加载文件系统的选择

在pyboard板子芯片本体内部有一套称为/flash微型文件系统，这个文件系统是占用微型处理器的"硬盘"资源的，如果插入一个Micro SD卡（TF卡），文件系统将会被Micro SD卡上的相关文件系统所替代，这个插入外部文件系统件取名为/sd。MicroPython系统需要调用加载至少一套文件系统完成用户程序的应用。

当pyboard启动时，根据引导设定来选择一套文件系统进行加载应用。默认的加载模式是如果没有Micro SD卡，MicroPython将加载内部文件系统/flash，否则将加载Micro SD卡中的/sd文件系统。

启动后，当前目录文件使用由上面两种途径选择后的载入文件（类似电脑启动选择系统引导）。如果有必要，您可以通过创建一个名为/flash/SKIPSD 的空文件来防止SD卡的使用。如果该文件存在，当pyboard启动时，就会跳过SD卡启动，pyboard将始终从内部文件系统启动，在这种情况下，SD卡不会被安装，但是您仍然可以在您的程序中使用os.mount来安装并使用它。(请注意，在旧版本的板上/flash 被称为0:/和/sd被称为1:/)在加载的文件系统中会用到2个文件：boot.py和main.py，pyboard通过USB线连接PC机建立通讯后，文件系统是以U盘形式呈现，可以将文件在电脑各个驱动器之间完成各种文件操作，同时可以使用任何文本编辑器完成对boot.py和main.py的编辑。在卸载pyboard板“U盘”前，一定要按照正常

正常引导和用户强制引导

正常引导：如果正常上电启动或者按复位键后，板子会按照引导序列进行正常模式启动，正常模式的引导序列为：boot.py会被首先执行，然后USB将会被配置（虚拟串口和“U盘”实现），最后main.py被执行。

用户强制引导：用户可以借助用户按键（USER）操作来替代这个正常引导。具体操作是：按着用户开关上电或者按一下复位键，这时注意不要松开用户按键，LED黄灯和LED绿灯组合会以二进制模式点亮，出现你想要的模式对应的LED状态时，迅速的松开按键，这时用户刚才选择的LED状态会快闪几下，然后进入对应模式或者功能。LED状态对应的模式或功能：

模式顺序	对应LED状态	模式名称	说明备注
模式1	仅有绿色LED亮起	正常模式	正常执行boot.py和main.py启动
模式2	仅有黄色LED亮起	安全模式	在启动时不运行任何脚本
模式3	绿色和黄色LED都亮起	恢复出厂设置	将文件系统重置为出厂状态然后启动安全模式

出现错误状态时LED指示：

用户可能会遇到两种错误状态LED指示。

1：如果红色LED和绿色LED交替闪烁，那么就表明Python脚本(如main.py)出现错误，这时可以使用REPL调试或者直接修改部署的文件。

2：如果4个LED灯都在缓慢地上下交替闪烁，那就表明出现了一个严重的故障，这个故障很难修复，用户可以进入强制引导模式选择模式3进行重置。

对文件系统中的文件操作时LED指示：

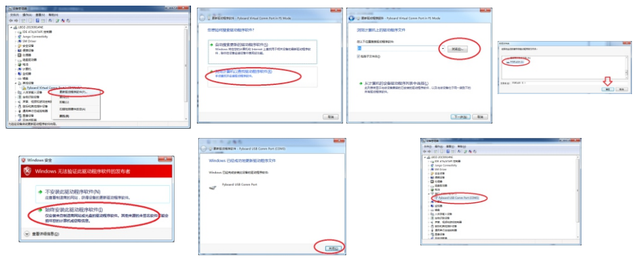
当pyboard与PC正常通信，并对文件系统进行操作时对其写入数据，红色LED亮起提示正在写入，写入过程可能是文件的保存，删除，粘贴等操作。

pyboard首次接入Windos指导：

通过MicroUSB线连接PC上电后可以看到两个变化，一个是电脑设备管理器出现了需要安装驱动的虚拟串口设备;另一个是“U盘”设备名称为PYBFLASH。

U盘中的有几个重要的文件boot.py、main.py、pybcdc.inf。如果板子正常启动，上电先会运行boot.py，然后再配置USB，最后运行main.py。

在设备管理器中，你可以找到没有成功安装驱动的设备，右击打开菜单，点击更新驱动程序软件->浏览计算机以查找驱动程序软件->点击浏览->选择插入的盘符->点击确定->点击下一步->点击始终安装此驱动程序软件->完成安装（虚拟串口安装完成后可以看到相应的串口号）。



DFU模式与更新固件

当用户获取到更新版本的Micropython pyboard固件或者由于其他原因需要更换固件时，pyboard需要进入DFU模式借助下载软件完成刷入固件的操作，类似于PC机更新Windows系统。步骤分为进入DFU模式和操作软件下载。

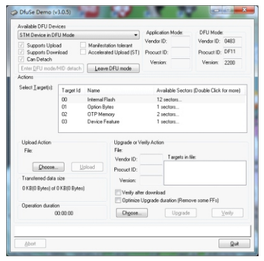
Q:如何进入DFU模式？

A:对于官方板，在连接PC机正常供电的情况下，需要使用金属导电物体例如镊子短接boot0引脚和3.3V引脚，然后按复位按键就可以进入DFU模式。为了方便用户进入DFU模式，我们创新设计了一个BOOT0按键，按着这个按键，再按一下复位键就可以完成进入DFU模式，这就避免了无短接工具或者用户已经在板子上焊接上了排针或排母的情况下无法下手的尴尬。

Q:使用什么软件在DFU模式下更新程序？

A:ST半导体公司提供了一套免费简单的下载软件：DfuSe。下载地址：http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1533/PF257916?s_searchtype= keyword

根据自己PC系统选择安装32位或者64位版本。



Q:下载到板子的DFU镜像文件在什么地方获取？

A:最新的DFU镜像文件获取地址：http://micropython.org/download/或者与我们取得联系获得。

Q:如何操作软件下载？

A:打开DfuSe软件点击Choose按键，选择之前下载好的DFU镜像文件保存路径，选择合适的镜像文件，点击Upgrade按键进行下载，等待下载完毕，按复位键或者点击Leave DFU mode按键离开DFU模式，进行正常模式。

编写和运行一个脚本：HELLO WORLD

在pyboard上运行一个Python脚本。毕竟，这才是它的全部意义！

连接到PC：连接pyboard到PC,遇到的情况可以参考上述文档解决，注意这是唯一的与编译平台通讯途径，不能出错。

当pyboard连接到你的PC时，它将启动并进入正常引导，绿色LED灯应该亮半秒或更短，当它熄灭时，意味着正常引导过程已经完成。

配置系统驱动呈现U盘：接下来就是要看你使用的什么PC系统了，安装相应的驱动完成U盘的识别。

现在可以通过U盘看到相应的文件。

所看到的驱动器被pyboard称为/flash，并且应该包含以下4个文件：

boot.py – 当pyboard启动时，该脚本将被执行。执行pyboard的各种配置选项。

main.py – 这是用户主要的用户程序实现文件.它在boot.py后执行。

README.txt – 包含了一些关于开始使用pyboard的非常基本的信息。

pybcdc.inf – 这是一个用于配置USB通讯的Windows驱动程序文件。

编辑main.py文件：

现在我们要写一段Python程序。任何文本编辑器都可以打开main.py文件。在Windows上，你可以使用记事本。在Mac和Linux上，使用你最熟悉的文本编辑器打开文件，会看到它只包含一行注释：

```
# main.py -- put your code here!
```

这段程序不会做任何事情，我们在接下来添加两段代码并添加注释。

```
# main.py -- put your code here!
import pyb       #添加pyb类库，因为我们要用到并控制LED模块
pyb.LED(4).on() #点亮蓝色LED
```

保存文档，会看的红色LED亮起，当熄灭时说明已经完成保存写入。

运行验证程序：按下pyboard上的RST键或者重新上电，绿色LED灯会很快闪几下，然后蓝色的LED灯会亮起，证明程序被运行了。

常用程序举例

通用控制指令

```
Pyb:pyboard专用模块
import pyb                          #包含pyb模块
pyb.repl_uart(pyb.UART(1, 9600)) #重置REPL的 UART(1),波特率为9600
pyb.wfi()                           #暂停CPU等待中断
pyb.freq()                          #获取CPU和总线频率
pyb.freq(60000000)                  #设置CPU和总线频率为60MHz
pyb.stop()                          #停止CPU等待外部中断
```

延时函数和时间应用

```
import time                         #包含时间函数
time.sleep(1)                       # 休眠1秒钟
time.sleep_ms(500)                  # 休眠500毫秒
time.sleep_us(10)                   # 休眠10微妙
start = time.ticks_ms()             # 获取毫秒计数器的值
delta = time.ticks_diff(time.ticks_ms(), start) #计算时差
```

板载LED应用

```
from pyb import LED                 #从pyb模块中添加LED功能
led = LED(1)                        #1红色2绿色3黄色4蓝色
led.toggle()                        #LED翻转
led.on()                            #LED点亮
led.off()                           #LED熄灭
# LED3和LED4支持PWM亮度为（0~255）
LED(4).intensity()                  # 获取强度
LED(4).intensity(128)               # 设置新强度为128
```

板载按键

```
from pyb import Switch              #从pyb模块中添加按键功能
sw = Switch()
sw.value()                          # 返回1或者0
sw.callback(lambda: pyb.LED(1).toggle()) #根据返回值进行动作
```

引脚和GPIO功能

```
from pyb import Pin                 #从pyb模块中植入Pin函数
p_out = Pin('X1', Pin.OUT_PP)       #初始化X1为输出
p_out.high()                        #X1为 高电平
p_out.low()                         #X1为低电平
p_in = Pin('X2', Pin.IN, Pin.PULL_UP)  #初始化X2为上拉输入
p_in.value()                        #判断获取输入的值为0或者1
```

舵机控制

```
from pyb import Servo               #从pyb模块中植入舵机函数
s1 = Servo(1)          # 启动位置1端口的舵机控制器（X1，VIN，GND）
s1.angle(45)                        # 正向迅速移动45度
s1.angle(-60, 1500)                 #在1500ms时间内移动到-60度
s1.speed(50)                        # 连续旋转舵机
```

外部中断

```
from pyb import Pin, ExtInt         #从pyb模块中植入引脚和外部中断函数
callback = lambda e: print("intr")
ext = ExtInt(Pin('Y1'), ExtInt.IRQ_RISING, Pin.PULL_NONE, callback)
```

定时器

```
from pyb import Timer               #从pyb模块中植入定时器函数
tim = Timer(1, freq=1000)
tim.counter()                       # 获取计数器的值
tim.freq(0.5)                       # 0.5 Hz
tim.callback(lambda t: pyb.LED(1).toggle())
```

RTC(实时时钟)

```
from pyb import RTC                 #从pyb模块中植入实时时钟函数
rtc = RTC()
rtc.datetime((2017, 8, 23, 1, 12, 48, 0, 0)) #设置时间和日期
rtc.datetime()                      #获取时间和日期
```

PWM（脉冲调幅控制）

```
from pyb import Pin, Timer          #从pyb模块中植入定时器函数
p = Pin('X1')                       # 在X1使用TIM2, CH1
tim = Timer(2, freq=1000)
ch = tim.channel(1, Timer.PWM, pin=p)
ch.pulse_width_percent(50)          #占空比为50%
```

ADC (模数转换)

```
from pyb import Pin, ADC            #从pyb模块中植入ADC函数
```

DAC (数模转换)

```
from pyb import Pin, DAC            #从pyb模块中植入DAC函数
dac = DAC(Pin('X5'))                #DAC引脚使用X5
dac.write(120)                      # 输出在0~255
```

UART串行通讯 (串口通讯)

```
from pyb import UART                #从pyb模块中植入串口函数
uart = UART(1, 9600)                #设置9600波特率
uart.write('hello')                 #输出hello
uart.read(5)                        # 读5个字节
```

SPI通讯

```
from pyb import SPI                 #从pyb模块中植入SPI函数
#设置SPI基本参数
spi = SPI(1, SPI.MASTER, baudrate=200000, polarity=1, phase=0)
spi.send('hello')                   #发送hello字符
spi.recv(5)                         #从总线接受5个字符
spi.send_recv('hello')              # 收到后反馈hello
```

I2C 总线通讯

```
from pyb import I2C                 #从pyb模块中植入I2C函数
#设置I2C最基本参数
i2c = I2C(1, I2C.MASTER, baudrate=100000)
i2c.scan()                          #返回从机列表
i2c.send('hello', 0x42)             # 发送给0x42地址5个字符
i2c.recv(5, 0x42)                   # 接收5个字符从0x42
i2c.mem_read(2, 0x42, 0x10)         #从器件地址0X42的从机的0X10寄存器    中读取2个字符
i2c.mem_write('xy', 0x42, 0x10)     # 写2个字符到器件地址为0x42器件的0x10寄存器
```

CAN总线 (局域网控制)

```
from pyb import CAN                 #从pyb模块中植入CAN函数
#设置基本模式和基本参数
can = CAN(1, CAN.LOOPBACK)
can.setfilter(0, CAN.LIST16, 0, (123, 124, 125, 126))
can.send('message!', 123)           # 使用id123发送message字节
can.recv(0)                         # 接收信息从 FIFO0
```

板载加速度

```
from pyb import Accel               #从pyb模块中植入加速度芯片函数
accel = Accel()
print(accel.x(), accel.y(), accel.z(), accel.tilt())
```

pyboard启动流程图