GPPU mini-school

basic slow control methods
with a single board computer

Sho Nagao

2021/04/29

Goal of this school

Making a simple positron emission tomography (PET) with cheap computers.
Learning very basic slow control methods.

Introduction

- Slow control
- Single-board computer
- Python

Topics

- LED operation (GPIO)
- Counter (GPIO)
- Thermometer (SPI)
- Power Supply (LAN)
- Stage control (RS-232C)

Schedule

1st day

Introduction

2nd

LED operation（Rm. 637）

3rd

Other operations（Rm. 637）

4th

Making slide & 5-10 min speach

Slow Control

A slow control system plays an important role in experiments. Operation and monitor of apparatus and writing logs are necessary.

Examples of the slow control

- Voltage, Current monitor
- Temperature, Humidity, Pressure monitor
- Position, Angle monitor
- PC storage monitor
- Camera
- Open-Close bend
- Making log
- etc...

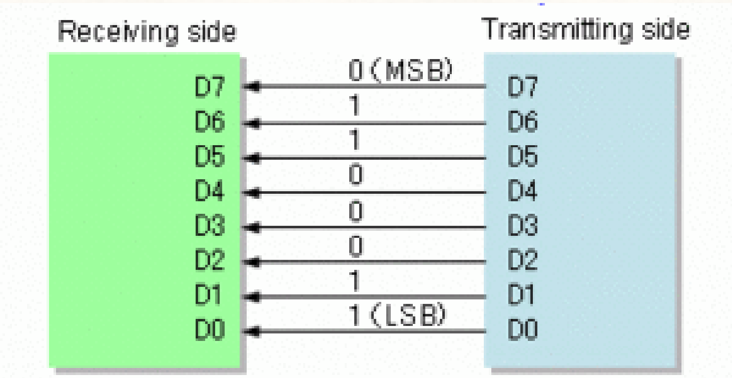
Serial Communication Interface

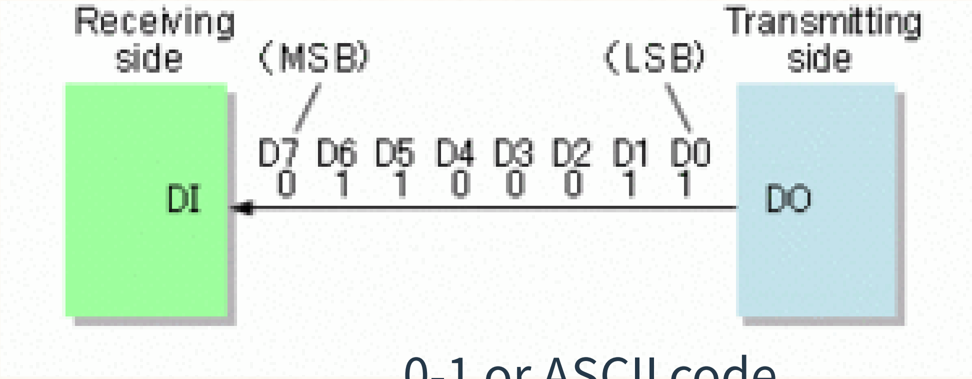
Serial vs. Parallel

Parallel link is used as a communication interface in the first age of PC communication.

Serial link becomes a standard communication because of faster com., less cable, less cross-talk

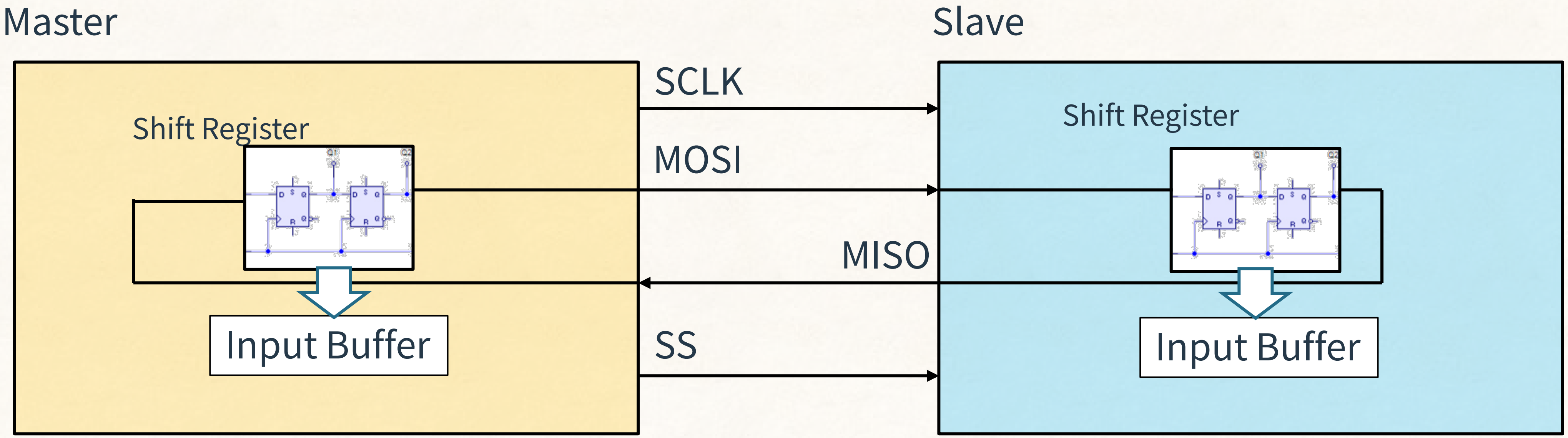
There are many serial communication specifications (SPI, I2C, UART, RS-232, RS-485, USB, Ethernet, SATA, PCI-express)





0-1 or ASCII code

SPI (Serial Peripheral Interface)



SPI is one of the simplest serial communication interface.

SPI is used for short distance (~1 m) communication. SPI is often used the communication between chips on a board.

(8 bit) data transfer.

Similar serial bus: I2C .

RS-232C

RS-232C is one of the serial communication interface.

Many apparatuses have RS-232C port to handle.

Maximum transmitted rate is 20 kbps.

Maximum cable length is ～15m.

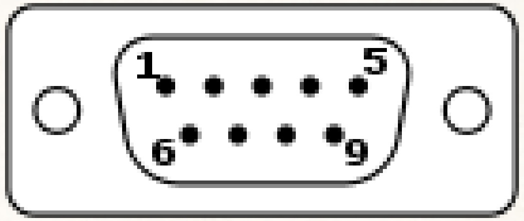
High-Level: -3～-25V

Low-Level: +3～+25V

Control of the communication rate (Baud rate) is necessary.

Similar interface: RS-422, RS-485

D-Sub 9 pins



Pin No.	Name	I/O	Detail
1	DCD	IN	Data Carrier Detect
2	RxD	IN	Received data
3	TxD	OUT	Transmitted data
4	DTR	OUT	Data terminal ready
5	GND	-	Common GND
6	DSR	IN	Data set ready
7	RTS	OUT	Request to send
8	CTS	IN	Clear to send
9	RI	IN	Ring indicator

Single-board Computer

A single-board computer is a complete computer built on a single circuit board with CPU, memory, storage, I/O etc...

Operation systems work on the SBC.

SBC is one of the most convenient tools developing a control system, IoT etc. with low cost.

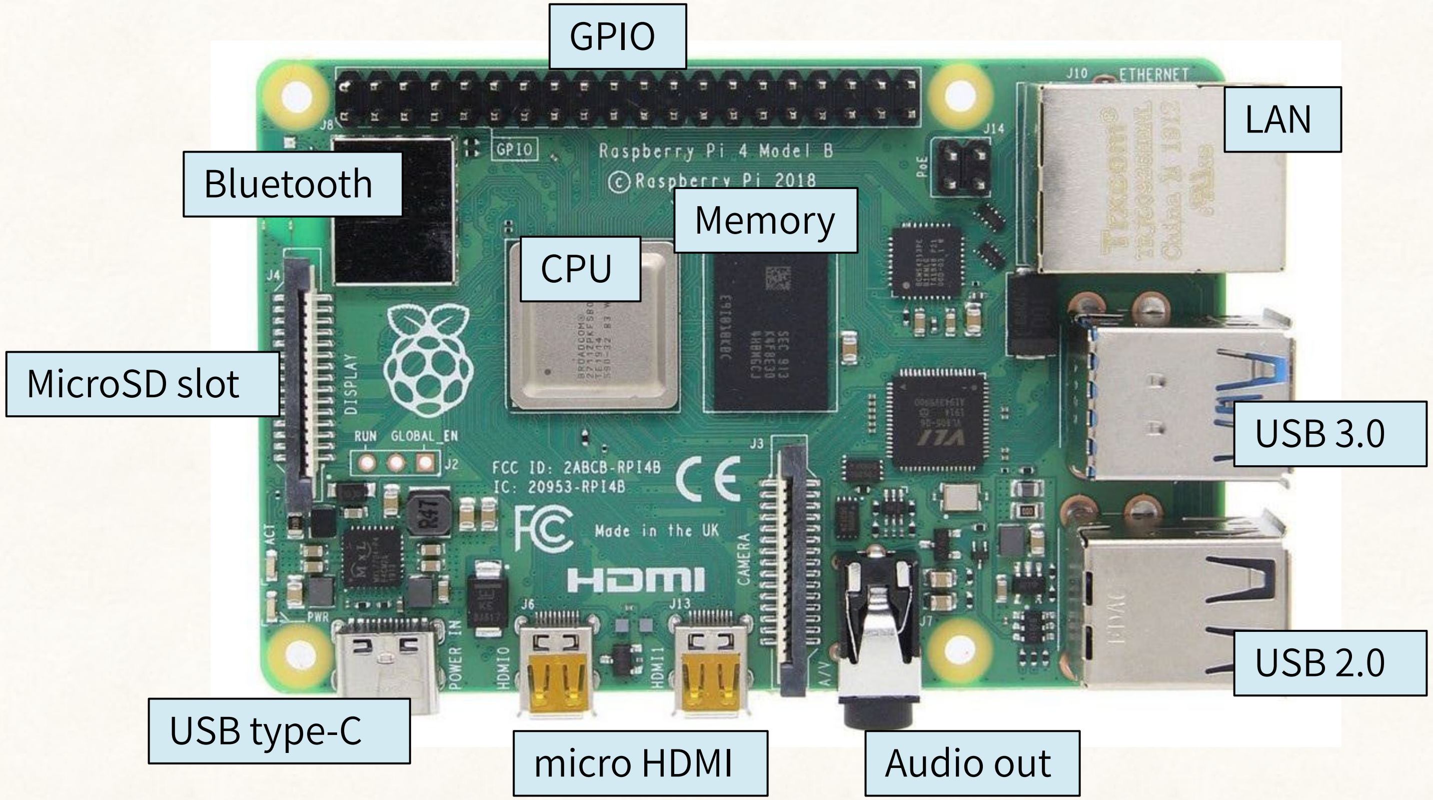
Species of SBC

- Raspberry pi (Raspberry pi foundation)
- Arduino (Arduino)
- Banana pi (Lemaker)
- Galileo (Intel)
- Others !

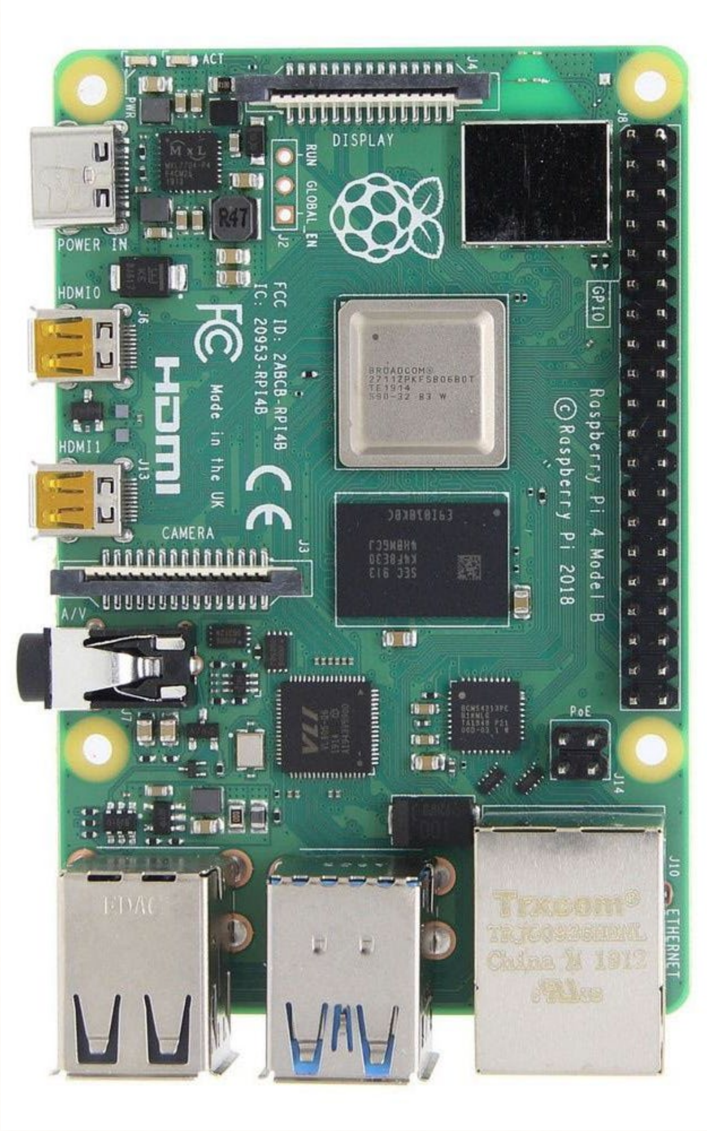
Raspberry Pi

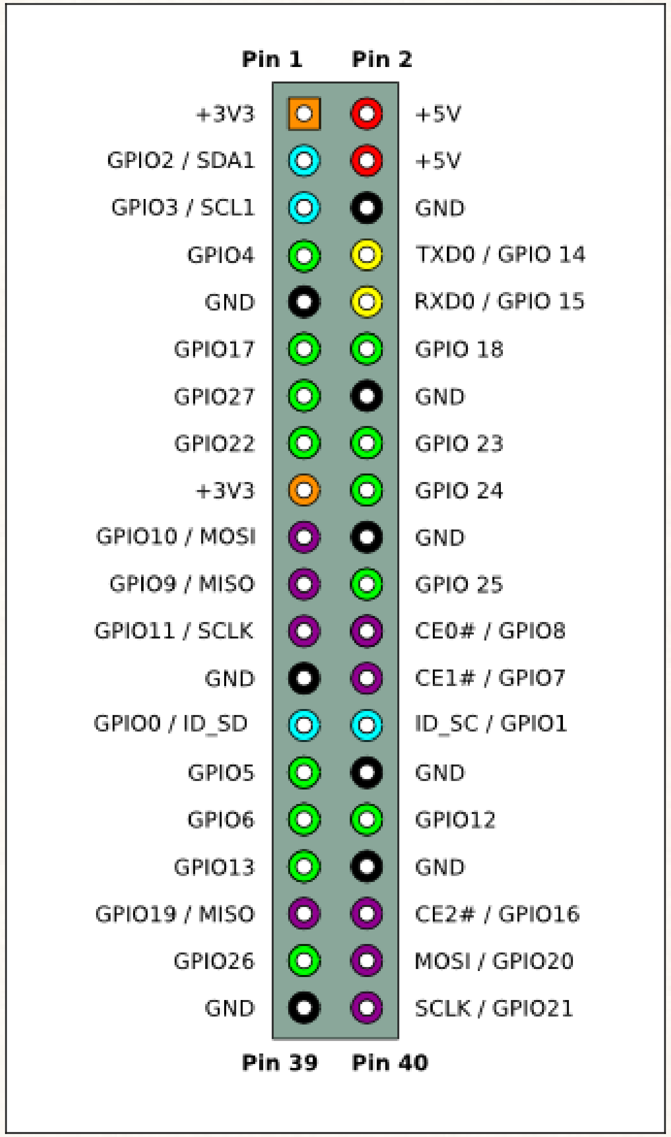
A raspberry pi is one of the most popular SBC. The raspberry pi opened the door of IoT after the first board launched in 2012.

- Species
    - Raspberry pi Zero W
        - $10
        - 1 GHz, single core
        - 512 MB
    - Raspberry pi 3 model B+
        - $35
        - 1.4 GHz, quad core
        - 1 GB
    - Raspberry pi 4 model B
        - $55
        - 1.5 GHz, quad core
        - 1〜4 GB



GPIO (General Purpose Input/Output)





	Pin 1	Pin 2	
+3V3	●	●	+5V
GPIO2 / SDA1	●	●	+5V
GPIO3 / SCL1	●	●	GND
GPIO4	●	●	TXD0 / GPIO 14
GND	●	●	RXD0 / GPIO 15
GPIO17	●	●	GPIO 18
GPIO27	●	●	GND
GPIO22	●	●	GPIO 23
+3V3	●	●	GPIO 24
GPIO10 / MOSI	●	●	GND
GPIO9 / MISO	●	●	GPIO 25
GPIO11 / SCLK	●	●	CE0# / GPIO8
GND	●	●	CE1# / GPIO7
GPIO0 / ID_SD	●	●	ID_SC / GPIO1
GPIO5	●	●	GND
GPIO6	●	●	GPIO12
GPIO13	●	●	GND
GPIO19 / MISO	●	●	CE2# / GPIO16
GPIO26	●	●	MOSI / GPIO20
GND	●	●	SCLK / GPIO21
	Pin 39	Pin 40	

There are a 40-pin GPIO header on Raspberry Pi boards.

Two +5V, another two +3.3V pins are present on the board.

GPIO pins designed as output pins of +3.3V as high-level and 0V as low-level.

These also designed as input pins.

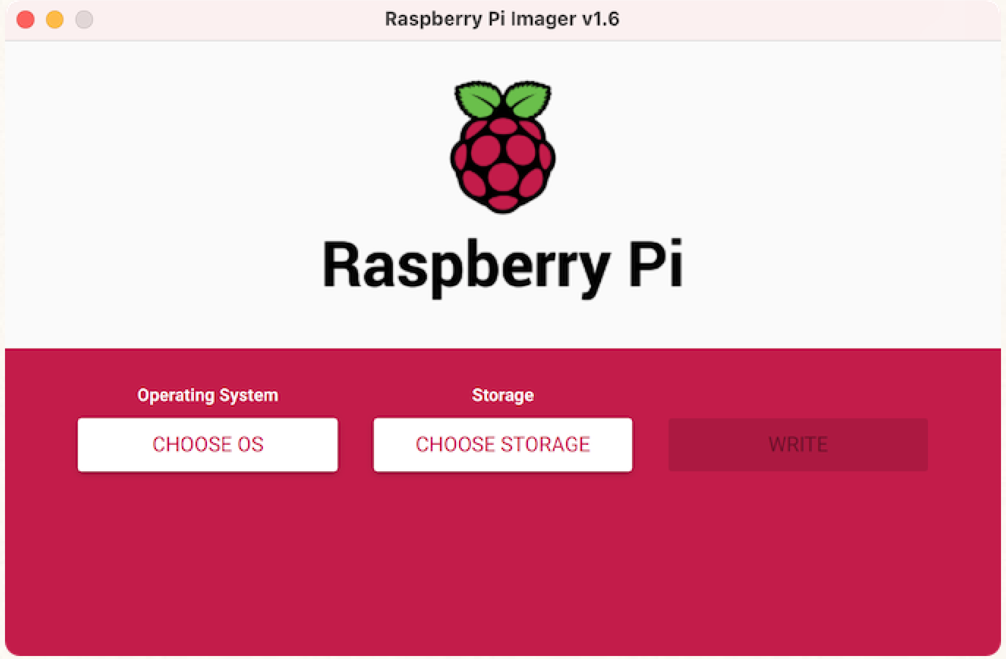
As well as simple I/O pins, GPIO pins can be used with several communication interface (PWM, SPI, I2C, RS, Serial)

Operating System

Raspberry pi doesn't operate the full operating system (e.g. Windows, RHEL) due to the limited machine power.

Raspberry Pi foundation have developed special operating system "Raspberry PI OS (Raspbian)" which based on Debian OS.

Other OS is also available, for example Windows10 IoT and Ubuntu MATE etc., though these are not full OS.



Python

Python is a high-level programing language. Simpler and less coding than C language are available. As there are many libraries from system operation to graphical user interface, Python becomes one of the most popular language in the world.

Standard output

```
print('Hello World')
print(val)
print(val, val)
print('1+1=%d' % 2)
```

Variables

```
i = 10
str = 'hello'
color = ['R', 'G', 'B']
```

If

```
if i == 0:
    i += 1
elif i <10:
    i += 0.5
else:
    pass
```

For

```
for i in range(10):
    print(i)
```

Exercise

LED Operation

7-seg LED operation

Thermometer

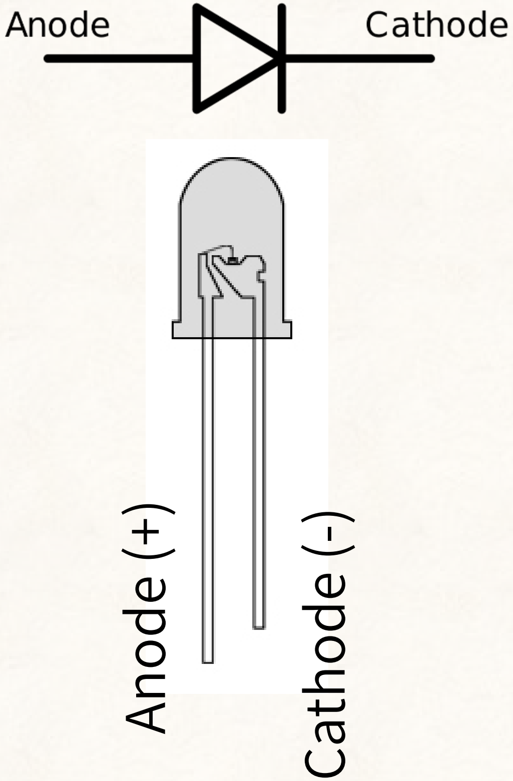
Power Supply

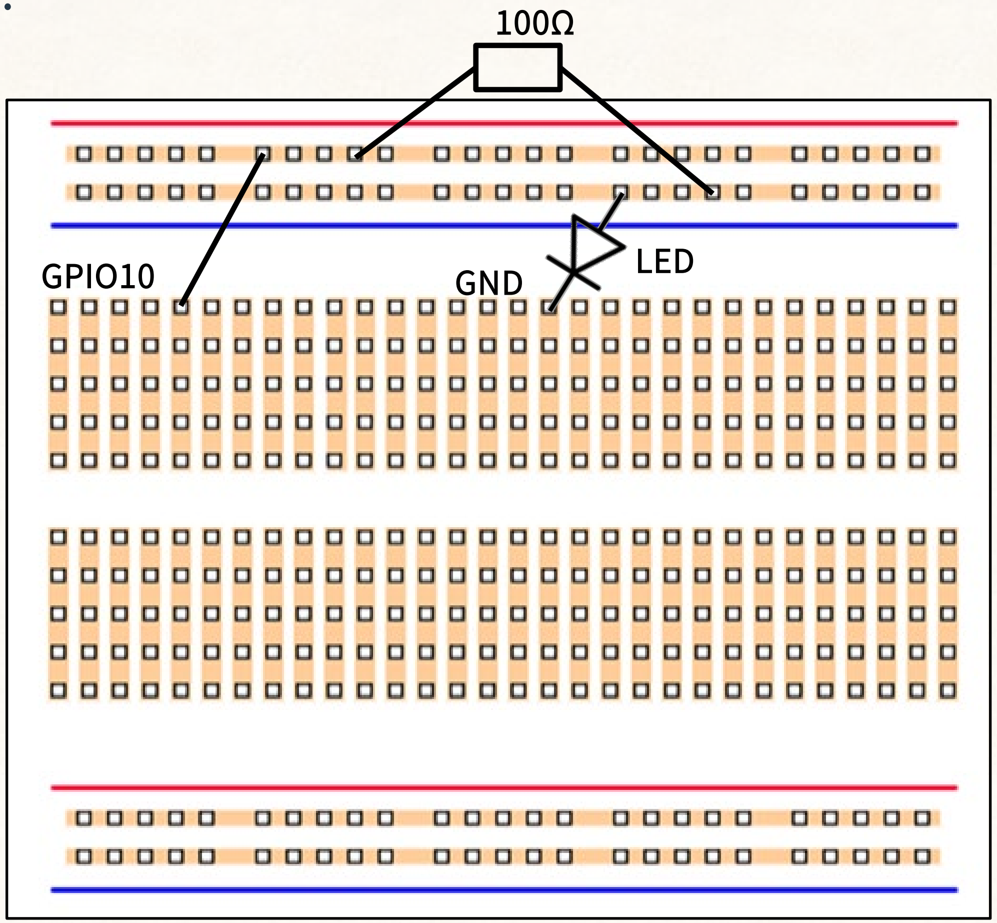
Stage

LED Operation

GPIO10 pin (Low: 0V, High: 3.3V) uses operating LED.

100Ω register inserts to control the current.



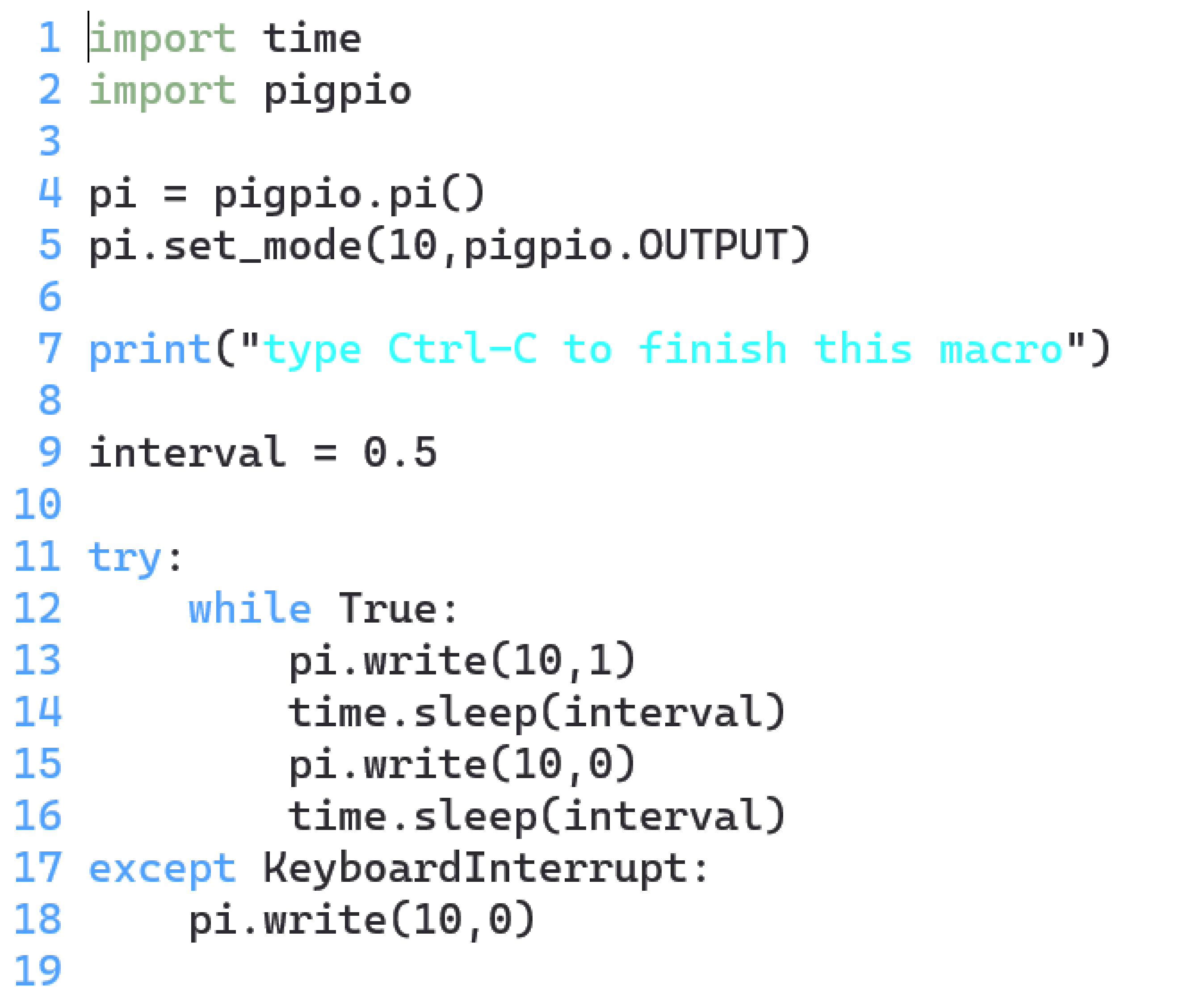


LED Operation with Python

1. Access to your raspberry pi with ssh.
2. Type commands

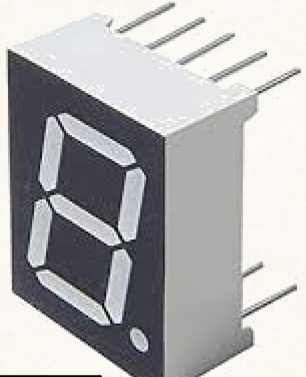
```
$ python
>> import pigpio
>> pi = pigpio.pi()
>> pi.set_mode(10, pigpio.OUTPUT)
>> pi.write(10, 1)
>> pi.write(10, 0)
```

led.py



```python
import time
import pigpio

pi = pigpio.pi()
pi.set_mode(10,pigpio.OUTPUT)

print("type Ctrl-C to finish this macro")

interval = 0.5

try:
    while True:
        pi.write(10,1)
        time.sleep(interval)
        pi.write(10,0)
        time.sleep(interval)
except KeyboardInterrupt:
    pi.write(10,0)

```

7-segment LED

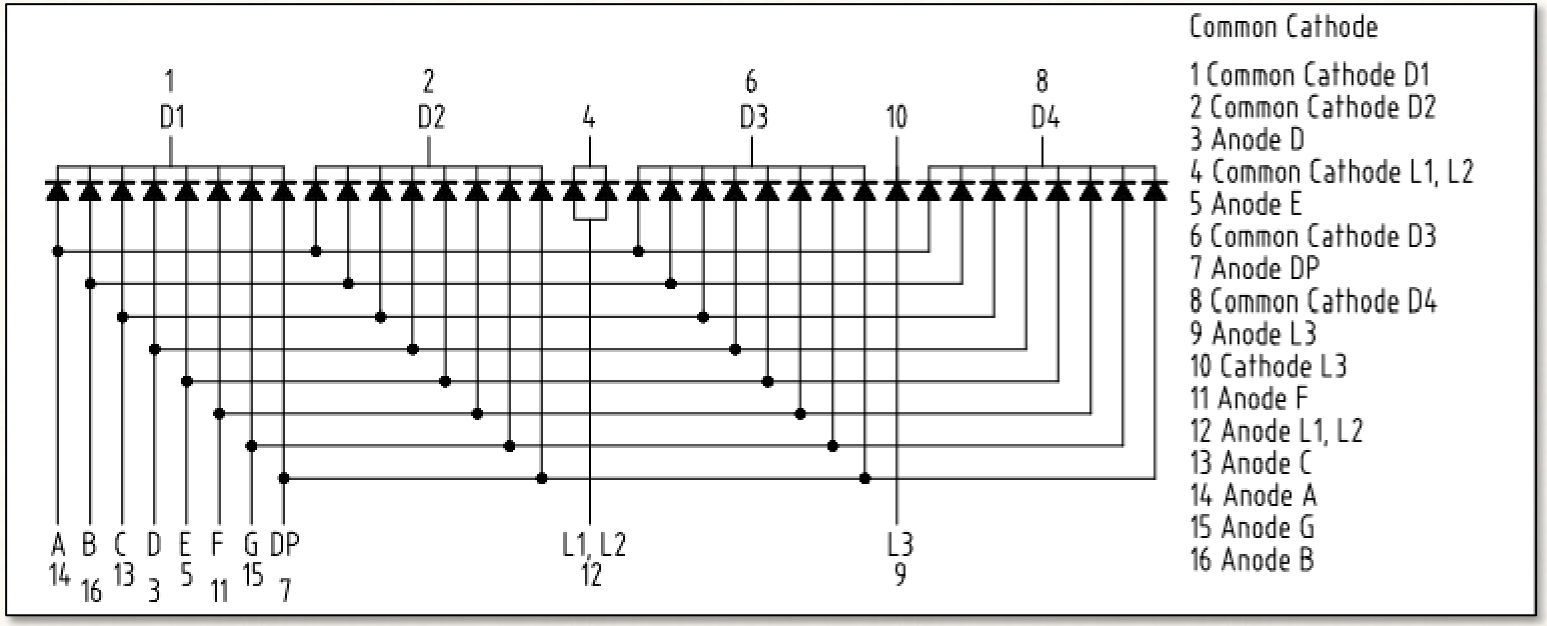


Shall we learn quicker GPIO operation with a 7-seg LED.

We will use 4-digits 7-segment LED.

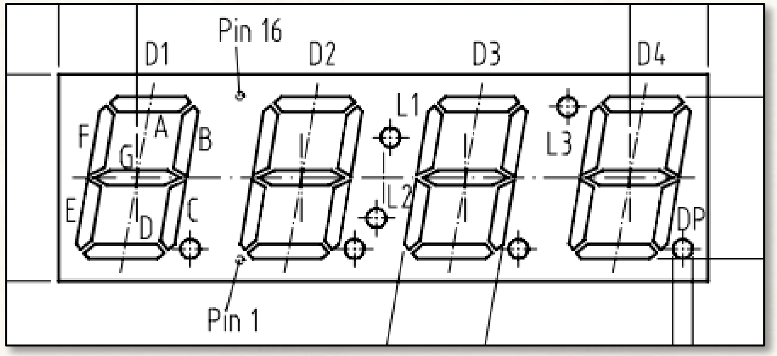
Connection

GPIO pin	LED pin
5	1
12	2
13	3
6	5
16	6
18	8
21	11
23	13
24	14
25	15
26	16



Common Cathode

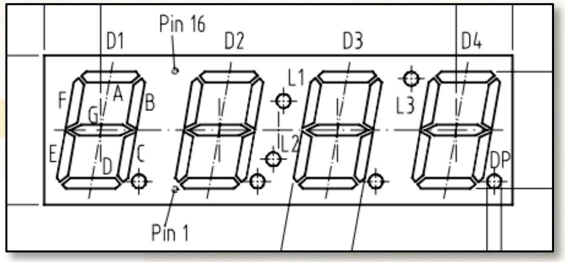
1 Common Cathode D1
2 Common Cathode D2
3 Anode D
4 Common Cathode L1, L2
5 Anode E
6 Common Cathode D3
7 Anode DP
8 Common Cathode D4
9 Anode L3
10 Cathode L3
11 Anode F
12 Anode L1, L2
13 Anode C
14 Anode A
15 Anode G
16 Anode B

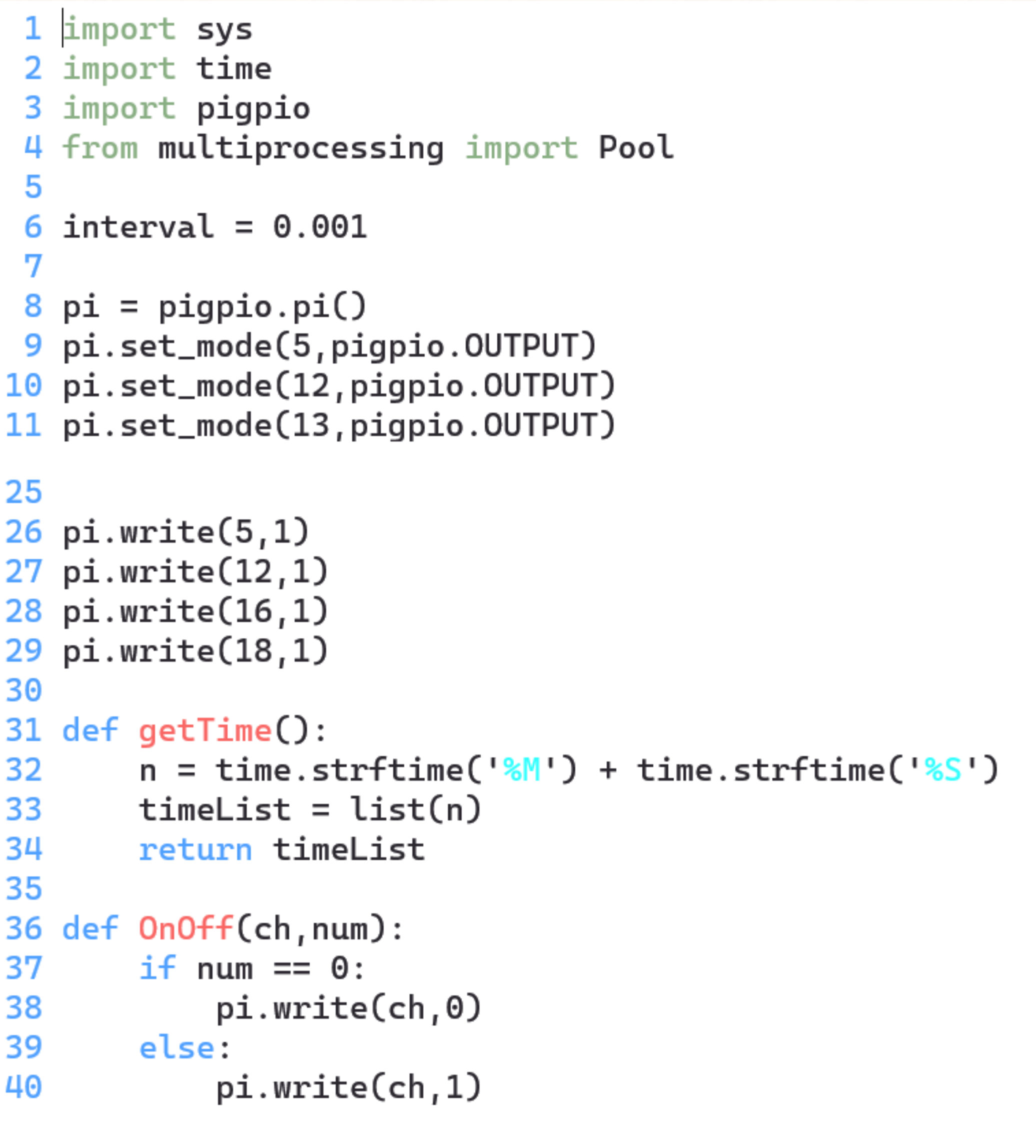


Check an operation with a macro.

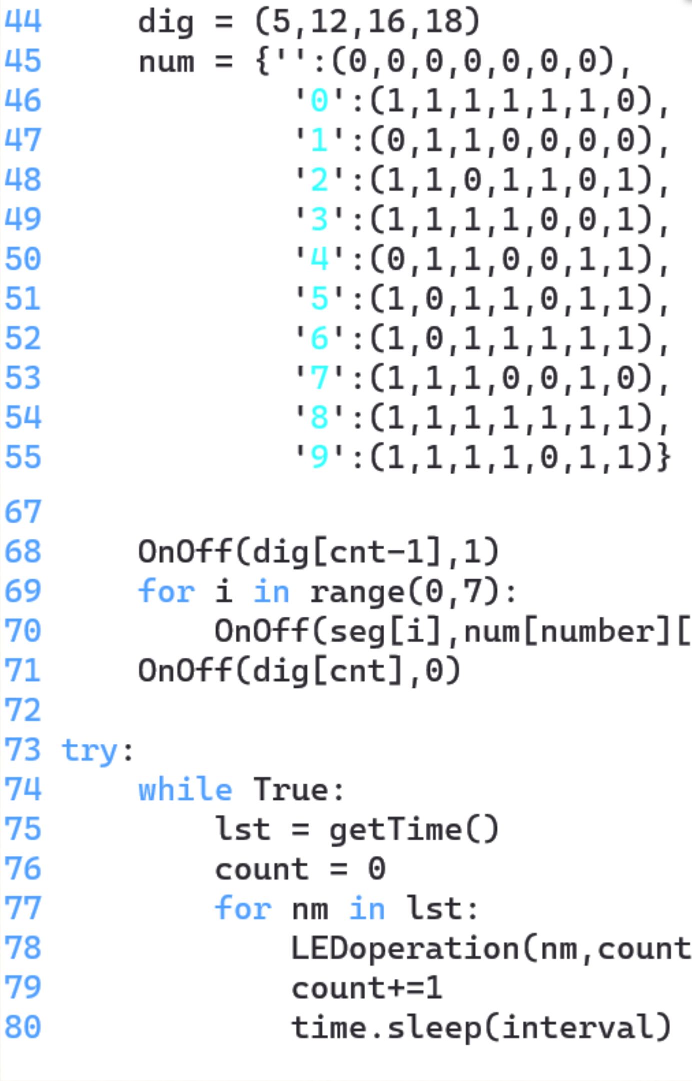
```
$ python 7-seg.py
```

7seg.py





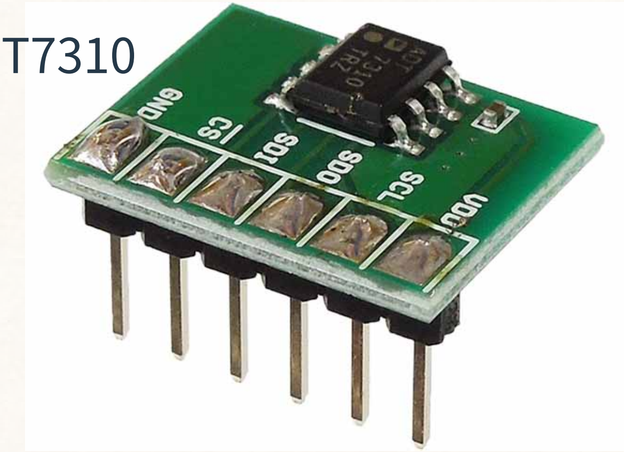
```python
import sys
import time
import pigpio
from multiprocessing import Pool

interval = 0.001

pi = pigpio.pi()
pi.set_mode(5,pigpio.OUTPUT)
pi.set_mode(12,pigpio.OUTPUT)
pi.set_mode(13,pigpio.OUTPUT)

pi.write(5,1)
pi.write(12,1)
pi.write(16,1)
pi.write(18,1)

def getTime():
    n = time.strftime('%M') + time.strftime('%S')
    timeList = list(n)
    return timeList

def OnOff(ch,num):
    if num == 0:
        pi.write(ch,0)
    else:
        pi.write(ch,1)
```



```python
def LEDoperation(number, cnt):
    seg = (24,26,23,13,6,21,25)
    dig = (5,12,16,18)
    num = {'':(0,0,0,0,0,0,0),
           '0':(1,1,1,1,1,1,0),
           '1':(0,1,1,0,0,0,0),
           '2':(1,1,0,1,1,0,1),
           '3':(1,1,1,1,0,0,1),
           '4':(0,1,1,0,0,1,1),
           '5':(1,0,1,1,0,1,1),
           '6':(1,0,1,1,1,1,1),
           '7':(1,1,1,0,0,1,0),
           '8':(1,1,1,1,1,1,1),
           '9':(1,1,1,1,0,1,1)}


    OnOff(dig[cnt-1],1)
    for i in range(0,7):
        OnOff(seg[i],num[number][i])
    OnOff(dig[cnt],0)

try:
    while True:
        lst = getTime()
        count = 0
        for nm in lst:
            LEDoperation(nm,count)
            count+=1
            time.sleep(interval)
```

Thermometer

ADT7310



16-bit (0.0078°C) or 13-bit (0.0625°C) temp sensor

GPIO pin	LED pin
+5 or +3.3V	VDD
GPIO11 (SPI_CLK)	SCL
GPIO9 (SPI_MOSI)	SDD
GPIO10 (SPI_MISO)	SDI
GPIO8 (SPI_CE0)	CS
GND	GND

Data-Sheet

https://www.analog.com/media/en/technical-documentation/data-sheets/ADT7310.pdf

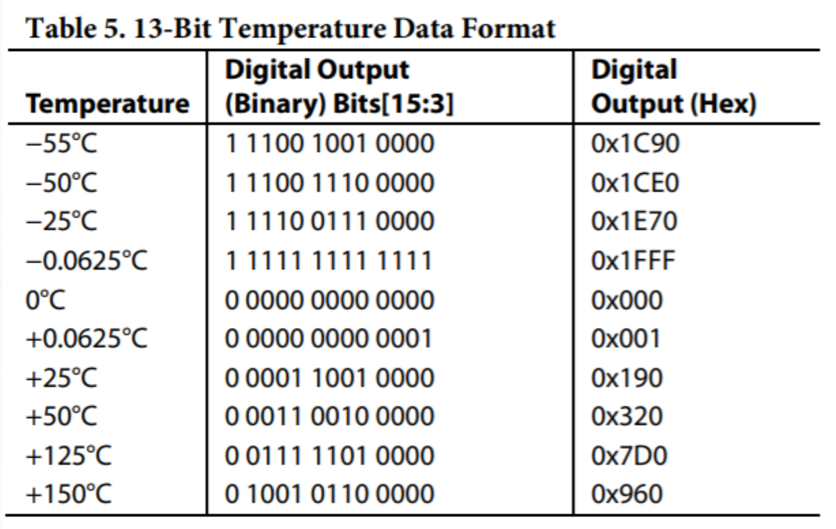
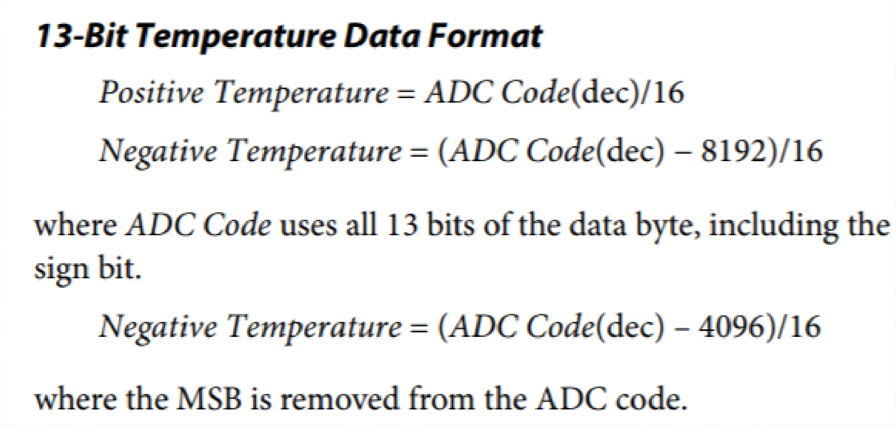


Table 5. 13-Bit Temperature Data Format

Temperature	Digital Output (Binary) Bits[15:3]	Digital Output (Hex)
−55°C	1 1100 1001 0000	0x1C90
−50°C	1 1100 1110 0000	0x1CE0
−25°C	1 1110 0111 0000	0x1E70
−0.0625°C	1 1111 1111 1111	0x1FFF
0°C	0 0000 0000 0000	0x000
+0.0625°C	0 0000 0000 0001	0x001
+25°C	0 0001 1001 0000	0x190
+50°C	0 0011 0010 0000	0x320
+125°C	0 0111 1101 0000	0x7D0
+150°C	0 1001 0110 0000	0x960



13-Bit Temperature Data Format

$$\textit{Positive Temperature} = \textit{ADC Code}(\text{dec})/16$$

$$\textit{Negative Temperature} = (\textit{ADC Code}(\text{dec}) - 8192)/16$$

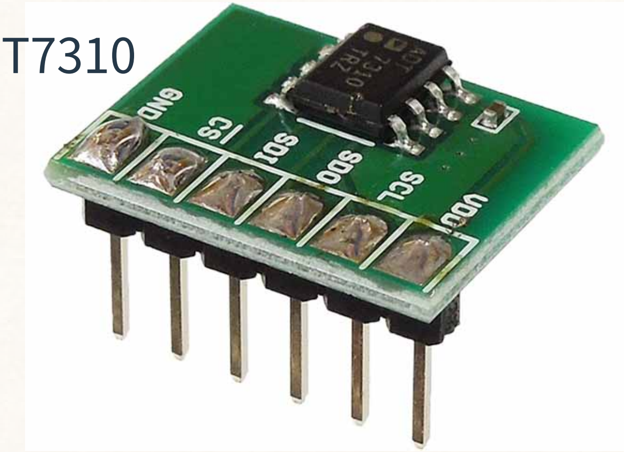
where *ADC Code* uses all 13 bits of the data byte, including the sign bit.

$$\textit{Negative Temperature} = (\textit{ADC Code}(\text{dec}) - 4096)/16$$

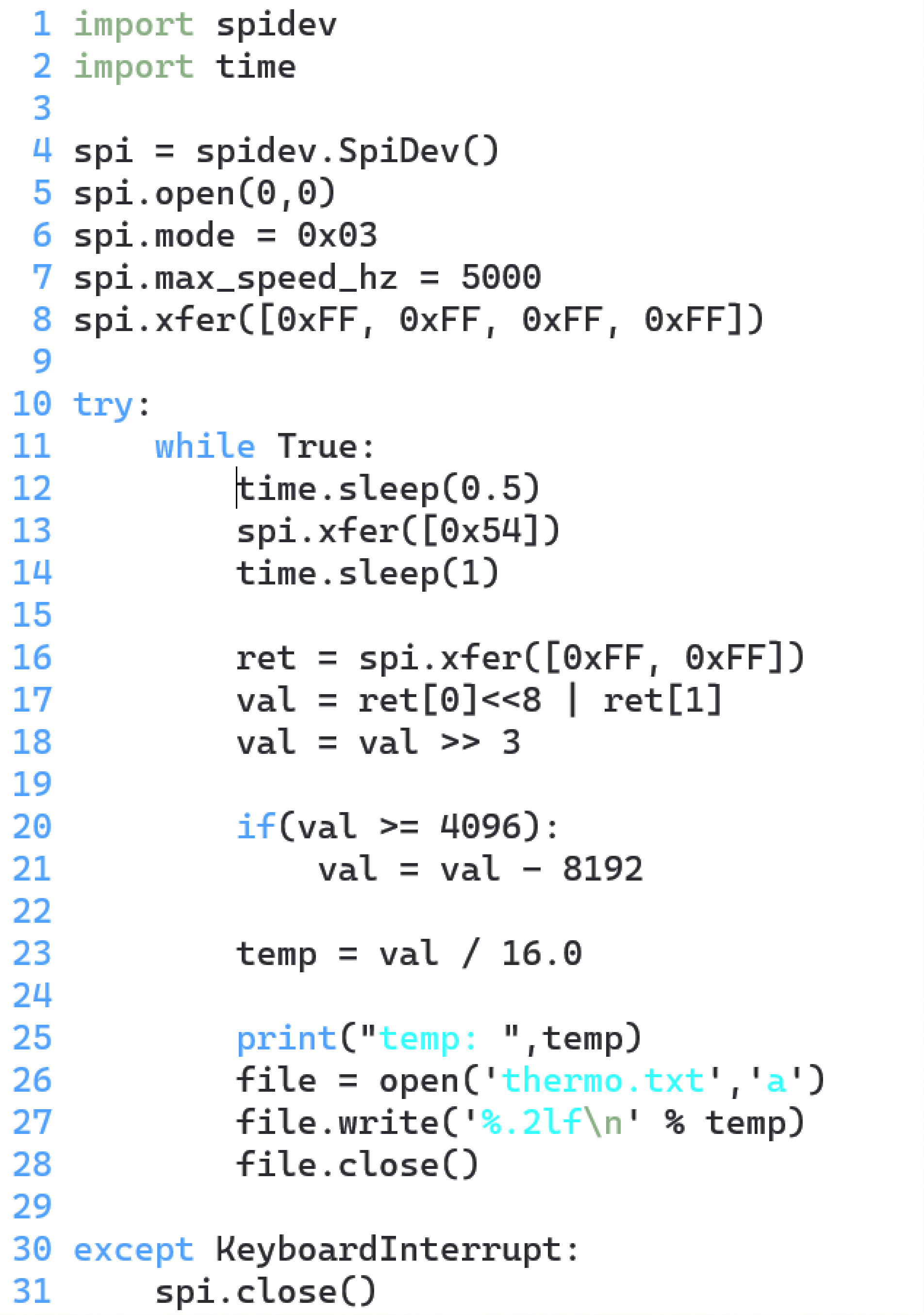
where the MSB is removed from the ADC code.

Thermometer

ADT7310



GPIO pin	LED pin
+5 or +3.3V	VDD
GPIO11 (SPI_CLK)	SCL
GPIO9 (SPI_MOSI)	SDD
GPIO10 (SPI_MISO)	SDI
GPIO8 (SPI_CE0)	CS
GND	GND



```
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)
spi.mode = 0x03
spi.max_speed_hz = 5000
spi.xfer([0xFF, 0xFF, 0xFF, 0xFF])

try:
    while True:
        time.sleep(0.5)
        spi.xfer([0x54])
        time.sleep(1)

        ret = spi.xfer([0xFF, 0xFF])
        val = ret[0]<<8 | ret[1]
        val = val >> 3

        if(val >= 4096):
            val = val - 8192

        temp = val / 16.0

        print("temp: ",temp)
        file = open('thermo.txt','a')
        file.write('%.2lf\n' % temp)
        file.close()

except KeyboardInterrupt:
    spi.close()
```

Power Supply (LAN communication)

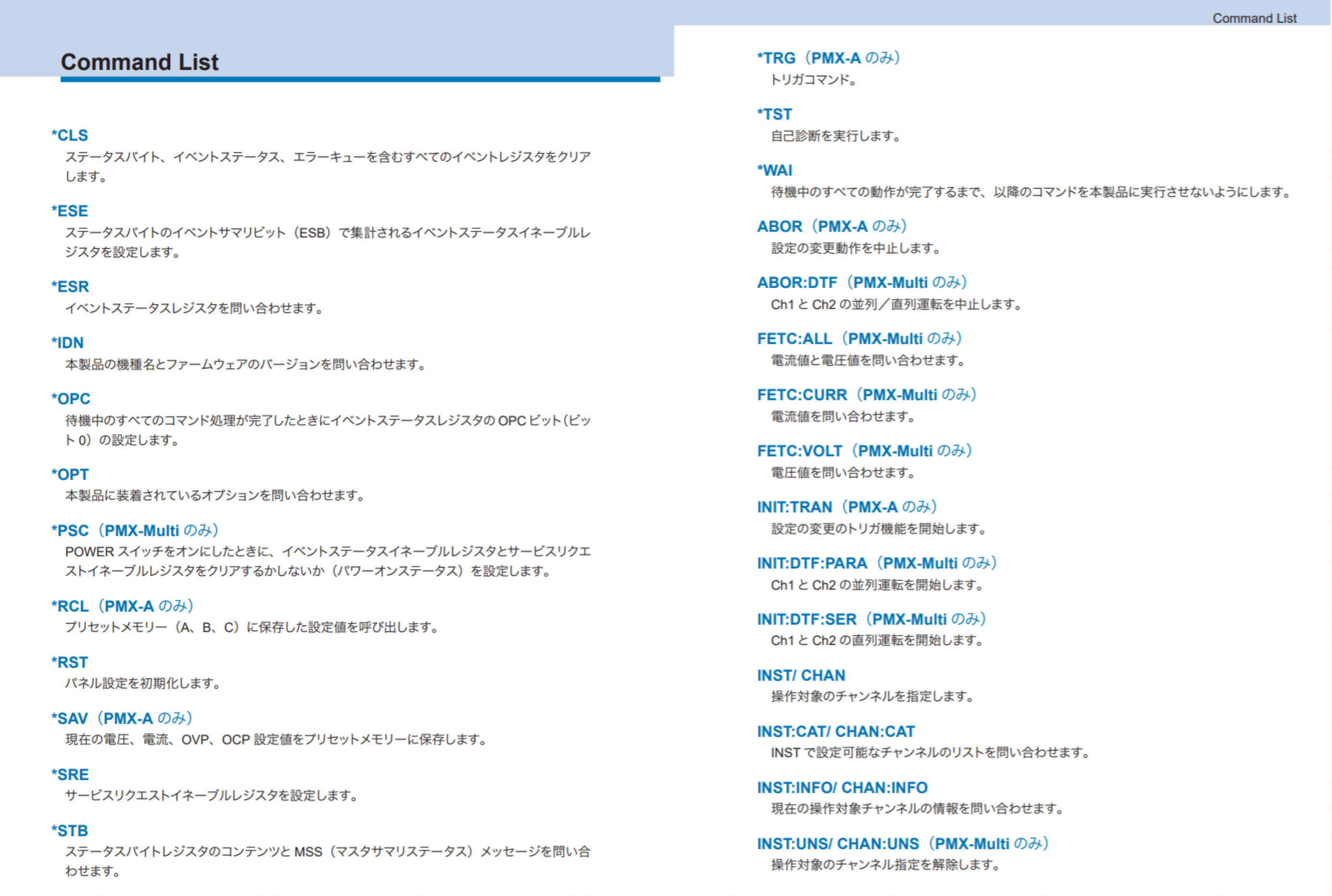
Kikusui PMX-A

S1
USB
RS232C
LAN
J1
AC INPUT 50/60Hz 150VA MAX
IF THIS COLUMN IS BLANK, THE UNIT IS WIRED IN 100V
SETTING SUPPLY
LINE VOLTAGE
STANDARD
100V
117V
200V
217V
234V
KIKUSUI ELECTRONICS CORP.
No.
MADE IN CHINA

Manual

https://manual.kikusui.co.jp/P/PMX_V2_J7.pdf

https://manual.kikusui.co.jp/P/PMX_IF_J2.pdf



Command List

Command List

*CLS
ステータスバイト、イベントステータス、エラーキューを含むすべてのイベントレジスタをクリアします。

*ESE
ステータスバイトのイベントサマリビット（ESB）で集計されるイベントステータスイネーブルレジスタを設定します。

*ESR
イベントステータスレジスタを問い合わせます。

*IDN
本製品の機種名とファームウェアのバージョンを問い合わせます。

*OPC
待機中のすべてのコマンド処理が完了したときにイベントステータスレジスタの OPC ビット（ビット 0）の設定します。

*OPT
本製品に装着されているオプションを問い合わせます。

*PSC（PMX-Multi のみ）
POWER スイッチをオンにしたときに、イベントステータスイネーブルレジスタとサービスリクエストイネーブルレジスタをクリアするかしないか（パワーオンステータス）を設定します。

*RCL（PMX-A のみ）
プリセットメモリー（A、B、C）に保存した設定値を呼び出します。

*RST
パネル設定を初期化します。

*SAV（PMX-A のみ）
現在の電圧、電流、OVP、OCP 設定値をプリセットメモリーに保存します。

*SRE
サービスリクエストイネーブルレジスタを設定します。

*STB
ステータスバイトレジスタのコンテンツと MSS（マスタサマリステータス）メッセージを問い合わせます。

*TRG（PMX-A のみ）
トリガコマンド。

*TST
自己診断を実行します。

*WAI
待機中のすべての動作が完了するまで、以降のコマンドを本製品に実行させないようにします。

ABOR（PMX-A のみ）
設定の変更動作を中止します。

ABOR:DTF（PMX-Multi のみ）
Ch1 と Ch2 の並列／直列運転を中止します。

FETC:ALL（PMX-Multi のみ）
電流値と電圧値を問い合わせます。

FETC:CURR（PMX-Multi のみ）
電流値を問い合わせます。

FETC:VOLT（PMX-Multi のみ）
電圧値を問い合わせます。

INIT:TRAN（PMX-A のみ）
設定の変更のトリガ機能を開始します。

INIT:DTF:PARA（PMX-Multi のみ）
Ch1 と Ch2 の並列運転を開始します。

INIT:DTF:SER（PMX-Multi のみ）
Ch1 と Ch2 の直列運転を開始します。

INST/ CHAN
操作対象のチャンネルを指定します。

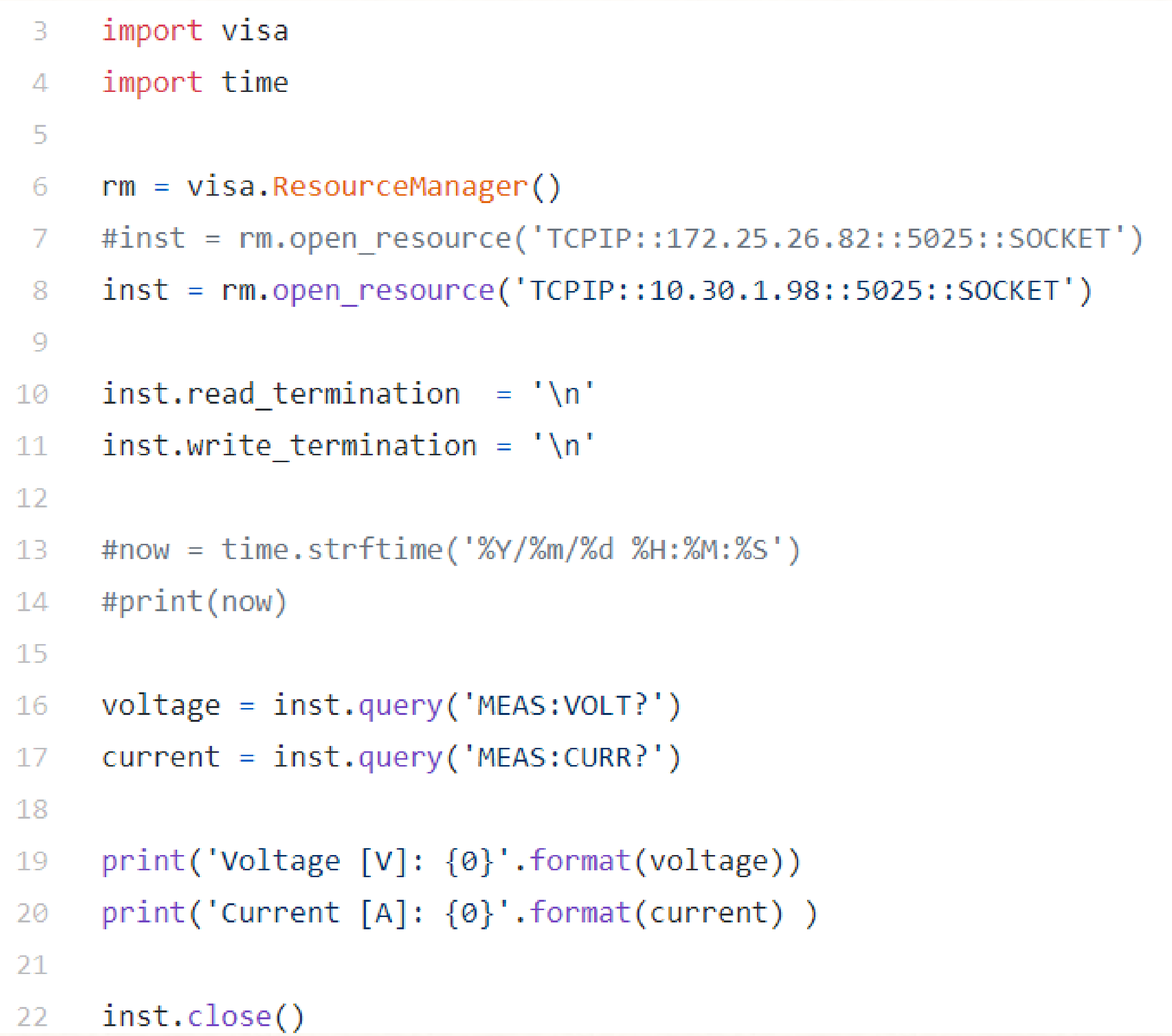
INST:CAT/ CHAN:CAT
INST で設定可能なチャンネルのリストを問い合わせます。

INST:INFO/ CHAN:INFO
現在の操作対象チャンネルの情報を問い合わせます。

INST:UNS/ CHAN:UNS（PMX-Multi のみ）
操作対象のチャンネル指定を解除します。

Power Supply

Kikusui PMX-A



```python
import visa
import time

rm = visa.ResourceManager()
#inst = rm.open_resource('TCPIP::172.25.26.82::5025::SOCKET')
inst = rm.open_resource('TCPIP::10.30.1.98::5025::SOCKET')

inst.read_termination  = '\n'
inst.write_termination = '\n'

#now = time.strftime('%Y/%m/%d %H:%M:%S')
#print(now)

voltage = inst.query('MEAS:VOLT?')
current = inst.query('MEAS:CURR?')

print('Voltage [V]: {0}'.format(voltage))
print('Current [A]: {0}'.format(current) )

inst.close()
```

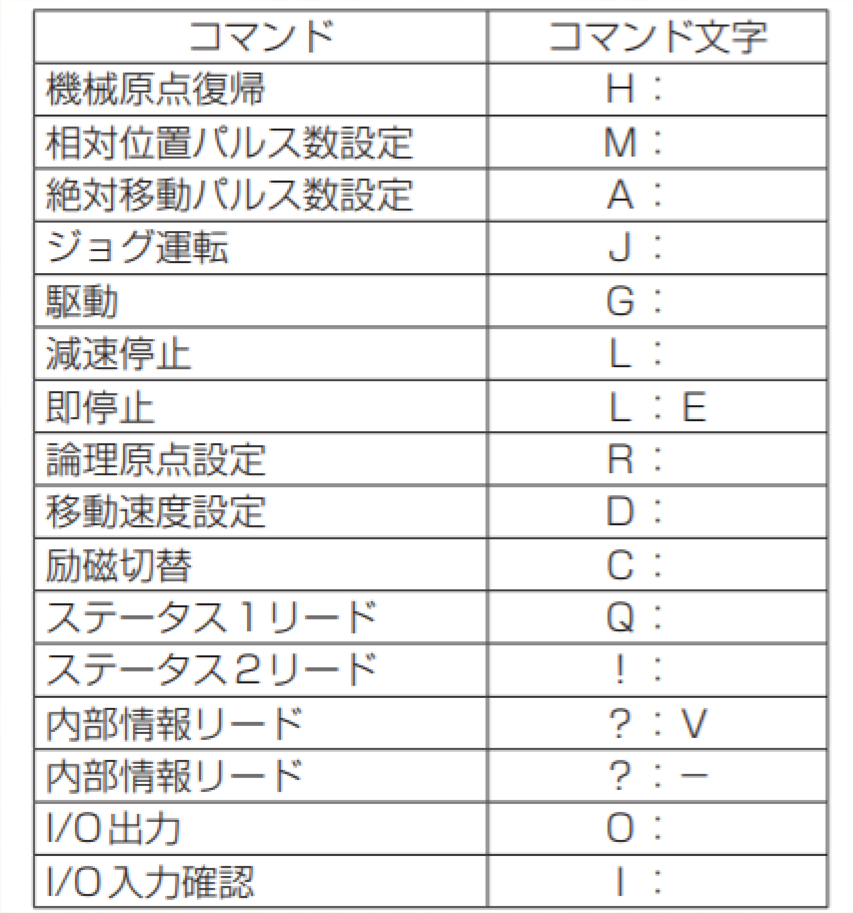
Stage (RS232-C)

SigmaKOKI
OSMS26

Controller (GSC-01)

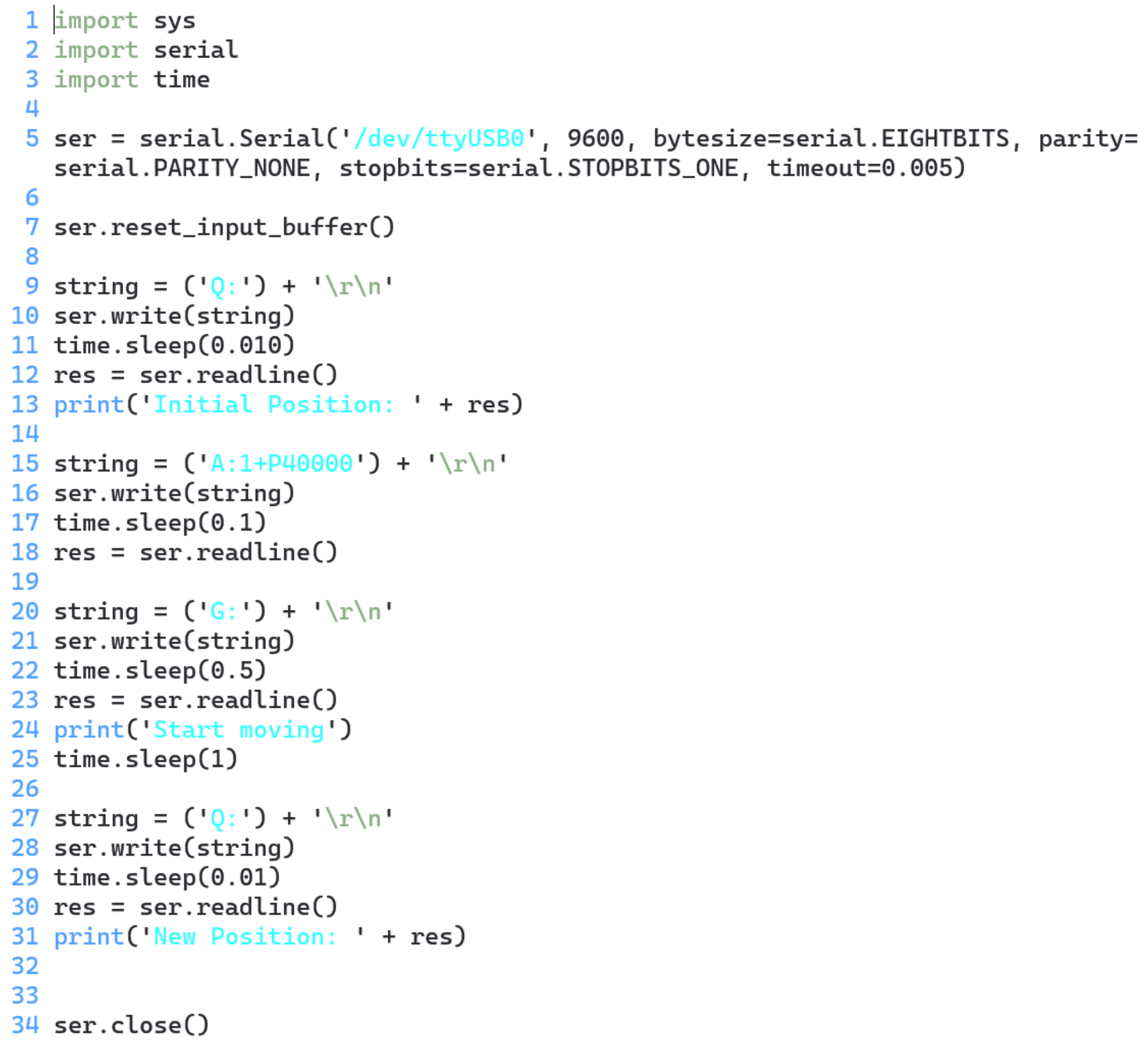
Manual

https://www.global-optosigma.com/jp/software/motorize/manual_jp/GSC-01.pdf



コマンド	コマンド文字
機械原点復帰	H：
相対位置パルス数設定	M：
絶対移動パルス数設定	A：
ジョグ運転	J：
駆動	G：
減速停止	L：
即停止	L：E
論理原点設定	R：
移動速度設定	D：
励磁切替	C：
ステータス１リード	Q：
ステータス２リード	！：
内部情報リード	？：V
内部情報リード	？：－
I/O出力	O：
I/O入力確認	I：

Stage (RS232-C)



```
import sys
import serial
import time

ser = serial.Serial('/dev/ttyUSB0', 9600, bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE, timeout=0.005)

ser.reset_input_buffer()

string = ('Q:') + '\r\n'
ser.write(string)
time.sleep(0.010)
res = ser.readline()
print('Initial Position: ' + res)

string = ('A:1+P40000') + '\r\n'
ser.write(string)
time.sleep(0.1)
res = ser.readline()

string = ('G:') + '\r\n'
ser.write(string)
time.sleep(0.5)
res = ser.readline()
print('Start moving')
time.sleep(1)

string = ('Q:') + '\r\n'
ser.write(string)
time.sleep(0.01)
res = ser.readline()
print('New Position: ' + res)


ser.close()
```

Summary

Introduction of Basic serial communication and control methods.

Let us try to operate with raspberry pi.