
牛艾科技

# 按键实验

Python base program

# 目录 Contents

# Pyboard的按键

RST按键：复位按键，如果按下重新擦写重启开发板，相当于将开发板断电再重启。

USR按键：供用户使用，可以通过声明一个按键对象进行控制。

# 按键开关

创建开关对象方法：

>>>sw=pyb.Switch()

当提示pyb不存在的错误时，是因为没有键入 import pyb 语句

利用按键对象可以得到按键状态:

>>>sw()

如果按键被按下打印True，松开则打印False

```
Terminal ready

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

# 回调函数和中断

按键回调函数： sw.callback()
该函数将在按键按下时创建一些东西，且使用了一个中断。

```
>>>sw.callback(lambda:ptint(‘pass’))
>>>sw.callback(lambda:pyb.LED(1).toggle())
>>> def f():
…
>>>sw.callback(f)
```

```
Terminal ready

>>> import pyb
>>> sw=pyb.Switch()
>>> sw.callback(lambda:print('press!'))
>>> press!

>>> sw.callback(lambda:pyb.LED(1).toggle())
>>> sw.callback(None)
>>>
```

按键摇骰子：

```
import pyb
import random
sw=pyb.Switch()#声明按键对象
def f():#创建函数
        print(random.randint(1,6))
        #生成1-6的随机数
        pyb.delay(500)
        #延时防抖
sw.callback(f)#按键回调函数
```

```
Terminal ready

>>> 6
1
4
4
6
3
2
6
1
5
5
6
4
3
2
```

注意：回调函数一定不能含有任何分配内存的定义（比如不能声明创建列表和元组）。回调函数越简单越好。如果确切需要定义列表，请在使用回调函数前定义并用一个全局变量存储（或者定义为局部变量并对其进行封装？）。如果需要多次复杂的计算，那么可以用按键回调设置一个标志供其他代码响应使用。

**中断的原理细节：**

当你调用了含有 sw.callback( )的函数时，按键将在其连接引脚产生一个边沿触发（下降沿）外部中断。这意味着芯片将监听该引脚的任何状态变换，且如下事情会发生：

1.当按键被按下时引脚将发生改变（电平由低到高？），芯片处理器将记录这种变化；

2.处理器完成当前机器指令，退出执行状态并保存当前的状态（将寄存器的内容推入栈中）。这将停止当前运行的任何代码，例如正在执行着的 python 脚本；

3.芯片开始执行与按键相关的特定外部中断触发处理。该处理指向你在 sw.callback( )函数中指定的函数功能并执行之；

4.直到回调函数执行完毕，控制主权将回到中断处理手上；

5.按键中断处理将返回，芯片处理器确认记录该中断被执行过；
6.芯片调回步骤 2 的状态；
7.继续执行开始时的代码，除了短暂的暂停，这部分代码看起来似乎没有被打断过。

当同一时间多个中断同时发生上述的过程将复杂得多。这种情况下拥有最高优先级别的中断将被首先执行，其他的中断按各自的优先级数序执行。按键中断的优先级最低。

# THANK YOU

Concise business business report PPT

牛艾科技