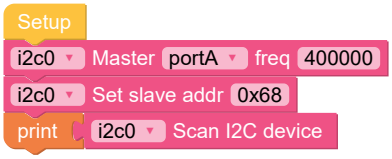
# I2C Master

## Example

Scan I2C device addresses and print them to the serial port.

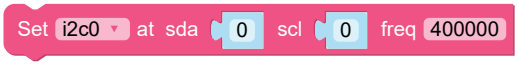



```
from m5stack import *
from m5stack_ui import *
from uiflow import *
import i2c_bus

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

i2c0 = i2c_bus.easyI2C(i2c_bus.PORTA, 0x00, freq=400000)
i2c0.addr = 0x68
print(i2c0.scan())
```

## API


```
i2c0 = i2c_bus.easyI2C(i2c_bus.PORTA, 0x00, freq=400000)
```

- Initialize an I2C bus on PORTA and set the communication frequency.

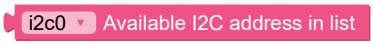



```
i2c0 = i2c_bus.easyI2C((0, 0), 0x00, freq=400000)
```

- Initialize an I2C bus on the specified GPIO pins, set the device address and communication frequency.


```
i2c0.addr = 0x68
```

- Set the I2C communication address.




```
str(i2c0.available())
```

- Check if devices are available on the I2C bus and convert the result to a string.


```
str(i2c0.scan())
```

- Scan for all I2C devices on the bus and convert the list of found device addresses to a string.


```
str(i2c0.read_u8(i2c0.scan()))
```

- Read one byte of data from the scanned I2C device.


```
str(i2c0.read_u16(0x00, byteorder="big"))
```

- Read 16-bit data from the specified I2C device address (in big-endian or little-endian order) and convert it to a string.


```
str(i2c0.read_reg(0x00, 0))
```

- Read a specified number of bytes from the specified I2C device address and register, and convert the data to a string.


```
str(i2c0.read_mem_data(0, 0, i2c_bus.UINT8LE))
```

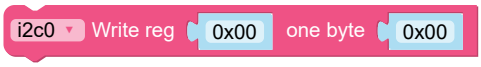
- Read data of a specified type from the specified I2C device address and register, and convert it to a string. Data types can be UINT8LE, UINT16LE, UINT32LE, etc.


```
str(i2c0.read_data(0, i2c_bus.UINT8LE))
```

- Read data of a specified type from the specified I2C device address and convert it to a string. Data types can be UINT8LE, UINT16LE, UINT32LE, etc.

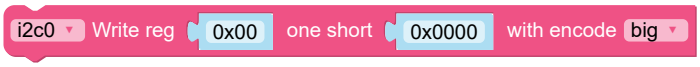

```
str([][0])
```

- Get the element at the specified index from a list and convert it to a string.




```
i2c0.write_u8(0x00, 0x00)
```

- Write one byte of data to the specified I2C device register.




```
i2c0.write_u16(0x00, 0x0000, byteorder="big")
```

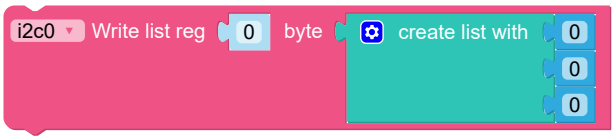
- Write 16-bit data to the specified I2C device register with big-endian or little-endian byte order.


```
i2c0.write_mem_data(0, 0, i2c_bus.UINT8LE)
```

- Write data of type UINT8LE to the register at address 0 of the I2C device. The data type can be UINT8LE, UINT16LE, UINT32LE, etc.


```
i2c0.write_data(0, i2c_bus.UINT8LE)
```

- Write data of a specified type to the I2C device.




```
i2c0.write_mem_list(0, [0, 0, 0])
```

- Write a list of byte data to the specified I2C device register.