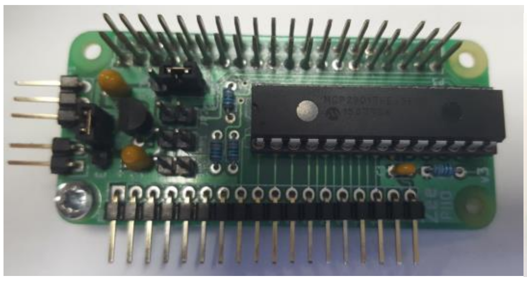
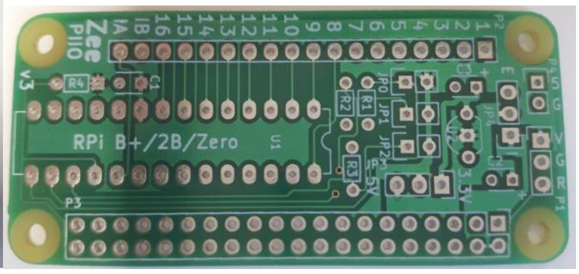
Zee PIIO Quick Start Guide

The Zee PIIO v3 board has been designed to make interfacing with Raspberry Pi (RPi) devices safe and easy.

The Zee PIIO has 16 I/O pins which can be programmed individually. A maximum of 8 Zee PIIO boards can be stacked together to expose a total of 128 I/O pins. Input pins on the Zee PIIO can safely be connected to sources with 3.3V or 5V outputs. Logical HIGH on output pins can be configured to output either 3.3V or 5V using a jumper. This feature is only available if you installed the optional voltage regulator. The Zee PIIO can also be configured to use the RPi as a power sources or to use an external power source if you require higher current.

Features

- 16-bit bidirectional I/O ports
- Three hardware address pins to allow up to 8 Zee PIIO devices on the bus
- 3.3V or 5V input and output logic
- External or Internal power source
- High-speed I2C™ interface
- Configurable interrupt output pins
- External reset input
- Operating voltage: 3.3V to 5.5V
- Works on any 40-PIN Raspberry Pi (B+/2B/Zero/3)
- All PRi PINS remain available for normal use

Absolute Maximum Ratings

Ambient temperature under bias	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on VDD with respect to VSS	-0.3V to +5.5V
Voltage on all other pins with respect to VSS (except VDD)	-0.6V to (VDD + 0.6V)
Total power dissipation	700 mW
Maximum current out of VSS pin*	150 mA
Maximum current into VDD pin*	125 mA
Input clamp current*	±20 mA
Output clamp current*	±20 mA
Maximum output current sunk by any output pin	25 mA
Maximum output current sourced by any output pin	25 mA

* *The current is dependent on the power source*

Part List

Part Number	Part Name	Description	Items
C1	100nF Cap		1
C2, C3	1uF Cap	Enable 3.3V output capability on P2 (optional)	2
R1, R2, R3, R4	100k Resistor		4
JP0, JP1, JP2	2-PIN Header SIL	Jumper headers to change the I2C address: Address range: 0x20 to 0x27	3
JP3	3-PIN Header SIL	Switch between 3.3V or 5V output capability on P2	2
JP4	3-PIN Header SIL	Jumper header to select RPi power or an external power source: E = External I = Internal	
P1	3-PIN Header R/A SIL	Power output and reset header: V = Vout (either 3.3V or 5V depending on JP3) G = Ground R = Reset (Ground RESET pin to reset device)	1
P2	18-PIN Header R/A SIL	16 x I/O pins 2 x Interrupt pins	1
P3	40-PIN Stacking Header DIL	Header connecting to the RPi	1
P4	2-PIN Header R/A SIL	External 5V power source input (optional)	1
U1	IC Socket 28-PIN Narrow		1
U1	MCP-23017		1
U2	MCP-1700-33	Enable 3.3V output capability on P2 (optional)	1
	2-PIN Jumpers	Jumpers for JP0 to JP4	5
	PCB		1
	Software		1

Table 1: Part list

Hardware Installation

- Solder the following components
  - R1 , R2, R3, R4
  - P1, P2, P4
  - U1 Socket (only solder the IC socket. The chip should be installed last.)
  - C1
  - C2, C3 (optional)
  - U2 (optional)
  - JP0, JP1, JP2, JP3, JP4
  - P3. The stackable header pins must be soldered on the top of the board. The stackable header can be difficult to fit through the holes due to the length of the header pins if the pins not being exactly straight.
- Install U1 in the IC socket
- Plug the Zee PIIO into the RPi

PRi Software Installation

- Install the newest version of Raspbian. In a headless environment the Rasbian Lite version is recommended.
- If you optionally want to disable ipv6 execute:

```
if grep -q ipv6.disable /boot/cmdline.txt; then echo IPv6 already disabled; else sudo /bin/sed -i -e 's/$/ ipv6.disable=1 /' /boot/cmdline.txt; fi
```

- Update the system by executing:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get dist-upgrade
```

- Install the required tools for I2C (Complete this step before enabling I2C using raspi-config – see below)

```
sudo apt-get install -y i2c-tools python-smbus wiringpi git
```

- Reboot the RPi

```
sudo reboot
```

- Configure the system using raspi-config

```
sudo raspi-config
```

  - Configure the Password, Hostname, Localisation Options, etc. as required
  - Enable I2C by
    - Select Interfacing Options
    - Select I2C
    - Enable I2C
  - Reboot the RPi once requested to do so

- Install Zee PIIO software from github

```
cd
git clone https://github.com/cactixxx/piio
```

Using the device

The RPi communicates via I2C with the Zee PIIO. I2C is a two wire communication protocol. A single RPi can communicate with up to 8 Zee PIIO devices at the same time. This is achieved by assigning a unique I2C address to each Zee PIIO using JP0, JP1 and JP2.

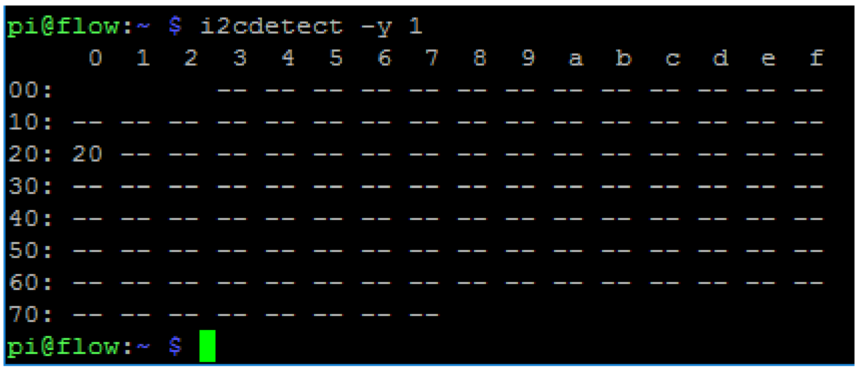
JP2	JP1	JP0	Address
OFF	OFF	OFF	0x20
OFF	OFF	ON	0x21
OFF	ON	OFF	0x22
OFF	ON	ON	0x23
ON	OFF	OFF	0x24
ON	OFF	ON	0x25
ON	ON	OFF	0x26
ON	ON	ON	0x27

Table 2: Board Address

To get the address of the connected I2C devices execute:

```
i2cdetect -y 1
```

With all jumpers (JP0 to JP2) disconnected the output should look as follows:



```
pi@flow:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@flow:~ $
```

Figure 1: Example i2cdetect screen output

This indicates that the I2C device if located at address 0x20.

Configuring I/O Pins

Before the Zee PIIO can be used the I/O pins must be programmed to indicate which ports are inputs and outputs. The Zee PIIO has two ports with 8 pins each. Port A is for pins 1-8 and Port B is for pins 9-16. The ports can be configured with the *i2cset* programme or with the easy to use *piio_init.sh* programme. In this guide the *piio_init.sh* programme will be explained. The *piio_init.sh* is a bash scripts that calls *i2cset.*

The *piio_init.sh* utility has the following command line options:

```
piio_init.sh address [port] [value]
```

The command line options descriptions are as follows:

Switch	Description
address	The address of the Zee PIIO board (e.g. 0x20). This is a mandatory value.
port	Either A or B, where A is pins 1-8 and B is pins 9-16. If no value is given all pins on both Port A and B will be configured as outputs.
value	The hex value to configure the port. The binary equivalent of the Hex digit will either set the pin as an input (1) or an output (0). If no value is given a value of 0x00 will be used. See the tables below for examples and more information.

Table 3: piio_init.sh command line options

Examples of using piio_init.sh:

Command	Description
piio_init.sh 0x20	Set all pins (1-16) as output pins
piio_init.sh 0x20 A	Set Port A pins (1-8) as output pins
piio_init.sh 0x20 B 0xFF	Set Port B pins (9-16) as input pins
piio_init.sh 0x20 A 0x22	Set Port A pins 2 and 6 as input pins, the rest are set as outputs
piio_init.sh 0x20 A 0x05	Set Port A pins 1 and 3 as input pins, the rest are set as outputs

Table 4: Examples of using piio_init.sh

The following is a hex single digit conversion table. Note that a 1 designates an input and 0 an output:

Hex Value	Binary Value	Pin 4	Pin 3	Pin 2	Pin 1
0 (0x00)	0000	0	0	0	0
1 (0x01)	0001	0	0	0	1
2 (0x02)	0010	0	0	1	0
3 (0x03)	0011	0	0	1	1
4 (0x04)	0100	0	1	0	0
5 (0x05)	0101	0	1	0	1
6 (0x06)	0110	0	1	1	0
7 (0x07)	0111	0	1	1	1
8 (0x08)	1000	1	0	0	0
9 (0x09)	1001	1	0	0	1
A (0x0A)	1010	1	0	1	0
B (0x0B)	1011	1	0	1	1
C (0x0C)	1100	1	1	0	0
D (0x0D)	1101	1	1	0	1
E (0x0E)	1110	1	1	1	0
F (0x0F)	1111	1	1	1	1

Table 5: Hex I/O conversion. 1 = Input, 0 = output

Writing to outputs

After the configuration of inputs and outputs using the *piio_init.sh* script the Zee PIIO can be used to switch outputs using the *piio_write.py* application. Help on using the piio_write.py application can be displayed by executing:

```
piio_write.py –h
```

A number of examples are given as part of the -h (or --help) option. As an additional example let's suppose we want to configure all pins on Port A as outputs and we want to switch on pins 1,3,5. The command to achieve this is:

```
cd ~/piio
./piio_init.sh 0x20 A
./piio_write.py –a 0x20 –o 1,3,5
```

Using the command line tools, bash scripts can be created for example:

```
#!/bin/bash

P='/home/pi/piio'                  #Set path to executables
$P/piio_init.sh 0x20 A             #Configure all pins 1-8 as outputs
$P/piio_write.py -a 0x20 -f 0      #Switch all outputs off
while :
do
   $P/piio_write.py -a 0x20 -o 1,3,5 -f 2,4
   sleep 1
   $P/piio_write.py -a 0x20 -o 2,4 -f 1,3,5
   sleep 1
done
```

Reading Inputs

The *piio_read* and *piio_read.py* applications can be used to read data in input pins. Help on using the and *piio_read* and *piio_write.py* application can be displayed by executing:

```
piio_read.py –h          #for help with the Python program
piio_read                #for help with the C program
```

Reading from inputs can be achieved using any of the following:

```
cd ~/piio
./piio_read 0x20                  #Read inputs in hex format (fast C program)
./piio_read.py –a 0x20            #Read all inputs in binary form
./piio_read.py –a 0x20 –p 1,2     #Read pins 1 and 2
./piio_read.py –a 0x20 –v         #Read all pins in verbose format
```