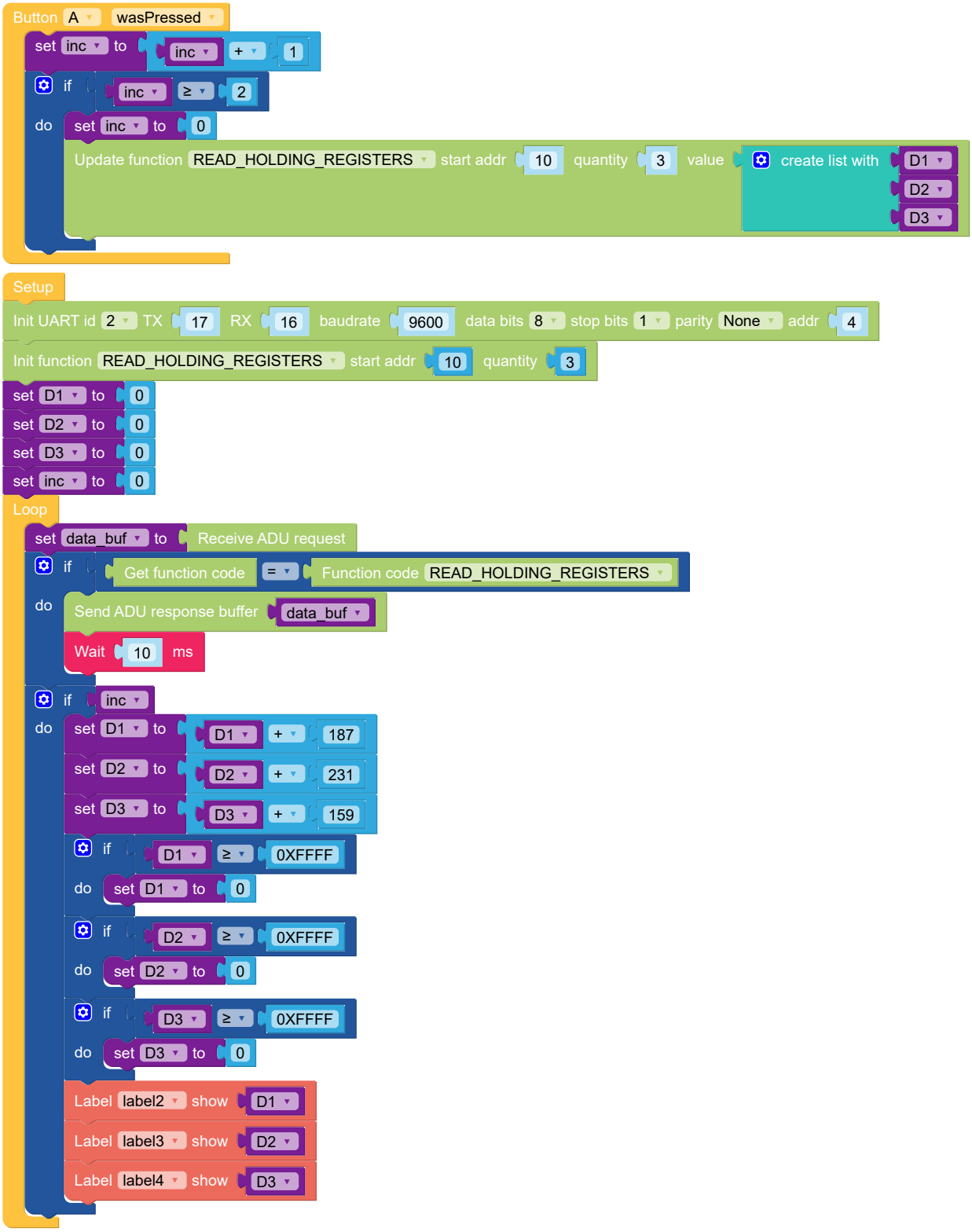
# Modbus Slave

## Example

The M5Stack device, as a Modbus RTU slave, provides three continuously increasing 16 bit register values (D1/D2/D3) to the master station through function code 03 (read hold register), and can trigger data updates through button A.




```
from m5stack import *
from m5ui import *
```

```
from uiflow import *
from modbus.slave.rtu import ModbusSlave
import time

setScreenColor(0x222222)

inc = None
D1 = None
D2 = None
D3 = None
data_buf = None

title0 = M5Title(title="Slave Read Holding Register", x=62, fgcolor=0xFFFFFF, bgcolor=0xff3d00)
label0 = M5TextBox(16, 112, "From Slave to Master Send Register Data ", lcd.FONT_Default, 0xFFFFFF, rotate=0
label1 = M5TextBox(41, 209, "Press A", lcd.FONT_Default, 0xFFFFFF, rotate=0)
label2 = M5TextBox(24, 55, "Text", lcd.FONT_DejaVu18, 0xFFFFFF, rotate=0)
label3 = M5TextBox(139, 55, "Text", lcd.FONT_DejaVu18, 0xFFFFFF, rotate=0)
label4 = M5TextBox(257, 55, "Text", lcd.FONT_DejaVu18, 0xFFFFFF, rotate=0)

def buttonA_wasPressed():
  global inc, D1, D2, D3, data_buf
  inc = inc + 1
  if inc >= 2:
    inc = 0
    modbus_s.update_process(3, 10, 3, [D1, D2, D3])
  pass
btnA.wasPressed(buttonA_wasPressed)


modbus_s = ModbusSlave(2, tx=17, rx=16, baudrate=9600, data_bits=8, stop_bits=1, parity=None, slaveID=4)
modbus_s.function_init(3, 10, 3)
D1 = 0
D2 = 0
D3 = 0
inc = 0
while True:
  data_buf = modbus_s.receive_req_create_pdu()
  if (modbus_s.find_function) == (3):
    modbus_s.create_slave_response(data_buf)
    wait_ms(10)
  if inc:
    D1 = D1 + 187
    D2 = D2 + 231
    D3 = D3 + 159
    if D1 >= 0XFFFF:
      D1 = 0
    if D2 >= 0XFFFF:
      D2 = 0
    if D3 >= 0XFFFF:
      D3 = 0
    label2.setText(str(D1))
    label3.setText(str(D2))
    label4.setText(str(D3))
  wait_ms(2)
```

## API

Function code READ_COILS_STATUS

```
print((str('code:') + str((1))))
```

- Obtain function code

Get address

```
print(modbus_s.find_address)
```

- Get device address

Get function code

```
print((str('code:') + str((modbus_s.find_function))))
```

- Obtain function code

Get quantity

```
print(modbus_s.find_quantity)
```

- Obtain data volume

Init UART id 1 TX 17 RX 16 baudrate 9600 data bits 7 stop bits 1 parity None addr 1

```
modbus_s = ModbusSlave(1, tx=17, rx=16, baudrate=9600, data_bits=7, stop_bits=1, parity=None, slaveID=1)
```

- Initialize UART parameters
  - Baud rate
  - Data bits
  - Checksum
  - Equipment address

Init function READ_COILS_STATUS start addr 0 quantity 0

```
modbus_s.function_init(1, 0, 0)
```

- Initialize coil status

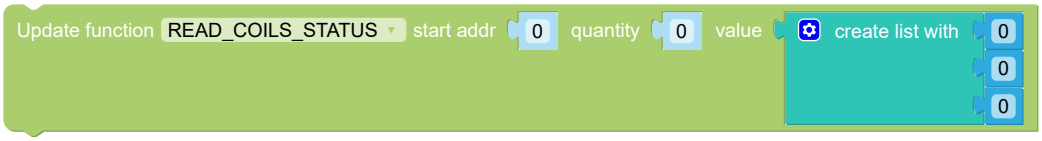
Receive ADU request

```
print((str('ADU:') + str((modbus_s.receive_req_create_pdu()))))
```

- Receive ADU request frames


```
modbus_s.create_slave_response()
```

- Send ADU response buffer




```
modbus_s.update_process(1, 0, 0, [0, 0, 0])
```

- Update functional parameters