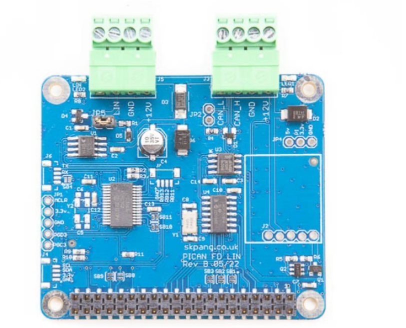


# SK Pang electronics RSP-PICANFDLIN PICAN FD and LIN-Bus Board for Raspberry Pi User Guide

Home » SK Pang electronics » SK Pang electronics RSP-PICANFDLIN PICAN FD and LIN-Bus Board for Raspberry Pi User Guide

Contents

SK Pang electronics

## SK Pang electronics RSP-PICANFDLIN PICAN FD and LIN-Bus Board for Raspberry Pi

## Introduction

This PiCAN FD board with LIN-bus. Classic CAN and CAN FD is provided by the Microchip MCP2518FD IC. LIN-bus is provided by a dsPIC33 micro-controller. Communication to the Pi is over UART on ttyS0 using ASCII text commands. Example LIN-bus GUI app is available written in Python3 and tkinter.
The firmware is updatable using the Microchip UnifiedHost java app. This requires the Raspberry Pi running in GUI mode.
Easy to install SocketCAN driver. Programming can be done in C or Python.
Optional 3A SMPS module which can power the PiCAN FD LIN-bus board and Raspberry Pi from 7 to 24v external supply

#### CAN Features

- Arbitration Bit Rate upto 1Mbps
- Data Bit Rate up to 8Mbps
- CAN FD Controller modes
- Mixed CAN2.0B and CANFD mode
- Conforms to ISO11898-1:2015
- High speed SPI Interface
- CAN connection via 4way screw terminal
- 120Ω terminator ready
- LED indicator (GPIO04)
- Four fixing holes, comply with Pi Hat standard
- SocketCAN driver, appears as can0 and can1 to application
- Interrupt RX on GPIO25

#### LIN-Bus Features

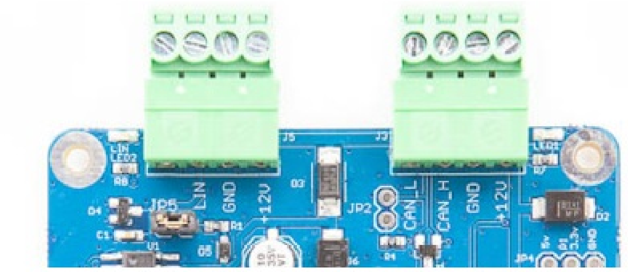
- LED indicator
- LIN master or slave. Setting via jumper
- LIN provided by dsPIC33 micro-controller with updatable firmware
- Communicate to the Pi via ASCII text commands on ttyS0

## Hardware Installation

Before installing the board make sure the Raspberry Pi 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 screw terminals on J3 and LIN on J5.



| J5 LIN-Bus | Function |
|---|---|
| 1 | |
| 2 | LIN |
| 3 | GND |
| 4 | +12v |

| J3 CAN-Bus | Function |
|---|---|
| 1 | CAN_L |
| 2 | CAN_H |
| 3 | GND |
| 4 | +12v |

### CAN-BUS 120W Terminator

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

### LEDs

There is a red LED (LED1) fitted to the board. This is connected to GPIO04. LED2 is LIN-bus indication which is controlled by the dsPIC33.

### Optional

SMPS. Switch mode power supply module option, this is a 5v module that can power the board and the Raspberry Pi. It has an input voltage range of 7v to 24v.

## Software Installation

It is best to start with a brand new Raspbian image. Download the latest from:
After first time boot up, do an update and upgrade first.

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

Add these lines to the end of file:

```
enable_uart=1
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 LIN board to your CAN network.
To send a CAN 2.0 message use :

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
```

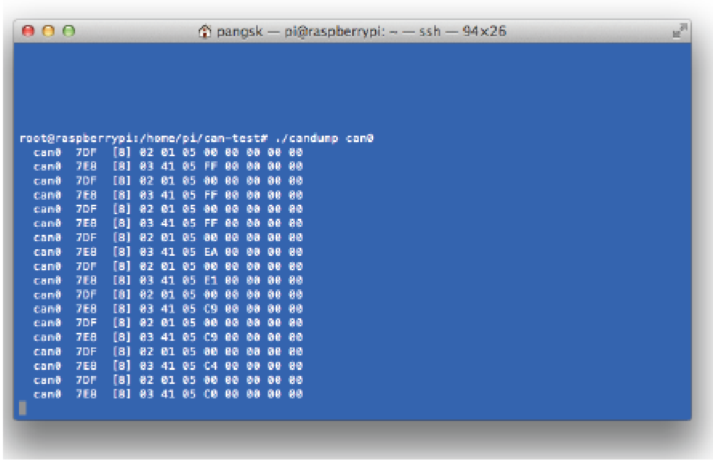
To send a CAN FD message with no BRS use :

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

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

```
candump can0
```

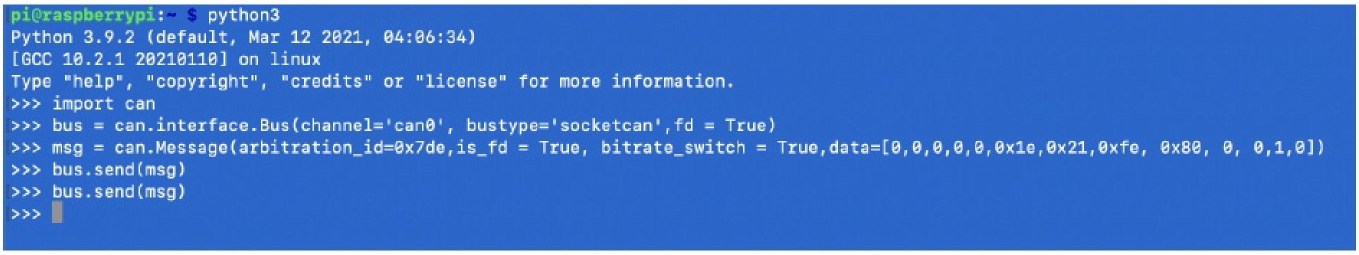
You should see something like this:



```
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
```

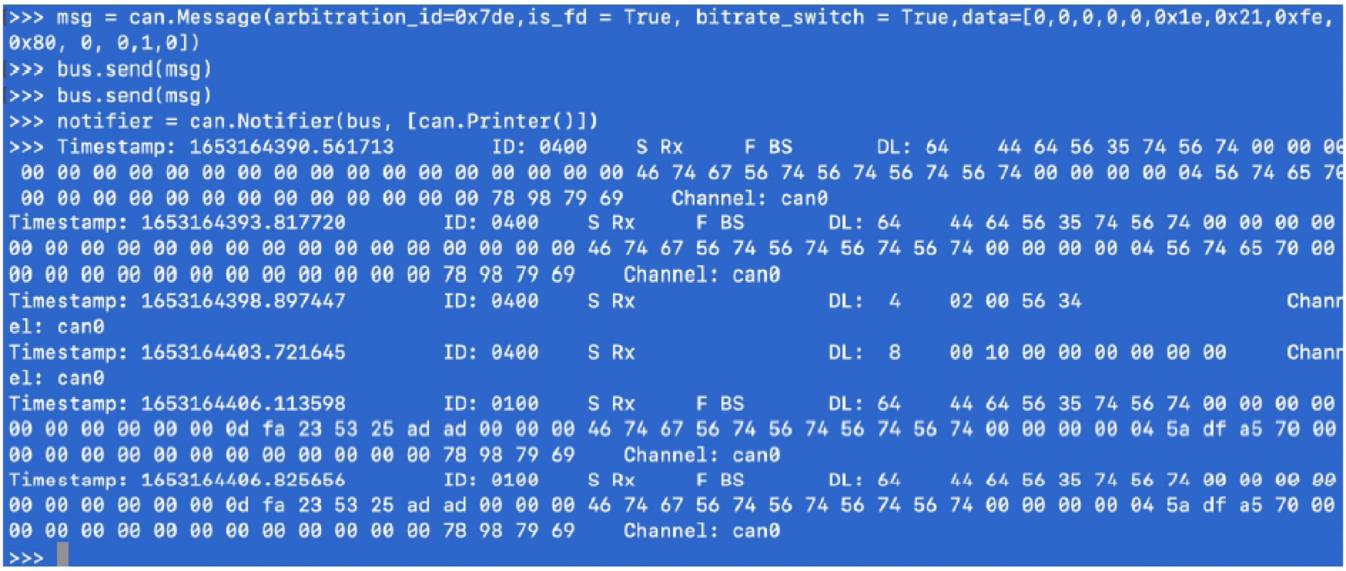
## Python Installation and Use

Ensure the driver fr 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 ad 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)



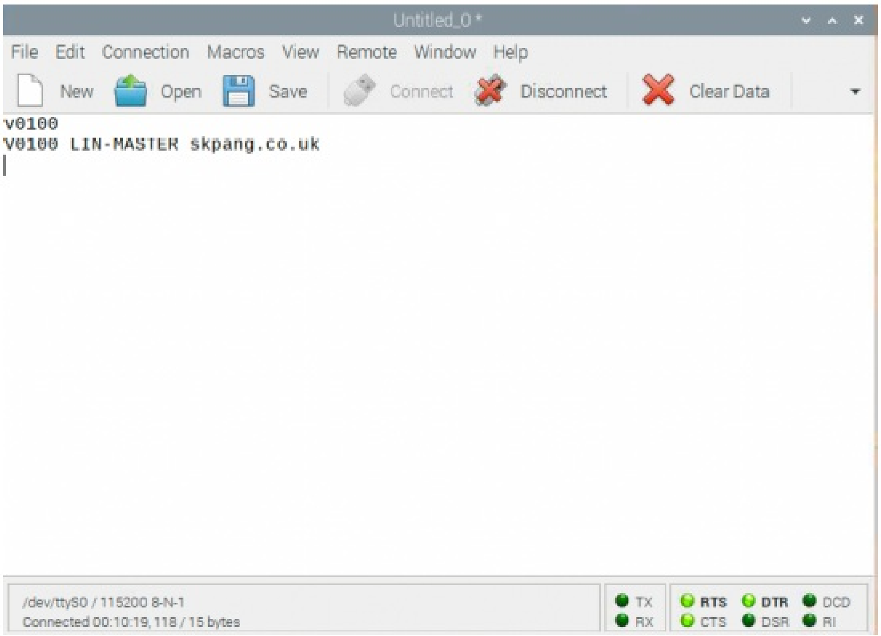
```
pi@raspberrypi:~ $ python3
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan',fd = True)
>>> msg = can.Message(arbitration_id=0x7de,is_fd = True, bitrate_switch = True,data=[0,0,0,0,0,0x1e,0x21,0xfe, 0x80, 0, 0,1,0])
>>> bus.send(msg)
>>> bus.send(msg)
>>> 
```

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



```
>>> 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)
>>> bus.send(msg)
>>> notifier = can.Notifier(bus, [can.Printer()])
>>> Timestamp: 1653164390.561713        ID: 0400    S Rx     F BS       DL: 64    44 64 56 35 74 56 74 00 00 0
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 46 74 67 56 74 56 74 56 74 56 74 00 00 00 00 04 56 74 65 7
 00 00 00 00 00 00 00 00 00 00 00 00 00 78 98 79 69    Channel: can0
Timestamp: 1653164393.817720        ID: 0400    S Rx     F BS       DL: 64    44 64 56 35 74 56 74 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 46 74 67 56 74 56 74 56 74 56 74 00 00 00 00 04 56 74 65 70 00
00 00 00 00 00 00 00 00 00 00 00 00 78 98 79 69    Channel: can0
Timestamp: 1653164398.897447        ID: 0400    S Rx                DL:  4    02 00 56 34                 Chann
el: can0
Timestamp: 1653164403.721645        ID: 0400    S Rx                DL:  8    00 10 00 00 00 00 00 00     Chann
el: can0
Timestamp: 1653164406.113598        ID: 0100    S Rx     F BS       DL: 64    44 64 56 35 74 56 74 00 00 00 00
00 00 00 00 00 00 0d fa 23 53 25 ad ad 00 00 00 46 74 67 56 74 56 74 56 74 56 74 00 00 00 00 04 5a df a5 70 00
00 00 00 00 00 00 00 00 00 00 00 00 78 98 79 69    Channel: can0
Timestamp: 1653164406.825656        ID: 0100    S Rx     F BS       DL: 64    44 64 56 35 74 56 74 00 00 00 00
00 00 00 00 00 00 0d fa 23 53 25 ad ad 00 00 00 46 74 67 56 74 56 74 56 74 56 74 00 00 00 00 04 5a df a5 70 00
00 00 00 00 00 00 00 00 00 00 00 00 78 98 79 69    Channel: can0
>>> 
```

Documentation for python-can can be found at :
[https://python-can.readthedocs.io/en/stable/index.html](https://python-can.readthedocs.io/en/stable/index.html)
More expamles in github:
[https://github.com/skpang/PiCAN-FD-Python-examples](https://github.com/skpang/PiCAN-FD-Python-examples)
1.9. LIN-Bus Usage
LIN-bus is communicated to the Pi over the UART on ttyS0 port at 115200 8-N-1 setting.
Using Coolterm for Raspberry Pi or a text base terminal. Set the port to /dev/ttyS0 / 115200 8-N-1 and connect.
On the terminal type in ‘v’ and press enter. Type in ‘V’ and press enter.
You should see a reply like this screen:



```
v0100
V0100 LIN-MASTER skpang.co.uk
```

/dev/ttyS0 / 115200 8-N-1
Connected 00:10:19, 118 / 15 bytes

**ASCII Command Set**

To control the LIN-bus port a simple ASCII command is sent from the Pi to the PICAN FD LIN board.
**Commands**
The command requires a carriage return character (0x0D). All values are in hex.

- V Get hardware version

- v Get firmware version
- O Open LIN port
- C Close LIN port
- S Set LIN baudrate
- S1 Set baudrate to 9600
- S2Set baudrate to 10400
- S3 Set baudrate to 19200 (default)
- Transmit a LIN frame with Classic checksum

  taaxdddddddddddddddd

  aa LIN address in hex

  x Data length 0 to 8 bytes

  dd Data byte value in hex
- T Transmit a LIN frame with Enhanced checksum

  Taaxdddddddddddddddd

  aa LIN address in hex

  x Data length 0 to 8 bytes

  dd Data byte value in hex
- M Monitor mode. All LIN traffic will be sent to the Pi with a prefix of M
- r Request/respond mode. An address value will be sent on the bus and respond will pass onto the Pi.

  raa aa LIN address in hex

Example 1. Controlling NCV7430 RGB LED
To set the NCV7430 first it needs to be initialized with these values 23 C0 00 00 7F. Command string:
O
t234C000007F
To set the LED to blue, sent this string 24 C0 00 00 10 31 00 00 FF
Command string:
t248C0000010310000ff
To set the LED to red, sent this string 24 C0 00 00 10 31 00 FF 00
Command string:
t248C00000103100ff00
Example 2. Reading a window mirror switch
To read the status of a window mirror switch we need to use the r command with an address of 0x49.
Command string:
O
r49
Expected respond:
M0A4900C000F8FFFFFF0BF1

- M Monitor command
- 0X 0x0A in hex means 10 bytes responds length including the address
- 49 Address in hex
- dd Data in hex

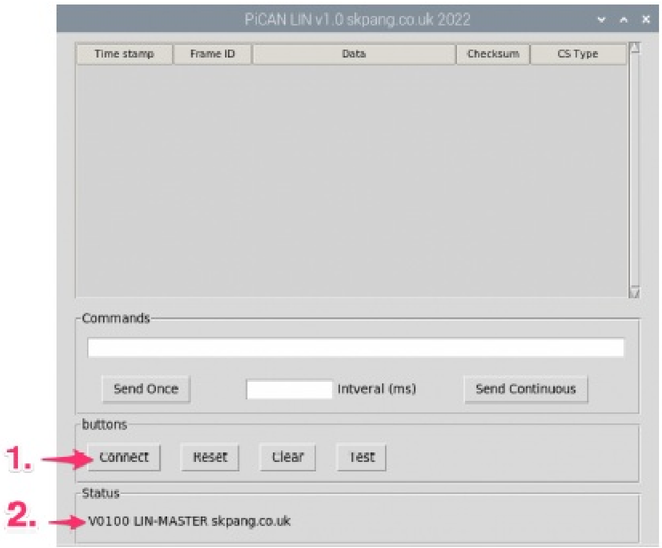
**LIN-BUS GUI**

A simple GUI interface is available to download from github:
https://github.com/skpang/PiCAN_LIN_GUI_demo

To run the program start a terminal and type in:

```
python3 pican-lin.py
```

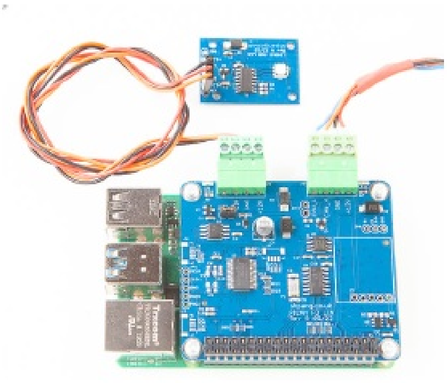
You should see a screen shown below:



```
V0100 LIN-MASTER skpang.co.uk
```

1. Click the Connect button to connect to the PiCAN FD LIN board.
2. Check the Status box is updated.

Example 1. Controlling NCV7430 RGB LED



To set the NCV7430 first it needs to be initialized with these values 23 C0 00 00 7F. First open the port. In the Commands box type in:

O

Click ‘Send Once’ then type in:

t234C000007F

Click ‘Send Once’

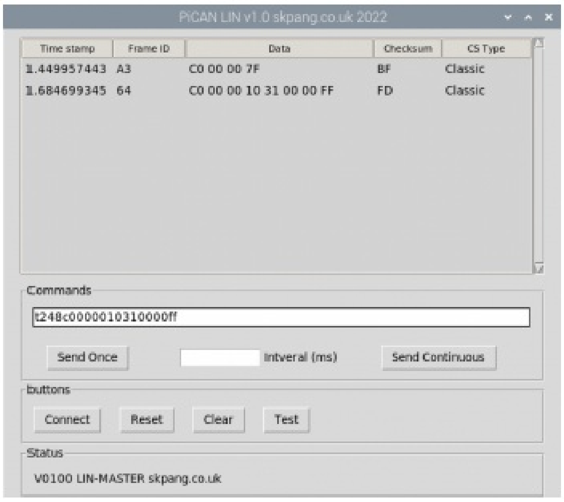
To set the LED to blue, sent this string 24 C0 00 00 10 31 00 00 FF

In the Commands box type in:

t248C0000010310000ff

Click ‘Send Once’

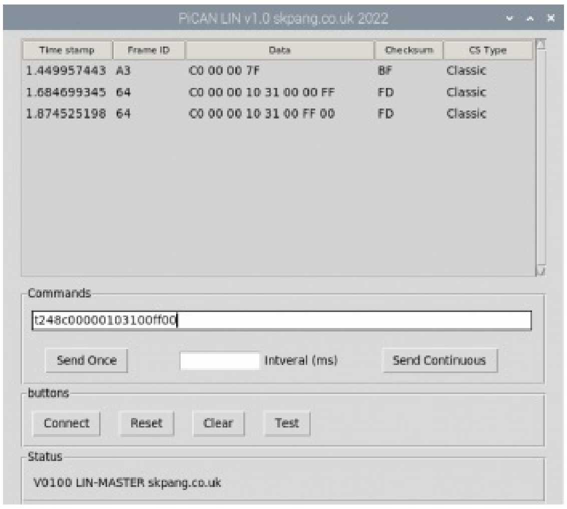
Check the LED turned blue.



| Time stamp | Frame ID | Data | Checksum | CS Type |
|---|---|---|---|---|
| 1.449957443 | A3 | C0 00 00 7F | BF | Classic |
| 1.684699345 | 64 | C0 00 00 10 31 00 00 FF | FD | Classic |

t248c0000010310000ff

V0100 LIN-MASTER skpang.co.uk



| Time stamp | Frame ID | Data | Checksum | CS Type |
|---|---|---|---|---|
| 1.449957443 | A3 | C0 00 00 7F | BF | Classic |
| 1.684699345 | 64 | C0 00 00 10 31 00 00 FF | FD | Classic |
| 1.874525198 | 64 | C0 00 00 10 31 00 FF 00 | FD | Classic |

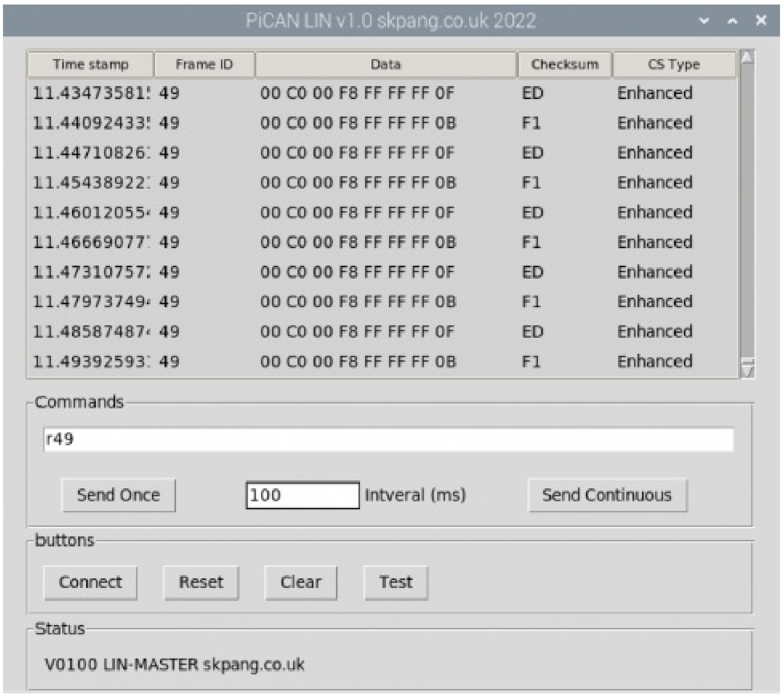
t248c00000103100ff00

V0100 LIN-MASTER skpang.co.uk

To set the LED to red, sent this string 24 C0 00 00 10 31 00 FF 00 In the Commands box type in:
t248C0000010310000ff
Click ‘Send Once’
Check the LED turned red.

Example 2. Reading a window mirror switch

To read the status of a window mirror switch type in the Commands box:
O
Click ‘Send Once’
r49
Click ‘Send Once’
100 in the Interval box and Click Start. You should see the request is sent out every 100ms with reply on the list box.
Press the window switches and you should see the data value return changes.



| Time stamp | Frame ID | Data | Checksum | CS Type |
|---|---|---|---|---|
| 11.43473581 | 49 | 00 C0 00 F8 FF FF FF 0F | ED | Enhanced |
| 11.44092433 | 49 | 00 C0 00 F8 FF FF FF 0B | F1 | Enhanced |
| 11.44710826 | 49 | 00 C0 00 F8 FF FF FF 0F | ED | Enhanced |
| 11.45438922 | 49 | 00 C0 00 F8 FF FF FF 0B | F1 | Enhanced |
| 11.46012055 | 49 | 00 C0 00 F8 FF FF FF 0F | ED | Enhanced |
| 11.46669077 | 49 | 00 C0 00 F8 FF FF FF 0B | F1 | Enhanced |
| 11.47310757 | 49 | 00 C0 00 F8 FF FF FF 0F | ED | Enhanced |
| 11.47973749 | 49 | 00 C0 00 F8 FF FF FF 0B | F1 | Enhanced |
| 11.48587487 | 49 | 00 C0 00 F8 FF FF FF 0F | ED | Enhanced |
| 11.49392593 | 49 | 00 C0 00 F8 FF FF FF 0B | F1 | Enhanced |

Commands

r49

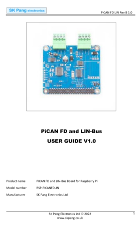
100 Intveral (ms)

buttons

Status

V0100 LIN-MASTER skpang.co.uk

## Documents / Resources

| | |
|---|---|
|  | SK Pang electronics RSP-PICANFDLIN PICAN FD and LIN-Bus Board for Raspberry Pi [pdf] User Guide<br>RSP-PICANFDLIN PICAN FD and LIN-Bus Board for Raspberry Pi, RSP-PICANFDLIN, PICAN FD and LIN-Bus Board for 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/PiCAN_LIN_GUI_demo
- GitHub - skpang/PiCAN-FD-Python-examples
- python-can 4.1.0 documentation