第八十四节： 中断与中断函数。

【84.1　中断。】

单片机的“中断”跟日常生活的“中断”差不多，你正在做“常事”的时候，突然遇到优先级更高的“急事”，这时你必须先暂停手上的“常事”，马上去处理突如其来的“急事”，处理完“急事”再返回来继续做“常事”。要理解单片机的“中断”，有六个关键点，第一点是“配置中断”，第二点是“做常事”，第三点是“中断请求”，第四点是“保护中断现场”，第五点是“处理急事”，第六点是“返回中断现场”。举个例子如下：

第一点：你老婆随时都会打电话给你，所以你把你的手机 24 小时都打开处于待机的状态。（配置中断）

第二点：你正在读一本书《道德经》（做常事）。

第三点：当你读到第 18 页的时候，你老婆突然给你打电话，让你去幼儿园帮接一下小孩（中断请求）。

第四点：你在第 18 页里夹了一张书签做标记（保护中断现场）。

第五点：你放下手上的书去幼儿园接小孩（处理急事）。

第六点：接了小孩，你回来继续打开《道德经》，找到书签标记的第 18 页（返回中断现场），继续阅读。

上述六点，在单片机的 C 语言里，“配置中断”放在主函数的初始化那里，“做常事”放在主函数的主循环里（main 函数内部的 while(1)循环），“中断请求”单片机内部硬件检测到符合发生中断的条件，“保护中断现场”是单片机内部硬件电路自动处理的（不需要我们软件干涉），“处理急事”是单片机自动跳转到另外开辟的一个特殊中断函数处理（自动跳转是单片机的硬件自动完成不需要我们软件干涉），执行完一次中断函数后单片机再自动跳转到主函数的主循环的现场点继续从现场点开始继续做常事（返回中断现场）。在这六点中，其中第四点的“保护中断现场”与第六点的“返回中断现场”是要特别强调的，单片机从 main 函数的主循环 while(1)准备跳转到中断函数之前，它会自动记录当前的位置（做好路标），以便处理完中断函数后再返回 main 函数的主循环 while(1)时，能找到之前的被中断跳转前的位置，这样就可以接上原来的步骤去处理原来的“常事”，在步骤上既不提前也不滞后恰到好处，中断就不会影响到常事的完整性。代码分布图的模板描述如下：

```
void main()
{
   配置中断;
   while(1)
   {
       处理常事;
   }
}

void 中断函数() interrupt 序号     //中断函数后缀带“interrupt 序号”特别修饰
{
   急事;
}
```

奇怪！上述代码，为什么“main 函数”与“中断函数”在软件上看不到任何关联，既不存在“main 函

数”调用“中断函数”，也不存在“中断函数”调用“main 函数”的情况，在观感上，“main 函数”与“中断函数”仿佛是隔离的毫无“物理连接”的，为什么单片机还能在“main 函数”与“中断函数”两者中切换自如？没错，确实，“main 函数”与“中断函数”在书写上是隔离的毫无关联的，但是它们之间之所以能相互切换，是因为背后有一只无形的手在自动操控这一切，这只手就是单片机硬件自身，这是一种特殊机制，也可以理解成一种特殊的游戏规则，我们只要遵守就好了，除了普通函数，其它凡是中断函数的，都不用跟 main 函数发生软件上的关联调用，它们之间的切换都是硬件自动完成的，这就是 main 函数与中断函数的特殊跳转机制（或者称为游戏规则也可以）。

【84.2　常用的中断函数有哪三类？】

单片机的中断有很多，但常用在项目上的有三类：

第一类是定时中断。配置中断后，使其每隔一段时间就产生一次中断，比如“1ms 一次的定时中断”几乎是所有的系统里的标配，因为它对程序框架起到一个时间节拍的作用。

第二类是通讯中断。比如串口每接收完一个字节就会产生一个中断通知我们去处理。

第三类是电平变化的中断。下降沿或者上升沿的中断，常常用在采集高速的脉冲信号。

【84.3　我们如何操控中断？】

刚才 84.1 提到“单片机硬件自动”这个概念，但是说它“硬件自动”并不意味着它不可控。单片机本身出厂的时候内部就携带了很多种类的中断，这些中断是否开启取决于你的“配置中断”代码，你要开启或者关闭某类中断，只需编写对应的“配置中断”代码就可以，而“配置中断”的代码本质就是填写某些寄存器数值。