1、按键扫描

按键扫描函数 `Key_Read()`放在定时器中进行扫描，10ms 执行一次，即 10ms 读取一次 I/O 状态，进行一次键值编码。读取到的键值是瞬态的，只能反映按键此刻的状态，无法反映按下按键抬起按键的稳定过程。

2、按键消抖

●三行代码第一行：`Key_Val = Key_Read();` 读取 10ms 更新一次的 I/O 电平状态，并存储在变量 `Key_Val`，可以理解为临时按键值。

●三行代码第三行：`Key_Old = Key_Val;`，`Key_Old` 为静态局部变量，**离开函数，值仍保留**：数据存储在静态存储区，在程序整个运行期间都不释放，且只能在该函数中调用。**将这次读取到的临时按键值 `Key_Val` 更新到 `Key_Old` 中，作为下一次的旧的按键值；概括说 `Key_Val` 与 `Key_Old` 为相差 10ms 的临时按键值。**

●三行代码第二行：`Key_Down = Key_Val & (Key_Old ^ Key_Val);`两个位操作：按位与，按位异或。

○首先 `Key_Old ^ Key_Val` 位操作针对二进制，二进制与十进制一 一对应。**按位异或：相同为 0，不同为 1**。由下图可知：`Key_Old` 与 `Key_Val` 可能出现的情况：(假设按下的是按键 4)

`Key_Old=0`, `Key_Val=0` 未按下 。`Key_Old ^ Key_Val=0`

`Key_Old=0`, `Key_Val=4` 按下过程中。`Key_Old ^ Key_Val=0100=4`

`Key_Old=4`, `Key_Val=4` 按下稳定期间。 `Key_Old ^ Key_Val=0000`

`Key_Old=4`, `Key_Val=0` 抬起过程中。 `Key_Old ^ Key_Val=0100=4`

再次说明由于是用定时器扫描，`Key_Old` 与 `Key_Val` 的按键值相差 10ms，不可能出现 `Key_Old=4`, `Key_Val=6` 两个按键值的情况。

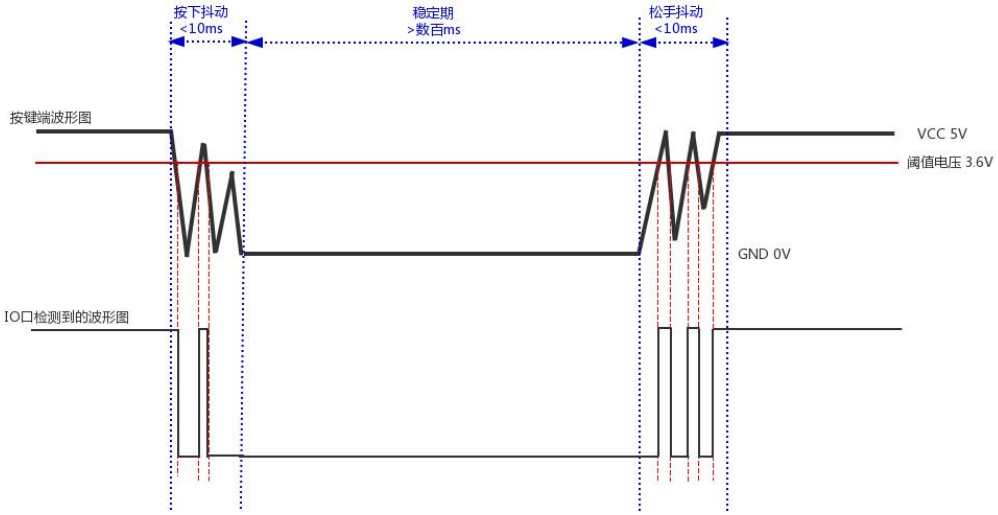
○`Key_Old ^ Key_Val` 的运算结果再`&`上 `Key_Val`

Key_Old	Key_Val	对应的按键过程	Key_Old ^ Key_Val	Key_Down
0	0	未按下	0	0
0	4	按下过程中	0100 (4)	4
4	4	按下稳定期间	0	0
4	0	抬起过程中	0100 (4)	0

所以由上表可知：三行代码第二行 `Key_Down = Key_Val & (Key_Old ^ Key_Val);` 最后的运算结果 `Key_Down` 只有在按键按下的过程中为按键值，持续时间大约 10ms。

●可以在原有三行代码的基础上再增加一行，来判断按键抬起的过程

```
Key_Up   = ~Key_Val & (Key_Old ^ Key_Val);
```



按键过程	Key_Down	Key_Up
未按下	0	0
按下过程中	相应的按键值	0
按下稳定期间	0	0
抬起过程中	0	抬起前的按键值

所以可以将 `Key_Down` 与 `Key_Up` 理解为**临时值**，只在按下或抬起过程中不为 0，又 **按键的扫描** `Key_Read()` 采用定时器进行扫描。10ms 扫描一次，数据 10ms 更新一次。