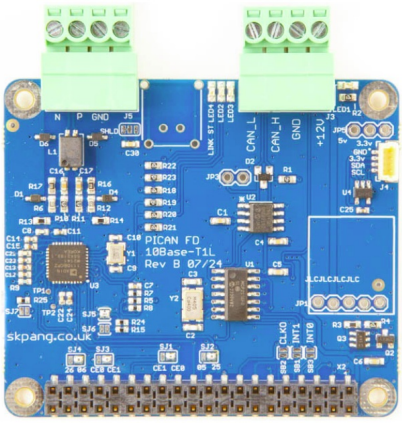
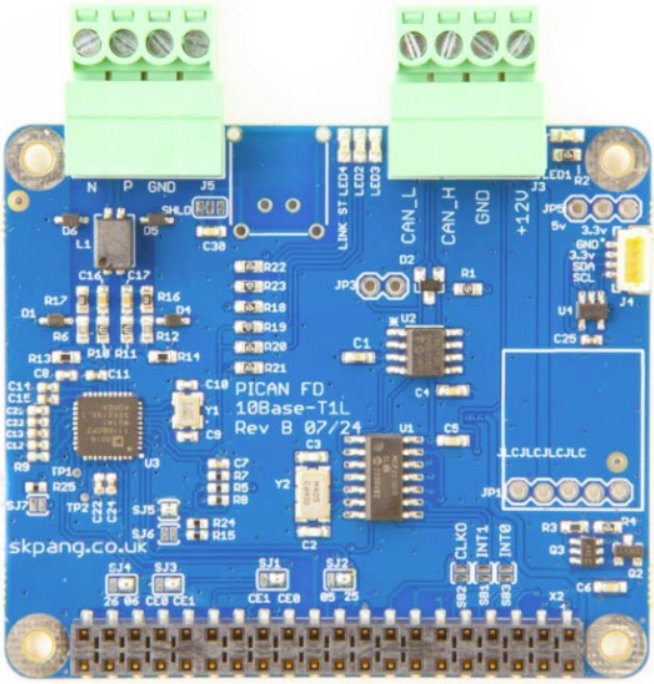


SK Pang electronics RSP-PICANFD-T1L PiCAN FD Board with 10 Base-T1L for Raspberry Pi User Guide

SK Pang electronics RSP-PICANFD-T1L PiCAN FD Board with 10 Base-T1L for Raspberry Pi User Guide



Contents

Introduction

This board has a 10Base-T1L Single Pair Ethernet (SPE) port and a CAN FD port.
The 10Base-T1L is provided by the Analog Devices' ADIN1110 chip. The CAN FD is provided by the Microchip MCP2518FD CAN controller. Connection is via 4 way plugable terminal. 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 for CAN applications.

CAN FD Features

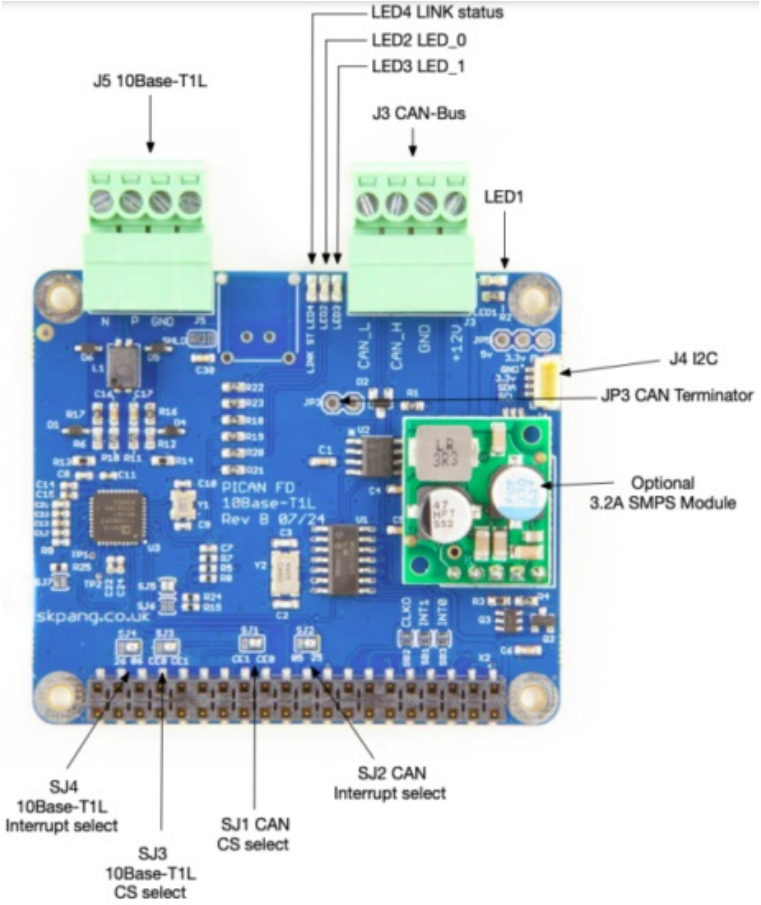
- Arbitration Bit Rate upto 1Mbps
- Data Bit Rate up to 8Mbps
- CAN FD Controller modes
- Mixed CAN2.0B and CANFD mode
- CAN2.0B mode
- Conforms to ISO11898-1:2015
- High speed SPI Interface
- CAN connection via 4 way pluggable terminal
- 120Ω terminator ready
- LED indicator
- Four fixing holes, comply with Pi Hat standard
- SocketCAN driver, appears as can0 to application
- Interrupt RX on GPIO25

10Base-T1L Features

- Analog Devices ADIN1110 MAC/PHY controller
- PHY designed to IEEE Std. 802.3cg – 2019
- Cable reach up to 1700 m with 1.0 V/2.4 V
- Integrated MAC with SPI
  - Supports OPEN Alliance 10BASE-T1x MAC-PHY serial interface
  - 16 MAC address filters
  - High and low priority queues with 28 kB buffer
  - Cut through or store forward operation

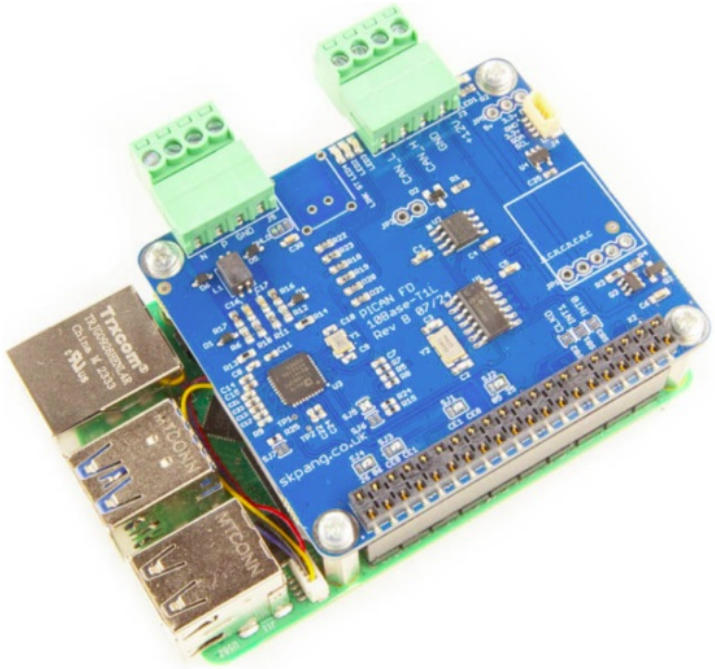
    - IEEE 1588 timestamp support
    - Statistics counters
- Low power consumption: 42 mW (dual-supply, 1.0 V p-p)
- Diagnostics
    - Cable fault detection with TDR
    - Link quality indicator with MSE
    - Frame generator and checker
    - Multiple loopback modes
    - IEEE test mode support
- Supports 1.0 V p-p and 2.4 V p-p transmit levels
- Auto negotiation
- Unique 48-bit MAC address provided by 24AA02E48 IC

Product Overview



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.



If the install is on the Raspberry Pi 5 with a fan fitted then this height extender is required:

https://www.skpang.co.uk/products/raspberry-5-header-extender-kit

CAN Bus connection
The CAN bus connection is on connector J3.

J3 pin no.	Function
1	CAN_L
2	CAN_H
3	GND
4	+12v

Note : The +12v In is only used on the board with SMPS option fitted.

10Base-T1L Connection
The 10Base-T1L connection is on connector J5.

J5 pin no.	Function
1	N
2	P
3	GND
4	

CAN Bus 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 Indicators
There are four LEDs fitted to the board.

LED1 red connected to GPIO22.

LED2 red connected to LED_0 of the ADIN1110

LED3 red connected to LED_1 of the ADIN1110

LED4 blue connected to LINK Status of the ADIN1110

Optional 3.2A SMPS
Switch mode power supply module, this is an optional 5v module that can power the Pi. It has an input voltage range of 8v to 26v. The total power drawn by the Pi and any other connected peripherals must be less then 3.2A

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/firmware/config.txt

Add these lines to the end of file:
dtparam=spi=on
dtparam=i2c_arm=on
dtoverlay=adin1110
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 .9

Connect the PiCAN2 to your CAN network via screw terminal or DB9.

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.

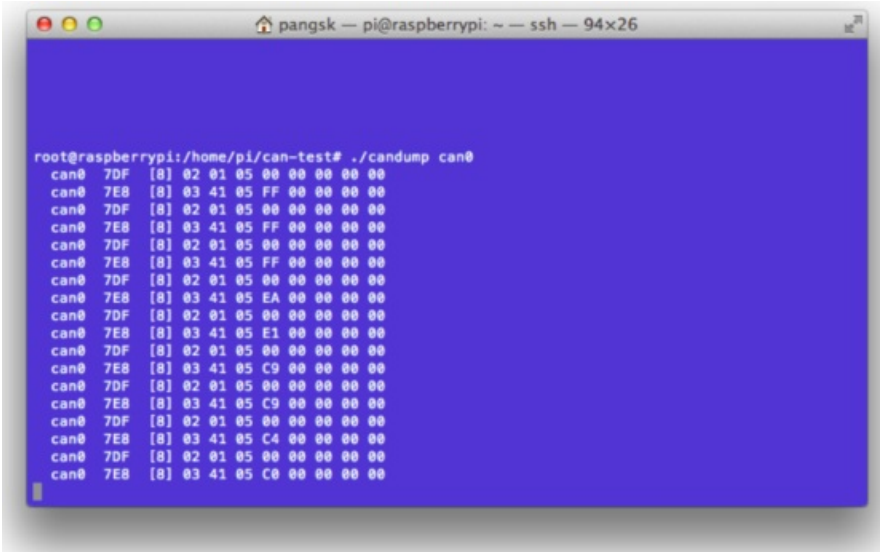
To send a CAN FD message with BRS use :
$ cansend can0 7df##15555555555555555

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:



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',fd = True) msg = can.Message(arbitration_id=0x7de,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 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 examples in github:

$ **https://github.com**/skpang/PiCAN-FD-Python-examples

10Base-T1L Driver Install

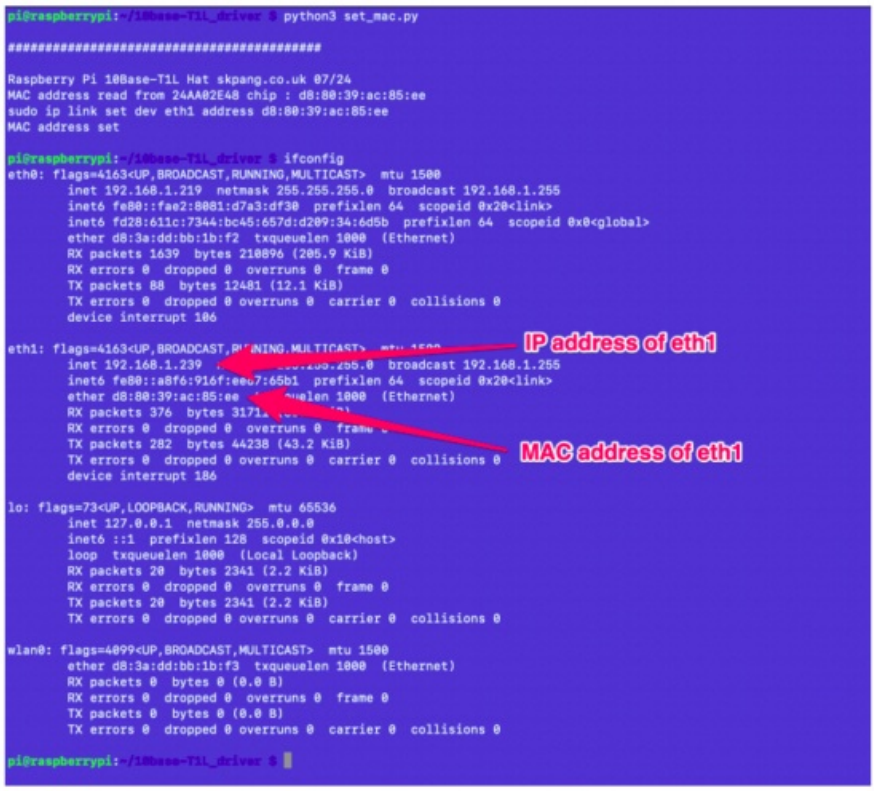
On the Raspberry Pi, at the prompt type in:

$ sudo apt install raspberrypi-kernel-headers
$ git clone https://github.com/skpang/10base-T1L_driver.git
$ cd 10base-T1L_driver/
$ make
$ sudo cp adin1110.ko /lib/modules/$(uname -r)/kernel/drivers/net

$ sudo depmod -a
$ sudo cp adin1110.dtbo /boot/overlays
$ sudo reboot
$ cd 10base-T1L_driver/
$ python3 set_mac.py

eth1 is the port of the 10base-T1L interface.

You should see something like this:



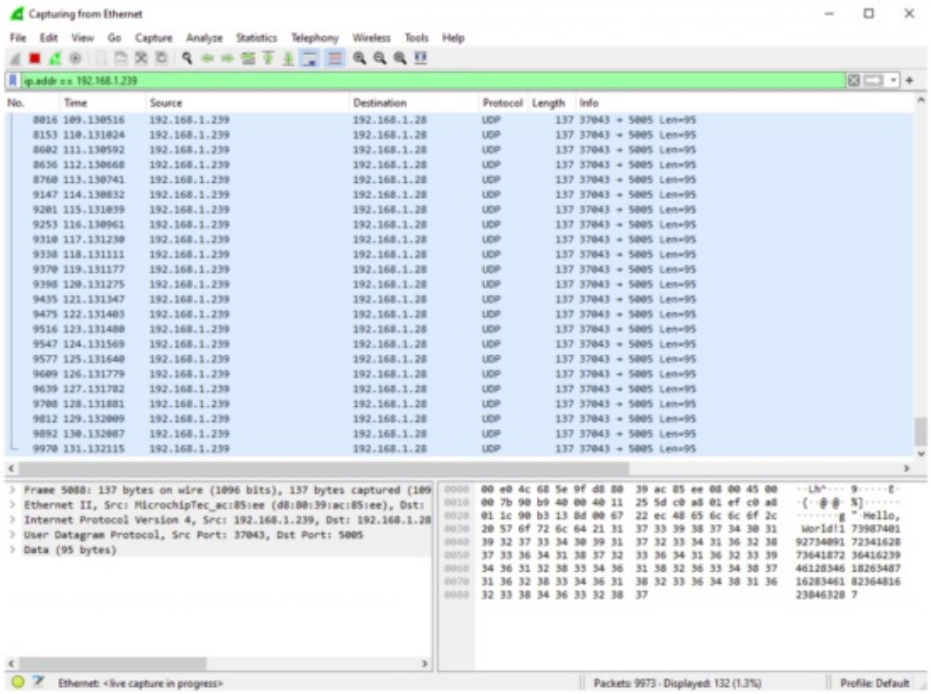
This shows eth1 is up.

Using a twisted pair wire, connect to J5 and the other end to a 10base-T1L media converter. Install Wireshark software on a PC. Assuming PC has an ip address of 192.168.1.28

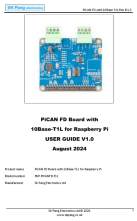
On the Raspberry Pi, at the prompt type in:

$ python3 10Base-T1S_UDP_demo.py

On the Windows machine start Wireshark software. You should see something like this



Documents / Resources

	SK Pang electronics RSP-PICANFD-T1L PiCAN FD Board with 10 Base-T1L for Raspberry Pi [pdf] User Guide RSP-PICANFD-T1L PiCAN FD Board with 10 Base-T1L for Raspberry Pi, RSP-PICANFD-T1L, PiCAN FD Board with 10 Base-T1L for Raspberry Pi, Raspberry Pi

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/10base-T1L_driver
- GitHub - skpang/PiCAN-FD-Python-examples
- python-can 4.4.2 documentation
- Raspberry 5 header extender kit — SK Pang Electronics Ltd
- User Manual