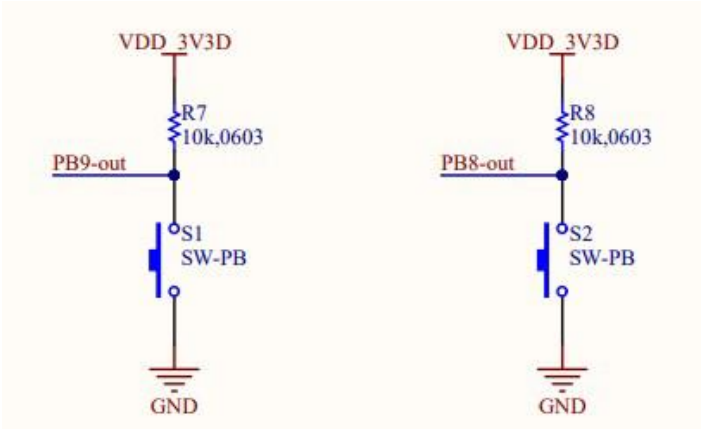
第 15 课，掌握硬件输入状态捕捉和外部中断处理

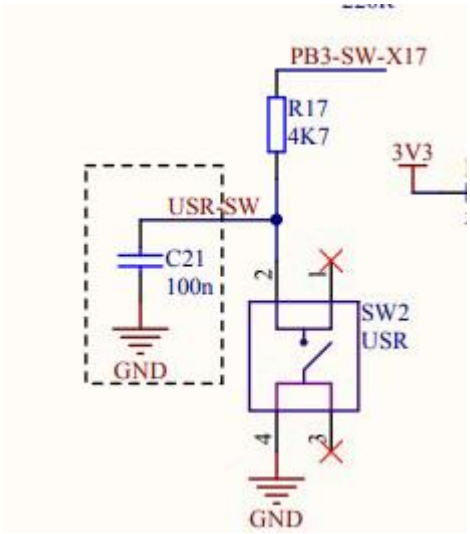
本课学习内容：

- 掌握 GPIO 输入功能的应用
- 理解中断概念，掌握中断的用途

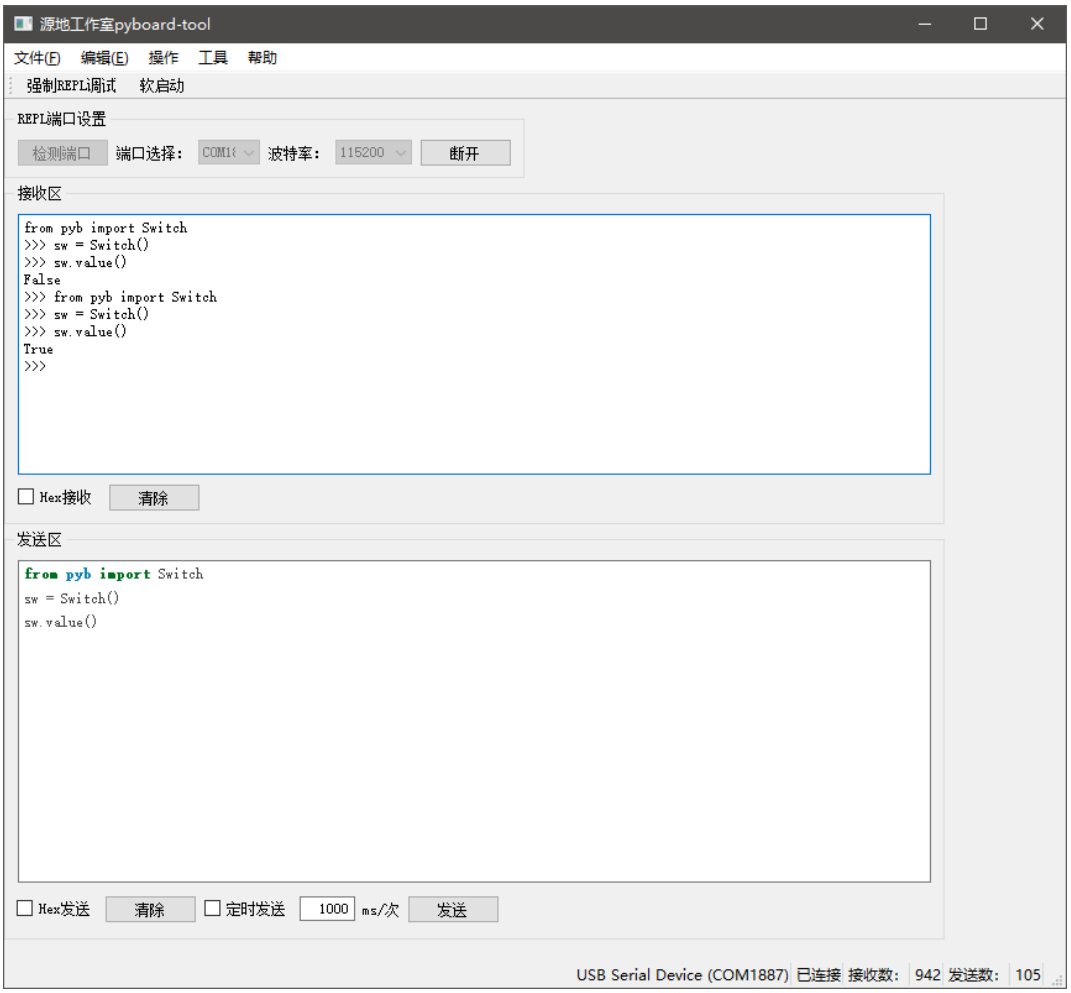
GPIO 输入功能是最基本的功能，任何一个 GPIO 口只有是不被固定占用，都可以使用 GPIO 的输入功能。我们 PYBminiDB 学习板上有三个按键，一个是在核心板上，标注为 USR 按键，另外两个是在底板上，按键 KEY1,KEY2。

首先看一下原理图，KEY1，KEY2,分析一下硬件电路，当按键按下 KEY1 或者 KEY2 时，对应的引脚应该为低电平，没有被按下或者平时状态为高电平，这样 MCU 就可以通过检查对应引脚的高低电平就能推测出按键是否被按下。在核心板板上的 USR 按键，使用是 MCU 内部的上拉，也是在按键按下的时候，变为低电平，虽然电路看起不太一样，但是原理是一样的，同时也要明白，MCU 的引脚可以通过设置变为内部上拉，也可以设置成内部下拉，也可以设置成既不是上拉也不是下拉的悬浮输入状态。





首先，我们通过 Pyboard-tool 的 REPL 交互状态，通过 REPL 发送命令查看这些按键是否被按下。测试 USR 按键是否按下。




```
from pyb import Switch
>>> sw = Switch()
>>> sw.value()
False
>>> from pyb import Switch
>>> sw = Switch()
>>> sw.value()
True
>>>
```

当没有按下发送询问指令后得到的信息为 False，当按下后返回的信息额为 True，证明被按下了。这里的 Switch 库函数就是为这个 USR 按键设计的，如果想用这个 USR 按键，需要首先引入这个库函数。

用户根据需要在合适的时候询问一下这个按键状态就能知道这个按键的状态，例如每隔 1s 检测一下，就可以知道按键状态。但是还有些应用场合，需要在按键按下后，必须立即处理按下按键后应该处理的事件，不能等待所谓的因循环造成的延时，这就用的了中断功能。

例程如下：

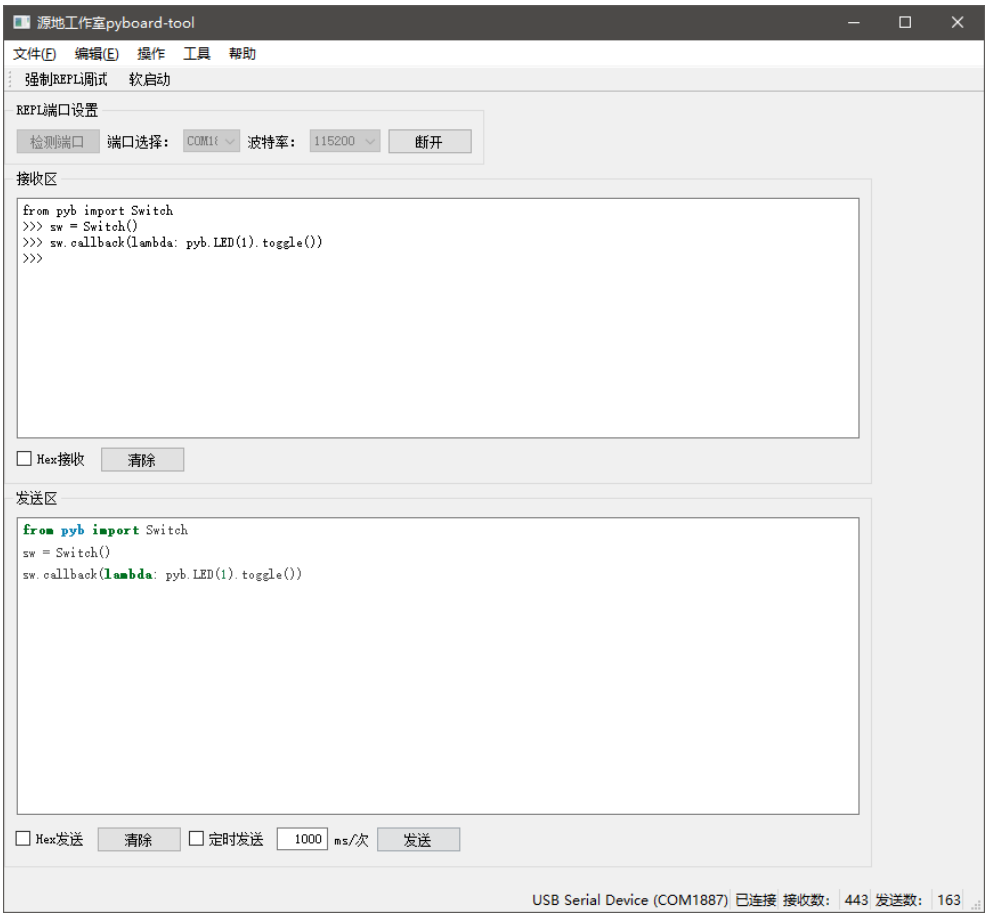
```
from pyb import Switch

sw = Switch()

sw.callback(lambda: pyb.LED(1).toggle())
```

该段程序就可以利用中断原理完成了一次中断事件处理，不论在这个基础程序上扩展再多的功能，只要是按键 USR 被按下，就会马上处理中断事件：lambda: pyb.LED(1).toggle()，即翻转 LED(1)灯状态，达到及时响应的目的。

我们在 Pyboard-tool 进行 REPL 测试一下。



不要按“**软启动**”，在点击发送后就可以测试，当我们按下 USR 按键时候，LED1 灯就会根据按下的事件改变自己状态，完成所谓的实时控制。

当然也可以在 main.py 脚本文件中完成上面的测试。上述我们只是讲解了 USR 这个特殊按键，下面介绍一下如何使用 KEY1（S1）,KEY2(S2)如何使用输入功能和中断输入功能。通过开头的电路图我们可以知道 PB9 和 PB8 是这里两个按键接入的引脚。

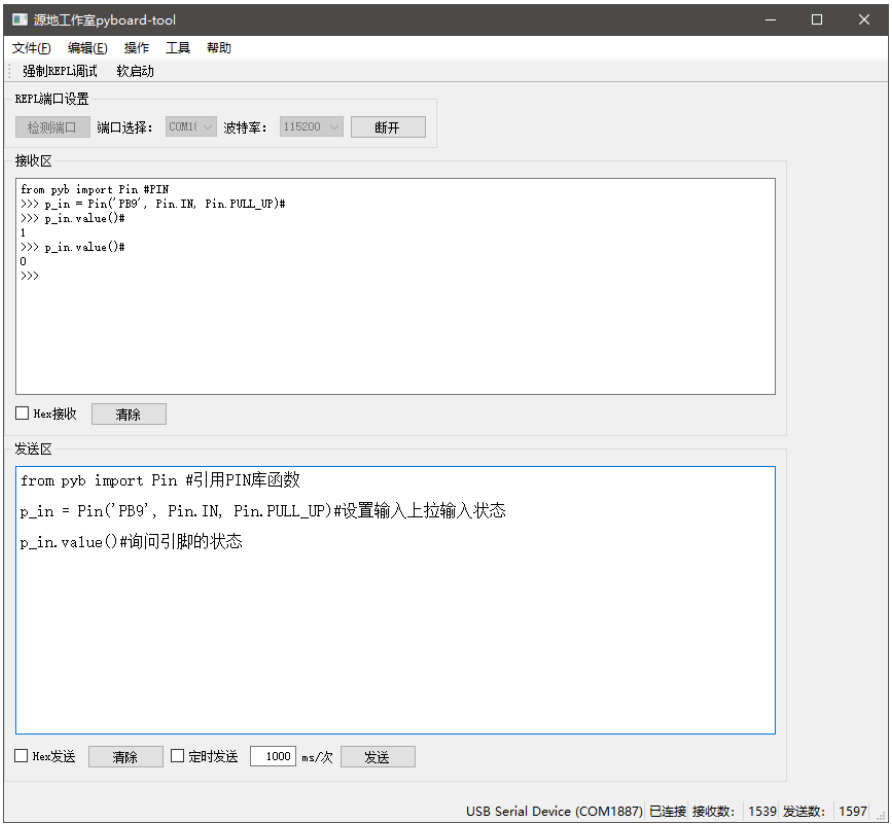
```
from pyb import Pin #引用 PIN 库函数

p_in = Pin('PB9', Pin.IN, Pin.PULL_UP)#设置输入上拉输入状态

p_in.value()#询问引脚的状态
```

同样我们先进行一下 Pyboard-tool 软件下的 REPL 测试。这是接收区的信息，我们可以看到没有按下时返回 1，按下时返回为 0。

```
from pyb import Pin
>>> p_in = Pin('PB9', Pin.IN, Pin.PULL_UP)
>>> p_in.value()
1
>>> p_in.value()
0
>>>
```



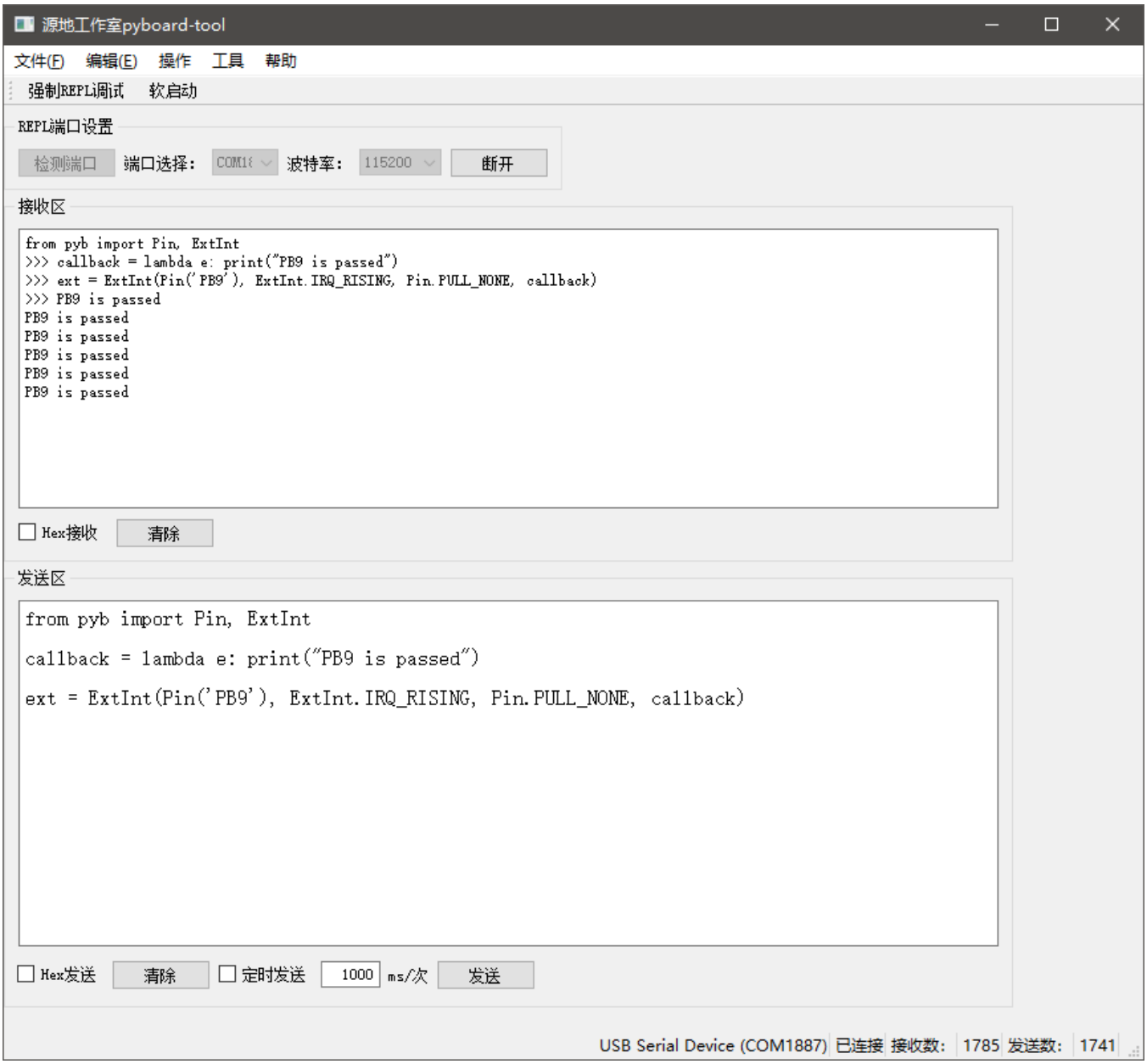
在 main.py 文件，我们也可以将上述脚本文件进行测试，这里就不做具体演示了。下面我们介绍如何使用普通的 GPIO 引脚，编写一个中断程序。

具体的例程函数为：

```
from pyb import Pin, ExtInt#应引用 Pin 库，和外部中断库

callback = lambda e: print("PB9 is passed")#编写中断处理程序

ext = ExtInt(Pin('PB9'), ExtInt.IRQ_RISING, Pin.PULL_NONE, callback)#初始化输入
```



在 REPL 点击发送后，就可以进行测试了，当我们按下 S1(PB9)会立即打印出“PB9 is passed”。当然也可以在 main.py 文件中进行程序的验证。

还有一种中断编程的脚本形式，还是通用的功能，当按下 S1 按键时，向 REPL 发送对应的信息。

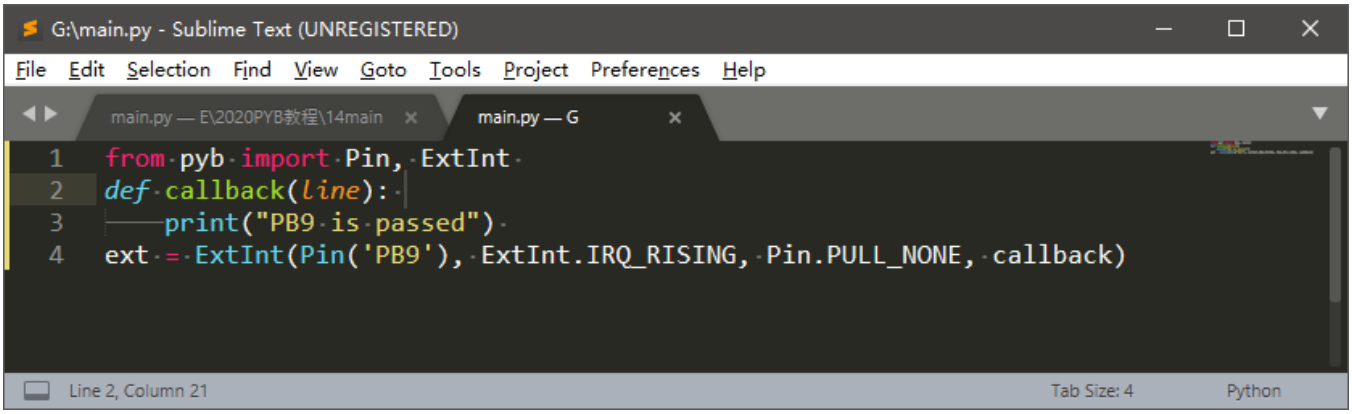
中断函数是以子函数形成编写的，推荐使用这种方法。

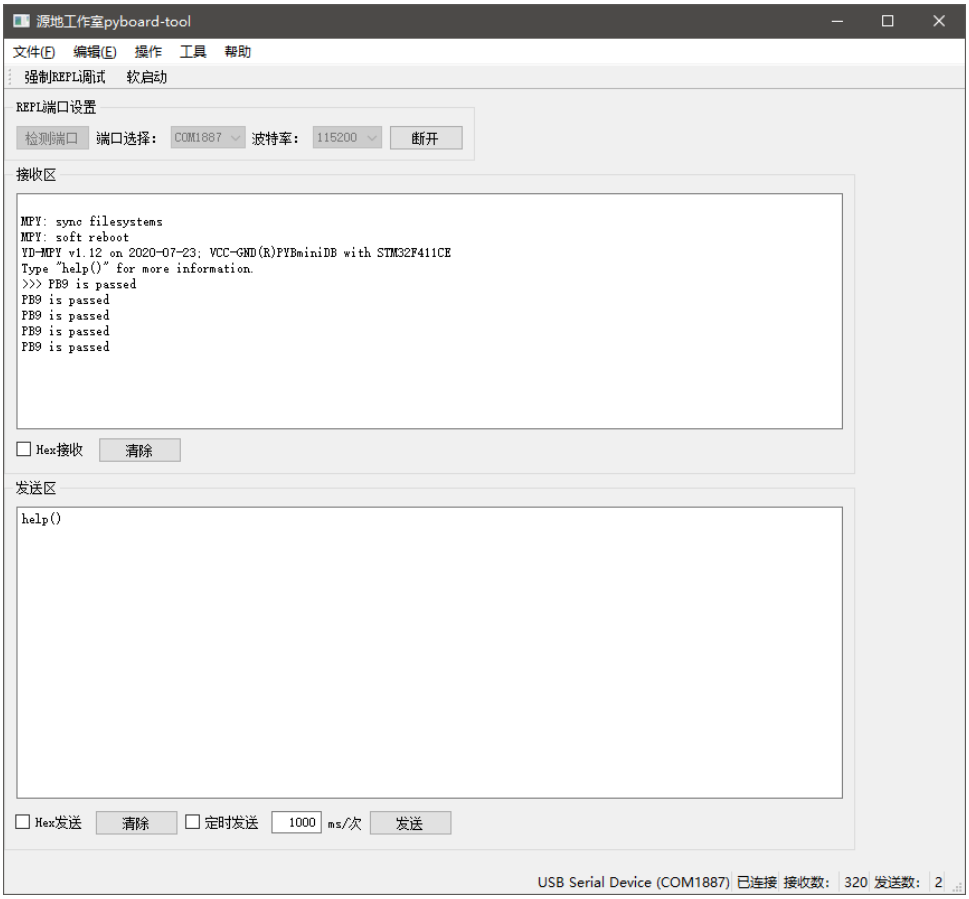
```
from pyb import Pin, ExtInt

def callback(line):

    print("PB9 is passed")

ext = ExtInt(Pin('PB9'), ExtInt.IRQ_RISING, Pin.PULL_NONE, callback)
```





总结一下语法：

```
from pyb import Pin


p_in = Pin('X2', Pin.IN, Pin.PULL_UP)  #X2 引脚赋值于 p_in 作为输入功能使用
p_in.value()                           # 获取输入值, 0 或者 1
#或者这样读取
p_in()

#外部中断例程
#通样可以使用 P+GPIO 号+数字代替 X Y 加数字进行控制例如 X1 可是使用 PA0 代替
from pyb import Pin, ExtInt

callback = lambda e: print("intr") #中断服务程序
ext = ExtInt(Pin('Y1'), ExtInt.IRQ_RISING, Pin.PULL_NONE, callback)#配置中断
```