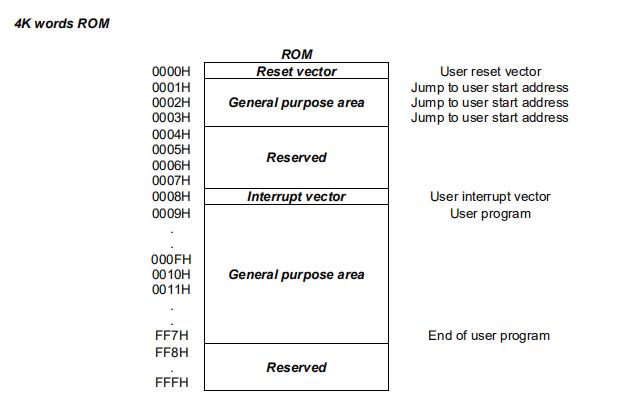
# ROM程序寄存器：

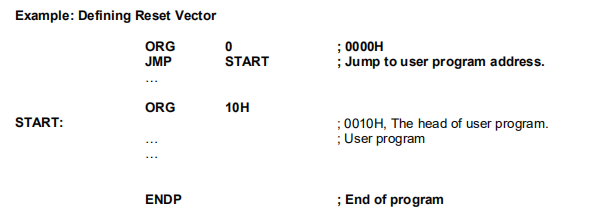


***General purpose area :y=一般用途区***

**RESET VECTOR (0000H) :一字节的区域用来执行系统重置**

**在重置电源或看门狗定时器溢出重置后，芯片将从地址0000h重新启动程序，所有系统寄存器将设置为默认值。**

地址0000H是一个使程序从头开始的“JMP”指令。  
地址0008H是中断向量。  
用户程序是一个循环例程的主要用途应用程序。

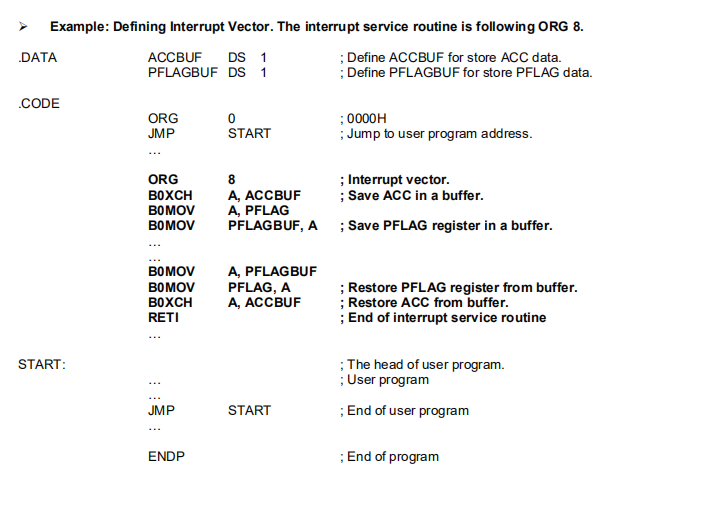


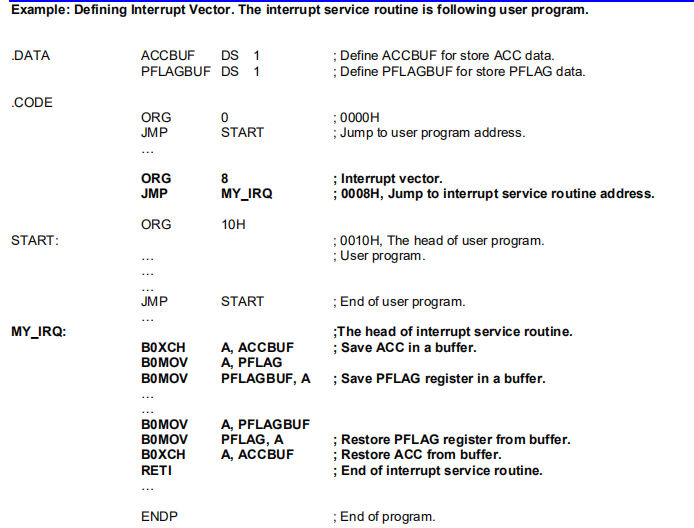
**INTERRUPT VECTOR (0008H) ：一个字节的地址区域用来执行中断请求**

**如果有中断服务执行，则程序计数器(PC)的值存储在堆栈缓冲区中，并跳转到程序内存的0008h来执行矢量中断。**

**注意：当中断发生时，用户必须通过程序保存和加载ACC和PFLAG寄存器。**

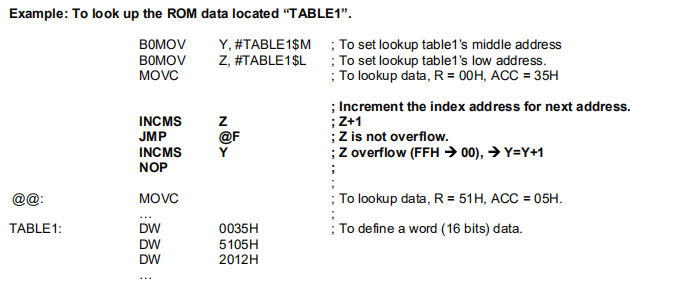
**用户必须定义中断向量。下面的例子展示了在程序内存中定义中断向量的方法。**





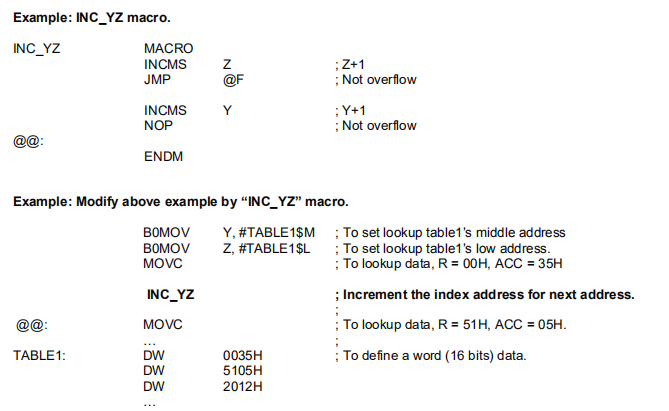
**查找表（LOOK-UP TABLE DESCRIPTION ）MOVC：**

**在ROM的数据查找功能中，Y寄存器指向中间字节地址（位8~位15），Z寄存器指向ROM的低字节地址（位0~位7）。在MOVC指令执行后，低字节数据将存储在ACC中，高字节数据将存储在R寄存器中。**



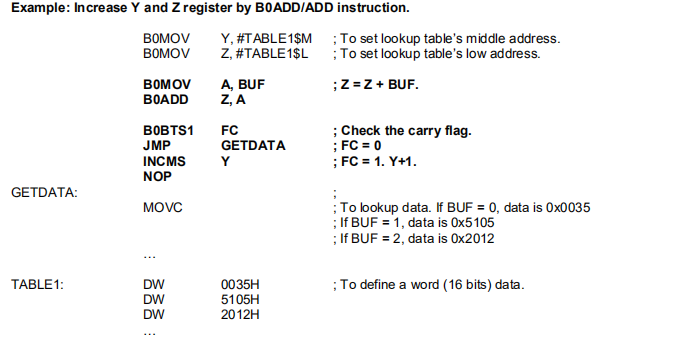
**注意：当Z寄存器跨越边界从0xFF到Y寄存器时，Y寄存器不会自动增加**

**0 x00。因此，用户必须注意这种情况，以避免查表错误。如果Z寄存器溢出，Y寄存器必须加1。**

**下面的INC\_YZ宏展示了一个简单的处理方法Y和Z自动注册。**

@F，就是向下转移到最近的 @@ 处；  
@B，就是向上转移到最近的 @@ 处。

**INCMS溢出则跳过下一条指令执行！**



**通过设置buf的值来控制查表的开始位置**

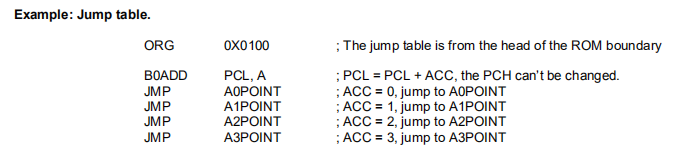
**跳转表操作(JUMP TABLE DESCRIPTION):**

**跳转表操作是一种多地址跳转功能。添加低字节程序计数器(PCL)和ACC值得到一个新的PCL。新的程序计数器(PC)指向一个串行跳转指令作为一个列表表。根据累加器(a)的值，很容易做出多跳程序。**

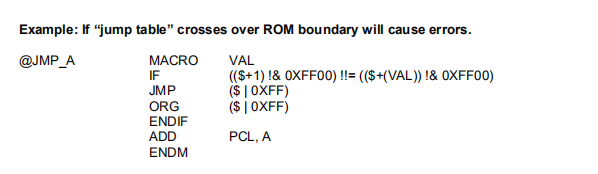
**注意：当执行“ADD PCL, A”后出现进位标志时，它不会影响PCH寄存器。用户必须检查跳转表是否跳过了ROM页边界或SONIX汇编软件生成的列表文件。如果跳转表跳过了ROM页的边界(例如，从xxFFH跳到xx00H)，则将跳转表移到下一个程序内存页(xx00H)的顶部。这里一页256字。**

**执行添加指令后PCL溢出时，程序计数器无法从PCL传输到PCH。**

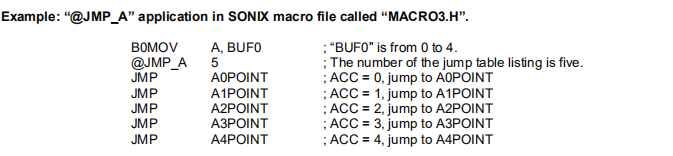
ORG 0 x0100;跳转表来自ROM边界的头部**（好习惯）**

  
在下面的示例中，跳转表从0x00FD开始。当执行B0ADD PCL, a时，如果ACC = 0或1，跳转表指向正确的地址。如果ACC大于1，则会导致错误，因为PCH不会自动增加1。  
我们可以看到，当ACC = 2时，PCL = 0，而PCH仍然保持在0。程序计数器(PC)将指向一个错误的地址0x0000并导致系统操作崩溃。检查跳转表是否越过边界是很重要的  
(xxFFH xx00H)。一个好的编码风格是把跳转表放在ROM边界的开始(例如0100H)。

SONIX提供了一个宏的安全跳转表函数。这个宏会检查ROM的边界，并自动将跳转表移动到正确的位置。这个宏的副作用可能会浪费一些ROM大小。



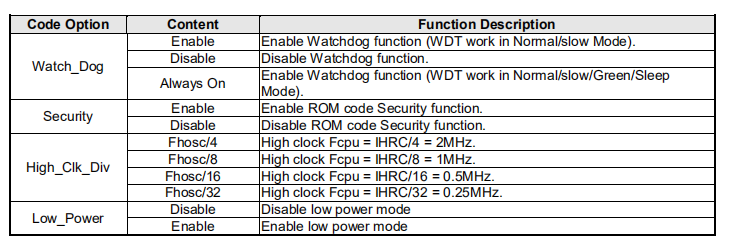
**注意：val代表跳转列表的个数！**



**如果跳转表的位置跨越了ROM边界(0x00FF~0x0100)，“@JMP\_A”宏将从下一个RAM边界(0x0100)开始调整跳转表例程。**

**校验和计算(CHECKSUM CALCULATION ):**

最后一个ROM地址是保留区域。用户应避免这些地址(最后的地址)计算校验和值。



Fcpu代码选项仅适用于高时钟。慢速模式的Fcpu为“Flosc/4”。

在高噪声环境下，强烈建议将Watch\_Dog设置为“Always\_On”。