

Module13.2 PPS

Example

Continuously read and print the output current, output voltage and MCU temperature (via serial port)

The diagram shows a Scratch-style block structure for the PPS module example. It starts with a 'Setup' section containing four blocks: 'Init device I2C address' with value '0x35' (range '0x01 ~ 0x7f'), 'Set output' with dropdown 'ENABLE', 'Set output voltage' with value '5.5' (range 'V, 0 ~ 30'), and 'Set output current' with value '1' (range 'A, 0 ~ 5'). This is followed by a 'Loop' section containing three 'print' blocks, each concatenated with a 'Get' block: 'output current: ' + 'Get output current (A, return float)', 'output voltage: ' + 'Get output voltage (V, return float)', and 'MCU temperature: ' + 'Get MCU temperature (°C, return float)'.

```
from m5stack import *
from m5stack_ui import *
from uiflow import *
import module

screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0xFFFFFF)

pps = module.get(module.PPS)

pps.init_i2c_address(0x35)
pps.setOutput(True)
pps.setOutputVoltage(5.5)
pps.setOutputCurrent(1)
while True:
    print((str('output current:') + str((pps.readOutputCurrent()))))
    print((str('output voltage:') + str((pps.readOutputVoltage()))))
    print((str('MCU temperature:') + str((pps.readMcuTemperature()))))
    wait_ms(2)
```

API

Get current I2C address (return int)

```
pps.getI2CAddress()
```

- Get the current device's I2C address

Init device I2C address `0x35` (0x01 ~ 0x7f)

```
pps.init_i2c_address(0x35)
```

- Initialize the device's I2C address

Get data update flag (return int)

```
pps.readDataUpdateFlag()
```

- Get the data update flag, returns an integer. This flag is usually used to indicate whether the data has been updated

Get input voltage (V, return float)

```
pps.readInputVoltage()
```

- Get the input voltage value in volts (V), returns a float. This value represents the currently measured input voltage

Get MCU temperature (°C, return float)

```
pps.readMcuTemperature()
```

- Get the MCU temperature value in degrees Celsius (°C), returns a float. This value represents the currently measured microcontroller temperature

Get module ID (return int)

```
pps.readModuleId()
```

- Get the module ID, returns an integer. This ID is used to uniquely identify the device or module

Get output current (A, return float)

```
pps.readOutputCurrent()
```

- Get the output current value in amperes (A), returns a float. This value represents the currently measured output current

Get output voltage (V, return float)

```
pps.readOutputVoltage()
```

- Get the output voltage value in volts (V), returns a float. This value represents the currently measured output voltage

Get PSU running mode (return int)

```
pps.readPsuRunningMode()
```

- Get the power module's running mode, returns an integer. This is usually used to indicate the current operating status of the power module

Get unique identifier (UID) (return bytearray)

```
pps.readUID()
```

- Get the unique identifier (UID), returns a byte array. The UID is used to uniquely identify the device or module

Set I2C address (0x01 ~ 0x7f)

```
pps.setI2CAddress()
```

- Set the device's I2C address

Set output

```
pps.setOutput(True)
```

- Set the output state. This function is usually used to control the device's output, such as turning a feature or module on or off

Set output current (A, 0 ~ 5)

```
pps.setOutputCurrent(1)
```

- Set the output current, range is 0 to 5 amperes. This is used to control the device's output current

Set output voltage (V, 0 ~ 30)

```
pps.setOutputVoltage(5.5)
```

- Set the device's output voltage. The range is 0 to 30 volts, used to adjust the device's output voltage value, controlling power supply or voltage level