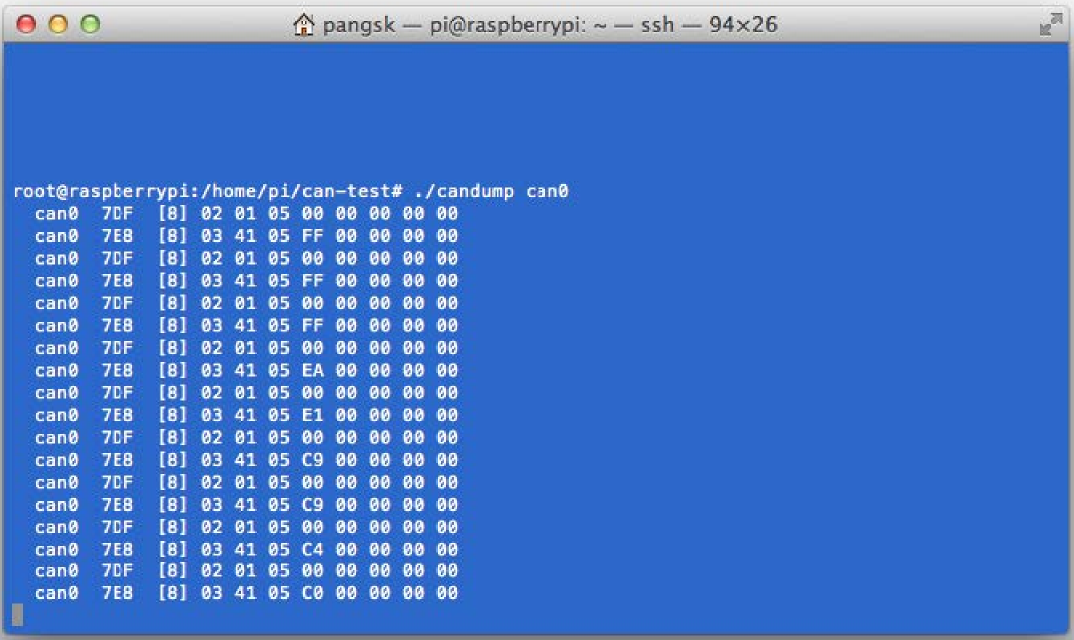
Installing CAN Utils

Install the CAN utils by:
```
sudo apt-get install can-utils
```

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 500kpbs / 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 .8 dsample-point .8
```
Connect the PiCAN FD Zero to your CAN network via the 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##15555555555555555
```
To send a CAN FD message with no BRS use :
```
cansend can0 7df##05555555555555555
```
Connect the PiCAN to a CAN-bus network and monitor traffic by using command:
```
candump can0
```

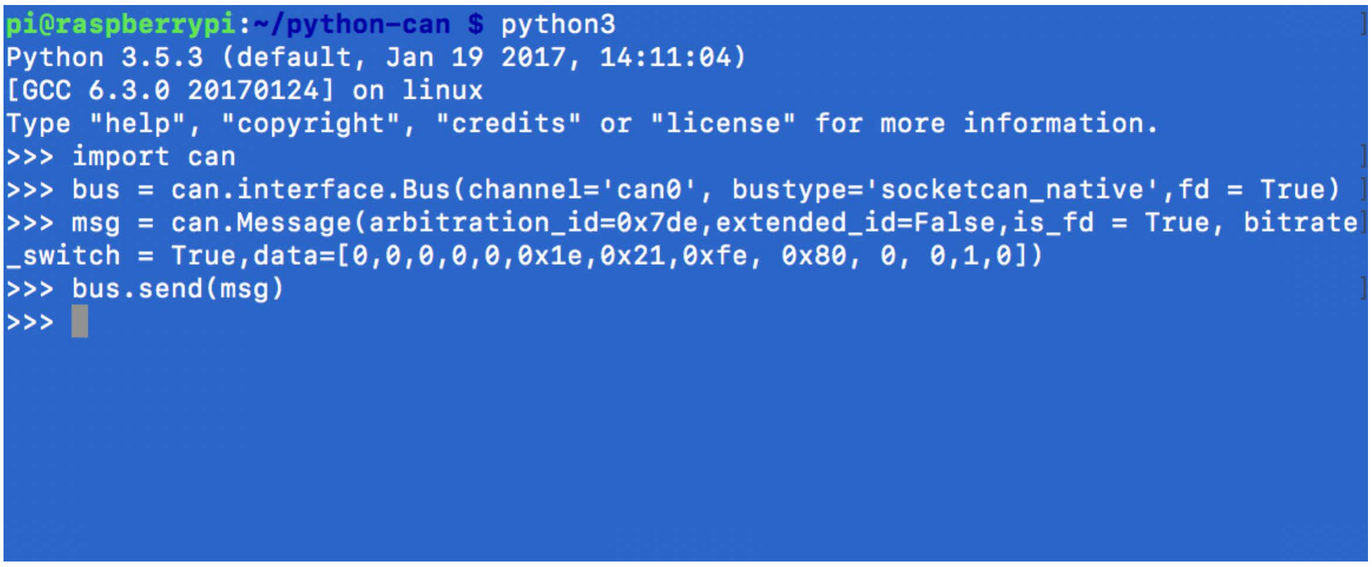
You should see something like this:



```
root@raspberrypi:/home/pi/can-test# ./candump can0
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 FF 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 FF 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 FF 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 EA 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 E1 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 C9 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 C9 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 C4 00 00 00 00
  can0  7DF   [8]  02 01 05 00 00 00 00 00
  can0  7E8   [8]  03 41 05 C0 00 00 00 00
```

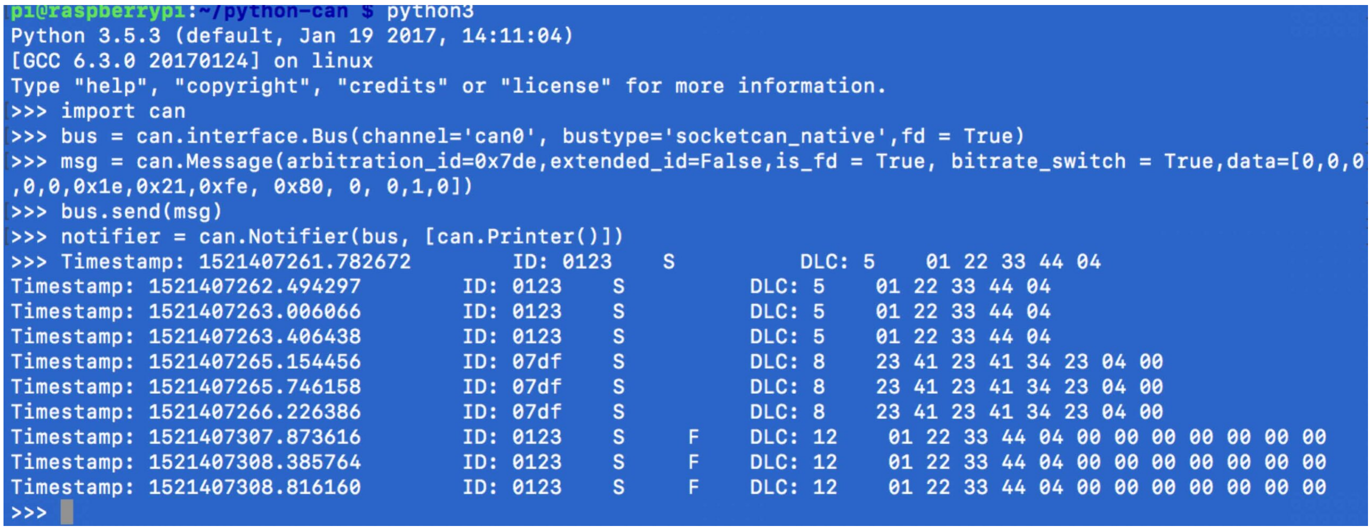
Python Installation and Use

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 .8 dsample-point .8
Now start python3 and try the transmit with CAN FD and BRS set.
python3
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)



```
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 receive messages and display on screen type in: notifier = can.Notifier(bus, [can.Printer()])

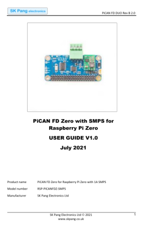


```
pi@raspberrypi:~/python-can $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan_native',fd = True)
>>> msg = can.Message(arbitration_id=0x7de,extended_id=False,is_fd = 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 00
Timestamp: 1521407308.385764        ID: 0123    S     F    DLC: 12    01 22 33 44 04 00 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 00
>>> 
```

Documentation for python-can can be found
at : https://python-can.readthedocs.io/en/stable/index.html
More expamles in github:
https://github.com/skpang/PiCAN-FD-Python-examples

SK Pang Electronics Ltd Ã“ 2021 www.skpang.co.uk

Documents / Resources

 SK Pang electronics PiCAN FD DUO Rev B 2.0 PiCAN FD Zero with SMPS for Raspberry Pi Zero USER GUIDE V1.0 July 2021 Product name PiCAN FD Zero for Raspberry Pi Zero with 1A SMPS Model number RSP-PICANFDZ-SMPS Manufacturer SK Pang Electronics Ltd SK Pang Electronics Ltd © 2021 www.skpang.co.uk 1	SK Pang electronics PiCAN FD Zero Raspberry Pi Zero [pdf] User Guide PiCAN FD Zero, Raspberry Pi Zero, PiCAN FD Zero Raspberry Pi Zero

References

- SK Pang Electronics Ltd - Electronic supply for engineer and hobbyist
- GitHub - hardbyte/python-can: The can package provides controller area network support for Python developers
- GitHub - skpang/PiCAN-FD-Python-examples
- python-can 4.1.0 documentation
- Raspberry Pi OS – Raspberry Pi