第 17 课，掌握 SPI 通讯实现与 25X40 芯片的通讯

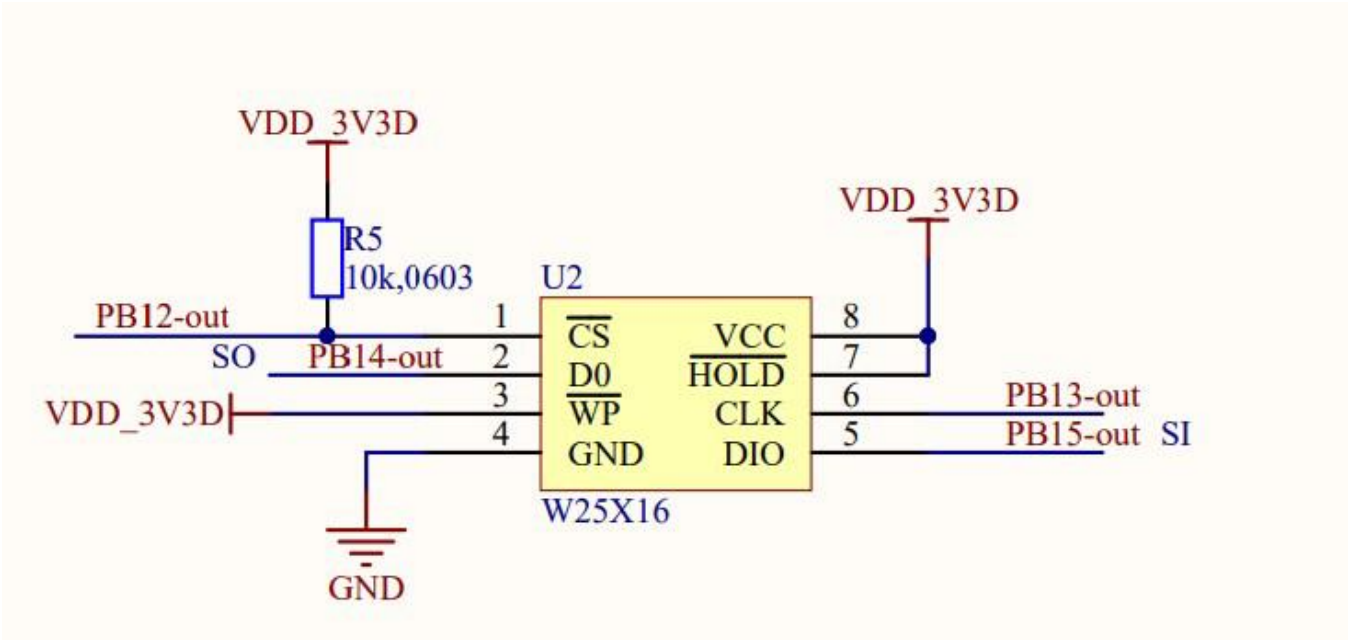
本课学习内容：

- 了解 SPI 是一种芯片级别直接的通讯
- 实现读取 25X40 的 ID

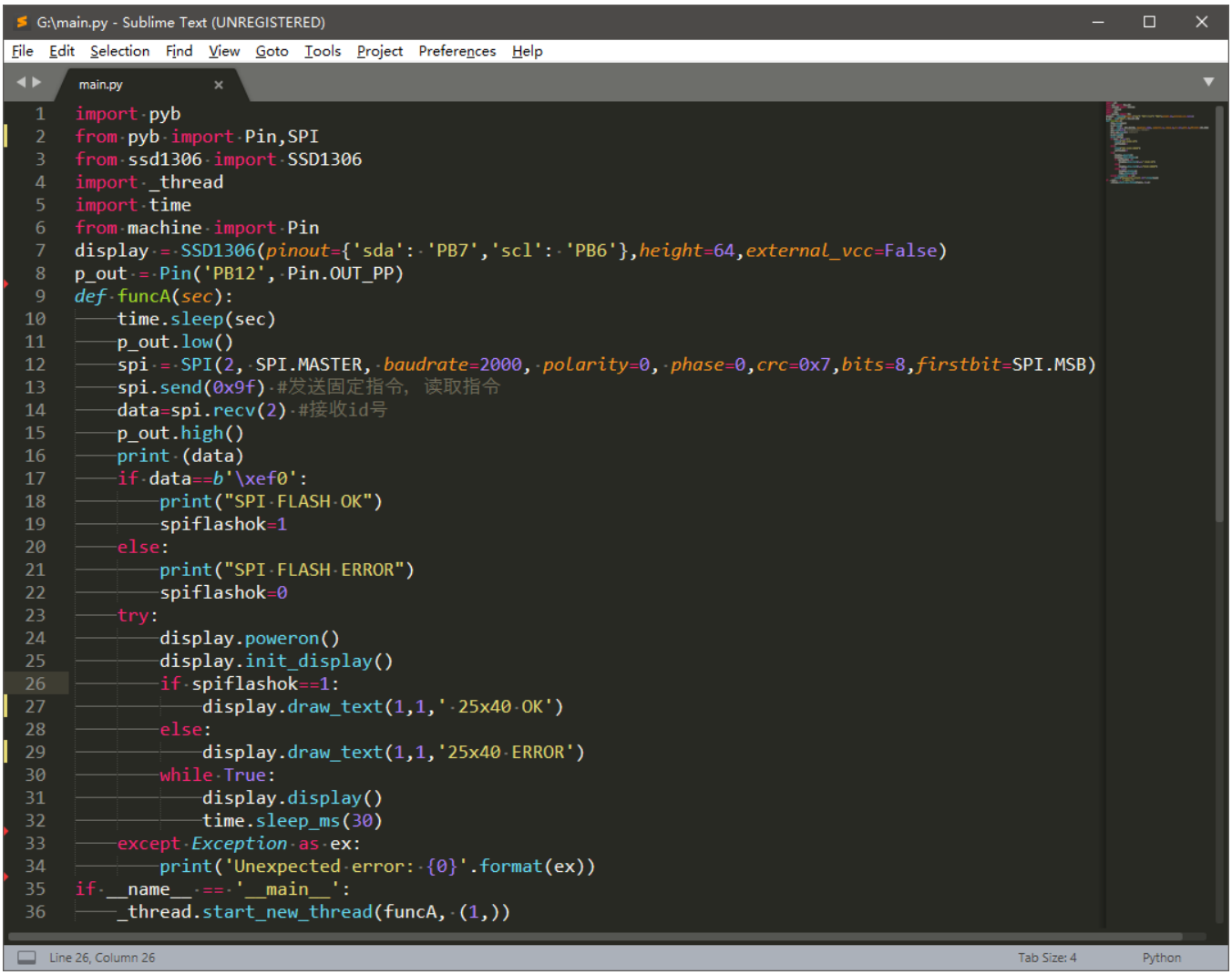
首先 25X4 它是个 Flash，Flash 是什么东西就不多说了（非易失性存储介质），分为 NOR 和 NAND 两种（NOR 和 NAND 的区别本篇不做介绍）。SPI 一种通信接口。那么严格的来说 SPI Flash 是一种使用 SPI 通信的 Flash，即，可能指 NOR 也可能是 NAND。但现在大部分情况默认下人们说的 SPI Flash 指的是 SPI NorFlash。早期 Norflash 的接口是 parallel 的形式，即把数据线和地址线并排与 IC 的管脚连接。但是后来发现不同容量的 Norflash 不能硬件上兼容（数据线和地址线的数量不一样），并且封装比较大，占用了较大的 PCB 板位置，所以后来逐渐被 SPI（串行接口）Norflash 所取代。同时不同容量的 SPI Norflash 管脚也兼容封装也更小。，至于现在很多人说起 NOR flash 直接都以 SPI flash 来代称。

25X40 在许多试验中都有出现。25X40 的应用主要在存储一些掉电后还要保存数据的场合，在上次运行时，保存的数据，在下一次运行时还能够调出。25X40 采用了 SPI 总线，是一种速度比 IIC 线总线更快的总线包括 MOSI,MISO,CLK,CS，更多知识请参考其他相关资料。如果您不想研究，也没有关系，Python 精髓就是在有时候不太明白时候不耽误使用，现在和今后您都可以只调用就是，不必花时间和精力去研究。

我们看一下原理图，W25X16(我们实际用的是 W25X40)器件硬件连接。 PB12 连接 CS，平时为高电平，工作选中时候为高电平，PB14 连接 SO(即 MISO),PB13 连接 CLK,PB15 连接 DIO(MOSI)。



以下程序实现功能：Pyboard 读取 25X40 的 ID 并判断读取的 ID,是否为手册中对应的 ID,以此判断 SPI 通讯是否正确，判断 IC 是是否有问题。




```
import pyb
from pyb import Pin,SPI
from ssd1306 import SSD1306
import _thread
import time
from machine import Pin
display = SSD1306(pinout={'sda': 'PB7','scl': 'PB6'},height=64,external_vcc=False)
p_out = Pin('PB12', Pin.OUT_PP)
def funcA(sec):
    time.sleep(sec)
    p_out.low()
    spi = SPI(2, SPI.MASTER, baudrate=2000, polarity=0, phase=0,crc=0x7,bits=8,firstbit=SPI.MSB)
    spi.send(0x9f) #发送固定指令, 读取指令
    data=spi.recv(2) #接收id号
    p_out.high()
    print (data)
    if data==b'\xef0':
        print("SPI FLASH OK")
        spiflashok=1
    else:
        print("SPI FLASH ERROR")
        spiflashok=0
    try:
        display.poweron()
        display.init_display()
        if spiflashok==1:
            display.draw_text(1,1,' 25x40 OK')
        else:
            display.draw_text(1,1,'25x40 ERROR')
        while True:
            display.display()
            time.sleep_ms(30)
    except Exception as ex:
        print('Unexpected error: {0}'.format(ex))
if __name__ == '__main__':
    _thread.start_new_thread(funcA, (1,))
```

注意测试该程序，需要安装上 OLED 显示屏，如若不然，会报错“Unexpected error: [Errno 5] EIO”该报错与我们程序中的 34 行有设置，有精力的小伙伴可以深入的研究一下，同时本节课程和上一节课程，都有用到

try：

……

except：

……

该语言是 Python 通用语法，为常用异常处理语言：

什么是异常？异常即是一个事件，该事件会在程序执行过程中发生，影响了程序的正常执行。

一般情况下，在 Python 无法正常处理程序时就会发生一个异常。异常是 Python 对象，表示一个错误。当 Python 脚本发生异常时我们需要捕获处理它，否则程序会终止执行。异常处理捕捉异常可以使用 try/except 语句。try/except 语句用来检测 try 语句块中的错误，从而让 except 语句捕获异常信息并处理。如果你不想在异常发生时结束你的程序，只需在 try 里捕获它。

以下为简单的 try....except...else 的语法：

```
try:
<语句>        #运行别的代码
except <名字>:
<语句>        #如果在 try 部份引发了'name'异常
except <名字>，<数据>:
<语句>        #如果引发了'name'异常，获得附加的数据
else:
<语句>        #如果没有异常发生
```

try 的工作原理是，当开始一个 try 语句后，python 就在当前程序的上下文中作标记，这样当异常出现时就可以回到这里，try 子句先执行，接下来会发生什么依赖于执行时是否出现异常。如果当 try 后的语句执行时发生异常，python 就跳回到 try 并执行第一个匹配该异常的 except 子句，异常处理完毕，控制流就通过整个 try 语句（除非在处理异常时又引发新的异常）。如果在 try 后的语句里发生了异常，却没有匹配的 except 子句，异常将被递交到上层的 try，或者到程序的最上层（这样将结束程序，并打印默认的出错信息）。

如果在 try 子句执行时没有发生异常，python 将执行 else 语句后的语句（如果有 else 的话），然后控制流通过整个 try 语句。

```
import pyb

from pyb import Pin,SPI

from ssd1306 import SSD1306

import _thread

import time

from machine import Pin

display = SSD1306(pinout={'sda': 'PB7','scl':
'PB6'},height=64,external_vcc=False)

p_out = Pin('PB12', Pin.OUT_PP)#SPI 的 CS 选择硬件引脚设置

def funcA(sec):

  time.sleep(sec)
```

```
    p_out.low()#拉低进入任务
    spi = SPI(2, SPI.MASTER, baudrate=2000, polarity=0,
phase=0,crc=0x7,bits=8,firstbit=SPI.MSB)
    spi.send(0x9f) #发送固定指令，读取指令
    data=spi.recv(2) #接收 id 号
    p_out.high()#结束通讯
    print (data)
    if data==b'\xef0':#判断是否是对的 ID
        print("SPI FLASH OK")#向 REPL 打印信息
        spiflashok=1#设置标志位
    else:
        print("SPI FLASH ERROR")#如果错误话，向 REPL 打印报错信息
        spiflashok=0#设置标志位
    try:
        display.poweron()#开启 OLED 显示
        display.init_display()#初始化 OLED 显示
        if spiflashok==1:#根据标志位显示不同的结果
            display.draw_text(1,1,' 25x40 OK')#OLED 显示正常时信息
        else:
            display.draw_text(1,1,'25x40 ERROR')#OLED 显示不正常时信息
        while True:
            display.display()#显示操作
            time.sleep_ms(30)#延时
    except Exception as ex:
        print('Unexpected error: {0}'.format(ex))#异常处理函数
if __name__ == '__main__':#程序入口
    _thread.start_new_thread(funcA, (1,))#线程 A
```