PiFace Digital I/O Documentation

Release 3.1.0

Thomas Preston

April 28, 2016

Contents

1 PiFace Digital **3**
- 1.1 Outputs . . . 3
- 1.2 Inputs . . . 4

2 Installation **5**
- 2.1 Emulator . . . 5

3 Examples **7**
- 3.1 Basic usage . . . 7
- 3.2 Interrupts . . . 8

4 Simple Web Control **9**
- 4.1 Controlling Output . . . 9
- 4.2 Changing Port . . . 9

5 Reference **11**

6 Indices and tables **15**

Python Module Index **17**

The pifacedigitalio Python module provides functions and classes for interacting with PiFace Digital.

Links:

- Blog
- GitHub
- PyPI

Contents:

CHAPTER 1

PiFace Digital

PiFace Digital has eight inputs, eight outputs, eight LED's, two relays and four switches.

1.1 Outputs

The eight output pins are located at the top of the board (near the LEDs). The outputs are open collectors, they can be thought of as switches connecting to ground. This offers greater flexibility so that PiFace Digital can control devices that operate using different voltages. The ninth pin provides 5V for connecting circuits to.

1.1.1 LEDs

The LED's are connected in parallel to each of the outputs. This means that when you set output pin 4 high, LED 4 illuminates.

```
>>> import pifacedigitalio
>>> pifacedigital = pifacedigitalio.PiFaceDigital()
>>> pifacedigital.output_pins[4].turn_on()  # this command does the same thing...
>>> pifacedigital.leds[4].turn_on()  # ...as this command
```

1.1.2 Relays

The two Relays are connected in parallel to output pins 0 and 1 respectively. When you set output pin 0 high, LED 0 illuminates and Relay 0 activates.

```
>>> import pifacedigitalio
>>> pifacedigital = pifacedigitalio.PiFaceDigital()
>>> pifacedigital.output_pins[0].turn_on()  # this command does the same thing...
>>> pifacedigital.leds[0].turn_on()  # ...as this command...
>>> pifacedigital.relays[0].turn_on()  # ...and this command...
```

When activated, the relay bridges the top and middle pins. When deactivated the bottom two are connected.

Relay activated:

```
[Top Pin   ]
[Common    ]
 Bottom Pin
```

Relay Deactivated:

```
 Top Pin
[Common    ]
[Bottom Pin]
```

1.2 Inputs

The eight input pins detect a connection to ground (provided as the ninth pin). The four switches are connected in parallel to the first four input pins. The inputs are *pulled up* to 5V. This can be turned off so that the inputs float.

```
>>> import pifacedigitalio
>>> pifacedigital = pifacedigitalio.PiFaceDigital()
>>>
>>> # without anything pressed
>>> pifacedigital.input_port.value
0
>>> pifacedigital.input_pins[0].value
0
>>> pifacedigital.switches[0].value
0
>>>
>>> # pressing the third switch
>>> pifacedigital.input_port.value
8
>>> bin(pifacedigital.input_port.value)
0b100
>>> pifacedigital.input_pins[2].value  # this command is the same as...
1
>>> pifacedigital.switches[2].value  # ...this command
1
```

CHAPTER 2

Installation

Make sure you are using the lastest version of Raspbian:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Install `pifacedigitalio` (for Python 3 and 2) with the following command:

```
$ sudo apt-get install python{,3}-pifacedigitalio
```

Test by running the `blink.py` program:

```
$ python3 /usr/share/doc/python3-pifacedigitalio/examples/blink.py
```

2.1 Emulator

Install `python3-pifacedigital-emulator` with the following command:

```
$ sudo apt-get install python3-pifacedigital-emulator
```

Start the emulator with:

```
$ pifacedigital-emulator
```

Note: You must be in an X11 session (`startx`).

CHAPTER 3

Examples

3.1 Basic usage

```
>>> import pifacedigitalio

>>> pfd = pifacedigitalio.PiFaceDigital() # creates a PiFace Digtal object

>>> pfd.leds[1].turn_on()     # turn on/set high the second LED
>>> pfd.leds[1].set_high()    # turn on/set high the second LED
>>> pfd.leds[2].toggle()      # toggle third LED
>>> pfd.switches[3].value     # check the logical status of switch3
0
>>> pfd.relays[0].value = 1   # turn on/set high the first relay

>>> pfd.output_pins[6].value = 1
>>> pfd.output_pins[6].turn_off()

>>> pfd.input_pins[6].value
0

>>> pfd.output_port.all_off()
>>> pfd.output_port.value = 0xAA
>>> pfd.output_port.toggle()

>>> bin(pfd.input_port.value)  # fourth switch pressed (logical input port)
'0b1000'

>>> bin(pfd.gpiob.value)  # fourth switch pressed (physical input port)
'0b11110111'

>>> pfd.deinit_board()  # disables interrupts and closes the file
```

Here are some functions you might want to use if objects aren't your thing:

```
>>> import pifacedigitalio as p
>>> p.init()
>>> p.digital_write(0, 1)     # writes pin0 high
>>> p.digital_write(5, 1, 2) # writes pin5 on board2 high
>>> p.digital_read(4)         # reads pin4 (on board0)
0
>>> p.digital_read(2, 3)      # reads pin2 (on board3)
1
>>> p.deinit()
```

3.2 Interrupts

Instead of polling for input we can use the `InputEventListener` to register actions that we wish to be called on certain input events.

```
>>> import pifacedigitalio
>>> def toggle_led0(event):
...     event.chip.leds[0].toggle()
...
>>> pifacedigital = pifacedigitalio.PiFaceDigital()
>>> listener = pifacedigitalio.InputEventListener(chip=pifacedigital)
>>> listener.register(0, pifacedigitalio.IODIR_FALLING_EDGE, toggle_led0)
>>> listener.activate()
```

When input 0 is pressed, led0 will be toggled. To stop the listener, call it's `deactivate` method:

```
>>> listener.deactivate()
```

The `Event` object has some interesting attributes. You can access them like so:

```
>>> import pifacedigitalio
>>> pifacedigital = pifacedigitalio.PiFaceDigital()
>>> listener = pifacedigitalio.InputEventListener(chip=pifacedigital)
>>> listener.register(0, pifacedigitalio.IODIR_RISING_EDGE, print)
>>> listener.activate()
```

This would print out the event informaion whenever you unpress switch 0:

```
interrupt_flag:    0b1
interrupt_capture: 0b11111111
pin_num:           0
direction:         1
chip:              <pifacedigitalio.core.PiFaceDigital object at 0xb682dab0>
timestamp:         1380893579.447889
```

CHAPTER 4

Simple Web Control

You can control PiFace Digital from a web browser (or any network enabled device) using the *simplewebcontrol.py* tool.

You can start the tool by running the following command on your Raspberry Pi:

```
$ python3 /usr/share/doc/python3-pifacedigitalio/examples/simplewebcontrol.py
```

This will start a simple web server on port 8000 which you can access using a web browser.

Type the following into the address bar of a browser on any machine in the local network:

```
http://192.168.1.3:8000
```

Note: Relace `192.168.1.3` with the IP address of your Raspberry Pi.

It will return a JSON object describing the current state of PiFace Digital:

```
{'input_port': 0, 'output_port': 0}
```

4.1 Controlling Output

You can set the output port using the URL:

```
http://192.168.1.61:8000/?output_port=0xaa
```

4.2 Changing Port

You can specify which port you would like `simplewebcontrol.py` to use by passing the port number as the first argument:

```
$ python3 /usr/share/doc/python3-pifacedigitalio/examples/simplewebcontrol.py 12345
```

CHAPTER 5

Reference

Note: Functions and classes in pifacedigitalio.core have been imported into the main namespace. `pifacedigitalio.digital_write()` is the same as *`pifacedigitalio.core.digital_write()`*.

class `pifacedigitalio.core.`**`InputEventListener`**(*chip=None*, *daemon=False*)

Listens for events on the input port and calls the mapped callback functions.

```
>>> def print_flag(event):
...     print(event.interrupt_flag)
...
>>> listener = pifacedigitalio.InputEventListener()
>>> listener.register(0, pifacedigitalio.IODIR_ON, print_flag)
>>> listener.activate()
```

class `pifacedigitalio.core.`**`PiFaceDigital`**(*hardware_addr=0*, *bus=0*, *chip_select=0*, *init_board=True*)

A PiFace Digital board.

Attribute input_pins – list containing `pifacecommon.mcp23s17.MCP23S17RegisterBitNeg`.

Attribute input_port – See `pifacecommon.mcp23s17.MCP23S17RegisterNeg`.

Attribute output_pins – list containing `pifacecommon.mcp23s17.MCP23S17RegisterBit`.

Attribute output_port – See `pifacecommon.mcp23s17.MCP23S17Register`.

Attribute leds – list containing `pifacecommon.mcp23s17.MCP23S17RegisterBit`.

Attribute relays – list containing `pifacecommon.mcp23s17.MCP23S17RegisterBit`.

Attribute switches – list containing `pifacecommon.mcp23s17.MCP23S17RegisterBit`.

Example:

```
>>> pfd = pifacedigitalio.PiFaceDigital()
>>> pfd.input_port.value
0
>>> pfd.output_port.value = 0xAA
>>> pfd.leds[5].turn_on()
```

`pifacedigitalio.core.`**`deinit`**(*bus=0*, *chip_select=0*)

Stops interrupts on all boards. Only required when using *`digital_read()`* and *`digital_write()`*.

Parameters

- **`bus`** (*`int`*) – SPI bus /dev/spidev<bus>.<chipselect> (default: {bus})

- **chip_select** (*int*) – SPI chip select /dev/spidev<bus>.<chipselect> (default: {chip})

`pifacedigitalio.core.digital_read`(*pin_num*, *hardware_addr=0*)

Returns the value of the input pin specified.

Note: This function is for familiarality with users of other types of IO board. Consider accessing the `input_pins` attribute of a PiFaceDigital object:

```
>>> pfd = PiFaceDigital(hardware_addr)
>>> pfd.input_pins[pin_num].value
0
```

Parameters

- **pin_num** (*int*) – The pin number to read.
- **hardware_addr** (*int*) – The board to read from (default: 0)

Returns int – value of the pin

`pifacedigitalio.core.digital_read_pullup`(*pin_num*, *hardware_addr=0*)

Returns the value of the input pullup specified.

Note: This function is for familiarality with users of other types of IO board. Consider accessing the `gppub` attribute of a PiFaceDigital object:

```
>>> pfd = PiFaceDigital(hardware_addr)
>>> hex(pfd.gppub.value)
0xff
>>> pfd.gppub.bits[pin_num].value
1
```

Parameters

- **pin_num** (*int*) – The pin number to read.
- **hardware_addr** (*int*) – The board to read from (default: 0)

Returns int – value of the pin

`pifacedigitalio.core.digital_write`(*pin_num*, *value*, *hardware_addr=0*)

Writes the value to the input pin specified.

Note: This function is for familiarality with users of other types of IO board. Consider accessing the `output_pins` attribute of a PiFaceDigital object:

```
>>> pfd = PiFaceDigital(hardware_addr)
>>> pfd.output_pins[pin_num].value = 1
```

Parameters

- **pin_num** (*int*) – The pin number to write to.
- **value** (*int*) – The value to write.
- **hardware_addr** (*int*) – The board to read from (default: 0)

`pifacedigitalio.core.`**`digital_write_pullup`**(*pin_num*, *value*, *hardware_addr=0*)

Writes the value to the input pullup specified.

Note: This function is for familiarality with users of other types of IO board. Consider accessing the `gppub` attribute of a PiFaceDigital object:

```
>>> pfd = PiFaceDigital(hardware_addr)
>>> hex(pfd.gppub.value)
0xff
>>> pfd.gppub.bits[pin_num].value = 1
```

Parameters

- **`pin_num`** (*int*) – The pin number to write to.
- **`value`** (*int*) – The value to write.
- **`hardware_addr`** (*int*) – The board to read from (default: 0)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

p

pifacedigitalio.core, 11

D

deinit() (in module pifacedigitalio.core), 11
digital_read() (in module pifacedigitalio.core), 12
digital_read_pullup() (in module pifacedigitalio.core), 12
digital_write() (in module pifacedigitalio.core), 12
digital_write_pullup() (in module pifacedigitalio.core), 13

I

InputEventListener (class in pifacedigitalio.core), 11

P

PiFaceDigital (class in pifacedigitalio.core), 11
pifacedigitalio.core (module), 11