第六章 实用程序设计

6．1 程序设计方法

程序设计就是用计算机所能接受的语言把解决问题的步骤描述出来，也就是编制计算机的程序。AVR 单片机程序设计语言有：C 编译高级语言和宏汇编汇编语言。

在设计应用系统时，软件的编制是重要环节。软件的质量直接影响整个系统功能的实现。

本章从应用的角度出发，介绍一些实用子程序，读者既可按需要改编调用，也可以吸收其设计方法，以便更好地设计出适合于自己系统的实用软件。

6．1．1 程序设计步骤

应用程序的设计因系统而异，因人而异。尽管如此，程序设计总是有共同特点及其规律的。在编写程序时，设计人员可以采取如下几个步骤：

（1）分析问题，明确所要解决问题的要求。将软件分成若干个相对独立的部分。根据功能关系和时序关系，设计出合理的软件总体结构。

（2）建立正确的数学模型。即根据功能要求，描述出各个输入和输出变量之间的数学关系，并确定采用的计算公式和计算方法。

（3）制定程序框图。根据所选择的计算方法，制定出运算的步骤和顺序，并画出程序框图。这不仅是程序设计的一个重要组成部分，而且是决定成败的关键部分。

（4）合理分配系统资源，包括程序 Flash、E2PROM、SRAM、定时器／计数器、中断、堆栈等。确定数据格式，分配好工作单元，进一步将程序框图画成详细的操作流程。

（5）根据程序的流程图和指令系统，编写出程序。注意在程序的有关位置处写上功能注释，提高程序的可读性。

（6）程序调试。通过编辑软件编辑出的源程序，必须用编译程序汇编后生成目标代码。如果源程序有语法错误，需修改源文件后继续编译，直到无语法错误为止。这之后利用目标码通过仿真器进行程序调试，排除设计和编程中的错误，直到成功。

(7)程序优化。使各功能程序实行模块化、子程序化，缩短程序的长度，加快运算速度和节省数据存储空间，减少程序执行的时间。

6．1．2 程序设计技术

1. 模块化程序设计

模块化程序设计是单片机应用中常用的一种程序设计技术。它是把有关功能完整的、较长的程序，分解为若干个功能相对独立的、较小的程序模块，各个程序模块分别进行设计、编程和调试，最后把各功能模块集成为所需的程序。

模块化程序设计的优点是，单个功能明确的程序模块的设计和调试比较方便、容易完成。一个模块可以为多个程序所共享，也可利用现成的程序模块。

2. 自上而下的程序设计

自上而下的程序设计时，先从主程序开始设计，从属的程序和子程序用符号来代替。主程序编好后，再编制各个从属程序和子程序，最后完成整个系统软件的设计。调试也按这个次序进行。

自上而下程序设计的优点是,比较习惯人们的日常思维，设计、调试和连接同时按一个线索进行，程序错误可以较早发现。缺点是修改比较麻烦。

3 软件抗干扰设计

用于生产现场的单片机应用系统。易受各种干扰侵袭，直接影响到系统的可靠性。因此，应用系统的抗干扰设计是非常重要的。

在实际情况中,针对不同的干扰后果，采用不同的软件对策。在实时数据采集系统中，为

了消除传感器通道中的干扰信号,可采用软件数据滤波，如算术平均法、比较舍取法、中值法、一阶递推数字滤波法等；在开关量控制系统中，为防止干扰进入系统，造成各种控制条件超差、输出失控，可采取软件冗余、程序自检等措施；为防止程序计数器失控，造成程序盲目运行或“死机”，可设置软件“看门狗”。来监视程序运行状态，也可在非程序区设置软件陷井,强行使程序拉回复位状态，重新启动。

6.2　应用程序举例

应用程序中包括一些算术运算、代码转换、数据传送、滤波算法、串行通信、A/D 转换子程序。根据需要，可以应用其中的一些子程序。

6.2.1　内部寄存器和位定义文件

AVR 单片机内部寄存器和位定义文件，用于定义器件内部的寄存器名和寄存器的位名。

在汇编程序文件中如包括了定义文件,则所有数据块中 I/O 寄存器名和 I/O 寄存器位名都能使用。寄存器名用 16 进制地述表示；寄存器位名用 0～7 位表示。

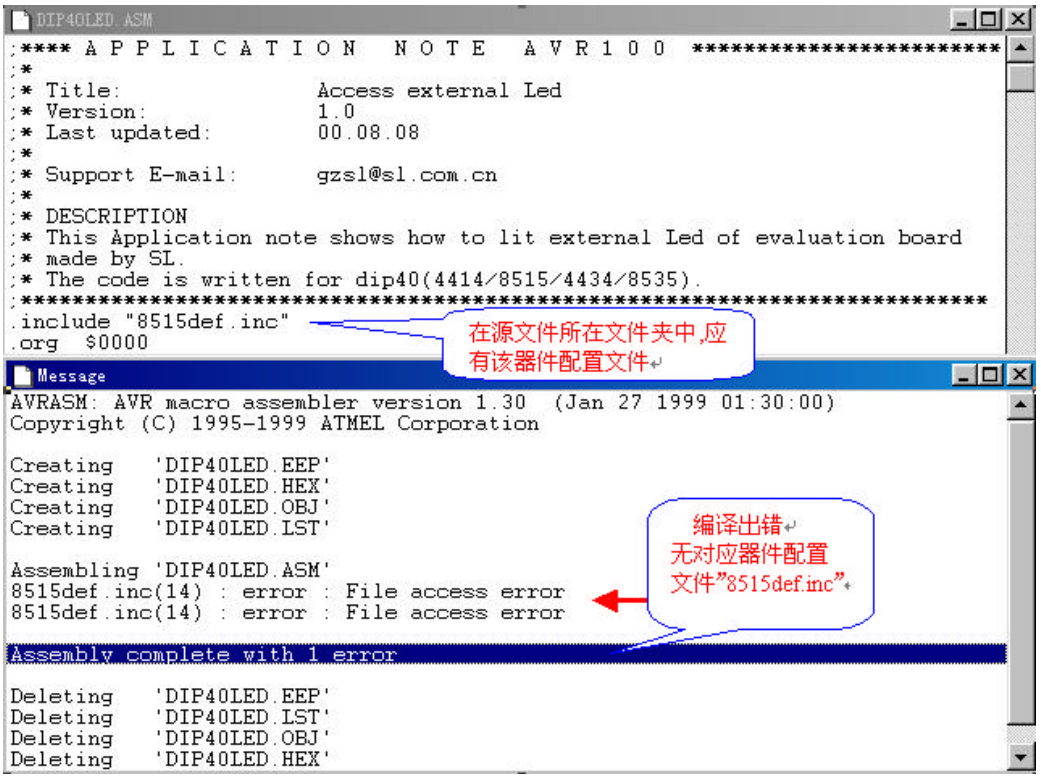
另外,被配置命名的 XL～ZH 6 个寄存器形成 3 个数据指针 X、Y 和 Z，作为内部 SRAM 高端 RAM 地址同样被定义。

注意，在指令中使用的位名是意义不同的。如”sbr”/”cbr”指令表示,若位置位/清零则跳行执行。

AT90S1200 单片机内部寄存器和位定义文件

详见光盘 AT90S1200 器件配置文件 *:\AVR\AVR\asmpack\appnotes\1200def.inc

在源文件编译时,源文件所在的文件夹中一定要有相应器件配置文件存在,不然将造成编译出错提示!



```
;*****************************************************************************
;* 适用于 A V R 系列的应用程序
;* 编号       :AVR000
;* 文件名         :"1200def.inc"
;* 标题       : AT90S1200 的寄存器/位定义
;* 日期           :99.01.28
;* 版本       :1.30
;* 技术支持热线 :+47 72 88 43 88 (ATMEL Norway)
;* 技术支持传真 :+47 72 88 43 99 (ATMEL Norway)
;* 技术支持 E-Mail   :avr@atmel.com
;* 目标 MCU       :AT90S1200
;* 说明
;* 当此文件包含在汇编文件中时，所有 I/O 寄存器名
;* 和 I/O 寄存器位名出现在数据书中以供使用。
;* 寄存器名用它们的十六进制地址代表。
;* 寄存器位名用它们的位编号(0-7)代表。
;* 请观察在指令"sbr"/"cbr" (置/清除寄存器中的位)和"sbrs"/"sbrc"
;* (如寄存器中的位被置 1/清除则跳过)等中使用位名的差别。
;* 以下示例说明了这一点:
;* in    r16,PORTB        ;读取 PORTB latch
;* sbr   r16,(1<<PB6)+(1<<PB5)    ; PB6 和 PB5 置 1 (使用伪装，而不是 bit#)
;* out   PORTB,r16        ;输出到 PORTB
;*
;* in    r16,TIFR         ;读取定时器中断标志寄存器
;* sbrc r16,TOV0          ;检查溢出标志(用 bit#)
;* rjmp TOV0_is_set       ;如为 1 则跳转
;* ...                    ;否则...
;*****************************************************************************

;***** 指定器件
.device AT90S1200

;***** I/O 寄存器定义
.equSREG    =$3f
.equGIMSK   =$3b
.equTIMSK   =$39
.equTIFR    =$38
.equMCUCR   =$35
.equTCCR0   =$33
.equTCNT0   =$32
.equWDTCR   =$21
.equEEAR    =$1e
.equEEDR    =$1d
.equEECR    =$1c
```

```
.equPORTB    =$18
.equDDRB     =$17
.equPINB     =$16
.equPORTD    =$12
.equDDRD     =$11
.equPIND     =$10
.equACSR     =$08
;***** 位定义
.equINT0     =6
.equT0IE0    =1
.equT0V0     =1
.equSE   =5
.equSM   =4
.equISC01    =1
.equISC00    =0
.equCS02     =2
.equCS01     =1
.equCS00     =0
.equWDE      =3
.equWDP2     =2
.equWDP1     =1
.equWDP0     =0
.equEEWE     =1
.equEERE     =0
.equPB7 =7
.equPB6 =6
.equPB5 =5
.equPB4 =4
.equPB3 =3
.equPB2 =2
.equPB1 =1
.equPB0 =0

.equDDB7     =7
.equDDB6     =6
.equDDB5     =5
.equDDB4     =4
.equDDB3     =3
.equDDB2     =2
.equDDB1     =1
.equDDB0     =0
.equPINB7    =7
.equPINB6    =6
.equPINB5    =5
```

```
.equPINB4    =4
.equPINB3    =3
.equPINB2    =2
.equPINB1    =1
.equPINB0    =0

.equPD6 =6
.equPD5 =5
.equPD4 =4
.equPD3 =3
.equPD2 =2
.equPD1 =1
.equPD0 =0
.equDDD6     =6
.equDDD5     =5
.equDDD4     =4
.equDDD3     =3
.equDDD2     =2
.equDDD1     =1
.equDDD0     =0
.equPIND6    =6
.equPIND5    =5
.equPIND4    =4
.equPIND3    =3
.equPIND2    =2
.equPIND1    =1
.equPIND0    =0
.equACD =7
.equACO =5
.equACI =4
.equACIE     =3
.equACIS1    =1
.equACIS0    =0
.equXRAMEND =0
.equE2END    =3F
.equFLASHEND=1FF
.equINT0addr=$001    ;外部中断 0 向量地址
.equOVF0addr=$002    ;溢出 0 中断向量地址
.equACIaddr =$003    ;模拟比较器中断向量地址
.defZL   =r30
```

二、AT90S2313 单片机内部寄存器和位定义文件

详见光盘 AT90S2313 器件配置文件 *:\AVR\AVR\asmpack\appnotes\2313def.inc

在源文件编译时,源文件所在的文件夹中一定要有相应器件配置文件存在,不然将造成编译出错!

三、AT90S4414 单片机内部寄存器和位定义文件

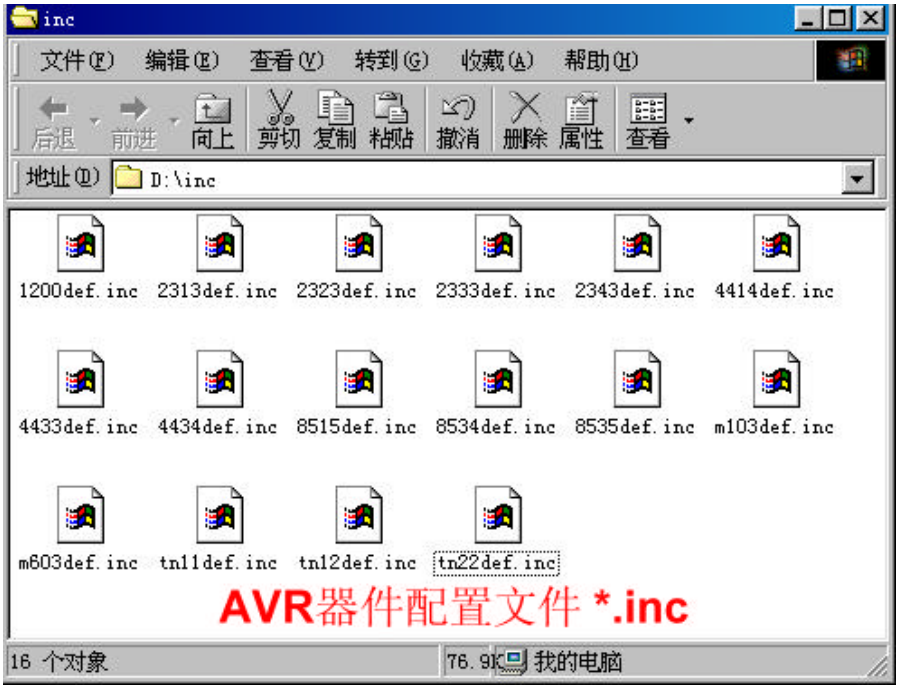
详见光盘 AT90S4414 器件配置文件 *:\AVR\AVR\asmpack\appnotes\4414def.inc
在源文件编译时,源文件所在的文件夹中一定要有相应器件配置文件存在,不然将造成编译出错!

四、AT90S8515 单片机内部寄存器和位定义文件

详见光盘 AT90S4414 器件配置文件 *:\AVR\AVR\asmpack\appnotes\8515def.inc
在源文件编译时,源文件所在的文件夹中一定要有相应器件配置文件存在,不然将造成编译出错!

五、AT90S8535 单片机内部寄存器和位定义文件

详见光盘 AT90S4414 器件配置文件 *:\AVR\AVR\asmpack\appnotes\8535def.inc
在源文件编译时,源文件所在的文件夹中一定要有相应器件配置文件存在,不然将造成编译出错!



6.2.2　访问内部 E2PROM

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR100.ASM
该应用程序说明如何读 E2PROM 数据和写数据到 E2PROM。其中包括：写 E2PROM 子程序“EEWrite”，读 E2PROM 子程序“EERead”，按序写 E2PROM 子程序“EEWrite_seq”、按序读 E2PROM 子程序“EERead_seq”和一个测试程序。

数据块传送

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR102.ASM
该应用程序说明,如何传送程序存储器的一个数据块到 SRAM 和把 SRAM 中的一个数据块传送到另一个 SRAM。其中包括,一个程序存储器块传送子程序”flash 2ram”、SRAM 块传送子程序”ram 2ram”和一个测试程序。

6.2.4　乘法和除法运算应用一

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR200.ASM
该应用程序列出了 8 X 8 位无符号乘法子程序，16X16 位无符号乘法子程序，8/8 位无符

号除法子程序，16／16 位无符号除法子程序和一个测试程序。

6.2.5　乘法和除法运算应用二

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR200B.ASM

该应用程序列出了 8 X 8 位无符号乘法子程序，8 X 8 位带符号乘,16X16 位无符号乘法子程序，16X16 位带符号乘,8/8 位无符号除法子程序，8/8 位带符号除,16／16 位无符号除法子程序,16／16 位带符号除和一个测试程序。

6.2.6　16 位运算

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR202.ASM

该应用程序列出了 16 位加法、16 位带立即数加法、16 位减法、16 位带立即数减法、16 位比较、16 位带立即数比较、16 位取反子程序和一个测试程序。

6.2.7　BCD 运算

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR204.ASM

该应用程序列出了 16 位二进制转换成 BCD、8 位二进制转换成 BCD、BCD 转换成 16 位二进制、BCD 转换成 8 位二进制、二位数压缩 BCD 加法、二位数压缩 BCD 减法和一个测试程序。

6.2.8　冒泡分类算法

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR220.ASM

该应用程序列出了如何用代码有效冒泡分类算法分类 SRAM 的数据块子程序和一个测试程序。

6.2.9　设置和使用模拟比较器

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR400.ASM

该应用程序列出设置和使用模拟比较器的子程序。

6.2.10　半双工中断方式 UART 应用一

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR304.ASM

该应用程序包含了一个非常有效的 UART 软件，并给出一个接收字符，然后返回一个应答的子程序。

6.2.11　半双工中断方式 UART 应用二

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR305.ASM

该应用程序给出一个怎样用 8 位定时器／计数器 0 和外部中断实现半双工 UAR 的软件。

6.2.12　8 位精度 A／D 转换器

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR401.ASM

该程序显示如何使用单片机片内模拟比较器和几个外部元件实现双斜率 A／D 转换器，包括一个测试程序，完成外循环的转换并输出到 8 位 LED 显示。

6.2.13　装载程序存储器

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR108.ASM

```
;* 标题:     装载程序存储器
;* 版本:     1.0
;* 最后更新:98.02.27
```

```
;* 目标器件:     AT90Sxx1x 及更高档的器件(带 SRAM 的器件)
;* 技术支持 E-mail: avr@atmel.com
;* 说明
;* 此应用程序说明怎样使用装载程序存储器(LPM)指令。
;* 此应用程序从程序存储器中一字节一字节地装入字符串 "Hello World"
;* 并输出到 B 口。
```

6.2.14　安装和使用相同模拟比较器

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR128.ASM

```
;* 标题:     安装和使用模拟比较器
;* 版本:     1.1
;* 最后更新:97.07.04
;* 目标器件:     AT90Sxxxx (带模拟比较器的器件)
;* 技术支持 E-mail: avr@atmel.com
;* 说明:
;* 此应用程序说明怎样激活和使用 AVR 自带的精密模拟比较器。
;* 此程序执行以下操作:
;* - 轮流检测模拟比较器的输出以等待正输出边沿
;* - 轮流检测中断标志以等待正输出边沿
;* - 比较器输出模式下中断使能。每次中断程序执行时将一个 16 位计数寄存器加 1。
```

6.2.15　CRC 程序存储的检查

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR235.ASM

```
;* 标题:     CRC 程序存储器校验
;* 版本:     1.0
;* 最后更新:98.06.15
;* 目标器件:     AT90Sxxxx (所有支持 LPM 指令的 AVR 器件，除了 AT90S1200)
;* 技术支持 E-mail: avr@atmel.com
;* 注意: 经常登陆 Atmel 的网站，www.atmel.com 以获取软件的最新版本。
;* 说明
;* 此程序说明利用一个简单的算法怎样来实现代码存储器内容的 CRC 计算。
;* 要产生 CRC 校验则寄存器“状态”装入 00 并调用"crc_gen"程序。
;* 校验的结果存于寄存器的第二字节（低字节）和第三字节（高字节）。
;* 要检查 CRC 校验寄存器“状态” 装入 FF 并调用"crc_gen"程序。
;* 校验的结果存于寄存器的第二字节（低字节）和第三字节（高字节）。
;* 如果校验的结果为 00 则程序代码是正确的，
;* 如果校验的结果不是 00 则程序代码有错误。
```

6.2.16　4X4 键区休眠触发方式

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR240.ASM

```
;* 标题:     4x4 键盘，按键唤醒
;* 版本:     1.1
;* 最后更新:98.10.22
;* 目标器件:     所有 AVR 器件
```

;* 技术支持 E-mail: avr@atmel.com
;* 说明
;* 此程序用来扫描一个 4 x 4 键盘并用休眠模式使 AVR 由按键唤醒。
;* 此设计使用最少的外部器件。
;* 测试程序唤醒 AVR 并扫描，当有键按下时根据所按键的键号点亮两个 LED 中的一个。
;* 外部中断线作唤醒之用。
;* 此例应用于 AT90S1200，但在向量、E2PROM 和堆栈指针上作一些适当的改动即适用于
;*N 任何 AVR 器件。
;* 定时采用一个 4 MHz 的时钟。
;* 一个查询表用于 E2PROM 中以使同样的结构被用于更高级的程序，例如 ASCII 输出到显示。

6.2.17　多工法驱动 LED 和 4X4 键区扫描

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR242.ASM
;* 标题:　　多路 LED 驱动和 4x4 键盘采样
;* 版本:　　1.0
;* 最后更新:98.07.24
;* 目标器件:所有 AVR 器件
;* 技术支持 E-mail: avr@atmel.com
;* 说明
;* 此程序包括一个提供 24 小时工业定时器或以 I/O 引脚作双重作用的实时时钟。
;* 通过 4 x 4 矩阵键盘输入，输出到多元四位 LED 显示并通过附加接口电路由两开/关输出
;* 驱动负载。
;* 此例中驱动的是 LED 负载但附加适当的器件就可驱动任何负载。
;* 触觉反馈是通过按键时一个音响器发出嘟嘟响声来实现。
;* 主程序允许通过键盘设定时钟，每 24 小时为每个负载进行开/关时间设定，
;* 实时时钟功能、键扫描功能和调整程序。
;* 此例以 AT90S1200 来显示怎样克服有限的 I/O，
;* 此例同样适用于任何 AVR 器件，仅需对矢量、E2PROM 和堆栈指针作适当的修改。
;* 定时采用一个 4.096 MHz 晶振（如果 178 代替 176 用在定时器装载次序中，4 MHz 晶振
;* 会产生-0.16%;
;* 的误差，但这可以在软件中以规则的时间间隔来调整）。
;* 查询表格用在 E2PROM 中解码显示数据，附加的字符用于定时和开/关设定显示及键盘转换表。
;* 如果您的应用用到 E2PROM，则表格可移到较大的 AVR 器件的 ROM 中。

6.2.18　I2C 总线

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR300.ASM
;* 标题　　: I2C (单)主器件执行
;* 版本　　: 1.0 (BETA)
;* 最后更新　　: 97.08.27
;* 目标器件　　: AT90Sxxxx (所有 AVR 器件)
;* 技术支持 email　: avr@atmel.com
;* 说明
;*　与 I2C 从器件通信的基本程序。
;*　在 I2C 总线上单一主器件执行受限制于单一总线控制器。

```
;*  绝大部分应用并不需要 I2C 总线提供的多主器件功能。
;*  到目前为止，单一主器件执行使用较少的资源并有较少的晶体频率依靠。
;*  特点:
;*  * 全部中断可用。可用于其他用途。
;*  * 支持正常和快速模式。
;*  * 支持 7-bit 和 10-bit 寻址。
;*  * 支持 AVR 全系列微控制器。
;*  I2C 主程序 :
;*  'i2c_start' -    发出起始条件并发送地址和传递方向。
;*  'i2c_rep_start' -    发出重复的起始条件并发送地址和传递方向。
;*  'i2c_do_transfer' - 根据 address/dir 字节中给出的方向发送或接收数据。
;*  'i2c_stop' -     通过发出停止条件终止数据传递。
;* 用法
;*  传递格式如 AVR300 文件中所述。
;*  (主代码中有一例)。
;* 注意
;*  I2C 程序可被非中断或中断程序的任意一个调用，非两者同时调用。
;* 统计
;*  代码量 : 81 个字 (最多)
;*  寄存器占用 : 4 高，0 低
;*  中断占用: 无
;*  其他占用: D 口的两个 I/O 引脚
;*  XTAL 范围    : N/A
```

6.2.19　I2C 工作

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR302.ASM

```
;* 标题      : I2C 从器件执行
;* 版本      : 1.2
;* 最后更新      : 97.07.17
;* 目标器件      : AT90Sxxxx (所有 AVR 器件)
;* 快速模式      : 仅 AT90S1200
;* 技术支持 email    : avr@atmel.com
;* 代码量            : 160 个字
;* 低寄存器占用      : 0
;* 高寄存器占用      : 5
;* 中断占用 : 外部中断 0,
;*   定时器/计数器 0 溢出中断
;* 说明
;*  此程序说明怎样实现 AVR AT90S1200 作为一个 I2C 从外部设备的应用。
;*  如使用其他 AT90S AVR 器件，则可能不得不用到其他的 I/O 引脚，堆栈指针也必须初始化。
;*  接收的数据存储在 r0 中，r0 中的数据在主器件读传递中被传输。
;*  这个简单的协议仅用于演示的目的。
;*  特点 :
;*  * 基于使用 INT0 和 TIM0 的中断。
```

```
;*  * 器件可接收任何的 7 位地址。（可扩展到 10 位）
;*  * 支持标准和快速模式。
;*  * 易插入“等待状态”。
;*  * 容量和速度优化。
;*  * 支持从闲置模式唤醒。
;*  参考应用程序 AVR302 文件以获得更详细的说明。
;* 用法
;*  在两个标记“插入用户代码”的地方插入用户代码。
;* 注意
;*  每一个中断之间最少会有一条指令被执行。
;* 统计
;*  代码量   : 160
;*  寄存器占用   : 5 高，0 低
;*  中断占用: EXT_INT0 和 TIM0_OVF
;*  端口占用: PD4(T0) 和 PD2(INT0)
;*  XTAL     : - I2C 快速模式      : 最低 16.0MHz
;*             - I2C 标准模式 : 最低  3.0MHz
```

6.2.20　SPI 软件

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVR320.ASM

```
;* 标题      : 软件 SPI 主器件
;* 版本      : 1.0
;* 最后更新      : 98.04.21
;* 目标器件      : AT90S1200
;* Easily modified for   : Any AVR microcontroller
;* 技术支持 E-mail   :avr@atmel.com
;* 说明
;*  这是一个集成了 8/16 位字，模式 0，主器件 SPI 的程序。
;*  它同时传输和接收 8/16 位字格式的 SPI 数据。
;*  数据首先发送和接收 MSB。
;*  一个寄存器组用来发送和接收；比如，当一位移出
;*  (被传输)，空余的位就用来存储新的位。
;*  这些程序是低电平接口程序，
;*  因而不包含一个命令结构；
;*  那是由联接的 SPI 外设决定的。
;*  因为有分离的 允许/禁止 和 读/写字程序，
;*  大块的数据只能在禁止/SS 之前通过多次调用
;*  RW_SPI 程序来发送。
;* 主程序:
;*  init_spi: 初始化口线用作 SPI。
;*            无需调用，返回。
;*  ena_spi: 迫使 SCK 为低，激活 /SS 信号。
;*            无需调用，返回。
;*  disa_spi: 令 /SS 信号为高 (去激活)。
```

```
;*           无需调用，返回。
;*  rw_spi:   发送/接收一个 8 位或 16 位数据字。
;*           在调用之前必须在(spi_hi,spi_lo)中设置数据发送;
;*           返回接收的数据到同一寄存器组中
;*           (如是 8 位，仅使用 spi_lo 寄存器)。
;* 变量:
;*  spi_hi 和 spi_lo 变量是高和低数据字节。
;*  它们可在寄存器文件中的任何地方。
;*  临时变量用来位计数，也用作高/低最小脉冲宽度定时。
;*  这必须位于上部寄存器
;*  因为使用了一条 IMMEDIATE 模式指令。
;* 备忘
;*  V1.098.04.21 (rgf) 发表
```

6.2.21　验证 SL-AVR 实验器及 90S1200 的口功能

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVRLED.ASM

该程序验证 OK-AVR 万用串行下载开发实验器及 AT90S1200 的 A 口、B 口 LED 灯亮灭程序,也同时验证实验器通讯接口连机正常与否,avrled2.asm 和 avrled3.asm 仅延时常数不同,所以 LED 闪动快慢不同;

6.2.22　验证 SL-AVR 实验器及 90S1200 的口功能

见光盘文件*:\AVR\AVR\asmpack\appnotes\AVRSTEP.ASM

该程序用模拟调试单步验证 AT90S1200 B 口、D 口的输出状态;

6.2.23 验证 SL-AVR 实验器及具有 DIP40 封装的口功能

见光盘文件*:\AVR\AVR\asmpack\appnotes\DIP40LED.ASM

该程序验证 OK-AVR 万用串行下载开发实验器及具有 DIP40 封装的 AT90S4414/AT90S8515/AT90S8535 等器件的 A 口、B 口、C 口、D 口 LED 灯亮灭程序,可修改延时常数;