中颖单片机入门与实战

张学峰　计万里　万清峰

张文臣　孙志俊　萧　斌

编著

中颖电子股份有限公司

前言

单片机又称单片微处理器，其应用已渗入到各行各异业，生产厂家亦从二十年前的寥寥几家发展到现在的几十家甚至更多。不同的厂家基于各自的架构平台，设计了不同功能特点的单片机，这就使得工程师们可以按照具体设计要求挑选最适合的一款芯片进行系统开发，既满足功能需求又能最大限度降低成本，提高了自己产品的性价比。

中颖单片机基于公司自有的 4-bit CPU IP(CPU60)发展起来，芯片采用的是程序内存和数据存储器在物理空间上完全独立的哈佛结构。程序内存和数据存储器地址和总线完全分开，可以使指令和数据有不同的数据宽度。同时由于读取指令和存取操作数可以同时进行（流水线作业），因而具有较高的执行效率。中颖设计工程师以此设计了 SH66XX, SH67XX 和 SH69XX 等一系列的单片机，涵盖了包括消费类，家电及来电显示电话的多方面应用，以其产品的多样化，优异的抗干扰性能，良好的性价比和及时的售后服务在竞争激烈的市场占有一席之地，并且每年的出货量在持续快速的增长中。中颖单片机能在短短数年间取得如此成绩及市场认可度，自有其道理。

考虑到目前中颖电子没有一本针对入门者的书籍，因此集合了中颖的 SA 与 FAE 共同来完成这本适合入门者或初次接触中颖产品的工程师的书籍。本书一共分为五个章节，包括中颖单片机介绍，指令集介绍，硬件资源介绍，开发工具介绍及应用实例。其中在硬件资源一章中具体介绍了中颖单片机各种功能模块的应用且辅以程序实例，让学习者更透彻地理解功能应用及扎实地掌握编程技巧。另外本书还在第五章选择了一些基础的应用实例，详细地介绍了其原理及编程方法，希望对初学者有所益助。

由于中颖还是第一次编辑这类书籍，时间上也比较匆忙，书中的错误和不妥之处在所难免，恳请各位读者批评指正。您可直接来电或 EMAIL 给我们与我们分享您的心得，帮助我们不断地完善数据提高服务质量。另外各位读者亦可从中颖股份有限公司网站主页 http://www.sinowealth.com 上获取更多的资料。在此还要感谢张学峰，计万里，万清峰，张文臣，孙志俊，朱金海及相关 SA, FAE 为此书编辑付出大量心力与时间。

萧斌
2005 年 10 月

目录

第一章

SinoWealth 4-bit 单片机基本介绍

1.1 Sino Wealth 4-bit单片机产品概述与分类

4-bit 单片机产品线是中颖公司（SinoWealth）众多产品线之一.其所有产品均基于中颖公司自有的 4-bitCPU IP(CPU60)发展起来的，产品系列齐全，应用场合广泛。

1.1.1 CPU的特点

每类 MCU 产品的应用场合都有所不同，对 CPU 的一些特性要求也有所差异。对应这些差异，CPU60 分为 CPU6610C，CPU6610D，CPU6610E 三种。其主要区别是在电路动静态结构和堆栈层数上。

CPU6610C：动态电路结构，堆栈的层数为 4 层。
CPU6610D：全静态电路结构，堆栈的层数为 4 层。
CPU6610E：全静态电路结构，堆栈的层数为 8 层。
每颗产品采用的 CPU 类型在产品的数据手册的首页均有标明。

1.1.2 存储器架构

SinoWealth 4-bit 单片机存储器架构采用的是适合单片机应用的哈佛结构。哈佛结构是一种将程序存储器和数据存储器在物理空间上完全独立,读取指令和存储数据的总线完全分开的一种存储器架构。中央处理器(CPU)首先到程序存储器中读取指令，进行解码，得到数据地址，再到相应的数据存储器中读取数据并进行下一步的操作。程序存储器和数据存储器地址和总线完全分开，可以使指令和数据有不同的数据宽度。同时由于读取指令和存取操作数可以同时进行（流水线作业），所以哈佛结构的处理器通常具有较高的执行效率。

1.1.3 内核设计的Pipeline流水线结构

SinoWealth 4-bit 单片机在内核设计方面是采用 Pipeline 流水线结构。程序指令的执行过程如下图 1-1：

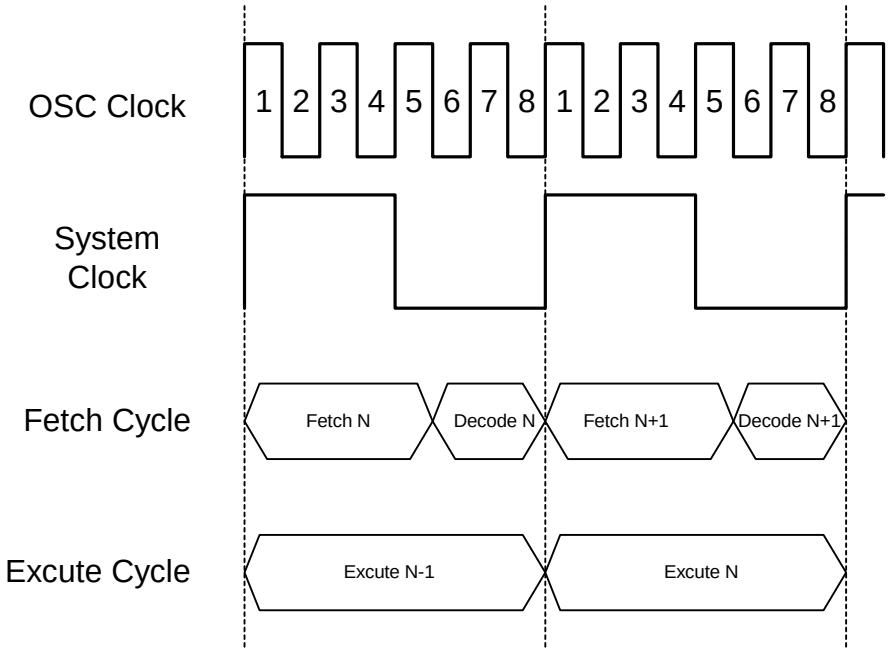


图 1-1 系统指令执行示意图

系统执行过程大致可分为读指令、指令译码、指令执行等几个阶段。一个 CPU 的系统周期包含 4 个机器周期。

在一个系统周期的第 1 到第 5 个机器时钟周期期间读第 N 条指令，在第 6 到第 8 个机器时钟周期内执行第 N 条指令的译码动作，同时在整个系统周期内执行第 N-1 条指令（上一条指令），如此循坏，在第二个系统周期内读入/译码第 N+1 条指令并执行第 N 条指令…

1.1.4 RISC结构的指令系统

SinoWealth 4 bit 单片机采用的是 RISC（精简指令集）结构的指令集。

任何指令均在一个系统时钟周期内完成。程序区任意位置取出的指令都是一条完整的指令，这些特点对于对实时性和抗干扰性要求都很高的 MCU 应用场合是很重要的。

1.1.5 SinoWealth 4-bit单片机产品分类

按照资源特性可分为：

- I/O 类(SH6x(P)2x 和 SH6x(P)3x 系列),如 SH6P20A，SH69P31 等；
- LCD 类(SH6x(P)5x 系列),如 SH69P54,SH69P58 等；
- ADC 类(如 SH6x(P)4x 系列)，如 SH69P44，SH69P48 等；
- PWM 类,如 SH69P44 等
- …

按照工作电压应用范围可分为：

- 低压类，如 SH6xLxx 系列,适合 1.5V 单节电池或太阳能电池等应用场合；
- 单电压+3.0V/+5.0V 类；
- 宽电压应用范围类；

- …

按照 ROM 类型可分为：

- MASK ROM 类(掩膜)；
- OTP(one time programming)类；

按照抗干扰等级可分为：

- SH65/66 系列，适合对系统抗干扰能力要求较低的应用场合；
- SH67 系列，适合对系统抗干扰能力要求较高的应用场合；
- SH69 系列，适合工业规格，对系统抗干扰能力要求极高的应用场合，如家电应用场合；

SinoWealth 4-bit 单片机系列产品较多，但由于其所有产品是基于同一颗 CPU，同一个开发环境(IDE)，同一套 RISC 指令集，系统架构简单明了，开发工具界面友好，资源配置丰富，所以在各个应用场合，从低端到高端，都有 SHxxxx 系列 MCU 的身影。

1.2 SinoWealth 4-bit单片机产品基本特性

4-bit 单片机所有产品的基本架构都是以 CPU60 为核心，配置必需的 ROM/RAM 模块，时钟产生电路，复位电路和依据特定的应用场合而配置的外围功能模块而组成，如图 1-2-1。

以下着重简单介绍一下 CPU，ROM 和 Register/RAM 和省电模式部分，其余部分将在后续的章节中介绍。

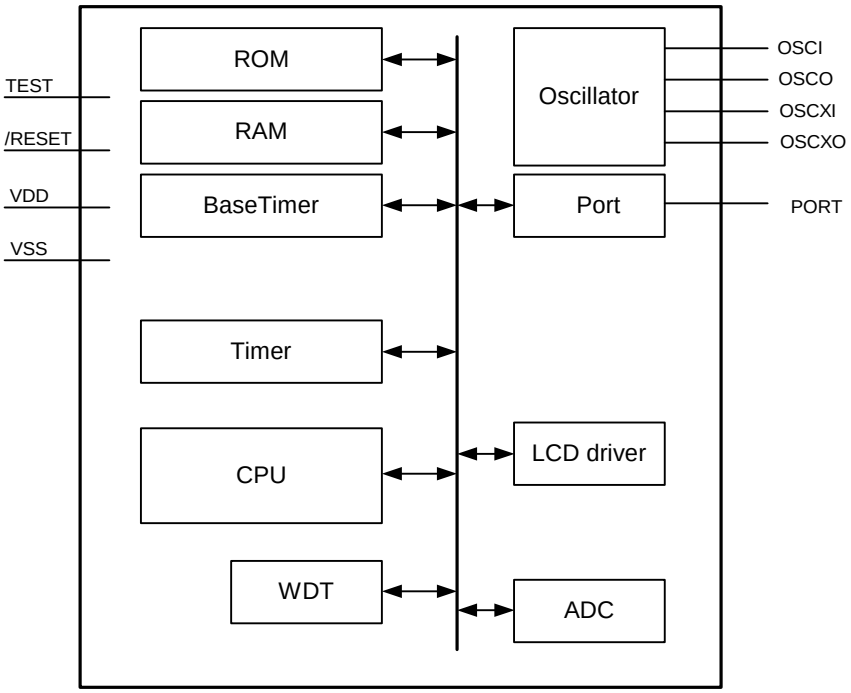


图 1-2-1 MCU 系统框图

1.2.1 CPU

CPU 包含以下功能模块：程序计数器，算术逻辑单元 (ALU)，进位标志，累加器，查表寄存器，数据指针 (INX，DPH，DPM，和 DPL)和堆栈。

PC (程序计数器)

程序计数器用于寻址 4K 程序的 ROM 空间。该计数器共有 12 位：页寄存器 (PC11)，和循环进位计数器 (PC10，PC9，PC8，PC7，PC6，PC5，PC4，PC3，PC2，PC1，PC0)。

通常每执行一条指令程序计数器的值加一，但在下列情况下除外：

(1) 当执行一条跳转指令时 (例如 JMP，BA0，BAC)；

(2) 当执行子程序调用指令时 (CALL)；

(3) 当发生中断时；

(4) 当芯片处于 INITIAL RESET 模式时。

程序计数器中装入相应的指令地址。将页寄存器中置为 1，无条件跳转指令 (JMP) 将指向高于 2K 的地址。

ALU和CY

ALU 执行算术和逻辑操作。ALU 有以下功能：

二进制加法/减法 (ADC，SBC，ADD，SUB，ADI，SBI)

加法/加法的十进制调整 (DAA，DAS)

逻辑操作 (AND，EOR，OR，ANDIM，EORIM，ORIM)

判断 (BA0，BA1，BA2，BA3，BAZ，BNZ，BC，BNC)

逻辑移位 (SHR)

进位标志 (CY) 保存了算术操作后 ALU 的溢出状态。在中断服务或子程序调用过程中，进位标志被压入堆栈，并遇 RTNI 指令后返回。RTNW 指令不影响进位标志。

累加器A

累加器 A 是一个四位寄存器，其中保存了算术逻辑单元的运算结果。它和 ALU 一起，能完成与系统寄存器，LCD RAM，或数据存储器之间的数据传送。

堆栈（Stack）

该组寄存器能在每次调用子程序或中断时按次序保存 CY 和 PC (11-0) 的值。它的结构为 13 位×4 层。最高位为 CY 保留。SH6xxx 系列单片机最多允许有 4 层或 8 层子程序调用和中断。当遇到返回指令 (RTNI/RTNW) 时，堆栈中的数据将按顺序返回至 PC 中。堆栈中的数据是按照先进后出的方式处理。4 层嵌套包括调用子程序和中断请求之和。注意如果调用子程序和中断请求的数量和大于 4，程序的执行将出现异常，此时堆栈最底部的数据将溢出，PC 值将被清零，程序被复位。

1.2.2 ROM

SH6xxx 的 ROM 字长为 16 位。

SH6xxx 的程序存储器最大寻址范围为 32K X 16 位，地址由$0000 到$7FFF。但由于 PC 值只有 12 位，最多只可以访问 4K 的 ROM 空间，所以在访问 ROM 时引入 BANK 的概念，跨 BANK 区域的操作称为翻 BANK。如表 1-2-1 我们将 SH6xxx 对应的 ROM 空间分为 BANK0，BANK1，…。

CPU Address	BANK Register							
	= 0	= 1	= 2	= 3	= 4	= 5	= 6	= 7
000 - 7FF	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)
800 - FFF	0800 - 0FFF (BANK 1)	1000 - 17FF (BANK 2)	1800 - 1FFF (BANK 3)	2000 - 27FF (BANK 4)	2800 - 2FFF (BANK 5)	3000 - 37FF (BANK 6)	3800 - 3FFF (BANK 7)	4000 - 47FF (BANK 8)
CPU Address	BANK Register							
	= 8	= 9	= A	= B	= C	= D	= E	
000 - 7FF	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	0000 - 07FF (BANK 0)	
800 - FFF	4800 - 4FFF (BANK 9)	5000 - 57FF (BANK 10)	5800 - 5FFF (BANK 11)	6000 - 67FF (BANK 12)	6800 - 6FFF (BANK 13)	7000 - 77FF (BANK 14)	7800 - 7FFF (BANK 15)	

表 1-2-1 SH6xxx ROM 的物理空间定义

翻 BANK 的动作只有 JMP 指令才可以完成，通过下面的例程可以清晰的看出翻 BANK 的动作如何实现，基本思路是先对 BANK Register 进行赋值，再执行 JMP Lable 指令即可跳到相应的位置。

例 1-2-1：BANK 之间的跳转

```
                ORG         0000H
                JMP         MAIN
                …
MAIN:           LDI         IE,0
                …
LOOPBNK0：                              ;此段程序位于物理空间 BANK0 中
                …
                LDI         BNK,00H
                JMP         LOOPBNK1    ;执行完毕，跳转至 BANK1 中的 LOOPBNK1 位置
                …
LOOPBNK1：                              ;此段程序位于物理空间 BANK1 中
                …
                LDI         BNK,01H
                JMP         LOOPBNK2    ;执行完毕，跳转至 BANK2 中的 LOOPBNK2 位置
                …
LOOPBNK2：                              ;此段程序位于物理空间 BANK2 中
                …
                LDI         BNK,02H
                JMP         LOOPBNK3    ;执行完毕，跳转至 BANK3 中的 LOOPBNK3 位置
                …
LOOPBNK3：                              ;此段程序位于物理空间 BANK3 中
                …
                LDI         BNK,00H
                JMP         LOOPBNK0    ;执行完毕，跳转至 BANK0 中的 LOOPBNK0 位置
                …
```

例 1-2-2：任何其它 BANK 与 BANK0 之间的跳转

如果 JMP Lable 指令中 Lable 的物理位置是位于 BANK0 中，则无论 JMP Lable 指令位于那个 BANK，都可以直接跳转，而不管 BANK Register 为何值。

```
                ORG         0000H
                JMP         MAIN
                …
MAIN:           LDI         IE,0
                …
LOOPBNK0：                              ;此段程序位于物理空间 BANK0 中
                …
                …
                …

                …
LOOPBNK1：                              ;此段程序位于物理空间 BANK1～15 任意位置
                …
                …
                JMP         LOOPBNK0    ;执行完毕，无论 BANK Register 为何值，跳转至 BANK0
                                        ;中的 LOOPBNK0 位置
                …
```

涉及 ROM 跳转的指令有 BAZ、BNZ、BC、BNC、BA0、BA1、BA2、BA3、CALL、RTNW、RTNI、JMP、TJMP 等 11 条，下面从指令的操作码来分别讨论其跳转的 ROM 地址访问范围。

a) BAZ、BNZ、BC、BNC、BA0、BA1、BA2、BA3、CALL 指令

SH6xxx 指令系统属于 RISC 结构，每一个操作均由一条指令，在一个系统周期内完成，每条指令的长度均为 16bits(1word)。

上述指令中操作码均为 5 位，操作数的长度为 11 位，而这些指令的操作数表示的是正

是待跳转的 ROM 绝对地址，地址范围为 2K，所以这些指令和指令要跳转的地址必须在同一个 BANK 中。

指令	指令码		寻址范围
	操作码 (5 bits)	操作数 (11 bits)	
BAZ Lable	10010	xxxxxxxxxxx	2K
BNZ Lable	10000	xxxxxxxxxxx	2K
BC Lable	10011	xxxxxxxxxxx	2K
BNC Lable	10001	xxxxxxxxxxx	2K
BA0 Lable	10100	xxxxxxxxxxx	2K
BA1 Lable	10101	xxxxxxxxxxx	2K
BA2 Lable	10110	xxxxxxxxxxx	2K
BA3 Lable	10111	xxxxxxxxxxx	2K
CALL Lable	11000	xxxxxxxxxxx	2K

b) TJMP 指令

TJMP 指令无操作数

指令	指令码	功能	寻址范围
TJMP	11110 1111 111 1111	PC←(PC11～8)(TBR)(A)	256

意义：

下一条执行指令的地址：

高三位 ⇐ 当前指令地址的高三位

中四位 ⇐ TBR 寄存器的值

低四位 ⇐ A 寄存器的值

相对于 TJMP 指令的位置而言，下一条执行指令的寻址范围为 8 bits，即 256.

c) RTNI, RTNW

当有 CALL 指令或有中断发生时，下一条指令的地址被自动压栈至 stack 中，RTNI/RTNW 正是与之对应的返回指令，返回时，PC 值被 stack 中保存的 PC 数据赋值。由于 Stack 中的 PC 值为 12bits，所以 RTNI/RTNW 指令的寻址范围为 4K。

中断服务程序的起始向量地址如表 1-2-2.

地址	说明
$000	上电复位/按键复位/低电压检测复位程序入口地址
$001	中断入口地址 1

$002	中断入口地址 2
$003	中断入口地址 3
$004	中断入口地址 4

表 1-2-2 中断服务程序的起始向量表

具体产品间中断向量的差异主要体现在：

a) SH6xxx 所有产品最多提供 4 个中断入口;
b) 由于每颗产品的应用定位不同，所以具体到每一颗的中断入口的具体数量有所差异；
c) 具体产品的中断入口的具体定义会有所差异；
d) 中断入口向量数目不等同于中断源的数目；

1.2.3 寄存器(Register) & RAM

SH6xxx 所有产品均内部集成了**寄存器**(Register)和 SRAM，依据产品定位和应用场合不同，集成的 Register 和 SRAM 数目都有所差异.

MCU 内部 CPU 的运行状态，I/O Port 的模式定义/切换，Timer 的模式定义，特殊模块的控制等都是通过操作 Register 来实现的。

一般而言，依据功能模块，Register 可以分为以下几类：

- 中断控制和中断标志类，如 IE，IRQ 等；
- I/O 模式定义和数据类，如 PACR，PORTA 等；
- 定时器（Timer）模式定义和数据类，如 TMOD，TL0，TH0 等；
- 间接寻址类，如 INX，TBR，DPH，DPM，DPL；

以上四类 register，属于 SH6xxx 产品的基本 Register，每颗产品都包含有。

- LCD 模式定义和控制类，如 LCDM，LCDCNTR 等；
- 其它功能模块寄存器，如 PWM/ADC/DAC/Tone Generator/R to F 等；

SRAM 是客户程序用于数据或运行变量的保存/暂存的随机存储单元，一般可以分为通用 RAM 和 LCD/LED RAM 两种，前者为单口 RAM，用作通用目的，后者为 LCD/LED 显示数据的存储区，一般为双口 RAM，一方面用户只需将需要显示的内容按照显示位置对应表写入对应的 LCD RAM 中，另一方面，由 MCU 内部 LCD/LED driver 硬体部分将 LCD/LED RAM 中的内容读出，产生对应的 LCD/LED 模拟输出信号，从 LCD/LED Segment/Common 输出脚输出。

通用 RAM 同基本的 Register 一样，是每一颗 SH6xxx 产品都包含有的，只是依据产品定位和应用场合不同，其数目或单元的数量不同而已，而 LCD/LED RAM 则包含在带 LCD 或 LED 驱动功能的产品中，同时依据驱动的 LCD 点数/LED 的段数，其数目或单元的数量有所不同。

另外依据 Register 或 SRAM 的控制对象的差异， Register 或 SRAM 的读写属性可能也有所不同，表现在两个方面：

- 有些 Register 或有些 SRAM 单元只有只写属性(write-only), 或只读属性(Read-only)，而大部分的 Register 和 SRAM 的属性是可读可写(R/W)；
- 同一个 Register 或 SRAM 位置，其读写属性是可读可写(R/W)，但是其读取的数据来源地(Source)和写入的数据目的地(Desnation)是不一样的；

所有 Register 和 SRAM 的读写属性在每颗产品的数据手册中均有详细而明确的定义。

1.2.4 省电待机模式(HALT和STOP)

中颖公司 SH6xxx 产品线所有的产品均提供两种省电模式，待机模式 1(HALT)和待机模式 2(STOP)。这两种模式的作用是使系统的功耗降低，并维持系统运行，以达到省电的目的。省电模式的进入是通过操作 HALT 指令和 STOP 指令来完成。

系统执行 HALT 指令后，MCU 将进入待机模式 1（HALT）。在 HALT 模式下，CPU 将停止工作。但是其周边电路（Timer，时基定时器，DAC, AGC, ADC,…）将继续工作。在执行 STOP 指令后，MCU 将进入待机模式 2（STOP)。在 STOP 模式下，除了看门狗定时器电路外，整个芯片（包括振荡器）将停止工作。较之 HALT 模式，STOP 更为省电。

在 HALT 模式下，发生任何中断，MCU 将被唤醒，并退出 HALT 模式。在 STOP 模式下，只有发生端口中断时，MCU 才能被唤醒，并退出 STOP 模式。

当系统从 HALT/STOP 模式唤醒退出时，首先执行相关中断服务子程序. 然后才会执行 HALT/STOP 指令后的下一条指令。

第二章

SinoWealth 4-bit 单片机指令系统

Sinowealth 4-bit SH6xxx 系列单片机的指令系统是一种简明易掌握，效率极高的 RISC (精简指令集)结构的指令系统。

SH6xxx 指令系统的指令数目共 43 条，均为单字节指令，其执行时间均相同，每一条指令的执行时间均为 1 个指令周期，指令周期=系统时钟周期=4×振荡器震荡周期，如系统时钟为 32.768kHz，则每条指令的执行时间=4/32768Hz=122.07us。

2.1 指令的分类

按照功能 SH6xxx 指令可以分成以下四大类：

- 算术运算指令，包括：
  - 加法指令：ADC，ADCM，ADD，ADDM，ADI，ADIM
  - 减法指令：SBC，SBCM，SUB，SUBM，SBI，SBIM
  - 十进制调整指令：DAA，DAS
- 逻辑运算指令，包括：
  - 与操作指令：AND，ANDM，ANDIM
  - 或操作指令：OR，ORM，ORIM
  - 异或操作指令：EOR，EORM，EORIM
  - 移位操作：SHR
- 数据传送指令，包括：
  - LDA，LDI，STA
- 流程控制指令，包括：
  - C（进/借位标志）标志操作指令：BC，BNC
  - A（累加器标志）位操作指令：BA0，BA1，BA2，BA3
  - A（累加器标志）判零操作指令：BAZ，BNZ
  - 子程序调用/返回操作指令：CALL，RTNW，RTNI
  - 无条件跳转操作指令：JMP，TJMP
- 省电模式操作指令，包括：
  - HALT，STOP
- NOP 指令
  - NOP

2.2 指令的格式

SH6xxx 指令的格式由操作码，操作数组成，具体格式为

操作码 [操作数 1]，[操作数 2]

其中，[ ]内的项目依指令的性质而使用，有的指令没有操作数，有的指令操作数为 1 或 2 个。操作码和操作数之间需以空格加以隔开，而操作数之间以‘,’隔开。

2.3 符号说明

在后续的指令描述中，以下符号将频繁出现，特加以说明：

PC	程序计数器，用于存放下一条要执行指令的地址
AC/A	累加器
CY	进位/借位标志
Mx	数据存储器(Data Memory)
bbb	RAM Bank
ST	堆栈（stack）
X	程序地址
I	立即数
&	逻辑 AND
|	逻辑 OR
^	逻辑 EOR

2.4 指令介绍

2.4.1 算术运算指令

加法指令

ADD

指令格式	ADD Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相加，结果仅存回 A 中
运算式	A←Mx+A
指令编码	00001 0bbb xxx xxxx
影响的标志位	CY

ADD 指令将 Data Memory(Mx)的内容和累加器 A 的内容相加，结果仅存回 A 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-1： 05H +06H

```
          :
LDI       20H,05H        ;$20H=05H,A=05H
LDI       21H,06H        ;$21H=06H,A=06H
LDA       20H,00H        ;A=05H
ADD       21H,0          ;A=0BH,CY=0,$21H=06H
          :
```

ADDM

指令格式	ADDM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相加，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx+A
指令编码	00001 1bbb xxx xxxx
影响的标志位	CY

ADDM 指令将 Data Memory(Mx)的内容和累加器 A 的内容相加，结果同时存回 A 和 Mx 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-2：
```
05H +06H
:
LDI     20H,05H         ;$20H=05H,A=05H
LDI     21H,06H         ;$21H=06H,A=06H
LDA     20H,00H         ;A=05H
ADDM    21H,0           ;A=0BH,$21H=0BH,CY=0
:
```

ADC

指令格式	ADC Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容及进位标记 CY 相加，结果仅存回 A 中
运算式	A ←Mx+A+CY
指令编码	00000 0bbb xxx xxxx
影响的标志位	CY

ADC 将 Data Memory(Mx)的内容和累加器 A 的内容及进位标记 CY 相加，结果仅存回 A 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-3：
```
05H +06H , CY=1
:               ;CY=1
LDI     20H,05H         ;$20H=05H,A=05H
LDI     21H,06H         ;$21H=06H,A=06H
LDA     20H,00H         ;A=05H
ADC     21H,0           ;A=0CH,$21H=06H,CY=0
:
```

ADCM

指令格式	ADCM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容及进位标记 CY 相加，结果同时存回 A 和 Mx 中
运算式	A ，Mx←Mx+A+CY
指令编码	00000 1bbb xxx xxxx
影响的标志位	CY

ADCM 将 Data Memory(Mx)的内容和累加器 A 的内容及进位标记 CY 相加，结果同时存回

A 和 Mx 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-4： 05H +06H , CY=1

```
        :               ;CY=1
LDI     20H,05H         ;$20H=05H,A=05H
LDI     21H,06H         ;$21H=06H,A=06H
LDA     20H,00H         ;A=05H
ADCM    21H,0           ;A=0CH,$21H=0CH,CY=0
        :
```

ADI

指令格式	ADI Mx，I
指令描述	将 Data Memory(Mx)的内容和立即数 I 相加，结果仅存回 A 中
运算式	A ←Mx + I
指令编码	01000 iiii xxx xxxx
影响的标志位	CY

ADI 将 Data Memory(Mx)的内容和立即数 I 相加，结果仅存回 A 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-5： $20H=05H , I=04H

```
        :
LDI     20H,05H         ;$20H=05H,A=05H,CY=0
ADI     20H,04H         ;A=09H,$20H=05H,CY=0
        :
```

ADIM

指令格式	ADIM Mx，I
指令描述	将 Data Memory(Mx) 的内容和立即数 I 相加，结果同时存回 A 和 Mx 中
运算式	A ，Mx←Mx+ I
指令编码	01001 iiii xxx xxxx
影响的标志位	CY

ADIM 将 Data Memory(Mx) 的内容和立即数 I 相加，结果同时存回 A 和 Mx 中，当结果超过 0FH 时，CY 会被置 1，否则置 0。

例 2-4-6： $20H=05H , I=04H

```
        :
LDI     20H,05H     ;$20H=05H,A=05H
ADIM    20H,04H     ;A=09H,$20H=09H,CY=0
        :
```

减法指令

SUB

指令格式	SUB Mx，bbb
指令描述	将 Data Memory(Mx)的内容减去累加器 A 的内容相加，结果仅存回 A 中
运算式	A←Mx - A
指令编码	00011 0bbb xxx xxxx
影响的标志位	CY

SUB 将 Data Memory(Mx)的内容减去累加器 A 的内容，结果仅存回 A 中，当 Mx ≥ A 时，CY 会被置 1，否则置 0。

例 2-4-7：
```
06H - 05H
:
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     20H,0           ;A=05H
SUB     21H,0           ;A=1,CY=1,$21H=06H
:
```

例 2-4-8：
```
05H - 06H
:
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     21H,0           ;A=06H
SUB     20H,0           ;A=0FH,CY=0,$20H=05H
                        :
```

SUBM

指令格式	SUBM Mx，bbb
指令描述	将 Data Memory(Mx)的内容减去累加器 A 的内容相加，结果同时存回 A 和 Mx 中
运算式	A,Mx←Mx - A
指令编码	00011 1bbb xxx xxxx
影响的标志位	CY

SUB 将将 Data Memory(Mx)的内容减去累加器 A 的内容，结果同时存回 A 和 Mx 中，当 Mx ≥ A 时，CY 会被置 1，否则置 0。

例 2-4-9：
```
06H - 05H
:
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     20H,0           ;A=05H
SUBM    21H,0           ;A=01H,CY=1,$21H=01H
:
```

例 2-4-10：05H - 06H

```
        :
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     21H,0           ;A=06H
SUBM    20H,0           ;A=0FH,CY=0,$20H=0FH
        :
```

SBC

指令格式	SBC Mx，bbb
指令描述	将 Data Memory(Mx)的内容减去累加器 A 的内容相加，再加上 CY，结果仅存回 A 中
运算式	A←Mx - A + CY
指令编码	00010 0bbb xxx xxxx
影响的标志位	CY

SBC 将 Data Memory(Mx)的内容减去累加器 A 的内容相加，再加上 CY，结果仅存回 A 中，当 Mx ≥ A 时，CY 会被置 1，否则置 0。

例 2-4-11： CY=0，06H - 05H=?

```
        :               ;CY=0
LDI     20H,05H         ;AC=05H,$20H=05H
LDI     21H,06H         ;AC=06H,$21H=06H
LDA     20H,0           ;AC=05H
SBC     21H,0           ;AC=01H,CY=0,$21H=06H
        :
```

例 2-4-12：CY=1，06H - 05H=?

```
        :               ;CY=1
LDI     20H,05H         ;AC=05H,$20H=05H
LDI     21H,06H         ;AC=06H,$21H=06H
LDA     20H,0           ;AC=05H
SBC     21H,0           ;AC=02H,CY=0,$21H=06H
        :
```

SBCM

指令格式	SBCM Mx，bbb
指令描述	将 Data Memory(Mx)的内容减去累加器 A 的内容相加，再加上 CY，结果同时存回 A 和 Mx 中
运算式	A,Mx←Mx - A + CY
指令编码	00010 1bbb xxx xxxx
影响的标志位	CY

SUB 将 Data Memory(Mx)的内容减去累加器 A 的内容，再加上 CY，结果同时存回 A 和 Mx 中，当 Mx ≥ A 时，CY 会被置 1，否则置 0。

例 2-4-13： CY=0，06H - 05H=?

```
        :               ;CY=0
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     20H,0           ;A=05H,CY=0
SBCM    21H,0           ;A=01H,CY=0,$21H=01H
        :
```

例 2-4-14： CY=1，06H - 05H=?

```
        :               ;CY=1
LDI     20H,05H         ;A=05H,$20H=05H
LDI     21H,06H         ;A=06H,$21H=06H
LDA     20H,0           ;A=05H,CY=1
SBCM    21H,0           ;A=02H,CY=0,$21H=02H
        :
```

SBI

指令格式	SBI Mx，I
指令描述	将 Data Memory(Mx)的内容减去立即数 I，结果仅存回 A 中
运算式	A←Mx - I
指令编码	01010 iiii xxx xxxx
影响的标志位	CY

SBI 将 Data Memory(Mx)的内容减去立即数 I，结果仅存回 A 中，当 Mx ≥ I 时，CY 会被置 1，否则置 0。

例 2-4-15： $20H=05H , I=04H

```
        :
LDI     20H,05H         ;$20H=05H,A=05H,CY=0
SBI     20H,04H         ;A=01H,$20H=05H,CY=1
        :
```

例 2-4-16： $20H=02H , I=07H

```
        :
LDI     20H,02H         ;$20H=02H,AC=02H,CY=0
SBI     20H,07H         ;AC=0BH,$20H=02H,CY=0
        :
```

SBIM

指令格式	SBIM Mx，I
指令描述	将 Data Memory(Mx)的内容减去立即数 I，结果同时存回 A 和 Mx 中
运算式	A,Mx←Mx - I
指令编码	01011 iiii xxx xxxx
影响的标志位	CY

SBIM 将 Data Memory(Mx)的内容减去立即数 I，结果同时存回 A 和 Mx 中，当 Mx ≥ I 时，CY 会被置 1，否则置 0。

例 2-4-17： $20H=05H , I=04H

```
        :
LDI     20H,05H         ;$20H=05H,A=05H
SBIM    20H,04H         ;A=01H,$20H=01H,CY=1
        :
```

例 2-4-18： $20H=02H , I=07H

```
        :
LDI     20H,02H         ;$20H=02H,A=02H
SBIM    20H,07H         ;A=0BH,$20H=0BH,CY=0
        :
```

十进制调整指令

DAA

指令格式	DAA Mx
指令描述	将相加后的 Data Memory(Mx)做十进制(BCD)调整，结果同时存回 A 和 Mx 中
运算式	A，Mx←调整 Mx
指令编码	11001 0110 xxx xxxx
影响的标志位	CY

DAA 将相加后的 Data Memory(Mx)做十进制(BCD)调整，结果同时存回 A 和 Mx 中，若 Mx 的值大于 9 或 CY=1，则将 Mx 的值加 6，CY 同时置 1。

例 2-4-19： 06H + 05H , and do DAA adjustment

```
        :
LDI     20H,06H         ;A=06H,$20H=06H
LDI     21H,05H         ;A=05H,$21H=05H
LDA     20H,0           ;A=06H
ADD     21H,0           ;A=0BH,CY=0
DAA     21H             ;A=01H,$21H=01H,CY=1
        :
```

DAS

指令格式	DAS Mx
指令描述	将相减后的 Data Memory(Mx)做十进制(BCD)调整，结果同时存回 A 和 Mx 中
运算式	A，Mx←调整 Mx
指令编码	11001 1010 xxx xxxx
影响的标志位	CY

DAA 将相减后的 Data Memory(Mx)做十进制(BCD)调整，结果同时存回 A 和 Mx 中，若 Mx 的值大于 9 或 CY=0，则将 Mx 的值加 0AH，CY 同时清为 0。

例 2-4-20： 05H - 06H , and do DAS adjustment

```
        :
LDI     20H, 06H        ;A=06H,$20H=06H
LDI     21H, 05H        ;A=05H,$21H=05H
LDA     20H,0           ;A=06H
SUB     21H,0           ;A=0FH,CY=0
DAS     21H             ;A=09H,$21H=09H,CY=0
        :
```

2.4.2 逻辑运算指令

与操作指令

AND

指令格式	AND Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相与，结果仅存回 A 中
运算式	A←Mx & A
指令编码	00110 0bbb xxx xxxx
影响的标志位	无

AND 将 Data Memory(Mx)的内容和累加器 A 的内容相与，结果仅存回 A 中。

例 2-4-21：06H & 05H

```
        :
LDI     20H,0110B       ;A=06H,$20H=06H
LDI     21H,0101B       ;A=05H,$21H=05H
AND     20H,0           ;A=0100B,$20H=06H
        :
```

ANDM

指令格式	ANDM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相与，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx & A
指令编码	00110 1bbb xxx xxxx
影响的标志位	无

ANDM 将 Data Memory(Mx)的内容和累加器 A 的内容相与，结果同时存回 A 和 Mx 中。

例 2-4-22：0110B & 0101B

```
        :
LDI     20H,0110B       ;A=0110B,$20H=0110B
LDI     21H,0101B       ;A=0101B,$21H=0101B
ANDM    20H,0           ;A=0100B,$20H=0100B
        :
```

ANDIM

指令格式	ANDIM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和立即数 I 相与，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx & A
指令编码	00110 1bbb xxx xxxx
影响的标志位	无

ANDIM 将 Data Memory(Mx)的内容和立即数 I 相与，结果同时存回 A 和 Mx 中。

例 2-4-23：0110B & 0101B

```
        :
LDI     20H,0110B       ;A=0110B,$20H=0110B
LDI     21H,0101B       ;A=0101B,$21H=0101B
ANDM    20H,0           ;A=0100B,$20H=0100B
        :
```

或操作指令

OR

指令格式	OR Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相或，结果仅存回 A 中
运算式	A←Mx ｜ A
指令编码	00101 0bbb xxx xxxx
影响的标志位	无

OR 将 Data Memory(Mx)的内容和累加器 A 的内容相或，结果仅存回 A 中。

例 2-4-24：0001B ｜ 0100B

```
        :
LDI     20H,0001B       ;$20H=0001B,AC=0001B
LDI     21H,0100B       ;$21H=0100B,AC=0100B
OR      20H,0           ;$20H=0001B,AC=0101B
        :
```

ORM

指令格式	ORM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相或，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx ｜ A
指令编码	00101 1bbb xxx xxxx
影响的标志位	无

ORM 指令将 Data Memory(Mx)的内容和累加器 A 的内容相或，结果同时存回 A 和 Mx 中。
例 2-4-25：0001B | 0100B

```
        :
LDI     20H,0001B       ;$20H=0001B,AC=0001B
LDI     21H,0100B       ;$21H=0100B,AC=0100B
ORM     20H,0           ;$20H=0101B,AC=0101B
        :
```

ORIM

指令格式	ORIM Mx，I
指令描述	将 Data Memory(Mx)的内容和立即数 I 相或，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx	I
指令编码	01101 iiii xxx xxxx
影响的标志位	无

ORIM 将 Data Memory(Mx)的内容和立即数 I 相或，结果同时存回 A 和 Mx 中。
例 2-4-26：Set bit 3 of the value of $20H to 1

```
ORIM   20H, 1000B
After execution: $20H=1xxxB
```

异或操作指令

EOR

指令格式	EOR Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相异或，结果仅存回 A 中
运算式	A←Mx ^ A
指令编码	00100 0bbb xxx xxxx
影响的标志位	无

EOR 将 Data Memory(Mx)的内容和累加器 A 的内容相异或，结果仅存回 A 中。
例 2-4-27：0011B ^ 0101B

```
        :
LDI     20H,0011B       ;$20H=0011B,AC=0011B
LDI     21H,0101B       ;$21H=0101B,AC=0101B
EOR     20H,0           ;$20H=0011B,AC=0110B
        :
```

EORM

指令格式	EORM Mx，bbb
指令描述	将 Data Memory(Mx)的内容和累加器 A 的内容相异或，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx ^ A
指令编码	00100 1bbb xxx xxxx
影响的标志位	无

EORM 指令将 Data Memory(Mx)的内容和累加器 A 的内容相异或，结果同时存回 A 和 Mx 中。

例 2-4-28：0011B ^ 0101B

```
        :
LDI     20H,0011B       ;$20H=0011B,AC=0011B
LDI     21H,0101B       ;$21H=0101B,AC=0101B
EORM    20H,0           ;$20H=0110B,AC=0110B
        :
```

EORIM

指令格式	EORIM Mx，I
指令描述	将 Data Memory(Mx)的内容和立即数 I 相异或，结果同时存回 A 和 Mx 中
运算式	A，Mx←Mx ^ I
指令编码	01100 iiii xxx xxxx
影响的标志位	无

EORIM 将 Data Memory(Mx)的内容和立即数 I 相异或，结果同时存回 A 和 Mx 中。

例 2-4-29：$20H=0011B，I=0101B

```
        :
LDI     20H,0011B       ;$20H=0011B,AC=0011B
EORIM   20H,0101B       ;$20H=0110B,AC=0110B
        :
```

移位操作指令

SHR

指令格式	SHR
指令描述	将累加器 A 的内容右移移位，0 移入 AC[3]，AC[0]移入 CY
运算式	A← A Shift 1 bit at right direction
指令编码	11110 0000 000 0000
影响的标志位	CY

SHR 指令将 A 的内容向右移动一位，

AC[0] → CY, AC[1] → AC[0], AC[2] → AC[1], AC[3] → AC[2], 0 → AC[3]

例 2-4-30：06 右移一位

```
        :
LDI     20H,0110B       ;$20H=0110B,AC=0110B
SHR                     ;AC=0011B, CY=0
        :
```

2.4.3 数据传送指令

LDA

指令格式	LDA Mx, bbb
指令描述	将 Data Memory(Mx)的内容传送至累加器 A 中
运算式	A←Mx
指令编码	00111 0bbb xxx xxxx
影响的标志位	无

LDA 指令将 Data Memory(Mx)的内容传送至累加器 A 中。

例 2-4-31：Load the value of $20H to AC

```
        :
LDI     20H,05H         ;$20H=05H,AC=05H
LDI     21H,0FH         ;$21H=0FH,AC=0FH
LDA     20H,0           ;AC=05H
        :
```

STA

指令格式	STA Mx, bbb
指令描述	将累加器 A 的内容传送至 Data Memory(Mx)中
运算式	Mx←A
指令编码	00111 1bbb xxx xxxx
影响的标志位	无

STA 将累加器 A 的内容传送至 Data Memory(Mx)中。

例 2-4-32：Save the value of AC in $21H

```
        :
LDI     20H, 05H        ;AC=05H, $20H=05H
STA     21H, 00H        ;$21H=05H
        :
```

LDI

指令格式	LDI Mx，I
指令描述	将立即数 I 写入 Data Memory(Mx)和累加器 A 中
运算式	A，Mx ← I
指令编码	01111 iiii xxx xxxx
影响的标志位	无

LDI 将立即数 I 写入 Data Memory(Mx)和累加器 A 中。

例 2-4-33：

```
LDI     20H，05H
```

执行结果： AC=05H，$20H=05H

2.4.4 流程控制指令

C(进位/借位标志)操作指令

BC

指令格式	BC X
指令描述	如果 CY=1，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if CY=1
指令编码	10011 xxxx xxx xxxx
影响的标志位	无

执行 BC 指令，如果 CY=1，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-34：If CY=1 then goto INC20H

```
                :
        LDI     20H,0FH         ;$20H=0FH,AC=0FH,CY=0
INC20H: SBIM    20H,01H         ;$20H,AC←$20H -1
        BC      INC20H          ;if CY=1,jump to INC20H
                :
```

BNC

指令格式	BNC X
指令描述	如果 CY=0，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if CY=0
指令编码	10001 xxxx xxx xxxx
影响的标志位	无

执行 BNC 指令，如果 CY=0，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-35：If CY=0 then goto INC20H

```
          :
          LDI     20H,0FH         ;$20H=0FH,AC=0FH,CY=0
INC20H:   SBIM    20H,01H         ;$20H,AC←$20H -1
          BNC     INC20H          ;if CY=0,jump to INC20H
          :
```

A(累加器标志)位操作指令

BA0

指令格式	BA0 X
指令描述	如果 AC[0]=1，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if AC[0]=1
指令编码	10100 xxxx xxx xxxx
影响的标志位	无

执行 BA0 指令，如果 AC[0]=1，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-36：If $20H(bit 0)=1 then goto INC21H

```
          :
          LDI     20H,0FH         ;$20H=0FH,AC=0FH
DEC20H:   SBIM    20H,01H         ;$20H,AC←$20H -1
          BA0     INC21H          ;if AC(bit0)=1,jump to INC21H
          JMP     DEC20H          ;else jump to DEC20H
          :
INC21H:   ADIM    21H,01H         ;$21H,AC←$21H+1
          :
```

BA1

指令格式	BA1 X
指令描述	如果 AC[1]=1，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if AC[1]=1
指令编码	10101 xxxx xxx xxxx
影响的标志位	无

执行 BA1 指令，如果 AC[1]=1，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-37：If $20H(bit 1)=1 then goto INC21H

```
          :
          LDI     20H,0FH         ;$20H=0FH,AC=0FH
DEC20H:   SBIM    20H,01H         ;$20H,AC←$20H -1
          BA1     INC21H          ;ifAC(bit1)=1,jump to INC21H
          JMP     DEC20H          ;else jump to DEC20H
          :
INC21H:   ADIM    21H,01H         ;$21H,AC←$21H+1
          :
```

BA2

指令格式	BA2 X
指令描述	如果 AC[2]=1，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if AC[2]=1
指令编码	10110 xxxx xxx xxxx
影响的标志位	无

执行 BA2 指令，如果 AC[2]=1，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-38：

```
                If $20H(bit 2)=1 then goto INC21H
                :
        LDI     20H,0FH     ;$20H=0FH,AC=0FH
DEC20H: SBIM    20H,01H     ;$20H,AC←$20H -1
        BA2     INC21H      ;if AC(bit2)=1,jump to INC21H
        JMP     DEC20H      ;else jump to DEC20H
                :
INC21H: ADIM    21H,01H     ;$21H,AC←$21H+1
                :
```

BA3

指令格式	BA3 X
指令描述	如果 AC[3]=1，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if AC[3]=1
指令编码	10111 xxxx xxx xxxx
影响的标志位	无

执行 BA3 指令，如果 AC[3]=1，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-39：

```
                If $20H(bit 3)=1 then goto INC21H
                :
        LDI     20H,0FH         ;$20H=0FH,AC=0FH
DEC20H: SBIM    20H,01H         ;$20H,AC←$20H -1
        BA3     INC21H          ;if AC(bit3)=1,jump to INC21H
        JMP     DEC20H          ;else jump to DEC20H
                :
INC21H: ADIM    21H,01H         ;$21H,AC←$21H+1
                :
```

A(累加器)判零操作指令

BAZ

指令格式	BAZ X
指令描述	如果 A=0，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if A=0
指令编码	10010 xxxx xxx xxxx
影响的标志位	无

执行 BAZ 指令，如果 A=0，则程序跳转到指定地址，否则继续执行下一条。

例 2-4-40：

```
        If ($20H=$20H-1)=00H then goto INC21H
                :
        LDI     20H,0FH         ;$20H=0FH
DEC20H: SBIM    20H,01H         ;AC,$20H ← $20H -1
        BAZ     INC21H          ;if AC=0 jump to INC21H
        JMP     DEC20H          ;else jump to DEC20H
                :
INC21H: ADIM    21H,01H         ;$21H+1
                :
```

BNZ

指令格式	BNZ X
指令描述	如果 A ≠ 0，则程序跳转到指定地址，否则继续执行下一条
运算式	PC ←X，if A ≠ 0
指令编码	10000 xxxx xxx xxxx
影响的标志位	无

执行 BNZ 指令，如果 A ≠ 0，则程序跳转到指定地址，否则继续执行下一条。

子程序调用/返回操作指令

CALL

指令格式	CALL X
指令描述	调用子程序
运算式	ST ←CY & (PC+1)，PC ← X
指令编码	11000 xxxx xxx xxxx
影响的标志位	无

CALL 指令用于调用子程序，执行此条指令时，CPU 首先进行现场保护，将当前的 CY 标记和 PC+1 值压入堆栈保存起来(子程序返回时自动弹栈，程序继续运行)，然后再跳转至指定的 ROM 地址 X（$000～$07FFH 或$0800～$0FFFH）处执行程序，其返回指令可用 RTNW 或 RTNI 两条指令。

在使用 CALL 指令时，堆栈的层数必须加以注意，目前 CPU6610C/D 只支持 4 层堆栈，而

CPU6610E 支持 8 层堆栈，而每颗产品使用的 CPU 类型在其对应规格书的首页都有明确的描述。

另外，CALL 指令的跳转范围限制在同一个 BANK 中，寻址范围为 2K，这一点可以结合第一章中有关 BANK 的描述部分加以理解。

RTNW

指令格式	RTNW H,L
指令描述	从子程序中返回上一层程序，H 参数赋值给 TBR，L 参数赋值给累加器 A，同时将堆栈的内容弹栈
运算式	PC ← ST，TBR ← H，A ← L
指令编码	11010 000h hhh LLLL
影响的标志位	无

RTNW 指令用于从子程序中返回上一层程序，执行此条指令时，CPU 首先进行现场恢复，将保存在堆栈中最顶层的 PC+1 值弹出，同时将该指令的操作数 H 赋值给 TBR 寄存器，将参数 L 赋值给累加器 A。

这一指令的主要作用有两点，一点可以实现固定资料表中数据的读取，第二点是可以实现程序的散转。

注意：执行此条指令时，弹栈的只是 PC 值，不包括 CY。

例 2-4-41：读取位于$302 处的数据表中的数据

```
TBR       EQU       0EH
TEMP      EQU       20H
                    :
                    :
001A      LDI       TBR, 00H    ;put index value (high nibble) 0 into TBR.
001B      LDI       TEMP, 02    ;put index value (low nibble) 2 into AC
001C      CALL      300H        ;call subprogram.
001D                :
                    :
                    :
          ORG       300H
0300      TJMP                  ; get destination address $0302H according to (PC11～PC8),TBR,AC
0301      RTNW      00H,01H
0302      RTNW      00H,02H     ;return to main program, H→TBR,L→AC
0303      RTNW      04H,05H
0304      RTNW      09H,08H
0305                :
```

RTNI

指令格式	RTNI
指令描述	从子程序中返回上一层程序，同时将堆栈的内容弹栈
运算式	PC ，CY ← ST
指令编码	11010 1000 000 0000
影响的标志位	CY

RTNI 指令主要用于从中断服务子程序中返回上一层程序，当然也可以用于一般子程序的返回。执行此条指令时，CPU 只是进行现场恢复，将保存在堆栈中最顶层的 CY 标记和 PC+1 值弹出。

无条件跳转操作指令

JMP

指令格式	JMP X
指令描述	无条件跳转至指定地址 X
运算式	PC ← X
指令编码	1110p xxxx xxx xxxx
影响的标志位	无

JMP 指令用于无条件程序跳转，由于 CPU 的寻址能力只能是 4K，所以 JMP 指令的寻址范围也为 4K，至于要跳转到 4K 以外的地方，则须借助翻 BANK 的操作模式，此部分在第一章 ROM 部分已经有详细的描述。此条指令类似 BASIC 程序中的 goto 语句和 8051 体系单片机中 SJMP/LJMP/JMP 指令。

例 2-4-42：PC=40H , Jump to 0E00H

```
        JMP   0E00H
```

执行结果：PC=0E00H

TJMP

指令格式	TJMP
指令描述	无条件跳转至指定地址,指定地址=（PC11～8）（TBR）（A）
运算式	PC ← （PC11～8）（TBR）（A）
指令编码	11110 1111 111 1111
影响的标志位	无

JMP 指令用于无条件程序跳转，但跳转的地址由(PC11～8)(TBR)(A)组成，程序范例可以参照 RTNW 指令。

例如当前 PC=300H，TBR=01H，A=02H，则组合成的新地址=011,0001,0010,即$312。

省电模式操作指令

HALT

指令格式	HALT
指令描述	CPU 停止工作，其它 Block 处于工作状态，包括振荡器
运算式	
指令编码	11011 0000 000 0000
影响的标志位	无

STOP

指令格式	STOP
指令描述	除了 STOP 模式唤醒相关电路外,其余所有电路均停止工作，系统进入最省电模式
运算式	
指令编码	11011 1000 000 0000
影响的标志位	无

HALT/STOP 指令主要用于强制使系统进入省电模式，一般而言，在 HALT 模式下，除 CPU 停止工作外，其余电路仍继续工作，包括振荡器部分，而 STOP 模式则除了 STOP 模式唤醒相关电路外，包括 CPU、振荡器等在内都停止工作。关于 HALT/STOP 模式的详细定义每颗产品可能有所不同，务必参照对应产品的规格书。

NOP 操作指令

NOP

指令格式	NOP
指令描述	CPU 执行一个空操作
运算式	
指令编码	11111 1111 111 1111
影响的标志位	无

2.5 伪指令

同其它通用的汇编语言编译器一样，SH6xxx 汇编器也支持伪指令操作。

SH6xxx 汇编器的伪指令目前共 13 条(随着编译器的升级，伪指令的数目会有所增加，具体可参见 Rice66 IDE 的 HELP 菜单)，这些伪指令可以伴随一些相应的操作数出现在语句行的命令部分，但伪指令不会生成指令码，它只是帮助汇编器进行过程控制，数据分配或格式化输出等。

SH6xxx 伪指令列表如下：

伪指令	作用
DW/DATA	在程序中初始化数据
END	程序结束标记
EQU	定义 DATA 类型的符号，并赋给一个初值
IF..ELSE..ENDIF	条件编译
INCLUDE	包含外部汇编文件
LIST	提供 LIST 文件选项，如 memory size, CPU type 等
ORG	程序初始地址设定
PAGE	在生成的 List 文件中插入换页符
RES	设定一段受保护的 Memory
SET	定义或设定一个变量类型的符号
SPAC	在生成的 List 文件中插入空行
TITLE	指定 List 文件的头部信息
ROMSIZE	定义编译的最大程序空间

DW/DATA: 在程序存储空间初始化常数

伪指令格式	[标号] DW [操作数 1], [操作数 2]… [标号] DATA [操作数 1], [操作数 2]…
指令描述	对程序存储空间进行数据的初始化，操作数部分为一组常量

例 2-5-1：

```
ORG     0350H
DW      10H,20H,30H
DATA    “This is a demo”
```

END: 程序结束标志

伪指令格式	[标号] END
指令描述	标明程序结束

例 2-5-2：

```
:
END     ; 程序结束
```

EQU: 定义 DATA 类型的符号，并赋给一个初值

伪指令格式	符号　EQU　常数/表达式
指令描述	定义一个 DATA 类型的符号常量,定义的符号就是常数或表达式的值，表达式中的符号只能是在此伪指令之前就已经定义过的符号。

例 2-5-3：

```
SYNC    EQU 19H
```

IF..ELSE..ENDIF: 条件编译

伪指令格式	IF 表达式
	　程序 1
	ELSE
	　程序 2
	ENDIF

例 2-5-4：

```
IF  1
    JMP 1FFh
ELSE
    JMP 0FFh
ENDIF
```

INCLUDE: 包含外部汇编文件

伪指令格式	[标号]　Include　“Filename”
指令描述	包含其它的外部汇编文件，文件名可以包含完整的路径

例 2-5-5：　INCLUDE　"REG.H"

LIST:提供生成的 List 文件选项，如 Memory Size，产品 type

伪指令格式	LIST P = < processor type > 　(66P20|66P22) LIST R = < radix > 　(DEC*|HEX|OCT) LIST F = < output format > 　(BIN16*|INHX16|INHX8M|INHX8S

例 2-5-6：

```
LIST P=66P22, R=HEX, F=INHX16
```

ORG: 程序初始地址设定

伪指令格式	[标号] ORG 常数/表达式
指令描述	设置程序初始地址

例 2-5-7：

```
ORG  100H
```

PAGE:在生成的 List 文件中插入换页符

伪指令格式	[标号] PAGE
指令描述	在生成的 List 文件中插入换页符

例 2-5-8：　　PAGE

RES: 设定一段受保护的程序空间

伪指令格式	[标号] RES 表达式
指令描述	设定一段受保护的程序空间

例 2-5-9：

```
0300H   RES 10H
```

执行结果：从当前地址 300H 到 300H+10H=310H 间的程序空间被保护起来，以作其它用途。

SET: 定义或设定一个符号，此符号为变量类型

伪指令格式	[符号] SET 常数/表达式
指令描述	定义或设定一个符号，此符号为变量类型，并且在后续的程序中可以通过 SET 改变符号的值

例 2-5-10：　FIVE SET 5

SPAC:在生成的 List 文件中插入空行

伪指令格式	SPAC 常数/表达式
指令描述	在生成的 List 文件中插入常数/表达式值数量的空行

例 2-5-11：　　SPAC 3

TITLE:在生成的 List 文件中插入头部信息

伪指令格式	[标号] TITLE “String”
指令描述	在生成的 List 文件中插入由 string 决定的头部信息

例 2-5-12：　TITLE　“TEST.ASM”

ROMSIZE: 定义编译的最大程序空间

伪指令格式	ROMSZIE=常数/表达式
指令描述	定义编译的最大程序空间

例 2-5-13：ROMSIZE = 0x400

2.6 宏的使用(MACRO)

宏的使用包括宏的定义和宏的调用，宏的定义由一系列指令或伪指令组成，在宏调用的地方将会插入指令或伪指令，宏在调用之前必须定义过。

2.6.1 宏的定义(macro definition)

宏的定义可以分成三个部分，宏头(macro header)，宏体(macro body)，宏结束(macro terminator)

2.6.1.1宏头(macro head)

宏头的格式为：

〈 符号〉 MACRO ［参数 1，参数 2，...］

其中符号为宏的名称，参数来自宏调用的参数。

宏名称的命名规则同对符号的规则约定，如果宏的名称和其它符号相同，编译器会报错。

2.6.1.2 宏体(macro body)

宏体紧跟在宏头定义之后，宏体可以由一系列的指令和伪指令组成，其中可能包括对参数的操作。宏在调用的时候将宏体的部分完全的替换到调用的地方。

2.6.1.3 宏结束(macro end)

宏以“ENDM”指示宏结束，格式为

ENDM

在 ENDM 和宏名称定义 MACRO 之间不能有另外的 MACRO，即宏不能嵌套。

2.6.2 宏的调用(macro call)

在宏定义好之后，就可以进行宏的调用，其格式为：

［标号］ 宏名 ［参数 1，参数 2，…］

其中，标号为程序地址，参数 1、参数 2 等等为传递给宏的参数，参数列表以最后的空白符或分号结束。

宏调用本身不占用程序空间，但替换了宏体之后，宏的实例从当前程序开始。逗号可以用来保留参数的位置，此时该参数为空，参数列表以空白符或分号结束。

2.6.3 参数

参数以字符形式传递到宏的实例中，因而符号是通过名字来传递而非符号的值，即直到宏展开之后参数的值才被计算出来，因而在宏中使用 SET 伪指令可以改变传递到宏中参数的值。

2.7 汇编程序的结构

汇编的源程序由一或多条语句(statements)或注释(Comment)组成。

每条语句分行书写并由指令码(mnemonics)，伪指令(directives)，宏(macros),，符号(symbols)，表达式(expressions)，常数(constants)组成。

2.7.1 汇编语句行的格式

一般，源程序的一条语句如以下形式书写：

[标号] 命令 [操作数] [注释] [CR]

1. 标号 (Label)

   标号为一条语句的第一部分，为可选，其书写规则是以字母开始，后面由字母，数字，下划线组成的字符串，标号以“:”结束。在保证每行的长度在 200 个字符以内即可。

   标号一般为程序中参考到的地址，标号的值等于该位置处的程序地址，标号可以单独占一行。

2. 命令 (Command)

   命令部分与其它部分以空白符分隔，命令可由操作码，伪指令和宏组成。对 SH6xxx 而言，操作码为 43 条，伪指令为 13 条。

3. 操作数(Operand)

   操作数紧跟在命令之后，与命令部分以空白符分隔，操作数部分依命令的不同，其个数有所不同，它们自己之间以“,”分隔。

4. 注释(Comment)

   注释部分在一语句行的最后，以分号“;”开头，这样分号至行末都被认为是注释，同时注释部分也支持 C++风格的行注释，注释以“//”开头。注释部分是可选项，并可单独占一行。

5. CR

   汇编语句以回车符结束。

2.7.2 常量 (Constants)

常量为字符，字符串或数字，编译器将其解释成一个固定的数值。编译器支持不同进制的数，如十六进制，十进制，八进制和二进制。十进制为编译器默认的进制。

1．字符或字符串常量

字符串常量以单引号或双引号开始，并以一匹配的引号结束。编译器会将引号内的字符转换成其对应的 ASCII 码值作为常量值。如‘A’，“This is a demo”。

2. 数字常量

数字常量由数字和字母组成，它们代表的值由其选择的进制决定，进制的指定由附加在数字之后的字符决定。

后缀字符	进制	组成字符	例子
B	二进制	0，1	1010B=10
O	八进制	0～7	0370=31
D	十进制	0～9	279D=279
H	十六进制	0～9，a～f/A～F	7FFH=2047

此外，对于十六进制，进制的指定也可以通过在数字前面加上前缀“0x”或“$”加以表示，如 0x7FF=$7FF=7FFH，数字中的 A～F 不区分大小写。

此外，对于十进制，进制的指定也可以通过在数字前面加上前缀“.”加以表示，如 123(默认格式)=123D=.123。

数字前都可以加单目运算符“+”或“-”来表示正值或负值。

2.7.3 符号 (Symbol)

符号在程序中表示一个值，它可以是符号常量，可以是变量，可以是地址标号。符号可以作为命令的操作数出现。

1. 符号名

符号名为用户定义的字母、数字或特殊符号组合在一起，在程序中作为某种标示出现。符号名的定义规则和程序标号类似，以字母开头，后面跟字母、数字、下划线或特殊字符“@”，编译器的默认值是区分符号的大小写的，但可以通过使用编译器时加上/I 开关量加以忽略。

2. 符号类型

符号类型提供符号的属性信息，它由符号的定义方式决定。符号类型分三种，分别为 DATA，VAR，ADDR 型，具体如下：

DATA 型：用户使用 EQU 伪指令定义的符号，符号的值为定值。

VAR 型：用户使用 SET 伪指令定义的符号，符号的值可变，可以在后面的程序中用 SET 更改。

ADDR 型：用户定义的标号地址或程序地址。

DATA 和 VAR 型的符号在定义时所参考到的符号必须是在定义点之前就定义过的符号，符号在程序中可以作为操作数或标号，但参考的符号必须在程序的其它地方有定义，否则编

译器会报错。

汇编器中有一个特殊的符号为“$”,它表示该处的程序地址，它可以应用到操作数部分或表达式中。如 JMP $-1。

2.7.4 表达式 (Expression)

表达式可以使用在源语句的操作数中。操作数由符号、常量或由运算符连接而成的符号、常量组成。

1. 运算符

表达式中使用的运算符分为算术运算符、逻辑运算符、关系运算符，位运算符等 4 类，具体为：

运算符	描述	例子
+	加/正	1+2+8
-	减/负	3-2
*	乘	4*8
/	除	9/5
%	求余	13%6
**	乘方	3**2
<<	左移	5<<2
>>	右移	3>>7
=	相等	X=y
! =	不等	X! =y
<=	小于等于	3 <= 5
>=	大于等于	5 >= 3
>	大于	7>4
<	小于	6<9
&&	逻辑与	X&&y
||	逻辑或	X||y
!	逻辑非	! X
&	位与	x & y
|	位或	x | y
^	位异或	X^ y
～	位取反	～X
()	括号	(x+y)

2. 优先级(priority)和结合性(associativity)

运算符的优先级和结合性遵从 C 语言的约定。除单目运算符“+”、“-”、“!”和“～”的结合性为右结合外，其余均为左结合。优先级和结合性的关系如下所示：

运算符	结合性	优先级
()	由左至右	最高
! ～ + -(单目运算符)	由右至左	
**	由左至右	
* / %	由左至右	
+ - (双目运算符)	由左至右	
<< >>	由左至右	
> >= < <=	由左至右	
= !=	由左至右	
&	由左至右	
^	由左至右	
|	由左至右	
&&	由左至右	
||	由左至右	最低

第三章

SinoWealth 4-bit 单片机硬件资源介绍

3.1 振荡器

中颖公司 SH6xxx 系列单片机提供多种振荡器类型为系统提供系统时钟信号源，是整个系统运行的引擎。振荡器类型，有石英晶体谐振器（Crystal），陶瓷谐振器（Ceramic），阻容振荡器（RC），芯片内建振荡器（internal RC）和外部输入时钟等。如此多种类的振荡器可以方便用户在实际应用中自主选择最适合的振荡器。

中颖单片机的工作频率是振荡器频率的 1/4，例如用 4MHz 的晶振时，单片机内部工作频率为 1MHz。

3.1.1 石英晶体谐振器（Crystal）和陶瓷谐振器（Ceramic）

石英晶体谐振器（Crystal）和陶瓷谐振器（Ceramic）是单片机最常用（标准）的时钟源。中颖公司 SH6xxx 系列单片机提供 OSCI 和 OSCO 管脚用于连接驱动外部石英晶体谐振器和陶瓷谐振器。

如图 3-1-1 所示：

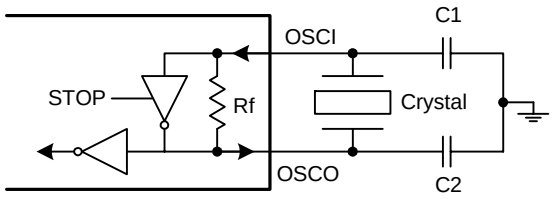


图 3-1-1 Crystal/Ceramic 振荡器示意图

中颖公司对不同频率范围的振荡器提供不同的增益以获得最佳的振荡效果。在实际开发过程中，用户需要注意选用的振荡器的频率范围，在填写掩膜(MASK)相关的查检表格(checklist)和烧写 OTP 时，按相应的频率范围选择项进行选择即可，十分方便。

在石英晶体谐振器和陶瓷谐振器的应用中，需要注意负载电容的选择。不同厂家生产的石英晶体谐振器和陶瓷谐振器的特性和品质都存在较大差异，在选用时，要了解该型号振荡器的关键指标，如等效电阻，厂家建议负载电容，频率飘差等。在实际电路中，也可以通过示波器观察振荡波形来判断振荡器是否工作在最佳状态。示波器在观察振荡波形时，只能观察 OSCO 管脚(Oscillator output)(不能检测 OSCI(Oscillator input)管脚，否则会导致振荡器停振)。应选择 100MHz 带宽以上的示波器探头，这种探头的输入阻抗高，容抗小，对振荡波形相对影响小。（由于探头上一般存在 10～20pF 的电容，所以观测时，适当减小在 OSCO 管脚的电容可以获得更接近实际的振荡波形）。工作良好的振荡波形应该是一个漂亮的正弦波，峰峰值应该大于电源电压的 70%。若峰峰值小于 70%，可适当减小 OSCI 及 OSCO 管脚上的外接负载电容。反之，若峰峰

值接近电源电压且振荡波形发生畸变，则可适当增加负载电容。

如常用的 4MHz 石英晶体谐振器，通常厂家建议的外接负载电容为 10～30pF 左右。若取中心值 15pF，则 C1，C2 各取 30pF 可得到其串联等效电容值 15pF。同时考虑到还另外存在的电路板分布电容，芯片管脚电容，晶体自身寄生电容等都会影响总电容值，故实际配置 C1，C2 时，可各取 20～15pF 左右。并且 C1，C2 使用瓷片电容为佳。

另外，工作电压和环境温度对振荡也存在影响，温度降低及电压降低都会使振荡减弱。所以，在设计定型时，要对最差环境进行考虑，保证电路对环境的兼容性。

3.1.2 外部阻容振荡器（RC）

外部阻容振荡器（RC）通过单片机的 OSCI 管脚外接电阻形成时钟电路，成本低，但相对来讲其振荡频率漂移范围较大。

如图 3-1-2 所示：

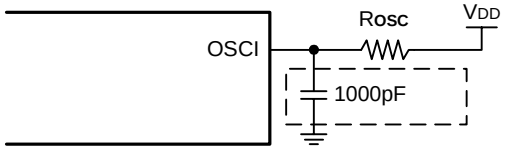


图 3-1-2 外部阻容振荡器示意图

其中，OSCI 管脚处对地所接电容是用于滤除电源噪声（不参与振荡），可以增强时钟频率的稳定性。选用的振荡电阻（ROSC）值可以在对应的中颖单片机规格书中的振荡频率与振荡电阻的关系图（R-F）中查到。在应用阻容振荡器（RC）时应该注意电源的波动对工作频率的稳定度的影响。供电电路自身产生的抖动和电路输出功率的剧烈变化等都会造成电源的波动。另外，阻容振荡器（RC）的一致性较差，同样阻值的振荡电阻（ROSC）在不同单片机上会获得不同的频率。在大批量产时，RC 振荡频率的一致性控制困难，所以对工作频率有一定要求的设计应避免使用。

3.1.3 芯片内建阻容振荡器（internal RC）

中颖单片机还提供芯片内建振荡器（internal RC）。这种振荡器完全在芯片内部实现振荡功能，无需任何外部组件。所以在有些芯片中 OSCI 和 OSCO 管脚可以复用为 I/O 端口，可以最大程度对管脚资源加以利用。其内建振荡器的工作频率都是固定的，一般有 2MHz，4MHz，6MHz 和 8MHz 可选。内建振荡器（internal RC）的频率稳定度和一致性要高于外部阻容振荡器 RC，频率范围可以控制在 2%～3%以内。此类振荡器被广泛应用于红外遥控器设计中，其频率漂移完全满足红外遥控载波的频漂规范。

3.1.4 外部输入时钟(External clock)

中颖单片机还可以通过 OSCI 管脚直接输入外部时钟信号作为工作时钟。外部时钟源可以是有源晶振,外部谐振时钟或其它单片机提供时钟。频率范围为 30KHz～ 8MHz。

3.1.5 双时钟振荡器单片机的使用

双时钟振荡器的类型和作用

部分中颖单片机包含两个振荡器（双时钟），其低频振荡器用于长时间计时和待机时 LCD 的显示，高频时钟用于执行需要快速执行的操作。双时钟的设计可以使产品在绝大多数的待机时间内，用最低的工作频率来维持必要的 LCD 显示输出、计时等功能，同时获得较低的功耗，使电池的使用寿命得到最大程度的延长。

中颖单片机的双时钟由低频时钟（OSC）和高频时钟（OSCX）组成,如图 3-1-3。

低频时钟（OSC）是一个在低频条件下工作的振荡器，低频振荡器一般有低频晶体（典型值 32.768kHz），RC（典型值 131kHz）振荡器等等，如果同时含有，一般由代码选项选择。

高频时钟（OSCX）是为高频操作设计的。一般提供的类型有：陶瓷振荡器（典型值 455KHz～4MHz），石英晶体谐振器(典型值 2MHz～8MHz)，外部 RC 振荡器（2MHz～8MHz)或内建 RC 振荡器(2MHz，4MHz，6MHz 或 8MHz），如果同时含有，同样由代码选项决定。

单片机通过软件控制寄存器完成高/低工作频率之间的相互切换，以达到在高频时钟下进行高速运算处理，在低频时钟下获得较低维持功耗的目的。

双时钟单片机在上电复位后，低频 OSC 首先开始振荡，并自动为系统提供时钟，同时 OSCX 为关闭状态。

在 WDT 复位初始化后，低频 OSC 振荡器打开而高频 OSCX 振荡器保持复位前的状态。

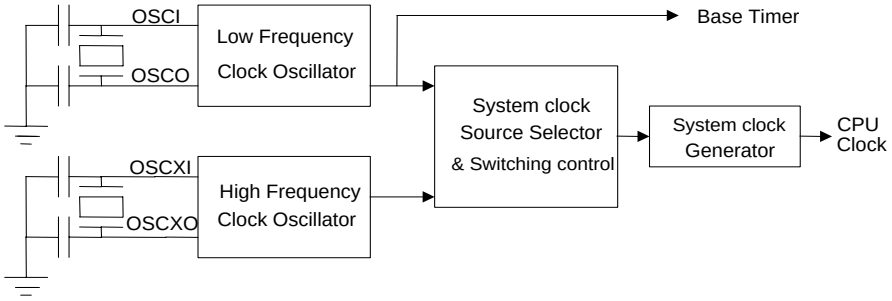


图3-1-3 双时钟振荡器框图

低频时钟振荡器 OSC 可以为 CPU 及其周边设备(LCD 液晶驱动)提供基本时钟脉冲。而高频时钟振荡器 OSCX 可以由软件或代码选项选择为陶瓷振荡器或 RC 振荡器作为 CPU 高频工作时的时钟。

如果芯片中硬件包含 OSCX 振荡器而实际应用中不使用，必须将其在代码选项中选择为陶瓷谐振器并将 OSCXI 接地，避免引脚悬空导致漏电。

双时钟振荡器的控制

双**时钟**振荡器的控制寄存器结构如下所示:

地址	第 3 位	第 2 位	第 1 位	第 0 位
$xx			OXM	OXON

其中：

OXON： OSCX 振荡器开/关。

0 ： 关闭高频时钟 OSCX 振荡器

1 ： 打开高频时钟 OSCX 振荡器

OXM： 切换系统时钟。

0 ： 选择低频时钟 OSC 作为系统时钟

1 ： 选择高频时钟 OSCX 作为系统时钟

双时钟荡器的编程注意事项

OSCX 振荡器从打开到稳定振荡至少需要 5ms 的时间。所以当 CPU 系统时钟振荡器由 OSC 切换到 OSCX 时,用户必须注意此时 OSCX 振荡器必须已经稳定。OSCX 振荡电路启动的时间长短会随选用振荡器的不同而存在差异（RC 振荡启动的时间较短，而陶瓷振荡器启动的时间较长）。因为低频振荡电路 OSC 一直为处于开启状态，所以 CPU 从高频切换回低频工作时无需延时等待，可以直接切换。并且系统电路设计中允许由 OSCX 转换到 OSC 的同时关闭 OSCX。在这种操作下，为了避免运行出错, CPU 将自动延迟一个指令周期再执行关闭 OSCX 的操作。

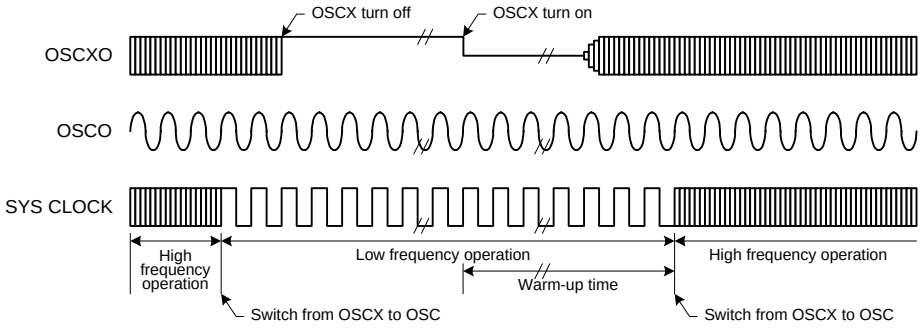


图 3-1-4 系统时钟切换时序图

为了使单片机振荡电路工作在最佳状态下，中颖公司推荐应用以下这些型号的谐振器，并配合表中推荐的负载电容参数。

谐振器负载电容选择

陶瓷谐振器			推荐型号	生产厂
频率	C1	C2		
455kHz	47～100pF	47～100pF	ZTB 455KHz	威克创通讯器材有限公司
			ZT 455E	深圳东光晶博电子有限公司
3.58MHz	-	-	ZTT 3.580M	威克创通讯器材有限公司
			ZT 3.58M*	深圳东光晶博电子有限公司
4MHz	-	-	ZTT 4.000M	威克创通讯器材有限公司
			ZT 4M*	深圳东光晶博电子有限公司

*- 已经内建有负载电容

晶体谐振器			推荐型号	生产厂
频率	C1	C2		
32.768kHz	5～12.5pF	5～12.5pF	Φ3x8 - 32.768KHz	威克创通讯器材有限公司
			DT 38 (Φ3x8)	KDS
4MHz	8～15pF	8～15pF	HC-49U/S 4.000MHz	威克创通讯器材有限公司
			49S-4.000M-F16E	深圳东光晶博电子有限公司
8MHz	8～15pF	8～15pF	HC-49U/S 8.000MHz	威克创通讯器材有限公司
			49S-8.000M-F16E	深圳东光晶博电子有限公司

注意事项:

表中负载电容为设计参考数据！在具体设计中，请注意印制板上的杂散电容，振荡电路工作的性能应在全应用电压和温度范围内测试通过。在应用陶瓷谐振器/晶体谐振器之前，用户需向谐振器生产厂要求相关应用参数以获得最佳性能。

更多的推荐谐振器生产厂目录，请登陆 http://www.sinowealth.com。

3.2 ROM/RAM

此部分在第一章关于 SinoWealth 4-bit 单片机产品基本特性部分已经有所描述，下面就 RAM 部分作一些补充。

内建数据存储器 RAM 包括通用数据存储器和系统寄存器。指令可以直接寻址访问数据存储器和系统寄存器。

系统寄存器用于完成各种控制功能,如 I/O 操作,Timer 操作,中断控制等。
地址位于$000-$01F 之间。若$000-$01F 无法放下全部的内容，则会扩展到$02F 和
$380-$3FF 之间。

数据存储器位于 $020($030) - $2FF 之间，LCD 显示数据存储器位于$300 - $37F。

下表 3-2-1 为一通用的系统寄存器定义表，SH6xxx 系列产品的系统寄存器定义基本参照此表。

地址	内容	说明
$00	Interrupt enable flags (IE)	中断设置寄存器
$01	Interrupt request flags (IRQ)	中断标志寄存器
$02	Timer0 Mode register (TM0)	Timer0 设置寄存器
$03	按型号定义不同功能	-
$04	Timer0 load/counter register (TL0)	Timer0 数据寄存器 (低位)
$05	Timer0 load/counter register (TH0)	Timer0 数据寄存器 (高位)
$06	Timer1 load/counter register (TL1)	Timer1 数据寄存器 (低位)
$07	Timer1 load/counter register (TH1)	Timer1 数据寄存器 (高位)
$08 - 0D	Port data	端口数据寄存器
$0E	Table Branch Register (TBR)	查表数据读写寄存器
$0F	Pseudo index register (INX)	索引寄存器
$10	Data pointer for INX low nibble (DPL)	索引地址寄存器低位（4 位）
$11	Data pointer for INX middle nibble (DPM)	索引地址寄存器中位（3 位）
$12	Data pointer for INX middle nibble (DPH)	索引地址寄存器高位（3 位）
$13	RAM Bank	RAM Bank 设置寄存器
$14	按型号定义不同功能	-
$18 - 1D	Port control	端口输入/输出方向设置寄存器
$1E	WDT register	看门狗设置/溢出控制寄存器
$1F	ROM Bank	ROM Bank 设置寄存器

表3-2-1 通用系统寄存器定义表

对具体寄存器内容的使用,请参考各章节的相关内容。

系统寄存器在上电复位，Reset 引脚复位，看门狗复位及低电压复位后，其存储的数据会发生变化。

用户数据存储器的数值在 Reset 引脚复位和看门狗复位的情况下是不会被改变的。

下表 3-2-2 为一通用的系统寄存器初始状态表，SH6xxx 系列产品的系统寄存器初始状态基本参照此表。

地址	第 3 位	第 2 位	第 1 位	第 0 位	上电复位 /Reset 引脚复位 / 低电压复位	WDT 复位
$00	IEAD	IET0	IET1	IEP	0000	0000
$01	IRQAD	IRQT0	IRQT1	IRQP	0000	0000
$02	-	T0M.2	T0M.1	T0M.0	-000	-uuu
$03	-	T1M.2	T1M.1	T1M.0	-000	-uuu
$04	T0L.3	T0L.2	T0L.1	T0L.0	xxxx	xxxx
$05	T0H.3	T0H.2	T0H.1	T0H.0	xxxx	xxxx
$06	T1L.3	T1L.2	T1L.1	T1L.0	xxxx	xxxx
$07	T1H.3	T1H.2	T1H.1	T1H.0	xxxx	xxxx
$08	PA.3	PA.2	PA.1	PA.0	0000	0000
$09	PB.3	PB.2	PB.1	PB.0	0000	0000
$0A	PC.3	PC.2	PC.1	PC.0	0000	0000
$0B	PD.3	PD.2	PD.1	PD.0	0000	0000
$0C	PE.3	PE.2	PE.1	PE.0	0000	0000
$0D	PF.3	PF.2	PF.1	PF.0	0000	0000
$0E	TBR.3	TBR.2	TBR.1	TBR.0	xxxx	uuuu
$0F	INX.3	INX.2	INX.1	INX.0	xxxx	uuuu
$10	DPL.3	DPL.2	DPL.1	DPL.0	xxxx	uuuu
$11	-	DPM.2	DPM.1	DPM.0	-xxx	-uuu
$12	-	DPH.2	DPH.1	DPH.0	-xxx	-uuu
$13～17	-	-	-	-	-	-
$18	PACR.3	PACR.2	PACR.1	PACR.0	0000	0000
$19	PBCR.3	PBCR.2	PBCR.1	PBCR.0	0000	0000
$1A	PCCR.3	PCCR.2	PCCR.1	PCCR.0	0000	0000
$1B - 1D	-	-	-	-	-	-
$1E	WD	WDT.2	WDT.1	WDT.0	0000	1000
$1F	BNK.3	BNK.2	BNK.1	BNK.0	0000	0000

表 3-2-2 通用系统寄存器初始状态表

说明： x = 不定，u = 未更改，- = 未使用位，读出值为'0'。

其它	复位后
程序计数器（PC）	$000
CY	不定
累加器（AC）	不定
数据存储器	不定

3.3 端口(I/O Port)结构及应用

端口（I/O）是单片机系统中最基本的一个资源， 用于外围状态的输入和控制信号的输出。不同结构的 I/O 在使用上是有差异的，所以在使用时，要事先了解该 I/O 的结构和定义，才能正确应用。

3.3.1 端口(I/O Port)结构

中颖公司 SH6xxx 系列单片机中，针对不同的应用，设计了几种不同的 I/O 结构。如：开漏电路(Open Drain)和 CMOS 结构。以下对这些 I/O 的结构和应用分别进行说明。

- CMOS 结构 I/O

中颖公司系列单片机典型的 COMS I/O 端口是一个标准的双向端口，即可设置为输入端口，又可设置为输出端口。端口在作为输出时，依据具体的应用定位，可以提供大小不等的驱动能力。端口在作为输入时，呈现极高的输入阻抗，对输入信号基本可视为开路或悬浮状态。

这些端口的输入或输出状态完全由用户通过软件设置相应控制寄存器进行控制。并且每个端口都互相独立。

CMOS I/O 端口的结构分为三种。

图 3-3-1 为 CMOS I/O 结构图 1：

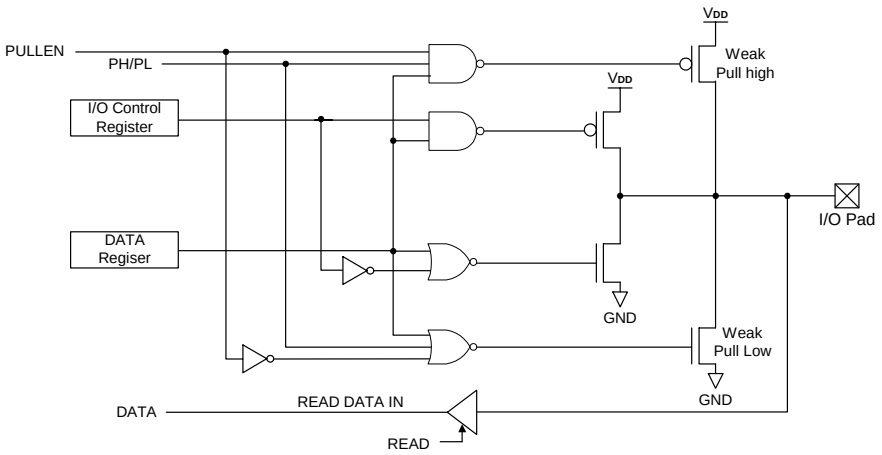


图 3-3-1 CMOS I/O 结构 1

特点：

1）此端口设计中有内建的上拉/下拉 CMOS 电阻（几十至几百千欧姆不等，依单片机型号而定），此功能仅在端口设置为输入状态时才能启用，在输出状态会自动关闭，且每个端口互相独立。

2）通过寄存器 PULLEN 控制位选择是否允许上拉/下拉电阻功能；通过寄存器 PH/PL 控制位选择上拉或下拉电阻功能。这些内建的上拉/下拉 CMOS 电阻在设计中可以节省电路板上的外部上拉/下拉电阻及电路板面积。

图 3-3-2 为 CMOS I/O 结构图 2：

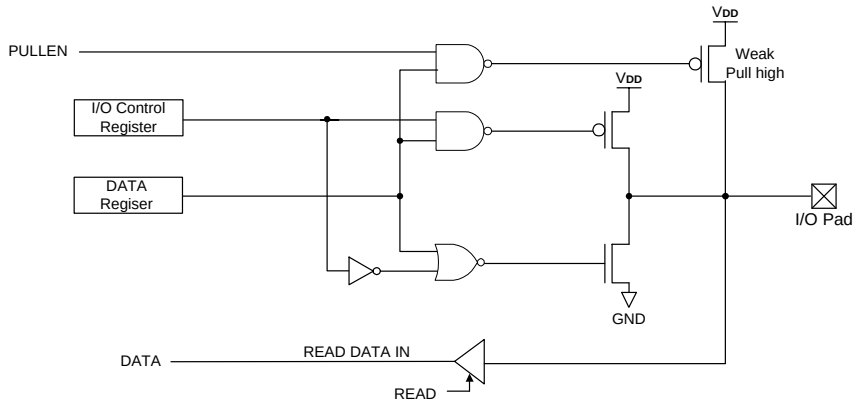


图 3-3-2 CMOS I/O 结构 2

此端口设计中只有内建的上拉 CMOS 电阻（几十至几百千欧姆不等，依单片机型号而定），此功能仅在端口设置为输入状态时才能启用，在输出状态会自动关闭，且每个端口互相独立。通过寄存器 PULLEN 控制位选择是否允许上拉电阻功能。

图 3-3-3 为 CMOS I/O 结构图 3：

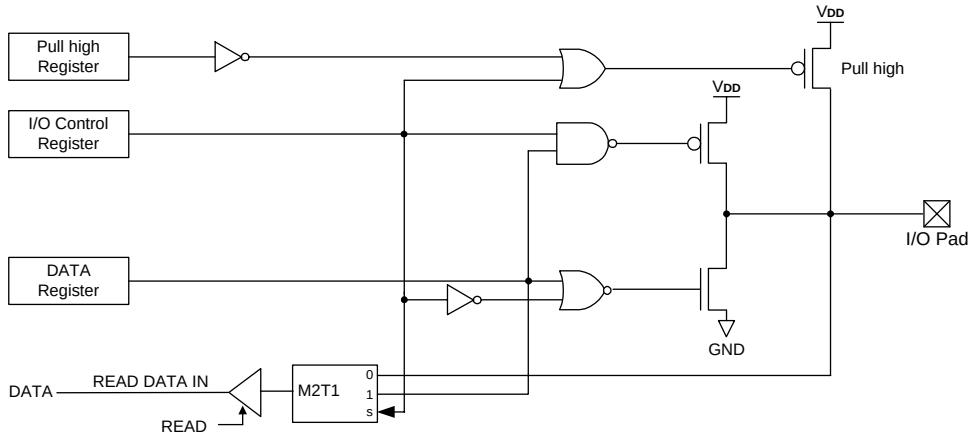


图 3-3-3 CMOS I/O 结构 3

此端口设计中有内建的上拉 CMOS 电阻（几十千欧姆），此功能仅在端口设置为输入状态时才能启用，在输出状态会自动关闭，且每个端口互相独立。 通过寄存器 PULLEN 控制位选择是否允许上拉电阻功能。

- 开漏结构(Open Drain) I/O

图 3-3-4 是开漏结构的 I/O 端口结构：

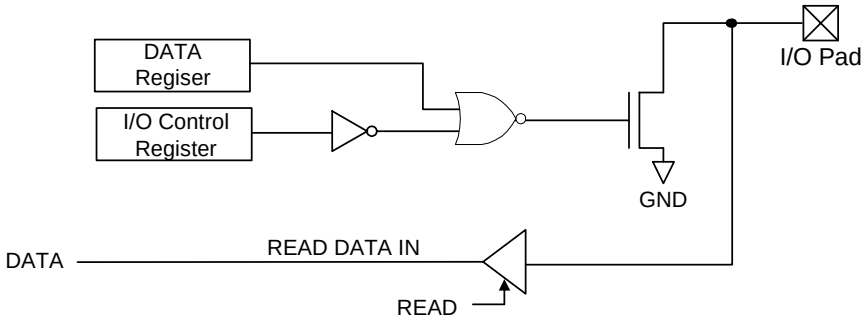


图 3-3-4 开漏结构 I/O 结构图

开漏结构端口输出“0”时，在引脚输出低电平。而输出“1”时，在引脚输出表现为高阻抗。如果要取得高电平则需要使用外部上拉电阻，如图 3-3-5。此外部高电平可以高于或低于单片机的工作电压，使用户可以根据外围电路需求取得需要的输出控制电平。同时这一特性可允许多个开漏结构端口并联使用，而不会造成冲突。

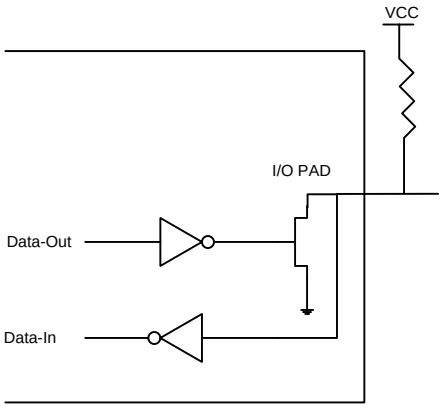


图 3-3-5 开漏结构端口的应用

3.3.2 端口(I/O Port)的操作

每个端口都有独立的输入/输出状态控制寄存器（PxCR)。其中 x 对应 A，B，C，D 等不同端口。通过用户软件对这些状态控制寄存器写入 0 或 1，就可以独立控制每一个端口的输入/输出状态，0 为输入状态，1 为输出状态。一旦向端口输入/输出状态控制寄存器（PxCR）写入数据后，端口将一直保持用户设定的状态，直至下一次被改写。

端口在系统复位后(上电复位、外部 RESET、低电压检测复位及看门狗溢出复位)的初始状态为输入悬空状态（数据寄存器复位为 0、PxCR 中数据复位为 0)。为了避免单片机复位时，端口的不确定状态对外部电路的工作状态产生影响，对处于输入悬空状态的端口，可以通过在外部加上拉/下拉电阻加以解决,使在系统复位开始到端口被重新设置期间,外部电路都能保持在正确的工作状态。

不使用的 I/O 端口不可以处于悬空输入状态。建议设置为输入，同时使用内部或外部上拉或下拉电阻，或设置为输出低电平。

相关寄存器介绍：

端口数据寄存器

地址	第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
$08	PA.3	PA.2	PA.1	PA.0	读/写	端口 A 数据寄存器

此寄存器除了可直接控制输出”0”或”1”于端口外,在大部分产品中还可以用于控制各个端口上拉或下拉电阻的开与关。

端口模式控制寄存器

地址	第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
$16	PACR.3	PACR.2	PACR.1	PACR.0	读/写	端口 A 控制寄存器 写入 0：设置为输入端口 写入 1：设置为输出端口

这个寄存器用于控制端口的输入或输出，每个 bit 都可独立地设为输入或输出。

上拉/下拉电阻控制寄存器

地址	第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
$13	PULLEN	PH/PL	X	X	读/写	端口上拉/下拉电阻控制 PULLEN=0：关闭上拉/下拉电阻功能 PULLEN=1：开启上拉/下拉电阻功能 PH/PL=0：选择下拉电阻 PH/PL=1：选择上拉电阻

此寄存器与端口数据寄存器配合控制上拉电阻的开与关。
下面以实例来说明 I/O 端口的设定。
用户可根据不同的应用选择开启上拉电阻或下拉电阻，下面以选择 PORT A 为例说明上下拉电阻的设定。

设 PORT A 为输入打开下拉

步骤 一 、 对端口模式寄存器 PA3OUT 写 “0” 把端口设为输入

步骤 二 、 对上下拉电阻控制寄存器 PULLEN 和 PH/PL 分别写 “1”和 “0”

步骤 三 、 对端口数据寄存器写 “0”

例 3-3-1：PORTA 设值

```
PORTA     EQU     08H          ;端口数据寄存器
PULLCTL   EQU     15H          ;上(下)拉电阻控制寄存器
PACR      EQU     13H          ;端口模式寄存器

          :
          :
LDI     PACR       ,00H       ;设端口 A 为输入
LDI     PULLCTL    ,08H       ;
LDI     PORTA      ,00H       ;打开端口 A 的下拉电阻
```

端口的读写操作

每个端口都有独立的端口数据寄存器（Px）。其中 x 对应 A，B，C，D 等不同端口。

当端口为输入状态时，对端口数据寄存器（Px）进行读操作可以直接获得引脚上的外部电平状态，对端口数据寄存器（Px）进行写操作则只会改写数据寄存器的内容而不会影响引脚的外部电平状态。

当端口为输出状态时，对端口数据寄存器（Px）进行写操作则会改写数据寄存器的内容，同时在引脚上加以输出。对端口数据寄存器（Px）进行读操作则分为两种：

1) 对于如“CMOS I/O 结构 3”中所示的端口，在输出状态下执行读操作时，读入的数据是数据寄存器（Px）中保存的数据，而非引脚上的电平状态。
2) 对于如“CMOS I/O 结构 1 ，2”中所示的端口，在输出状态下执行读操作时，读入的数据是引脚上的外部电平状态，而非数据寄存器（Px）内数据。

针对上述状况(在输出状态下的读操作的数据来源存在差异)，程序中直接对 I/O 端口进行逻辑指令操作需要慎重处理，因为 I/O 逻辑操作指令的执行过程是先执行读操作，再执行逻辑运算，最后执行写操作，也称为 Read-Modify-Write 指令。

端口的输出驱动能力

每个端口的输出驱动能力由组成 I/O 结构的 P 沟道和 N 沟道场效应管决定。在输出为空负载时，输出高电平接近 VDD，输出低电平接近 0V。依不同型号，其驱动能力达从几毫安至上百毫安不等（请参考具体型号单片机规格书）。

在一些低功耗设计的应用场合中，具备较强输出驱动能力的端口在电路中可以作为部分外围器件的电源使用，输出“1”开始供电，输出“0”，切断外围电路供电，达到省电目的(Power down)。中颖公司 SH69，SH67 系列单片机中,I/O 端口都内置静电（ESD）保护电路，以保护单片机不受静电损伤及干扰程序运行。

3.3.3 特殊应用实例

I/O 端口用于市电（交流电）过零检测

单片机的 I/O 端口存在着内置的 ESD 保护二极管，其作用在于将施于 I/O 端口的外部输入电压钳位在 VDD～0V 范围之内，以保护单片机不被损坏。在确保输入电流不超过 1mA 时，此特性可以用于市电（交流电）的过零检测。电路如下图 3-3-6：

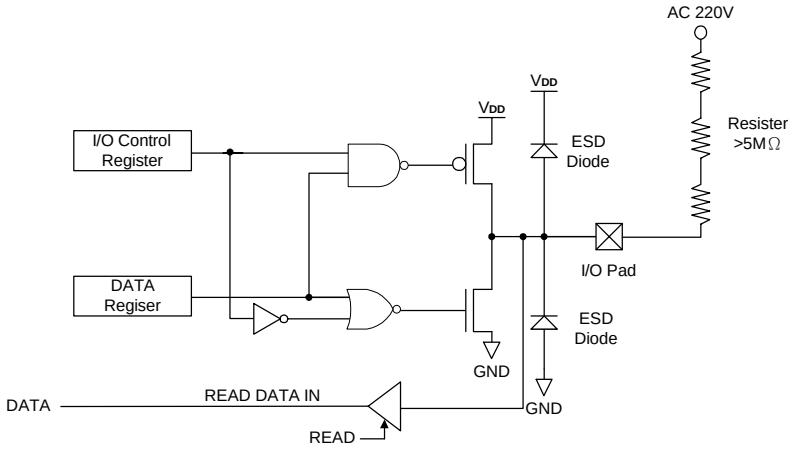


图 3-3-6 I/O 端口用于市电（交流电）过零检测原理示意图

将 I/O 端口和市电（交流电）通过 5 兆～10 兆欧姆的限流电阻直接连接，当市电高于单片机工作电压 VDD 时，I/O 端口通过 ESD 保护二极管使输入电压钳位在 VDD 处，I/O 读入状态为 1。当市电由高于 1/2 VDD 转为低于 1/2 VDD 时，I/O 读入状态也由 1 变 0，此时可以认为是交流电的过零状态。当市电电压低于 0V 后时，I/O 端口通过 ESD 保护二极管使输入电压钳位在 0V 处，I/O 读入状态为 0。

这样就可以通过读入 I/O 端口数据的变化得到交流电的过零点。程序可以通过端口中断或定时扫描端口的处理方式来实现交流电的过零检测功能。

应用中需注意：

1. 交流电一般会掺杂较大的噪声，中断或扫描的处理中要考虑增加去抖动处理才不会多计入过零点。
2. 限流电阻应采用多个电阻串联，以防单个电阻的耐压不够导致损坏限流电阻。
3. 开漏结构的端口因为没有对 VDD 的 ESD 保护二极管，所以不能作此应用。

交流电过零点开关灯泡的应用实例

1）电路设计

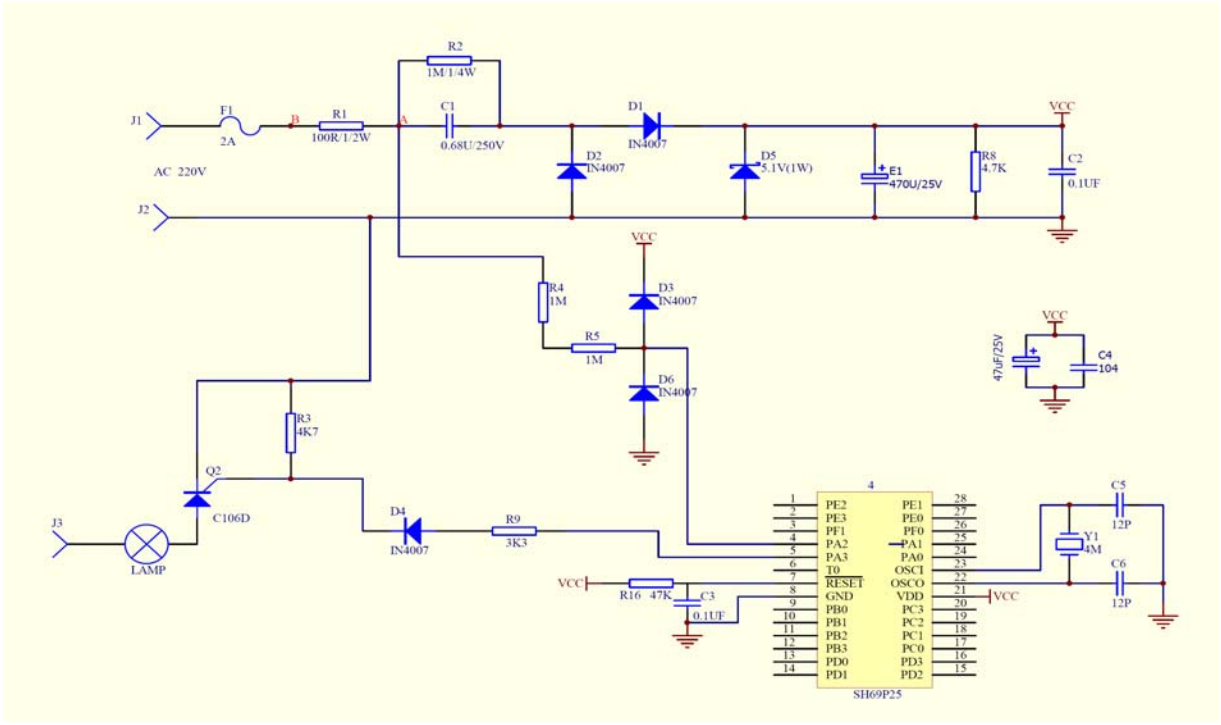


图 3-3-7 交流电过零点开关灯泡电路

电路采用交流电源，4M 晶振，交流电通过阻容降压（R2 和 C1）后，通过二极管 D1 和 D2 进行半波整流，然后再经稳压管 D5 稳压到 5.1V，提供 IC 工作的电压。

可控硅 C106D 是单向可控硅，当芯片检测到交流电过零时，通过 PA.3 口控制可控硅打开/关闭灯泡，图中电阻 R3 和二极管 D4 起保护作用，J3 接于图中 B 点。

交流电经过保险管 F1 和限流电阻 R1 后，再经过两 2 个串联的 1MΩ限流电阻 R4 和 R5 流入 PA.2 口，这样流入 PA.2 口的最大电流仅约为 0.16mA， 远远小于该 I/O 口的最大灌电流；直接接到 PA.2 口上的两个二极管 D3 和 D6（通常芯片 I/O 引脚内部已经有这两个二极管，此例中为了更加安全，在外部也增加了这两个二极管），可以保证输入到 PA.2 口的电压值在芯片的正常工作范围内。

例 3-3-2：交流电过零点开关灯泡

程序主要功能：在 TIMER0 中检测交流过零信号，每当检测到过零时，对可控硅控制口输出进行取反,从而控制灯泡在交流过零时开关。

```
;*************************************************************
;       系统寄存器的定义
;*************************************************************
IE              EQU     00H         ;中断使能标志
IRQ             EQU     01H         ;中断请求标志
TM0             EQU     02H         ;定时器 0 模式寄存器
TL0             EQU     04H         ;定时器 0 装入/计数值低 4 位
TH0             EQU     05H         ;定时器 0 装入/计数值高 4 位
PORTA           EQU     08H         ;PORT A 数据寄存器
```

```
SETTING         EQU       15H
                                    ;BIT3:端口内部上/下拉电阻使能控制
                                    ;BIT2:端口内部上/下拉电阻控制
                                    ;BIT1:端口 B 端口 C 上升沿/下降沿中断控制
PAOUT           EQU       16H       ;PORT A 输入/输出状态控制寄存器
WDT             EQU       1EH       ;看门狗定时器
;*************************************************************
;              用户寄存器的定义
;*************************************************************
AC_BACKUP       EQU       20H       ;各中断中备份累加器 AC 的值
WORK_FLAG0      EQU       21H       ;BIT0:交流过零标志
                                    ;BIT1～3:未使用
PORTA_B         EQU       22H       ;PORTA 口备份寄存器
;*************************************************************
;                    向量地址区域
;*************************************************************
          ORG       00H
          JMP       RESET                   ;跳转到复位服务子程序的入口地址
          RTNI                                    ;保留
          JMP       TIMER0INT                   ;跳转到定时器 0 服务子程序的入口地址
          RTNI                                    ;保留
          RTNI                                    ;端口 B&C 中断服务子程序的入口地址
;*************************************************************
;                定时器 0 中断服务子程序
;*************************************************************
TIMER0INT:
          STA       AC_BACKUP,00H           ;保护累加器 AC 的值
          ANDIM     IRQ,1011B                   ;清除定时器 0 中断请求标志
PASS_ZERO_CHECK_PART:                       ;过零检测部分
          ANDIM     PAOUT,1011B                 ;设置 PA.2 做为输入口
          NOP
          NOP
          LDA       PORTA ,00H              ;读取 PA.2 的值
          BA2       PASS_ZERO_CHECK_PART10
          ANDIM     WORK_FLAG0,1110B        ;设置本次读取 PA.2 的状态为 0
          JMP       PASS_ZERO_CHECK_PART99
PASS_ZERO_CHECK_PART10:
          LDA       WORK_FLAG0,00H          ;检测 PA.2 口上次的状态
           BA0      PASS_ZERO_CHECK_PART99
          ORIM      WORK_FLAG0,0001B        ;设置交流过零标志

          EORIM     PORTA_B,1000B               ;打开/关闭可控硅输出
          STA       PORTA,00H
          ORIM      PAOUT,1000B
          JMP       PASS_ZERO_CHECK_PART99
PASS_ZERO_CHECK_PART99:
TIMER0INT_OUT:
          LDI       IE,0100B                    ;打开定时器 0 中断使能标志:进入中断后,硬件会自
                                                ;动清除中断使能标志,所以中断服务子程序执行后
                                                ;要手动恢复中断使能标志
          LDA       AC_BACKUP,00H           ;恢复累加器 AC 的值
          RTNI
;*************************************************************
;                  程序初始化部分
;*************************************************************
RESET:
```

```
        LDI     IE,00H                  ;清除中断使能标志
        LDI     TM0,0FH                 ;设置定时器 0 的模式,此处设定为 1 分频
        LDI     TL0,06H                 ;设定定时器 0 的定时时间,工作频率为 4MHZ,此处
                                        ;设定定时时间为 250 微秒
        LDI     TH0,00H                 ;先装入定时器 0 计数值低 4 位,再装入定时器 0 计
                                        ;数值高 4 位
        LDI     PORTA,00H               ;设置 PORTA 口为低电平
        LDI     PAOUT,1011B             ;设置 PA.3 作为输出口，PA.2 作为输入口
        LDI     PORTA_B,00H             ;初始化 PORTA 口备份寄存器
        LDI     AC_BACKUP,00H           ;初始化累加器 AC 的备份寄存器
        LDI     WORK_FLAG0,00H          ;初始化 WORK_FLAG0 寄存器
        LDI     IRQ,00H                 ;清除中段请求标志
        LDI     IE,0100B                ;打开定时器 0 中断使能标志
        JMP     MAIN_LOOP_PART          ;跳转到主程序部分
;************************************************************
;               主程序部分
;************************************************************
MAIN_LOOP_PART:
        LDI     IE,0100B                ;打开定时器 0 中断使能标志
        LDI     WDT,0FH                 ;清看门狗寄存器，防止看门狗溢出
        NOP
        HALT                            ;进入 HALT 低功耗模式
        NOP
        NOP
        JMP     MAIN_LOOP_PART
        END                             ;程序结束
```

2）程序运行时过零检测口 PA.2 和可控硅控制口 PA.3 的波型如图 3-3-8。

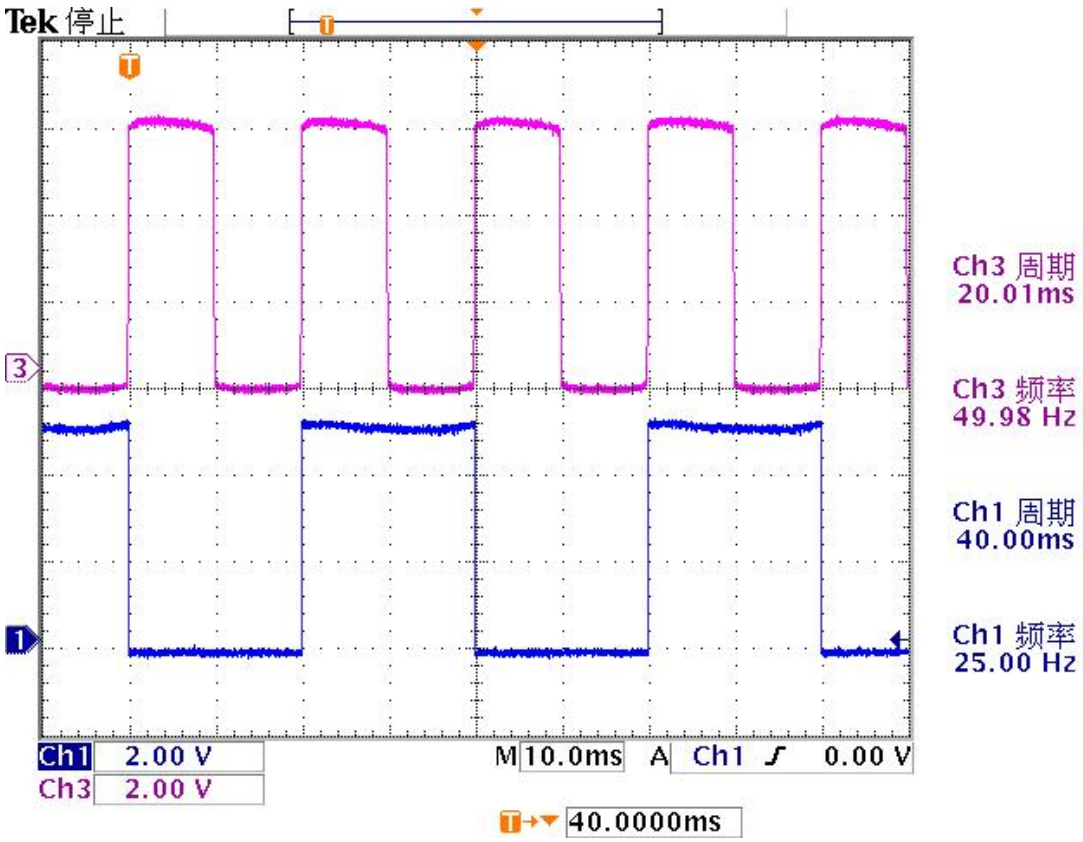


图 3-3-8 PA.2 & PA.3 波形图

图中 CH3 波型为过零检测口 PA.2 的输入波型，CH1 的波型为可控硅控制口 PA.3 的输出波型。

从图中可以看到过零信号的周期约为 20.01ms,频率约为 49.98HZ，可控硅控制口 PA.3 输出波型的周期为 40.00ms，频率为 25.00HZ，与程序设计的要求完全符合。

用 I/O 测量电阻值

在日常生活中，我们时常需要通过温度来控制一些对象，而温度参数大都通过 A/D 模块采样来获得。对于高精度的控制对象，我们肯定需要通过 A/D 模块采样来求得参数，在一些精度不需要太高，采样参数很少，而且芯片成本要低（没有 AD 模块）的情况下，我们可以通过 I/O 来测量参数。例如房间空气调节器或者空调器遥控器，需要做一个实际环境温度的测量，在精度不要求很高的情况下，完全可以通过 I/O 来测量温度参数。

1） 测量原理

首先我们熟悉一下温度传感器。通常温度传感器是负系数的，即温度升高，电阻值降低，如果温度降低，则传感器的电阻值增大。然后我们可以通过阻容 RC 测量充放电时间来间接求得电阻值。考虑到环境变化下，电阻电容的参数都会有偏移，所以我们采用比较的方式来测量电阻值。电路中我们采用一个高精度的金属膜电阻作为参考电阻，然后用所测试的电阻的充电时间同参考

电阻的充电时间比较来求取电阻值。

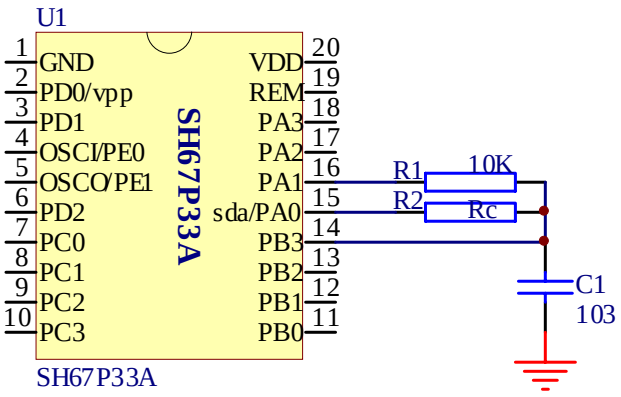


图 3-3-9 SH67P33A I/O 测温原理图

在实际应用中，我们要考虑到测量的误差，所以我们可以在求取参数时采用采样多次后求取平均值的方法来降低误差。通常情况下我们采样 10-20 次左右就可以了。在求取了时间参数后，我们就可以采用运算或者查表的方式来求取电阻值。同时我们采用芯片内部定时器来计数以减少误差，这比软件计数要准备和精度高。

RC 测温度基本原理是，通过电阻向 C 充电，用 TMR0，或 TMR1，或其它办法检查充电的时间，有了时间就算出电阻值，然后就可以查表计算出温度。由于单个 RC 电路要受 R、C 的误差限制，一般的是用分别用 2 个电阻（一个是热敏电阻，一个是标准电阻）对同一个 C 充电。

2 个电阻分别接一个 I/O 口，分别对 C 充电。在一个端口作充电时间测量的时候，另一个端口要设置成输入，这样这个端口就成高阻状态而减少对测量端口的干扰。在充电测试完毕后一定要将 RC 彻底的放电再进行下一次的充电时间测量。

公式是：$R_{热敏}=T_{热敏充电时间}*R_{标准}/T_{标准充电时间}$。

可以看出，公式排除了电容 C 的误差影响，精度提高了，并且对电容的选择要求就低了。对标准电阻要选择温度系数低的金属膜精密电阻。

设计过程中要注意电阻、电容值的选择，保证充电时间要落在记时器里面。为了保证计算方便，一般的是标准电阻选所测温度对应的热敏电阻值范围内的中间值。

另：C 在一次充电完成后要及时彻底放电。在 R、C 连接点并一个 I/O 口，C 充电时候，I/O 口设成输入并记录电平跳高，放电就设成输出并置 0

2）计算方法：

$T_{ref} = k \times R_f \times C$ (1)

$T_{rt} = k \times R_t \times C$ (2)

由方程式（1）、（2）可得到：

T_{ref} / T_{rt} = R_f / R_t

由此可以算出 R_t 的值，再经查表即可得到温度值。

3）用 RC 充放电测温之局限性：

由于随着 VDD 变动电阻值也相应变化，因此当 VDD 下降时温度测量误差将会增大，而且在高温情况下误差较明显。

4）测量精度的控制

时间的测量我们采用芯片内部的定时器技术来获取。在测量之前我们可以先估算一下测试的结果值，然后确定需要多少位寄存器来计数。

5）程序设计

下面例程是按照上面原理图来测试图中的 R_c，例程中 AD 采样的精度设置为 12 位。参照上面原理图计算，参考电阻的时间参数大概是：$10K*10^4*10^{-12}$=100us。所以将定时器设计成 1us 计数，考虑到测试电阻值的大小，设计为 3 个寄存器保留参数值。这样精度就是 12 位了。

例 3-3-3 用 I/O 测量电阻

```
;;;;;;;;;;;;;;;system define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 IE              EQU        00H        ;中断允许标志
 IRQ             EQU        01H        ;中断请求标志
 PA              EQU        08H        ;PORT A 数据寄存器
 PB              EQU        09H        ; PORTB 数据寄存器
 PC              EQU        0AH        ; PORT C 数据寄存器
 PD              EQU        0BH        ; PORT D 数据寄存器
 PE              EQU        0CH        ; PORT E 数据寄存器
 PACR            EQU        16H        ; PORT A 输入输出控制寄存器
 PBCR            EQU        17H        ; PORT B 输入输出控制寄存器
 PCCR            EQU        18H        ; PORT C 输入输出控制寄存器
 PDCR            EQU        19H        ; PORT D 输入输出控制寄存器
 PECR            EQU        1AH        ; PORT E 输入输出控制寄存器
 PMOD            EQU        12H        ;BIT0-BIT2：载波输出时钟预分频设定
                                       ;BIT3：下拉允许控制位
 DPL             EQU        10H        ;数据指针低位
 DPM             EQU        11H        ;数据指针中间位
 DPH             EQU        12H        ;数据指针高位
 INX             EQU        0FH        ;间接索引寄存器
;;;;;;;;;;;;;;;; ram define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ACCBK            EQU        20H        ;备份 ACC 寄存器
DELAYL           EQU        21H        ;延时函数输入参数低位
DELAYM           EQU        DELAYL+1   ;延时函数输入参数中位
DELAYH           EQU        DELAYL+2   ;延时函数输入参数高位

TEMP             EQU        24H        ;临时使用寄存器
TEMPL            EQU        25H
TEMPH            EQU        26H
```

```
DPL_BK          EQU     27H             ;移位函数临时寄存器
DPM_BK          EQU     DPL_BK+1        ;移位函数临时寄存器
PARA_BYTE_BK    EQU     DPL_BK+2        ;移位函数临时寄存器
PARA_BYTE       EQU     DPL_BK+3        ;移位函数输入参数，移位字节
PARA_BIT        EQU     DPL_BK+4        ;移位函数输入参数，移位位数
TEMPSUB         EQU     DPL_BK+5        ;移位函数临时寄存器

PBBUF           EQU     2DH             ;PORTB 输出缓冲寄存器
T0SUM           EQU     2EH             ;TIMER0 溢出计数器
ADTIME          EQU     2FH             ;AD 采样计数器

REFER_T0L       EQU     30H             ;参考电阻当次采样定时器低位寄存器
REFER_T0H       EQU     31H             ;参考电阻当次采样定时器高位寄存器
REFER_T0SUM     EQU     32H             ;参考电阻当次采样定时器溢出寄存器
TEST_T0L        EQU     34H             ;测试电阻当次采样定时器低位寄存器
TEST_T0H        EQU     35H             ;测试电阻当次采样定时器高位寄存器
TEST_T0SUM      EQU     36H             ;测试电阻当次采样定时器溢出寄存器

REFER_SUML      EQU     38H             ;参考电阻采样平均值低位
REFER_SUMM      EQU     39H             ;参考电阻采样平均值中位
REFER_SUMH      EQU     3AH             ;参考电阻采样平均值高位
TEST_SUML       EQU     3CH             ;测试电阻采样平均值低位
TEST_SUMM       EQU     3DH             ;测试电阻采样平均值中位
TEST_SUMH       EQU     3EH             ;测试电阻采样平均值高位

DIVL            EQU     44H             ;除数
DIVM            EQU     DIVL+1
DIVH            EQU     DIVL+2
BDIV_PL         EQU     DIVL+3
BDIVL           EQU     DIVL+4          ;被除数
BDIVM           EQU     DIVL+5
BDIVH           EQU     DIVL+6

RESULT_PL       EQU     DIVL+7          ;结果的小数部分
RESULTL         EQU     DIVL+8          ;商的整数低位
RESULTM         EQU     DIVL+9          ;商中整数间位
RESULTH         EQU     DIVL+0AH        ;商的整数高位
;;;;;;;;;;;;;;;; port define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
REFER           EQU     02H             ;PA1
TEST            EQU     01H             ;PA0
SAMPLE          EQU     08H             ;PB2
ADPORTCR        EQU     PBCR            ;AD 采样端口控制寄存器
ADPORT          EQU     PB              ;AD 采样端口
IET0            EQU     04H
SCALE1          EQU     0111B
SCALE8          EQU     0100B
SCALE2048       EQU     0000B
            ORG     0000H
            JMP     RESET                   ;复位入口地址
            JMP     ONLYRET
            JMP     T0ISR                   ;定时器 T0 中断入口地址
            JMP     ONLYRET
            JMP     PBCISR                  ;PBC 中断入口地址

PBCISR:
```

```
        STA         ACCBK,00H
        JMP         TORET

TOISR:
        STA         ACCBK,00H               ;备份 ACC 寄存器
        LDI         WDT,08H
        ADIM        TOSUM,01H               ;定时器溢出寄存器加 1

TORET:
        LDI         IRQ,00H                 ;清除中断请求标志
        LDI         IE,IET0                 ;允许定时器溢出中断
        LDA         ACCBK,00H               ;恢复 ACC 寄存器的值
ONLYRET:
        RTNI                                ;退出中断子程序

RESET:
        NOP                                 ;上电稳定空跑 4 条指令
        NOP
        NOP
        NOP
        LDI         WDT,08H                 ; 清除 watchdog 计数
        LDI         IRQ,00H                 ;清除中断请求标志
        LDI         IE,00H                  ;不允许中断

        LDI         REM,0H                  ;初始化设置端口
        LDI         PACR,0FH
        LDI         PBCR,0FH
        LDI         PCCR,0FH
        LDI         PDCR,06H
        LDI         PECR,0FH
        LDI         PA,00H
        LDI         PB,00H
        LDI         PC,00H
        LDI         PD,00H
        LDI         PE,00H
        LDI         PMOD,0101B              ;不允许下拉电阻

CLEARRAM:                                   ;清 0  RAM 020H-4FH 区
        LDI         DPL,00H
        LDI         DPM,02H
        LDI         DPH,00H
?CLRINX:
        LDI         INX,00H
        ADIM        DPL,01H
        BNC         ?CLRINX
        ADIM        DPM,01H
        ANDIM       DPM,07H
        SBI         DPM,05H
        BNC         ?CLRINX

ADSAMPLE:
        LDI         ADPORTCR,1111B          ;设定采样端为输出
        LDI         ADPORT,0000B            ;输出低电平放电

        LDI         WDT,08H                 ;清除 watchdog 计数
        LDI         PACR,1111B              ;将电容 C 彻底放电
```

```
LDI         PA,00H
CALL        PARA10MS
LDA         ADPORT,00H              ;检测是否放电完毕，没有继续放电
BA2         $-3

LDI         ADPORTCR,1011B          ;设定采样端为输入
LDI         ADPORT,0000B            ;输入端拉低

LDI         TOSUM,00H               ;清 0 定时器溢出计数器
LDI         WDT,08H                 ;清除 watchdog 计数
LDI         PACR,0010B              ;将参考电阻端为输出，其它端口输入
LDI         PA,0010B                ;参考电阻端输出高电平充电
LDI         TOM,SCALE1              ;定时器预分频设为 1us
LDI         TOL,00H                 ;定时器清 0
LDI         TOH,00H
LDI         IE,IET0                 ;允许定时器中断
NOP
LDA         ADPORT,00H              ;检测是否充电完毕
BA2         $+2                     ;充电完就再检测一次确认
JMP         $-3                     ;没有就继续充电
NOP
LDA         ADPORT,00H              ;检测是否充电完毕
BA2         $+2                     ;充电完毕就读取参数
JMP         $-7                     ;没有就继续充电
LDA         TOH,00H                 ;保存充电参数
STA         REFER_TOH,00H
LDA         TOL,00H
STA         REFER_TOL,00H
ADDM        REFER_SUML,00H
LDA         REFER_TOH,00H
ADCM        REFER_SUMM,00H
LDA         TOSUM,00H
STA         REFER_TOSUM,00H
ADCM        REFER_SUMH,00H
LDI         TOSUM,00H

LDI         ADPORTCR,1111B
LDI         ADPORT,0000B
LDI         WDT,08H
LDI         PACR,1111B
LDI         PA,00H
CALL        PARA10MS
LDA         ADPORT,00H
BA2         $-3

LDI         ADPORTCR,1011B
LDI         ADPORT,0000B

LDI         TOSUM,00H
LDI         WDT,08H
LDI         PACR,0001B
LDI         PA,0001B
LDI         TOL,00H
LDI         TOH,00H
NOP
LDA         ADPORT,00H
```

```
        BA2         $+2
        JMP         $-3
        NOP
        LDA         ADPORT
        BA2         $+2
        JMP         $-7
        LDA         T0H,00H
        STA         TEST_T0H,00H
        LDA         T0L,00H
        STA         TEST_T0L,00H
        ADDM        TEST_SUML,00H
        LDA         TEST_T0H,00H
        ADCM        TEST_SUMM,00H
        LDA         T0SUM,00H
        STA         TEST_T0SUM,00H
        ADCM        TEST_SUMH,00H

        ADIM        ADTIME,01H                  ;AD 采样技术其加 1
        SBI         ADTIME,01H                  ;判断是否是第一次采样
        BAZ         ADSAMPLE                    ;是第一次就不求平均值
        SBI         ADTIME,15                   ;是否已经采样了 15 次
        BC          TEST_OK                     ;是就退出采样

        LDI         DPL,REFER_SUML & 0FH        ;求参考电阻充电时间平均值
        LDI         DPM,(REFER_SUML >> 4) & 0FH
        LDI         PARA_BYTE,03H
        LDI         PARA_BIT,01H
        CALL        SHRC_N
        LDI         DPL,TEST_SUML & 0FH         ;求测试电阻充电时间平均值
        LDI         DPM,(TEST_SUML >> 4) & 0FH
        LDI         PARA_BYTE,03H
        LDI         PARA_BIT,01H
        CALL        SHRC_N
        JMP         ADSAMPLE                    ;继续下一次采样

TEST_OK:
        LDA         REFER_SUML,00H              ;用除法求取测试电阻同参考电阻的比值
        STA         DIVL,00H
        LDA         REFER_SUMM,00H
        STA         DIVM,00H
        LDA         REFER_SUMH,00H
        STA         DIVH,00H
        LDA         TEST_SUML,00H
        STA         BDIVL,00H
        LDA         TEST_SUMM,00H
        STA         BDIVM,00H
        LDA         TEST_SUMH,00H
        STA         BDIVH,00H
        CALL        DIV
TOHALT:
        LDI         T0M,SCALE2048               ; 定时器预分频设为 2ms
        LDI         T0L,06H                     ;设定每 500ms 采样一次
        LDI         T0H,00H

        LDI         WDT,08H                     ; 清除 watchdog 计数
        LDI         IRQ,00H
```

```
        LDI         IE,IET0
        NOP
        HALT
        NOP
        NOP
        NOP
        JMP         RESET
;=================================================
;FunctionName:SHRC_N
;Input:DPL,DPM,DPH,PARA_BYTE(连续移位字节数),PARA_BIT(移位位数)
;Output:
;Temporary:TEMPSUB,DPL_BK,DPM_BK,PARA_BYTE_BK
;CalledSubroutine:
;Description:将指定同一 bank 中的连续存储单元右移指定位数
;=================================================
SHRC_N:
        LDA         DPL,00H                 ;备份 rdpl 参数-->zDPLBK,zDPMBK
        STA         DPL_BK,00H
        LDA         DPM,00H                 ;备份 rdpm 参数
        STA         DPM_BK,00H
        LDA         PARA_BYTE,00H           ;备份 zpara_byte 参数
        STA         PARA_BYTE_BK,00H
        BAZ         SHFRET                  ;如果移位字节数=0 就退出函数
        LDA         PARA_BIT,00H
        BAZ         SHFRET                  ;如果移位位数=0 就退出函数
?SHRFIRST:
        LDA         INX,00H
        SHR                                 ;右移间接指定寄存器 1 位
        STA         INX,00H
        SBIM        PARA_BYTE,01H           ;移位字节数减 1
        BAZ         ?SHRNEXT                ;结果=0 就转到移位下一位
        ADIM        DPL,01H                 ;间接寄存器指针低位+1
        EOR         DPL,00H
        ADCM        DPM,00H
?SHRLOOP:
        LDI         TEMPSUB,00H
        LDA         INX,00H
        SHR                                 ;右移间接指定寄存器 1 位
        STA         INX,00H
        BNC         $+2
        LDI         TEMPSUB,08H             ;如果进位 C=1, ztempsub=8
        SBIM        DPL,01H                 ;返回上一寄存器
        EOR         DPL,00H
        SBCM        DPM,00H
        LDA         TEMPSUB,00H             ;将高一字节的低位加入
        ADDM        INX,00H
        SBIM        PARA_BYTE,01H           ;移位字节数减 1
        BAZ         ?SHRNEXT                ;结果=0 就转到移位下一位
        ADIM        DPL,02H
        EOR         DPL,00H
        ADCM        DPM,00H
        JMP         ?SHRLOOP
?SHRNEXT:
        SBIM        PARA_BIT,01H            ;移位数减 1
        BAZ         SHFRET                  ;结果=0 就退出函数
        LDA         DPL_BK,00H              ;恢复要移位寄存器的地址
```

```
        STA         DPL,00H
        LDA         DPM_BK,00H
        STA         DPM,00H
        LDA         PARA_BYTE_BK,00H            ;恢复要连续移位的字节数
        STA         PARA_BYTE,00H
        JMP         ?SHRFIRST
SHFRET:
        RTNI

PARA10MS:                                       ;延时 10ms 参数
        LDI         DELAYL,02H
        LDI         DELAYM,0EH
        LDI         ELAYH,04H
TIME_DELAY:                                     ;调用函数延时 7X+2 条指令周期
        LDI         WDT,08H                     ; 清除 watchdog 计数
        SBIM        DELAYL,01H                  ;
        EOR         DELAYL,00H                  ;
        SBCM        DELAYM,00H                      ;
        EOR         DELAYM,00H                  ;
        SBCM        DELAYH,00H                      ;
        BC          TIME_DELAY                  ;
        RTNI                                    ;
;================================================
;FunctionName:DIV
;Input:
;Output:RESULT_PL,RESULTL,RESULTM,RESULTH
;Temporary:
;CalledSubroutine:
;Description:除法子程序，精确到一个小数字
;================================================
DIV:
        LDI         WDT,08H                     ;CLEAR WATCHDOG TIMER
        LDI         RESULTL,00H
        LDI         RESULTM,00H
        LDI         RESULTH,00H
        LDI         BDIV_PL,00H
        LDI         RESULT_PL,00H
?DSUB:
        LDA         DIVL,00H
        SUBM        BDIV_PL,00H
        LDA         DIVM,00H
        SBCM        BDIVL,00H
        LDA         DIVH,00H
        SBCM        BDIVM,00H
        EOR         BDIVM,00H
        SBCM        BDIVH,00H
        BNC         DIV_RET
        ADIM        RESULT_PL,01H
        EOR         RESULT_PL,00H
        ADCM        RESULTL,00H
        EOR         RESULTL,00H
        ADCM        RESULTM,00H
        EOR         RESULTM,00H
        ADCM        RESULTH,00H
        JMP         ?DSUB
DIV_RET:
```

```
RTNI

END
```

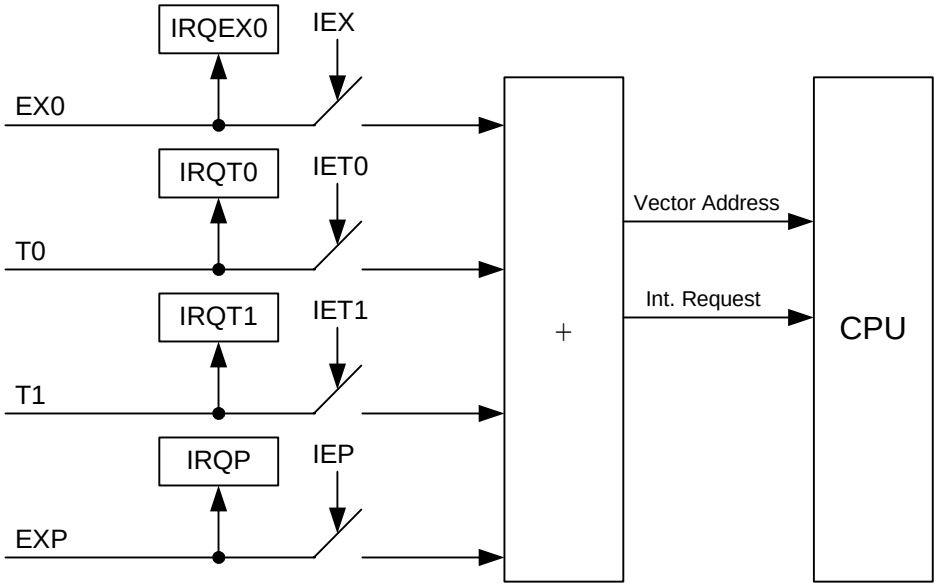
3.4 中断系统(Interrupt)

中断系统在单片机系统中起着非常重要的作用。SH6xxx 系列单片机的中断系统，功能强大，驱动事件种类涵盖各种应用场合的需求。

在叙述 SH6xxx 的中断系统之前，先介绍一下两个概念，中断向量和中断源。

中断向量(Interrupt Vector)是指在有中断事件发生且该中断功能为开放的前提下，系统首先暂停正常的程序，将用于返回时的 PC 值和 CY 压入堆栈，然后将依据定义好的该中断事件的程序入口地址将程序转向运行中断服务程序，那么该中断程序入口地址就称为中断向量。而中断源是指能够使触发系统产生一个中断请求的外部或内部的信号或激励。

3.4.1 中断向量和中断源的关系



EX0: External Interrupt 0

T0: Timer Interrupt 0

T1: Timer Interrupt 1

EXP: External Port Interrupt

图 3-4-1 中断向量和中断源结构示意图

CPU6610(C/D/E)中断系统最多可提供 4 个中断向量：

向量地址	对应的中断源	优先级
$0001	外部中断 0	取决于用户程序
$0002	定时器中断 0	
$0003	定时器中断 1	
$0004	Port 中断	

上图和上表列出了 SH6xxx 系列的 CPU 的中断方块图，从中可以看出：

b）CPU 最多可以提供 4 个中断向量；

c）对应的每一中断均有一个中断允许位 IExx；

d）有中断事件产生时，不管对应的中断允许位打开与否，中断标志位均被置起来，所以在也可以通过查询的方式判断有无中断产生；

四个中断源的优先级是相同的，所以一般情况下，不会出现中断嵌套的现象(中断嵌套可以通过软件实现)；

3.4.2 中断响应过程

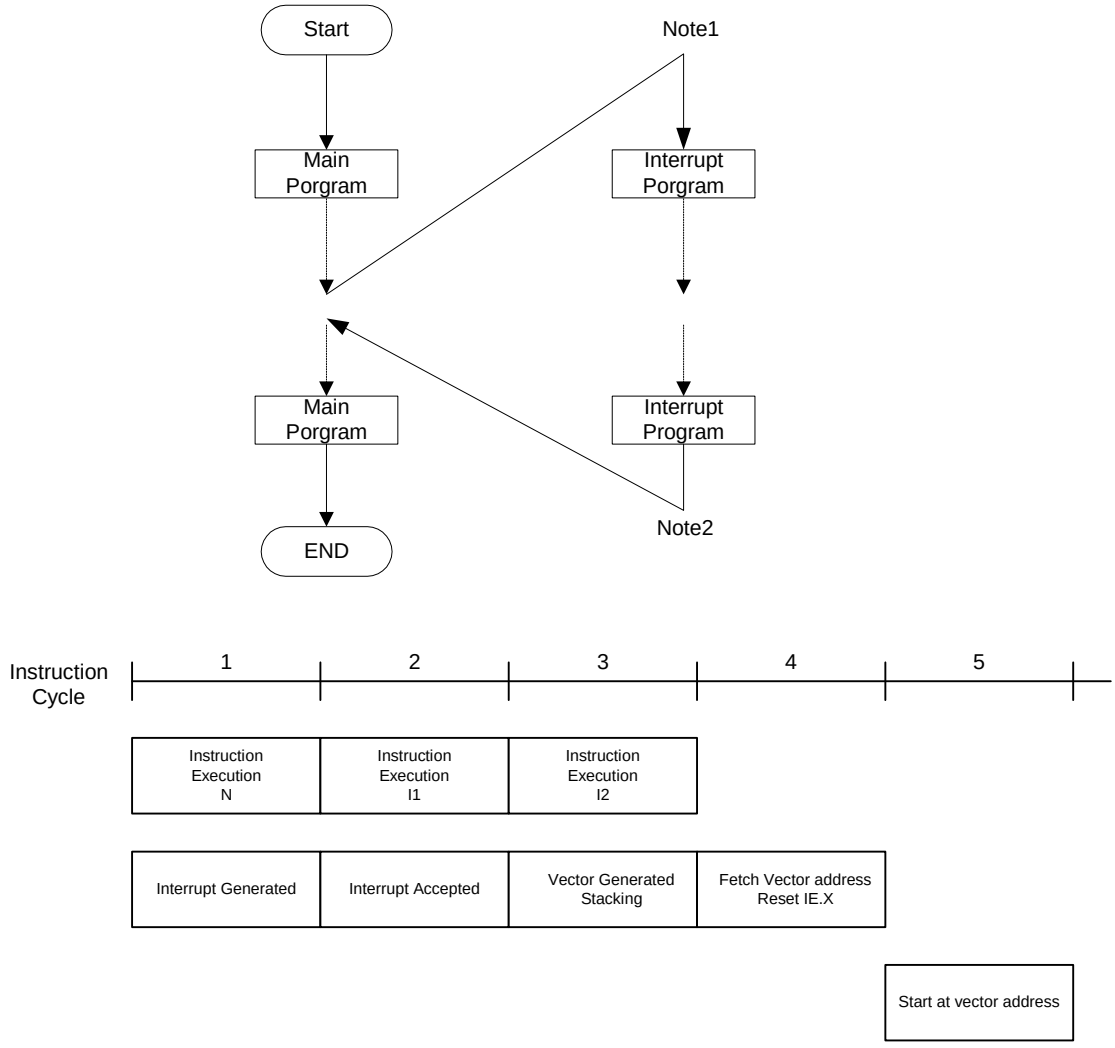


图 3-4-2 中断响应过程示意图

系统内部在以下三种情况都会产生硬件复位，程序开始地址被复位为 0000H，除了有特殊说明的寄存器外，其它寄存器恢复为初始化值：

a) 初始或重新接上电源(POR, Power on reset)；

b) 在 RESET 管脚上输入 Reset 脉冲；

c) 看门狗电路溢出(WatchDog overflow)；

d) 其它可以产生硬件复位的事件发生，如低电压检测检测电路(LVR, Low voltage reset)等，

此部分电路在各自的 datasheet 中均有详细描述；
由图 3-4-2 可以看出，程序正常运行时，若有中断产生，系统会中断正常的程序执行，强制跳转至中断服务程序，中断服务程序执行完后，再返回主程序中断处，继续执行。
要满足系统能够响应所指定的中断，程序中必须将该中断对应的中断允许位置为 ON。
上图中在 Note1 处系统的响应为：
a) 中断响应时间(从中断事件产生到系统开始响应的时间间隔)为 4 个系统时钟；
b) 主程序中断处的下一条待执行指令的地址(PC 值)和当前 CY 值被硬件压入堆栈；
c) 所有的中断允许位被硬件清 0；
上图中在 Note2 处系统的响应为：
a) 中断返回时间(从中断返回指令执行到系统开始执行主程序的时间间隔)为 2～3 个系统时钟；
b) 主程序中断处的下一条待执行指令的地址(PC 值)和当前 CY 值被硬件弹出堆栈；
c) 返回主程序，继续执行被中断的操作；

3.4.3 中断源

SH6xxx 中断部分控制寄存器和中断标志寄存器如下表：

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	备注
$00	IEX	IET0	IET1	IEP	R/W	中断控制寄存器
$01	IRQEX0	IRQT0	IRQT1	IRQP	R/W	中断标志寄存器

各控制位和标志位的意义如下：
外部中断 0
IEX：外部中断 0 中断允许位
　　1：中断允许
　　0：中断禁止
IRQEX0：外部中断 0 中断标志位
　　1：外部中断 0 有事件产生
　　0：外部中断 0 没有事件产生
外部中断 0 在 SH6xxx 系列产品中一般被定义为某一 I/O 口的下降沿中断，如 PORTA.3 或 PORTA.0。

外部 Port 中断(PBC 中断)
IEP：外部 Port 中断中断允许位
　　1：中断允许
　　0：中断禁止
IRQEXP：外部 Port 中断中断标志位
　　1：外部 Port 中断有事件产生
　　0：外部 Port 中断没有事件产生
外部中断 0 在 SH6xxx 系列产品中一般被定义为某一组，两组甚至更多组 I/O 口的下降沿中断，如 PORTB/C，一般也称 PBC 中断。
外部 PORT 中断和外部中断 0 其实都属于外部的 PORT 中断，但两者是有区别的：
a) 外部中断 0 的信号来源是一个单独的 I/O Port，其功能与其它的 I/O Port 没有任何关系；
b) 外部 Port 中断的信号来源是一组或两组或者更多组的 I/O Port，某一个 Port 的功能与其它的 I/O Port 的状态有密切的关系，如下图 3-4-3：

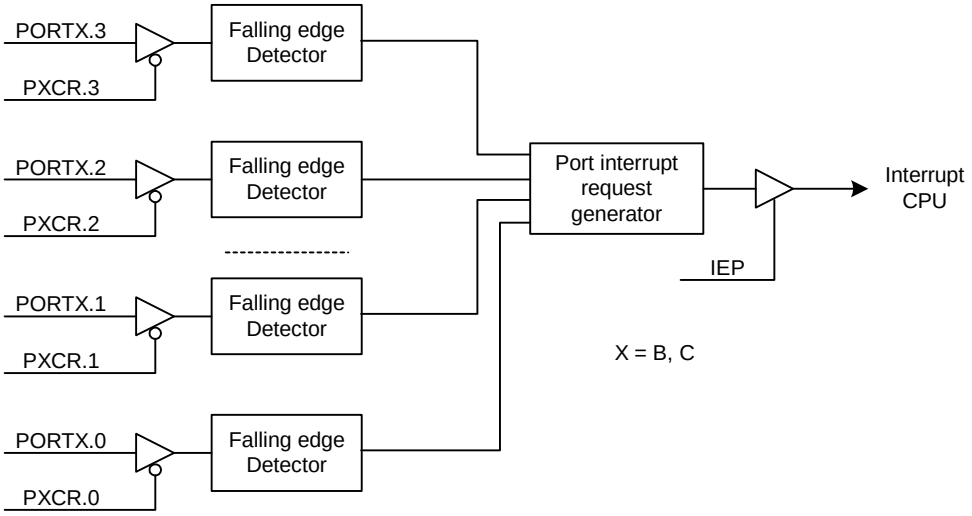


图 3-4-3 外部 Port 中断系统结构图

由图中可以看出：

- I/O port 只有在输入状态下，才可以作为外部 Port 中断源；
- 所有作为外部 Port 中断源的 I/O Port 信号之间为”与”(AND)的关系，此关系意味着某一 I/O port 的中断得以响应的条件是其余相关 Port 必须保持在高电平状态，否则将不能响应；
- 此例中外部 Port 中断的中断源为 PortB3～0，PortC3～0，共 8 个；

定时器中断 0

IET0：定时器中断 0 中断允许位

1：中断允许

0：中断禁止

IRQT0：定时器中断 0 中断标志位

1：定时器中断 0 有溢出产生

0：定时器中断 0 没有溢出

一般而言，在 SH6xxx 产品中，定时器中断 0 定义为 Timer0 的中断，。

定时器中断 1

IET1：定时器中断 1 中断允许位

1：中断允许

0：中断禁止

IRQT1：定时器中断 1 中断标志位

1：定时器中断 1 有溢出产生

0：定时器中断 1 没有溢出

一般而言，在 SH6xxx 产品中，定时器中断 1 定义为 Timer1 或 BaseTimer 的中断。

3.4.4 中断源的扩展

由于 SH6xxx 系列产品的 CPU 中断源和中断向量只有 4 个(EX0，T0，T1，EXP),而每颗具体的产品其中断源的类型和数量均有所不同，此时就存在一个中断源扩展的问题，即几个中断源共享一个中断向量。当这些中断发生时，其中断入口地址是一致的，具体是哪一个中断源产生的中断，就需要程序响应中断后，通过查询每个事件的独立标志位来判断。

例如：SH67K93

Addr	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$00	IEX0	IET0	IET1	IEP	R/W	Interrupt enable flags
$01	IRQEX0	IRQT0	IRQT1	IRQP	R/W	Interrupt request flags
$17		IRQ_ FSKTX	IRQ_ FSKIN	IRQ_PA0	R/W	Bit0: the interrupt flag of the PORTA0 Bit1: the interrupt flag of the FSK CMP Bit2: the interrupt flag of the FSK generator

在 SH67K93 产品中，中断源较多，有：

a) PortA.0;

b) FSK 信号输入沿中断;

c) FSK 产生器中断;

d) Timer0 定时中断;

e) Timer1 定时中断;

f) PortC/D/E 沿中断;

中断源数目有 6 个，大于 4 个，所以在 SH67K93 中就进行了中断源的扩展。

具体为 PortA.0 沿中断/FSK 信号输入沿中断/ FSK 产生器中断共享一个中断向量 EX0，当三个中断源中有任一事件产生时，首先其对应的 IRQ 标志位会被置起，在 IEX0=1 的情况下，CPU 响应外部中断 0 中断，程序再通过查询 IRQ_FSKTX、IRQ_FSKIN 和 IRQ_PA0 的状态即可判别具体的中断源。

Timer0 和 Timer1 的中断没有扩展。外部 Port 中断也进行了扩展,共支持 PortC/D/E 12 个 I/O port 的沿中断。

3.4.5 中断编程注意事项

a) 中断允许位（IE）在系统进入中断服务程序后自动由硬件清零，如果在中断服务程序中 IE 置起过早，可能由于 IRQ 在 IE 置起前没有清零，或在 IE 置起后，中断服务程序退出前又有中断事件产生，就会导致中断嵌套，当堆栈的层数超过限制时将导致系统复位，所以最好是在中断返回指令 RTNI 前才将 IE 置起;

例如：

```
    :
LDI  IRQ, 0    ;clear the interrupt flag
LDI  IE, 0FH   ;set the interrupt enable control bit for next interrupt
RTNI
    :
```

b) 为了避免同一次中断多次响应，在中断服务程序退出前，对应的中断标志位一定要用软件清

零。否则会发生中断服务程序刚退出，又会立即进入的现象；

例如：

```
        :
LDI  IRQ, 0    ;clear the interrupt flag
LDI  IE, 0FH   ;set the interrupt enable control bit for next interrupt
RTNI
        :
```

c) 在 IE 被置起指令的后续两个指令周期内发生的中断，将不会被 CPU 响应；

d) 如果 IE 置起指令是放在中断程序之外，而且程序中有 HALT 或 STOP 指令，则 HALT/STOP 指令与 IE 置起指令尽量靠近，否则会出现程序无法退出 HALT/STOP 模式而出现“假死机”。

例如：

程序：

```
        :
LDI  IE, 0FH      ; enable interrupt
NOP
NOP
HALT
```

如果在两条 NOP 指令之后，立刻有中断事件产生或 IRQ 中断标志不等于 0，CPU 将立即响应对应的中断，IE 被硬件清零。当程序从中断服务程序返回后，CPU 开始执行 HALT/STOP 指令。由于 IE 已被清零，所有的中断都被禁止，系统也无法由中断唤醒，一直处于 HALT/STOP 模式，呈现一种假死机现象。

解决办法：将 HALT/STOP 指令紧靠 LDI IE，0FH 指令之后。

e) 由于系统响应中断或进入子程序时，只有 PC& CY 被硬件压栈，所以对其它的一些重要参数，如累加器 A 的内容需要用户通过软件自行处理现场保护的工作；

3.4.6 PORT电平的变化产生中断应用实例

在中颖的单片机中，有两种中断可以让 PORT 电平的变化来产生中断：一个是外部中断（INT0 和 INT1），另一个是 PORT 中断。其中外部中断可以使用上升沿中断和下降沿中断，PORT 中断也可以，而且有些芯片的 PORT 中断还是可以单独控制的。

现在对这两种中断方式的应用进行详细说明。

- **PORT 中断**

中颖所有的单片机都带有 PORT 中断，只是因各个单片机的不同而有所不同，比如能作为 PORT 中断的 I/O 口有所不同，能使用上升沿中断或下降沿中断而有所不同，使用方法有所不同，等等。以下所要用来描述 PORT 中断的使用方法的单片机 SH67P54 的 PORT 中断有 PORTB 和 PORTC 总共 8 个 I/O 口可以用来做 PORT 中断，可以设置使用上升沿中断也可以设置使用下降沿中断，还有，SH67P54 的 PORT 中断的 I/O 口不是单独控制的。

当单片机进入 HALT 模式或是 STOP 模式，若是 PORT 中断被允许，当发生 PORT 中断时，单片机将在 HALT 模式或是 STOP 模式下被唤醒。

使用按键来唤醒 STOP 模式时，要注意一下系统时钟对按键扫描的影响。因为单片机进入 STOP

模式后，振荡器将停止工作；当单片机从 STOP 模式被唤醒后，振荡器开始工作，但是达到稳定需要一段时间，所以在单片机中就做了一个预热定时器来等待运行稳定后才开始执行指令。当使用低频振荡器时，预热定时器得到的等待时间会比较长，比如使用 32.768KHz 晶振时预热定时器计得的时间为 $2^{12}\times$（1/32768Hz）＝125ms，这样的结果是很可能在 STOP 模式唤醒后开始工作时已经扫描不到按键，因为等待的时间太长，若是按键已经松开，虽然 STOP 模式是被唤醒的，但是可能扫不到按键了。

使用 PORT 中断的方法是：

使用上升沿 PORT 中断

1. 将用于 PORT 中断的 PORT 口设置为输入口
2. 使能内部下拉电阻并往 PORT 口数据寄存器里边写 0 打开下拉电阻，若是没有内部下拉电阻，要在这些 PORT 口外部电路接下拉电阻以避免输入口悬空
3. 设置 PORT 中断为上升沿产生中断
4. 所有用于产生 PORT 中断的输入口状态都为低电平且 PORT 中断打开之后，当某一个用于产生 PORT 中断的输入口电平由低电平跳成高电平时，将会向系统发出 PORT 中断请求，产生 PORT 中断

使用下降沿 PORT 中断

1. 将用于 PORT 中断的 PORT 口设置为输入口
2. 使能内部上拉电阻并往 PORT 口数据寄存器里边写 1 打开上拉电阻，若是没有内部上拉电阻，要在这些 PORT 口外部电路接上拉电阻以避免输入口悬空
3. 设置 PORT 中断为下降沿产生中断
4. 所有用于产生 PORT 中断的输入口状态都为高电平且 PORT 中断打开之后，当某一个用于产生 PORT 中断的输入口电平由高电平跳成低电平时，将会向系统发出 PORT 中断请求，产生 PORT 中断

需要注意的是，由于口中断不是单独控制的，所以要使用上升沿中断时必须保证所有用于产生 PORT 中断的输入口在来中断信号前都为低电平，而要使用下降沿中断时必须保证所有用于产生 PORT 中断的输入口在来中断信号前都为高电平。

另外，还有些单片机的 PORT 中断会受输出口的影响，这些芯片要求使用上升沿中断时，在中断前要保证所有能用于 PORT 中断的 I/O 口（不管是输入口还是输出口中）都必须是低电平，而使用下降沿中断时，在中断前要保证所有能用于 PORT 中断的 I/O 口（不管是输入口还是输出口）都必须是高电平，而且输出口产生的电平变化也会导致 PORT 中断产生，使用时要注意一下这些单片机：SH66L08、SH66L10、SH66L12、SH66N12、SH66P14、SH66P13A、SH6613、SH66P14A、SH66P14、SH66P20A、SH6620A、SH66P22A 和 SH6622A。

下面举一个例子来说明一下 PORT 中断的使用：上电时 LED 无显示，之后每次按键按下则按下的按键所对应的 LED 亮起。

- 电路原理图

电路原理图如图 3-4-4 所示，使用 SH67P54 作为控制芯片，使用 32.768KHz 晶振作为低频主振荡器，PORTB 口作为输入口，接四个按键，PB0～PB3 分别接到 K1～K4 四个按键，使用内部上拉电阻。PORTD 口作为输出口，PD0～PD3 分别通过三极管进行电流放大来控制 L1～L4 四个 LED，对按键按下的状态进行显示。

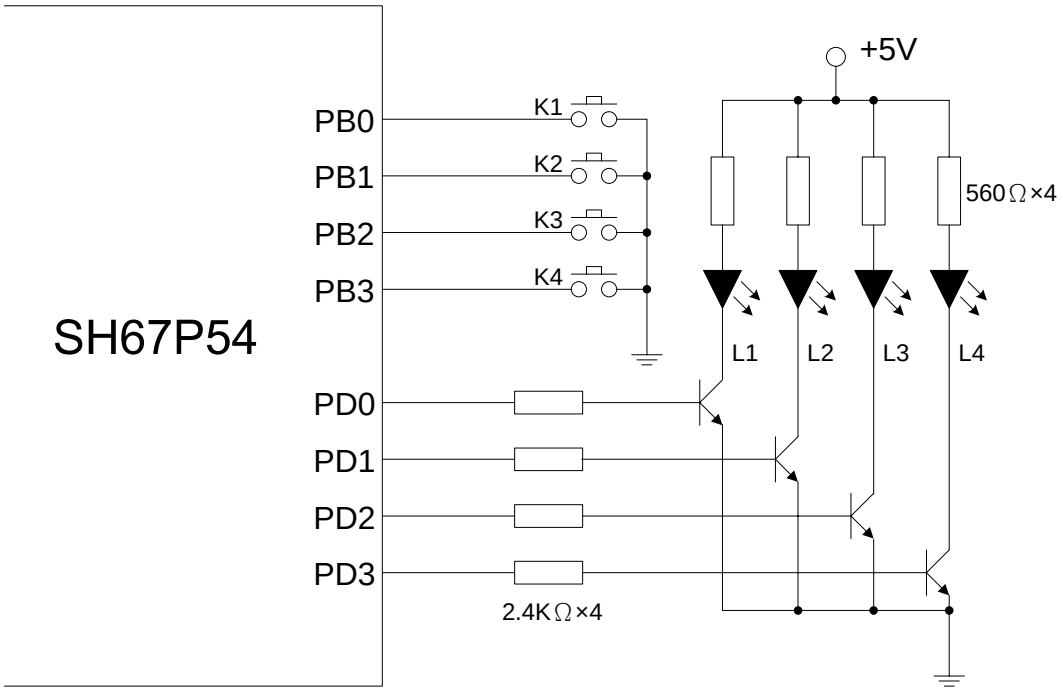


图 3-4-4　　SH67P54 的 PORT 中断电路原理图

- 程序设计

上电后设置好与 PORT 中断相关的各个寄存器后，进入 HALT 模式，等待按键按下时产生 PORT 中断将单片机从 HALT 模式唤醒，马上进行按键扫描（使用延时去抖动方法），扫描到按键后将对应的 LED 点亮。

例 3-4-1 PORT 电平的变化产生中断

```
LIST  P=67P54
ROMSIZE=4096
;***********************************************
; 系统寄存器 (Bank0)
;***********************************************
IE          EQU   00H         ;中断使能标志
IRQ         EQU   01H         ;中断请求标志
PORTB       EQU   09H         ;PORTB 数据寄存器
PORTC       EQU   0AH         ;PORTC 数据寄存器
PORTD       EQU   0BH         ;PORTD 数据寄存器
TBR         EQU   0EH         ;查表寄存器
INX         EQU   0FH         ;间接寻址伪索引寄存器
DPL         EQU   10H         ;INX 数据指针低四位
DPM         EQU   11H         ;INX 数据指针中三位
DPH         EQU   12H         ;INX 数据指针高三位
SETTING     EQU   13H         ;位 0: 外部中断(PA0)上升沿/下降沿设置
                              ;位 1: PB&PC 中断上升沿/下降沿设置
                              ;位 2: 端口上拉电阻/下拉电阻设置
                              ;位 3: 端口上拉电阻/下拉电阻使能控制位
PBCR        EQU   17H         ;PORTB 输入/输出控制寄存器
PDCR        EQU   19H         ;PORTD 输入/输出控制寄存器
```

```
;***********************************************
; 用户定义寄存器(Bank0)
;***********************************************
AC_BAK      EQU   20H                       ;AC 值备份寄存器
TMP         EQU   21H                       ;临时寄存器
KEY_ST      EQU   22H                       ;按键按下状态
;---------------------------------
CT1         EQU   23H                       ;记数值 1(用于延时)
CT0         EQU   24H                       ;记数值 0(用于延时)
;***********************************************
; 程序
;***********************************************
            ORG         0000H
            JMP         RESET
            RTNI
            RTNI
            RTNI
            JMP         PORT_ISP            ;PORT 中断服务程序入口地址
;*****************************************
; 子程序: PORT 中断服务程序
;*****************************************
PORT_ISP:
            STA         AC_BAK,00H          ;备份 AC 值
            LDI         IRQ,00H             ;清中断请求标志
PORT_ISP_END:
            LDA         AC_BAK,00H          ;取出 AC 值
            RTNI                            ;返回
;*****************************************
; 上电程序
;*****************************************
RESET:
            NOP
;--------------------------------------
; 清用户寄存器
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
            LDI         DPH,00H             ;从$20 开始
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TMP,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
POWER_RESET_3:
            SBI         DPH,02H             ;到$16F 结束
            BNZ         POWER_RESET_1
            SBI         DPM,07H
            BNZ         POWER_RESET_1
;--------------------------------------
; 初始化系统寄存器
SYSTEM_INITIAL:
            ;初始化 I/O 口
```

```
          LDI       PORTB,00H
          LDI       PBCR,00H            ;设置 PORTB 口作为输入口
          LDI       PORTD,00H
          LDI       PDCR,0FH            ;设置 PORTD 口作为输出口，输出低电平
;*********************************************
; 主程序
MAIN:
;*************************************
; 模块：PORT 中断设置
;*************************************
PORT_INT_SET:
          LDI       SETTING,1100B       ;端口上拉电阻使能，PORT 中断设置为下降沿中断
          LDI       PORTB,0FH           ;打开 PORTB 口上拉电阻
          LDI       PORTC,0FH           ;打开 PORTC 口上拉电阻
          NOP                           ;等待稳定
          NOP
          NOP
          LDI       IRQ,00H             ;清中断请求标志
          LDI       IE,0001B            ;打开 PORT 中断
          HALT                          ;进入 HALT 模式
          NOP                           ;单片机从 HALT 模式被 PORT 中断唤醒
          NOP
;*************************************
; 检测按键状态
KEY_CK:
          LDA       PORTB,00H           ;读 PORTB 口状态
          STA       TMP,00H             ;暂存于临时寄存器
          SBI       TMP,0FH
          BAZ       PORT_INT_SET        ;全为高电平，则为干扰，回到 HALT 模式
          CALL      DELAY               ;调用延时子程序延时 40ms 以去抖动
          LDA       PORTB,00H           ;再读 PORTB 口
          SUB       TMP,00H             ;比较前后两次电平状态
          BNZ       PORT_INT_SET        ;不相等则为按键抖动，回到 HALT 模式
                                        ;确认有按键按下
          EORIM     TMP,0FH                 ;将读到的值取反，得到正逻辑的键值
          STA       KEY_ST,00H          ;把键值送给按键状态寄存器
KEY_CK_END:
;---------------------------------
; 显示按键状态
DISP:
          LDA       KEY_ST,00H
          STA       PORTD,00H           ;将按键状态送到 PORTD 口，通过 LED 进行显示
DISP_END:
;*************************************
          JMP       MAIN                ;返回主程序
;*************************************
; 子程序：延时大约 40ms
;*************************************
DELAY:
          LDI       CT1,09H
          LDI       CT0,0EH
          SBIM      CT0,01H
          BC        $-1
          SBIM      CT1,01H
          BC        $-4
          RTNI
```

```
;*********************************************
        END
```

■ 可单独控制的 PORT 中断

中颖在新出的单片机中有些单片机的 PORT 中断是可以通过寄存器设置单独控制的，比如 SH69P48 这个单片机的寄存器中有 PDIEN、PDIF、PBIEN 和 PBIF 这四个寄存器可以用来控制 PORT 中断的使用。另外，这些单片机大部分还能通过寄存器单独控制内部上拉电阻使能。

其实这些单片机的 PORT 中断跟其它单片机的 PORT 中断基本上是一样的，不一样的只是用来做 PORT 中断的各个端口可以单独控制。

当单片机进入 HALT 模式或是 STOP 模式，若是 PORT 中断被允许，当发生 PORT 中断时，单片机将在 HALT 模式或是 STOP 模式下被唤醒。同样，使用按键通过 PORT 中断来唤醒 STOP 模式要注意系统时钟对按键扫描的影响。

使用单独控制的 PORT 中断的方法是：

使用上升沿 PORT 中断

1. 将用于 PORT 中断的 PORT 口设置为输入口
2. 使能内部下拉电阻并往 PORT 口数据寄存器里边写 0 打开下拉电阻，若是没有内部下拉电阻，要在这些 PORT 口外部电路接下拉电阻以避免输入口悬空
3. 设置 PORT 中断为上升沿产生中断
4. 将当前所要检测的 PORT 中断的单独控制 PORT 中断请求标志位（PxIF）清零，再打开当前所要检测的 PORT 中断的单独控制 PORT 中断使能标志（PxIEN 相应的位置写 1）
5. 清 PORT 中断请求标志，打开 PORT 中断
6. 不管其它的端口电平如何，当当前所要检测的 PORT 中断的输入口电平由低电平跳成高电平时，在 PxIF 的相应位置将自动置 1，将会向系统发出 PORT 中断请求，产生 PORT 中断

使用下降沿 PORT 中断

1. 将用于 PORT 中断的 PORT 口设置为输入口
2. 使能内部上拉电阻并往 PORT 口数据寄存器里边写 1 打开上拉电阻，若是没有内部上拉电阻，要在这些 PORT 口外部电路接上拉电阻以避免输入口悬空
3. 设置 PORT 中断为下降沿产生中断
4. 将当前所要检测的 PORT 中断的单独控制 PORT 中断请求标志位（PxIF）清零，再打开当前所要检测的 PORT 中断的单独控制 PORT 中断使能标志（PxIEN 相应的位置写 1）
5. 清 PORT 中断请求标志，打开 PORT 中断
6. 不管其它的端口电平如何，当当前所要检测的 PORT 中断的输入口电平由高电平跳成低电平时，在 PxIF 的相应位置将自动置 1，将会向系统发出 PORT 中断请求，产生 PORT 中断

PORT 中断能够单独控制的好处是：使用上升沿中断时，能作为 PORT 中断的输入口在来中断信号之前无需全部保持为低电平，只要单独控制 PORT 中断使用寄存器使能的相应的 I/O 口有上升沿信号，则相应的 PxIF 就会置 1，PORT 中断允许的话就会产生中断，而且中断后无需等待本次中断的输入口回到低电平，只要是单独控制 PORT 中断使能寄存器使能的相应的输入口再来一

个上升沿信号的话而 PORT 中断又被打开的话还能再产生中断；使用下降沿中断时，能作为 PORT 中断的输入口在来中断信号之前无需全部保持为高电平，只要单独控制 PORT 中断使用寄存器使能的相应的 I/O 口有上升沿信号，则相应的 PxIF 就会置 1，PORT 中断允许的话就会产生中断，而且中断后无需等待本次中断的输入口回到高电平，只要是单独控制 PORT 中断使能寄存器使能的相应的输入口再来一个下降沿信号的话而 PORT 中断又被打开的话还能再产生中断。

以 SH69P48 为例来说明一下这种类型的 PORT 中断的使用：上电时 LED 无显示，之后每次按键按下则当时为按下状态的按键所对应的 LED 亮起。

- 电路原理图

电路原理图如图 3-4-5 所示，使用 SH69P48 作为控制芯片，使用 4MHz 晶振作为主振荡器，PORTB 口作为输入口，接四个按键，PB0～PB3 分别接到 K1～K4 四个按键，使用内部上拉电阻。PORTD 口作为输出口，PD0～PD3 分别通过三极管进行电流放大来控制 L1～L4 四个 LED，对按键按下的状态进行显示。

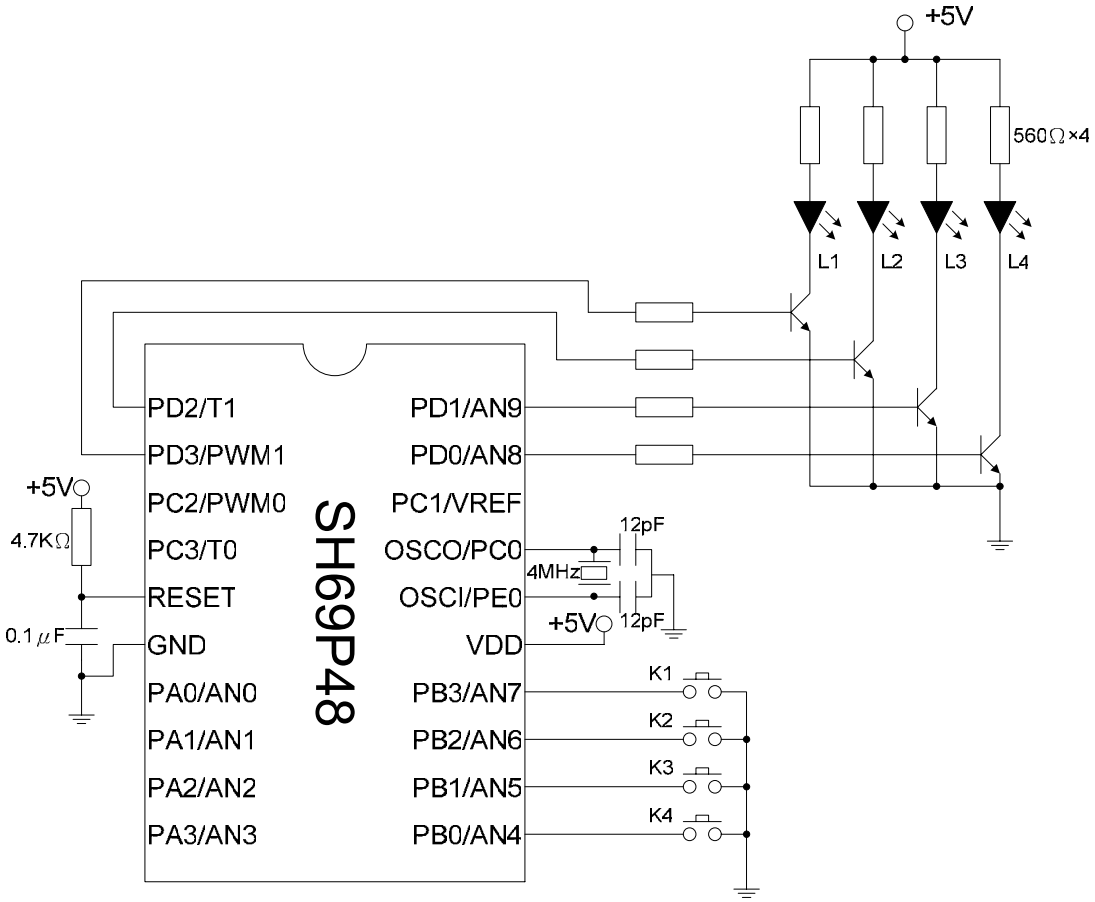


图 3-4-5　　SH69P48 的 PORT 中断电路原理图

- 程序设计

由于只使用到 PORTB 口作为 PORT 中断口，所以在 PBIEN 中写 0FH、而 PDIEN 中写 00H 就可以不管 PORTD 口的电平状态了。由于系统使用 4M 晶振作为主振荡器，预热定时器所产生的等待时间很短，大概为 1ms，所以进入 STOP 模式不会影响到按键扫描。上电后设置好要使用的 PORT 中断的相关的寄存器后，进入 STOP 模式，等待按键按下时产生的 PORT 中断将单片机从 STOP 模式唤醒，马上进行按键扫描（使用延时去抖动方法），并将扫描到的按键状态

送到 LED 进行显示。

例 3-4-2 可单独控制的 PORT 中断

```
LIST  P=69P48
ROMSIZE=4096
;************************************************
; 系统寄存器(Bank0)
;************************************************
IE          EQU   00H         ;中断使能标志
IRQ         EQU   01H         ;中断请求标志
PORTB       EQU   09H         ;PORTB 数据寄存器
PORTD       EQU   0BH         ;PORTD 数据寄存器
TBR         EQU   0EH         ;查表寄存器
INX         EQU   0FH         ;间接寻址伪索引寄存器
DPL         EQU   10H         ;INX 数据指针低四位
DPM         EQU   11H         ;INX 数据指针中三位
DPH         EQU   12H         ;INX 数据指针高三位
PBCR        EQU   19H         ;PORTB 输入/输出控制寄存器
PDCR        EQU   1BH         ;PORTD 输入/输出控制寄存器
;************************************************
; 系统寄存器(Bank7)
;************************************************
PDIEN       EQU   04H         ;PORTD 口中断使能标志
PDIF        EQU   05H         ;PORTD 口中断请求标志
PBIEN       EQU   06H         ;PORTB 口中断使能标志
PBIF        EQU   07H         ;PORTB 口中断请求标志
PPBCR       EQU   09H         ;PORTB 口上拉电阻控制寄存器
PPDCR       EQU   0BH         ;PORTD 口上拉电阻控制寄存器
;************************************************
; 用户定义寄存器(Bank0)
;************************************************
AC_BAK      EQU   30H         ;AC 值备份寄存器
TMP         EQU   31H         ;临时寄存器
KEY_ST      EQU   32H         ;按键按下状态
;--------------------------------
PARM        EQU   33H         ;延时时间长度寄存器
CT2         EQU   34H         ;计数值 2(用于延时)
CT1         EQU   35H         ;计数值 1(用于延时)
CT0         EQU   36H         ;计数值 0(用于延时)
;************************************************
; 程序
;************************************************
            ORG         0000H
            JMP         RESET
            RTNI
            RTNI
            RTNI
            JMP         PORT_ISP              ;PORT 口中断服务程序入口地址
;*****************************************
; 子程序: PORT 口中断服务程序
;*****************************************
PORT_ISP:
            STA         AC_BAK,00H            ;备份 AC 值
            LDI         IRQ,00H               ;清中断请求标志
```

```
PORT_ISP_END:
          LDA       AC_BAK,00H          ;取出 AC 值
          RTNI                          ;返回
;****************************************
; 上电程序
;****************************************
RESET:
          NOP
;--------------------------------------
; 清用户寄存器
POWER_RESET:
          LDI       DPL,00H
          LDI       DPM,03H
          LDI       DPH,00H             ;从$30 开始
POWER_RESET_1:
          LDI       INX,00H
          ADIM      DPL,01H
          LDI       TMP,00H
          ADCM      DPM,00H
          BA3       POWER_RESET_2
          JMP       POWER_RESET_3
POWER_RESET_2:
          ADIM      DPH,01H
POWER_RESET_3:
          SBI       DPH,01H             ;到$EF 结束
          BNZ       POWER_RESET_1
          SBI       DPM,07H
          BNZ       POWER_RESET_1
;--------------------------------------
; 初始化系统寄存器
SYSTEM_INITIAL:
          ;初始化 I/O 口
          LDI       PORTB,00H
          LDI       PBCR,00H            ;设置 PORTB 口作为输入口
          LDI       PORTD,00H
          LDI       PDCR,0FH            ;设置 PORTD 口作为输出口, 输出低电平
;****************************************************
; 主程序
MAIN:
;************************************
; 模块: PORT 口中断设置
;************************************
PORT_INT_SET:
          LDI       TBR,0FH
          STA       PPBCR,07H           ;PORTB 内部上拉电阻使能
          LDI       PORTB,0FH           ;打开 PORTB 内部上拉电阻
          NOP                           ;等待稳定
          NOP
          NOP
          LDI       TBR,00H
          STA       PBIF,07H            ;清 PORTB 中断请求标志
          LDI       TBR,0FH
          STA       PBIEN,07H           ;PORTB 中断使能
          LDI       IRQ,00H             ;清中断请求标志
          LDI       IE,0001B            ;打开 PORT 中断
          STOP                          ;进入 STOP 模式
```

```
        NOP                             ;单片机从 STOP 模式被 PORT 中断唤醒, PBIF 被写入了中断请求标志
        NOP
;************************************
; 检测按键状态
KEY_CK:
        LDA     PORTB,00H               ;读 PORTB 口状态
        STA     TMP,00H                 ;暂存于临时寄存器
        SBI     TMP,0FH
        BAZ     PORT_INT_SET            ;全为高电平, 则为干扰, 回到 STOP 模式
        ;将延时 6*6.2ms=37.2ms
        LDI     PARM,06H
        CALL    DELAY                   ;调用延时子程序以去抖动
        LDA     PORTB,00H               ;再读 PORTB 口
        SUB     TMP,00H                 ;比较前后两次电平状态
        BNZ     PORT_INT_SET            ;不相等则为按键抖动, 回到 STOP 模式
        ;确认有按键按下
        EORIM   TMP,0FH                 ;将读到的值取反, 得到正逻辑的键值
        STA     KEY_ST,00H              ;把键值送给按键状态寄存器
KEY_CK_END:
;--------------------------------
; 显示按键状态
DISP:
        LDA     KEY_ST,00H
        STA     PORTD,00H               ;将按键状态送到 PORTD 口, 通过 LED 进行显示
DISP_END:
;************************************
        JMP     MAIN                    ;返回主程序
;************************************
; 子程序: 延时大约(PARM*6.2)ms
;************************************
DELAY:
        SBIM    PARM,01H                ;判断 PARM>0
        BC      $+2
        RTNI                            ;返回主程序
        LDI     CT2,0AH
        LDI     CT1,0FH
        LDI     CT0,0FH
        SBIM    CT0,01H
        BC      $-1
        SBIM    CT1,01H
        BC      $-4
        SBIM    CT2,01H
        BC      $-7
        JMP     DELAY
;************************************************
        END
```

■ 外部中断（INT0）

中颖的单片机中有些芯片带有外部中断 INT0，比如 SH6612、SH6614 和 SH67P54 等等，甚至还有芯片带有 INT0 和 INT1 两个外部中断，比如 SH69K20A。INT0 中断使用 PA0 口，INT1 中断使用 PA3 口。INT1 的使用和 INT0 是一样的，下面以 INT0 中断来描述外部中断的使用。

当单片机进入 HALT 模式或是 STOP 模式，若是外部中断被允许，当发生外部中断时，单片机

将在 HALT 模式或是 STOP 模式下被唤醒。

当然，跟 PORT 中断一样，使用按键通过外部中断来唤醒 STOP 模式，仍然要注意系统时钟对按键扫描的影响，具体参考 PORT 中断的描述。

使用外部中断的设置跟 PORT 中断的设置差不多，只不过外部中断跟其它 I/O 没有关系，不用注意其它 I/O 口的状态。使用外部中断的方法是：

使用上升沿外部中断

1. 将外部中断的 PORT 口设置为输入口
2. 使能内部下拉电阻并往外部中断的 PORT 口数据寄存器里边写 0 打开下拉电阻，若是没有内部下拉电阻，要在外部中断口的外部电路接下拉电阻以避免输入口悬空
3. 设置外部中断为上升沿产生中断
4. 外部中断的输入口为低电平且外部中断打开之后，当外部中断的输入口电平由低电平跳成高电平时，将会向系统发出外部中断请求，产生外部中断

使用下降沿外部中断

1. 将外部中断的 PORT 口设置为输入口
2. 使能内部上拉电阻并往外部中断的 PORT 口数据寄存器里边写 1 打开上拉电阻，若是没有内部上拉电阻，要在外部中断口外部电路接上拉电阻以避免输入口悬空
3. 设置外部中断为下降沿产生中断
4. 外部中断的输入口为高电平且外部中断打开之后，当外部中断的输入口电平由高电平跳成低电平时，将会向系统发出外部中断请求，产生外部中断

跟 PORT 中断一样，这里举一个例子来说明一下外部中断的使用：上电时 LED 无显示，之后每次按键按下则按下的按键所对应的 LED 亮起。

- 电路原理图

电路原理图如图 3-4-6 所示，使用 SH67P54 作为控制芯片，使用 32.768KHz 晶振作为低频主振荡器，PORTB 口作为输入口，接四个按键，PB0～PB3 分别接到 K1～K4 四个按键，使用内部上拉电阻。PORTB 口通过连接两个二输入一输出的与门再接到一个二输入一输出的与门输出到外部中断 INT0 所使用的 PA0 口，当没有按键按下的时间，由于 PORTB 使用内部上拉电阻，所以 PORTB 四个 I/O 口的电平都为高电平，通过三个与门的逻辑电路后到达 PA0 的电平就是高电平，而当有按键按下时，PORTB 口有 I/O 为低电平，通过三个与门的逻辑电路到达 PA0 的电平就是低电平，所以通过这样的逻辑电路可以得到只要的按键按下的时候在 PA0 就会产生一个从高电平到低电平的下降沿，这样就可以使用下降沿的 INT0 中断来检测按键的按下。PORTD 口作为输出口，PD0～PD3 分别通过三极管进行电流放大来控制 L1～L4 四个 LED，对按键按下的状态进行显示。

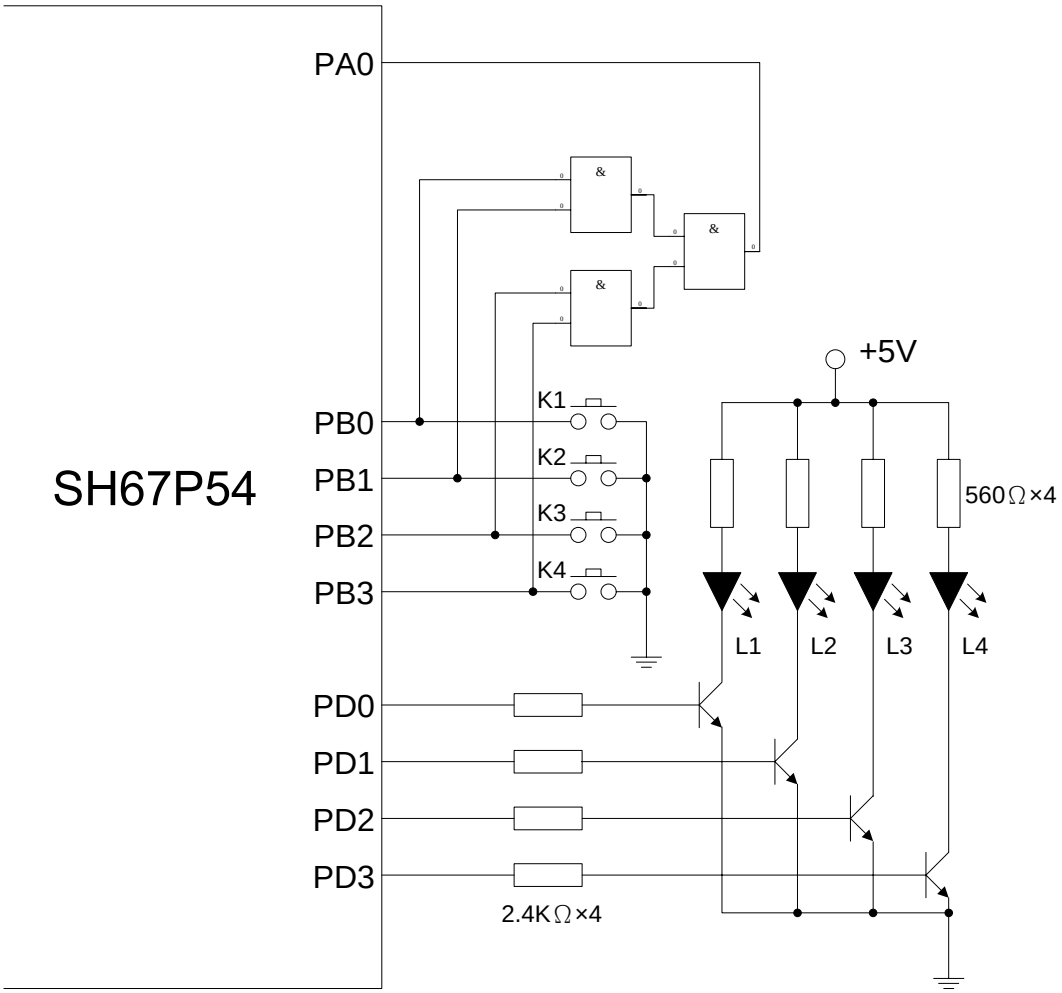


图 3-4-6　　SH67P54 的 INT0 中断电路原理图

- 程序设计

上电后设置好与 INT0 中断相关的各个寄存器后，进入 HALT 模式，等待按键按下时通过逻辑电路而产生的外部中断信号进入外部中断口，导致外部中断产生将单片机从 HALT 模式唤醒，马上进行按键扫描（使用延时去抖动方法），扫描到按键后将对应的 LED 点亮。

例 3-4-3 外部中断（INT0）

```
LIST  P=67P54
ROMSIZE=4096
;************************************************
; 系统寄存器 (Bank0)
;************************************************
IE          EQU   00H         ;中断使能标志
IRQ         EQU   01H         ;中断请求标志
PORTA       EQU   08H         ;PORTA 数据寄存器
PORTB       EQU   09H         ;PORTB 数据寄存器
PORTD       EQU   0BH         ;PORTD 数据寄存器
TBR         EQU   0EH         ;查表寄存器
INX         EQU   0FH         ;间接寻址伪索引寄存器
```

```
DPL         EQU   10H         ;INX 数据指针低四位
DPM         EQU   11H         ;INX 数据指针中三位
DPH         EQU   12H         ;INX 数据指针高三位
SETTING     EQU   13H         ;位 0: 外部中断(PA0)上升沿/下降沿设置
                              ;位 1: PB&PC 中断上升沿/下降沿设置
                              ;位 2: 端口上拉电阻/下拉电阻设置
                              ;位 3: 端口上拉电阻/下拉电阻使能控制位
PACR        EQU   16H         ;PORTA 输入/输出控制寄存器
PBCR        EQU   17H         ;PORTB 输入/输出控制寄存器
PDCR        EQU   19H         ;PORTD 输入/输出控制寄存器
;************************************************
; 用户定义寄存器(Bank0)
;************************************************
AC_BAK      EQU   20H         ;AC 值备份寄存器
TMP         EQU   21H         ;临时寄存器
KEY_ST      EQU   22H         ;按键按下状态
;--------------------------------
CT1         EQU   23H         ;记数值 1(用于延时)
CT0         EQU   24H         ;记数值 0(用于延时)
;************************************************
; 程序
;************************************************
            ORG         0000H
            JMP         RESET
            JMP         INT0_ISP                ;外部中断服务程序入口地址
            RTNI
            RTNI
            RTNI
;****************************************
; 子程序: 外部中断 INT0 服务程序
;****************************************
INT0_ISP:
            STA         AC_BAK,00H      ;备份 AC 值
            LDI         IRQ,00H                 ;清中断请求标志
INT0_ISP_END:
            LDA         AC_BAK,00H      ;取出 AC 值
            RTNI                                ;返回
;****************************************
; 上电程序
;****************************************
RESET:
            NOP
;--------------------------------------
; 清用户寄存器
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
            LDI         DPH,00H                 ;从$20 开始
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TMP,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
```

```
          ADIM      DPH,01H
POWER_RESET_3:
          SBI       DPH,02H             ;到$16F 结束
          BNZ       POWER_RESET_1
          SBI       DPM,07H
          BNZ       POWER_RESET_1
;-------------------------------------
; 初始化系统寄存器
SYSTEM_INITIAL:
          ;初始化 I/O 口
          LDI       PORTA,00H
          LDI       PACR,00H            ;设置 PORTA 口作为输入口
          LDI       PORTB,00H
          LDI       PBCR,00H            ;设置 PORTB 口作为输入口
          LDI       PORTD,00H
          LDI       PDCR,0FH            ;设置 PORTD 口作为输出口, 输出低电平
;*********************************************
; 主程序
MAIN:
;*************************************
; 模块: 外部中断 INT0 设置
;*************************************
INT0_SET:
          LDI       SETTING,1100B       ;端口上拉电阻使能, 外部中断为下降沿中断
          LDI       PORTB,0FH           ;打开 PORTB 上拉电阻
          NOP                           ;等待稳定
          NOP
          NOP
          LDI       IRQ,00H             ;清中断请求标志
          LDI       IE,1000B            ;打开外部中断
          HALT                          ;进入 HALT 模式
          NOP                           ;单片机从 HALT 模式被外部中断唤醒
          NOP
;*************************************
; 检测按键状态
KEY_CK:
          LDA       PORTB,00H       ;读 PORTB 口状态
          STA       TMP,00H             ;暂存于临时寄存器
          SBI       TMP,0FH
          BAZ       INT0_SET            ;全为高电平, 则为干扰, 回到 HALT 模式
          CALL      DELAY           ;调用延时子程序延时 40ms 以去抖动
          LDA       PORTB,00H       ;再读 PORTB 口
          SUB       TMP,00H             ;比较前后两次电平状态
          BNZ       INT0_SET            ;不相等则为按键抖动, 回到 HALT 模式
          ;确认有按键按下
          EORIM TMP,0FH             ;将读到的值取反, 得到正逻辑的键值
          STA       KEY_ST,00H      ;把键值送给按键状态寄存器
KEY_CK_END:
;---------------------------------
; 显示按键状态
DISP:
          LDA       KEY_ST,00H
          STA       PORTD,00H       ;将按键状态送到 PORTD 口, 通过 LED 进行显示
DISP_END:
;*************************************
          JMP       MAIN                ;返回主程序
```

```
;************************************
; 子程序: 延时大约 40ms
;************************************
DELAY:
          LDI       CT1,09H
          LDI       CT0,0EH
          SBIM      CT0,01H
          BC        $-1
          SBIM      CT1,01H
          BC        $-4
          RTNI
;************************************************
          END
```

3.5 定时器/计数器

与其它种类的单片机一样，SH6xxx 单片机也集成了若干定时器/计数器。定时器/计数器是单片机应用中使用最广泛的资源之一。

定时器从电路结构来讲，是由一组脉冲计数器组成，定时器和计数器是一个相对的概念。一般而言，如果脉冲计数器的时钟来源与系统内部的系统时钟或振荡器时钟，则称之为定时器，因为其时钟是固定的，可以用作时间基准。如果其时钟来源于系统外部的脉冲，则称之为计数器。

SH6xxx 产品中的定时器种类一般包括：

- 8-bit Genernal Timer0
- 8-bit Genernal Timer1
- BaseTimer
- WatchDog Timer
- Warmup Timer

下面就这些 Timer 做一些具体介绍。

3.5.1 Timer0和Timer1

Timer0/1 一般有以下特点(具体产品 Timer0 或 Timer1 的特点在此基础上可能有所增加或删减)：

- 8 位定时/计数器，一般 Timer1 只用作定时器；
- 数据计数器可读可写；
- 8 种预分频系数；
- 定时/计数器时钟来源可以选择内部时钟和外部脉冲；
- 计数值由$FF 到$00 时产生中断溢出(overflow)；
- 对于外部事件可选择边沿触发

以下为简化的定时器框图：

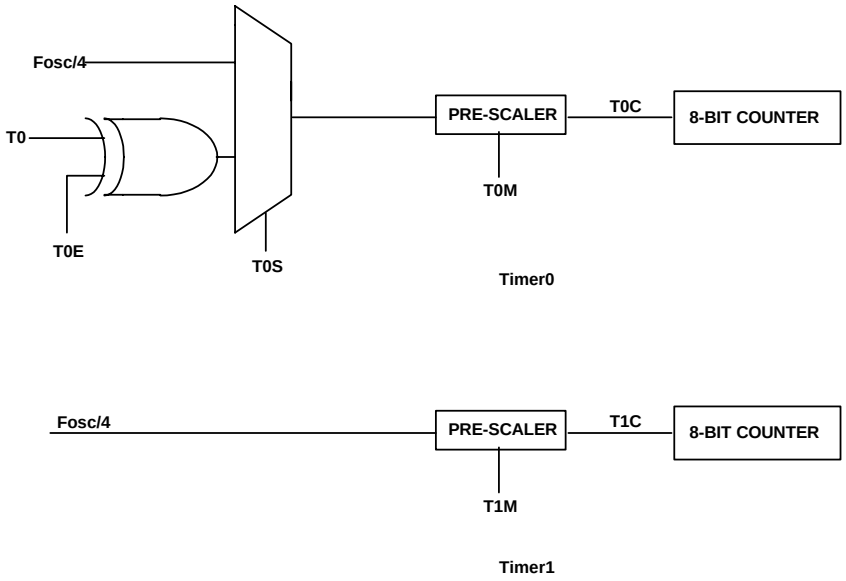


图 3-5-1 Timer0/1 结构框图

3.5.1.1 结构配置和操作

定时器0和定时器1分别由一个8位只写定时寄存器 (TL0L/TL1L, TL0H/TL1H) 和一个8位只读计数器 (TC0L/TC1L, TC0H/TC1H) 组成，每个寄存器又由低四位和高四位组成。

对计数器初始化时, 将数据写入定时寄存器(TL0L, TL0H) 中就可以了。 寄存器的编程方法: 先写入低四位数据再写入高四位数据。当计数器中写入高四位数据或者计数器从$FF到$00计数溢出时, 计数器将会自动装入定时寄存器的值。

如图 3-5-2，由于寄存器高四位寄存器 H 控制着实际的读和写操作信号。 所以在操作寄存器时请遵循以下原则:

写操作时，先写低四位,后写高四位以更新计数器;

读操作时，先读高四位,再读低四位。

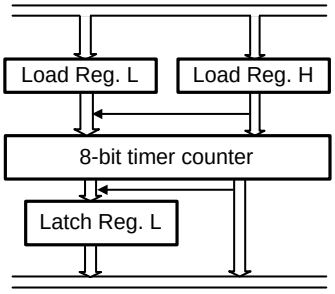


图 3-5-2 Timer0/1 操作示意图

3.5.1.2 定时器0或定时器1中断

当计数器由$FF计数到$00时会溢出,定时器溢出将会产生一个内部中断请求信号.如果中断使能标志为“1”,系统将处理定时器中断服务子程序。该功能也可用于在HALT模式下唤醒CPU。

3.5.1.3 定时器0工作模式寄存器

通过对定时器 0/1 工作模式寄存器 (TM0/TM1) 的设置，定时器可编程为多个不同分频比的分频器。由 8 位计数器对分频器输出的脉冲进行计数。定时器工作模式寄存器 (TM0/TM1) 是一个 3 位的寄存器，用于定时器的控制，如下表所示。这些工作模式寄存器将选择定时器的输入脉冲源。

定时器 0 工作模式寄存器

TM0.2	TM0.1	TM0.0	预分频器分频比	比例 N

0	0	0	$/2^{11}$	2048（初始值）
0	0	1	$/2^{9}$	512
0	1	0	$/2^{7}$	128
0	1	1	$/2^{5}$	32
1	0	0	$/2^{3}$	8
1	0	1	$/2^{2}$	4
1	1	0	$/2^{1}$	2
1	1	1	$/2^{0}$	1

3.5.1.4 外部时钟/事件T0作为TMR0的时钟源

当一个外部时钟/事件用作TM0的输入时，外部时钟源被CPU系统时钟同步. 因此，外部时钟源必须遵循以下原则. 由于TOC从TOM输出，并且TOC在系统时钟的每个指令周期里采样得到. 因此，对TOC来讲其高电平必须保持至少2倍 t_{osc}的时间而低电平也必须保持至少2倍 t_{osc}时间. 当选择分频器分频比为$/2^0$时，TOC与系统时钟的输入相同. 其条件如下:

TOH = TOCH = T0高电平时间 ≥ 2 t_{osc} + ΔT

TOL = TOCL = T0低电平时间 ≥ 2 t_{osc} + ΔT

Note: ΔT = 40ns

当选择其它的分频比时，TM0通过异步脉动计数器来标度，且预分频器的输出信号是对称的.

那么:

$$TOC\text{高电平时间} = TOC\text{低电平时间} = \frac{N*T0}{2}$$

其中

T0 = 定时器0输入周期

N = 预分频器的值

因此，满足条件是:

$$\frac{N*T0}{2} \geq 2\ t_{osc} + \Delta T,\ \text{or}\ T0 \geq \frac{4*t_{OSC} + 2\Delta T}{N}$$

上述条件仅限于T0周期用作定时器时. 该等式对脉宽没有限制. 概括如下:

$$T0 = \text{定时器 0 周期} \geq \frac{4 * t_{osc} + 2\Delta T}{N}$$

3.5.2 Base Timer(时基定时器)

时基定时器（BaseTimer）是在通用定时器基础上进行简化的一类 Timer。

BaseTimer 根据其设定的模式，为系统提供时基信号，用于系统的计时或给系统提供一个时间基准信号。

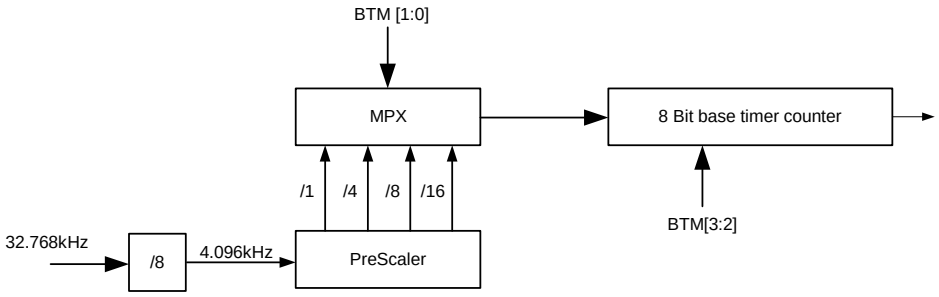


图 3-5-3 BaseTimer 结构图示意图

BaseTimer 模式寄存器：

BTM3	BTM2	BTM1	BTM0	分频比	定时时间(秒)	时钟源
1	0	0	0	/1	1/16	32.768kHz
1	0	0	1	/4	1/4	32.768kHz
1	0	1	0	/8	1/2	32.768kHz
1	0	1	1	/16	1	32.768kHz
0	0	X	X	BaseTimer 禁止使用		
0	1					
1	1					

上图和上表中，BaseTimer 的时钟源为 32.768kHz，其实这只是一个原则而已。BaseTimer 的时钟源每个产品可能有所不同，具体的时钟源请参考具体产品的数据手册。

系统复位开始运行，在 BTM[3:2]被设置为 10 后，BaseTimer 被打开，开始对内部时钟进行计数。当计数值达到$FF 时，在下一个时钟输入以后，计数器的值将等于 $00 并产生溢出,使 BaseTimer 中断请求标志置为 1。同时如果 BaseTimer 的中断允许位为 ON，则 CPU 会立即响应溢出中断。

在 BaseTimer 计数过程中，程序不能读写 BaseTimer 内部的计数值。

BaseTimer 定时时间=1 / (时钟源 / 8 /分频值)

3.5.3 WatchDog Timer (看门狗定时器)

参见 3.10 部分

3.5.4 Warmup Timer(预热定时器)

所有的单片机系统初次上电后，其振荡器首先开始起振，其振荡过程与振荡器的类型有关，一般而言，RC 类型的振荡器起振时间较短(几十到几百 ms)，而晶振或陶瓷振荡器的起振时间较长(几百毫秒到 1～2 秒)。

在 SH6xxx 系统中，晶振或陶瓷振荡器是采用负反馈结构，RC 振荡器是利用 RC 充放电的特性。

由于振荡器在刚起振到振荡稳定的过程中，振荡器产生的振荡波形很不稳定，振幅和频率变化极大。这样的波形如果直接用作系统的时钟，会导致系统运行不稳定，甚至出现“死机”。为了消除这一影响，在系统中就引入了 warmup timer(预热定时器)的概念。从振荡器开始起振开始，其产生的振荡时钟首先作为 warmup Timer 的输入，只有当 warmup timer 接收到一定数量的稳定的振荡时钟后，振荡器的振荡时钟才输出到系统中，作为系统时钟，系统开始运行。这样就消除了振荡器起振时的不稳定状态，保证系统正常可靠的运行。

一般而言，warmup timer 的启动发生在以下情况：

- 初始上电阶段(POR,Power on reset)；
- 外部有 Reset 信号输入时；
- 系统内部产生了硬件 Reset 信号时；
- 系统从省电模式返回时(系统进入省电模式，在该模式下，振荡器为关闭状态)，如从 STOP 模式返回时；

由于每个产品所采用的振荡器类型不同，使用条件不同，warmup timer 的定时时间(即时钟的计数溢出值)均有所差异，具体的数据在每个产品的数据手册的 warmup timer 部分均有详细列出。

3.6 LCD Driver(液晶驱动器)

在单片机的应用中，人机界面占据相当重要的地位。人机界面主要包括事件输入和结果指示，事件输入包括键盘输入，通讯接口，事件中断等，结果指示包括 LED/LCD 显示、通讯接口、外围设备操作等。而在这些人机界面当中，LCD 显示技术由于其具有界面友好，成本较低等特点而在很多应用场合得以广泛应用。

我们在第一章 SH6xxx 单片机分类中就介绍过，LCD 类单片机是 SH6xxx 单片机产品线的一个重要类别。

3.6.1 LCD的显示原理

在讲解 LCD driver 之前，我们先就 LCD 的显示原理作一简单的介绍。

LCD(Liquid Crystal Display)是利用液晶分子的物理结构和光学特性进行显示的一种技术。液晶分子的特性：

- 液晶分子是介于固体和液体之间的一种棒状结构的大分子物质；
- 在自然形态，具有光学各向异性的特点，在电(磁)场作用下，呈各向同性特点；

下面以直视型简单多路 TN/STN LCD Panel(液晶显示面板)的基本结构介绍 LCD 的基本显示原理，示意图如图 3-6-1：

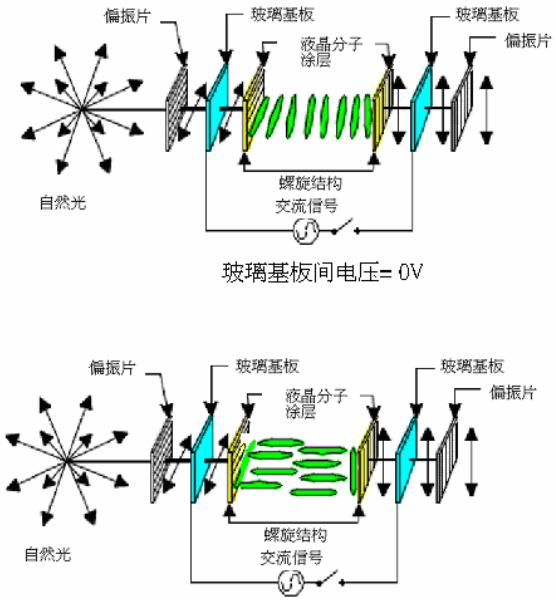


图 3-6-1 LCD 的基本显示原理

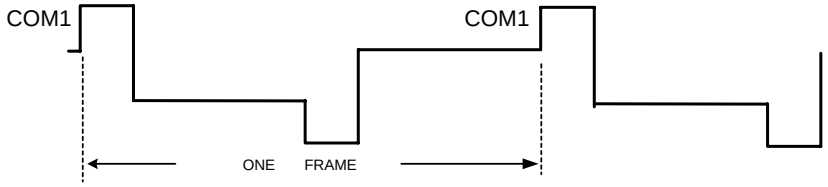
整个 LCD Panel 由上下玻璃基板和偏振片组成，在上下玻璃之间，按照螺旋结构将液晶分子有规律的进行涂层。液晶面板的电极是通过一种 ITO 的金属化合物蚀刻在上下玻璃基板上。如图所示，液晶分子的排列为螺旋结构，对光线具有旋旋光性，上下偏振片的偏振角度相互垂直。在上下基板间的电压为 0 时，自然光通过偏振片后，只有与偏振片方向相同的光线得以进入液晶分子的螺旋结构的涂层中，由于螺旋结构的的旋旋光性，将入射光线的方向旋转 90 度后照射到另一端的偏振片上，由于上下偏振片的偏振角度相互垂直，这样入射光线通过另一端的偏振片完全的射出，光线完全进入观察者的眼中，看到的效果就为白色。而在上下基板间的电压为一交流电压时，液晶分子的螺旋结构在电(磁)场的作用下，变成了同向排列结构，对光线的方向没有作任何旋转，而上下偏振片的偏振角度相互垂直，这样入射光线就无法通过另一端的偏振片射出，光线无法进入观察者的眼中，看到的效果就为黑色。这样通过在上下玻璃基板电极间施加不同的交流电压，即可实现液晶显示的两种基本状态亮(On)和暗(Off)。

在实际的液晶模以驱动电压中，有几个参数非常关键：

- 交流电压，液晶分子是需要交流信号来驱动的，长时间的直流电压加在液晶分子两端，会影响液晶分子的电气化学特性，引起显示模糊，寿命的减少，其破

坏性为不可恢复；

- 扫描频率，直接驱动液晶分子的交流电压的频率一般在 60～100Hz 之间，具体是依据 LCD Panel 的面积和设计而定，频率过高，会导致驱动功耗的增加，频率过低，会导致显示闪烁，同时如果扫描频率同光源的频率之间有倍数关系，则显示也会有闪烁现象出现。



3-6-2 帧频(Frame)示意图

- 液晶分子是一种电压积分型材料，它的扭曲程度(透光性)仅仅和极板间电压的有效值有关，和充电波形无关。电压的有效值用 COM/SEG 之间的电压差值的均方根 VRMS 表示：

$$V(RMS)=\sqrt{\frac{1}{T}\int_0^T[V(t)]^2\,dt}$$

LCD 显示黑白(透光和不透光)的电压有效值的分界电压称为开启电压 Vth，当电压有效值超过 Vth，螺旋结构的旋光角度加大，透光率急剧变化，透明度急剧上升。反之，则透明度急剧下降。光线的透射率与交流电压的有效值的关系如图 3-6-3：

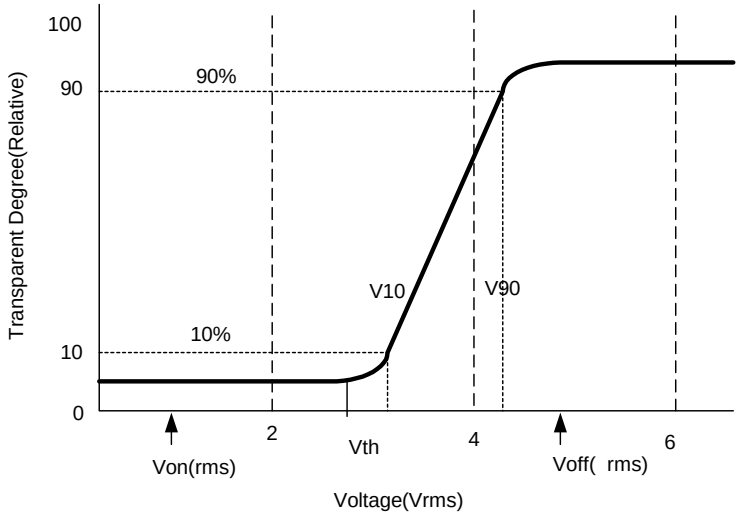


图 3-6-3 光线的透射率与交流电压的有效值的关系图

LCD 类单片机内嵌的 LCD driver(液晶驱动器)，正是通过系统的控制，按照用户定义的显示图案，产生点亮 LCD(Liquid Crystal Display，液晶)所需的模拟驱动波形，接到 LCD Panel(液晶显示屏)上点亮对应的像素而达到显示的效果。

- 占空比(Duty)

该项参数一般也称为 Duty 数或 COM 数。由于 STN/TN 的 LCD 一般是采用时分动态扫

描的驱动模式，在此模式下，每个 COM 的有效选通时间与整个扫描周期的比值即占空比(Duty)是固定的，等于 1/COM 数。

- 偏置(Bias)

LCD 的 SEG/COM 的驱动波形为模拟信号，而各档模拟电压相对于 LCD 输出的最高电压的比例称为偏置，而一般来讲，Bias 是以最低一档与输出最高电压的比值来表示，如图 3-6-4 所示(1/4 Duty,1/3 Bias)：

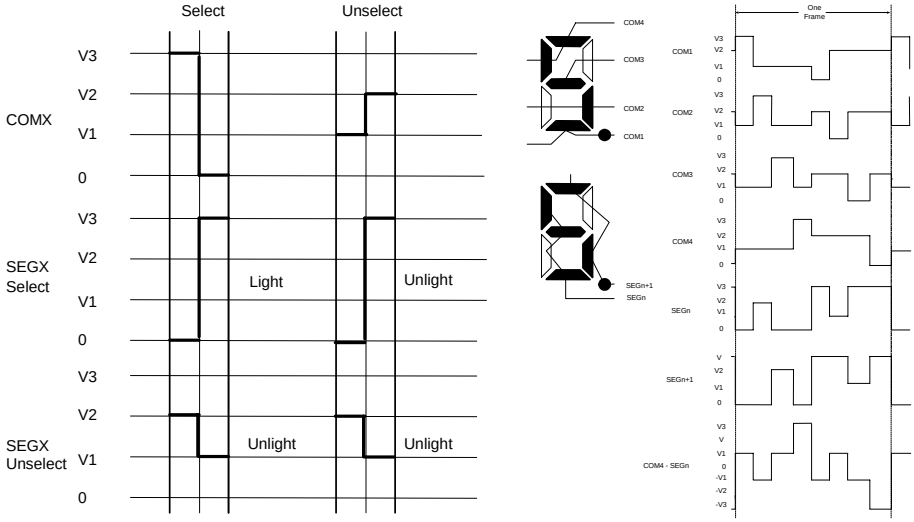


图 3-6-4 LCD driver 驱动波形图

该图对应的是 1/4 duty，1/3 bias 的液晶驱动波形，COM 数为 4，每个 COM 的有效选通时间与整个扫描周期的比值(Duty)=1/4，驱动波形的模拟电压共分 3 档，V3 位输出最高电压，V2，V1 为输出中间电压，并且 V1/V3=1/3，所以上述波形图对应的 Duty=1/4，Bias=1/3。

一般而言，Bias 和 Duty 之间是有一定关系的，duty 数越多，每根 COM 对应的扫描时间变短，而要达到同样的显示亮度和显示对比度，VON 的电压就要提高，选电平和非选电平的差异需要加大，即 Bias 需要加大，Duty 和 Bias 间有一经验公式，即

$$\text{Bias} = 1/(\sqrt{\text{Duty}} + 1)$$

3.6.2 LCD驱动器的电源

液晶驱动波形为由若干档直流电平组合而成的模拟波形，各档直流电平的比例关系反映驱动波形的 Bias 比例关系，各档电平的具体幅值取决于 LCD Panel 的液晶特性和 Duty 数的多少。图 3-6-5 为一 LCD 驱动电源部分的示意图：

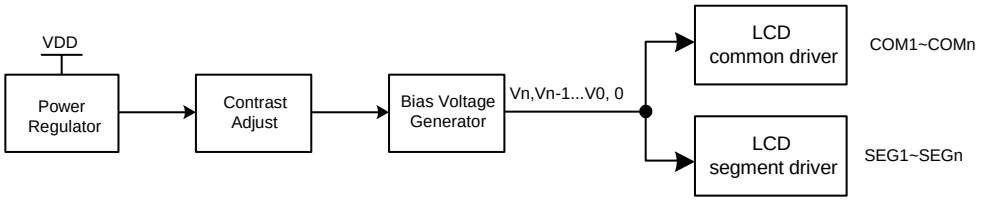


图 3-6-5 LCD 驱动电源部分的示意图

电源调整器部分(Power Regulator)：

产生 LCD 驱动所需的最高直流电平，一般分为三种：

a) LCD 驱动所需的最高直流电平等于外部输入电源 VDD 的，此部分就直接将 VDD 输入至后续电路；

b) LCD 驱动所需的最高直流电平大于外部输入电源 VDD，且不需要稳压输出的，如固定等于 1.5VDD 或 2.0VDD，此部分通常做法是将外部输入电源 VDD 通过升压电路(pump)升至所需的电压，输入至后续电路；

c) LCD 驱动所需的最高直流电平大于外部输入电源 VDD，且需要稳压输出的，即驱动所需的最高直流电平不随 VDD 的变化而变化的，如要求 VDD =2.4～5.5V 全电压范围里，VLCD 的输出电压都保持不变，此部分通常做法是首先产生一个误差范围符合要求的电压基准源，然后将此电压基准源比例放大至所需的电压，同时外部输入电源 VDD 通过升压电路(pump)升至一定的电压，如 2VDD，作为比例放大部分的电源。如图 3-6-6：

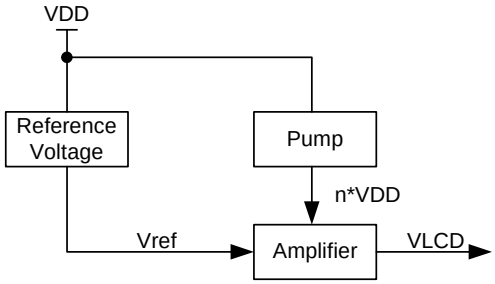


图 3-6-6 电源调整器部分结构示意图

对比度/亮度调整部分(Contrast Adjustment)：

通过对比度/亮度控制寄存器，调节输出的 LCD 驱动电压。

设置此部分的目的有三个：

a) 同一颗单片机适配的 LCD Panel 的选择余地较大，LCD panel 的工作电压(额定电压)处于 LCD 驱动器输出的最高电压和最低电压之间即可；

b) 可以有效的消除 LCD Panel 在制作过程中工作电压的偏移，特别是 TN/STN 等对成本要求较严格的 LCD Panel，其最佳工作电压与设计工作电压间的偏移较大；

c) 有些产品的 LCD driver 无电源调整电路，其 LCD 输出的最高电压(VLCD)与外部输入电源跟随变化。在实际产品中，特别是使用电池作为电源的应用场合，外部输入电源随着使用时间的加长会慢慢降低，LCD 输出的电压和 LCD Panel 的对比

度也会随之降低，这时保持 LCD Panel 的对比度不变，就可以通过调节对比度/亮度控制寄存器进行调节。

此部分依据每颗单片机产品的定位不同，有所差异，有些产品无部分，包含此部分的在调节档数或调节精度上也有所差异。

偏置电压产生部分(Bias Voltage Generator)：

LCD driver 输出的最高电压通过偏置电压产生电路，根据选择的偏置设置，产生 LCD 交流驱动波形所需要的其它几档偏置电压(VLCD,Vn,Vn-1,…V1,V0)，提供给后续的 COM/SEG 波形产生电路。

此部分的实现方式一般分为两种：

a) 电阻分压结构，即依据 Bias 的设置，选择合适的分压电阻，产生需要的直流分压电平，如图 3-6-7；

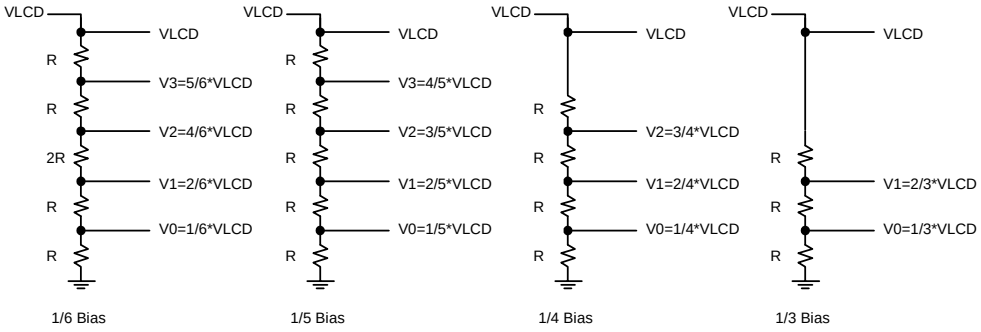


图 3-6-7 电阻分压结构的偏压电路示意图

b) 电容结构，这时一种较为特殊的 LCD driver 的电源结构，在这种结构下，电压调整部分和电压偏置部分是整合在一起的，电源升压部分是直接按照 Bias 的设置产生 LCD driver 需要的直流分压电平，如图中，VP2 是 2 倍的 VDD，VP1 是 3 倍的 VDD。

在此结构下，如图 3-6-8 所示的外接电容一般情况是必须要的，否则仅仅依靠芯片内的电容，其驱动能力较差。

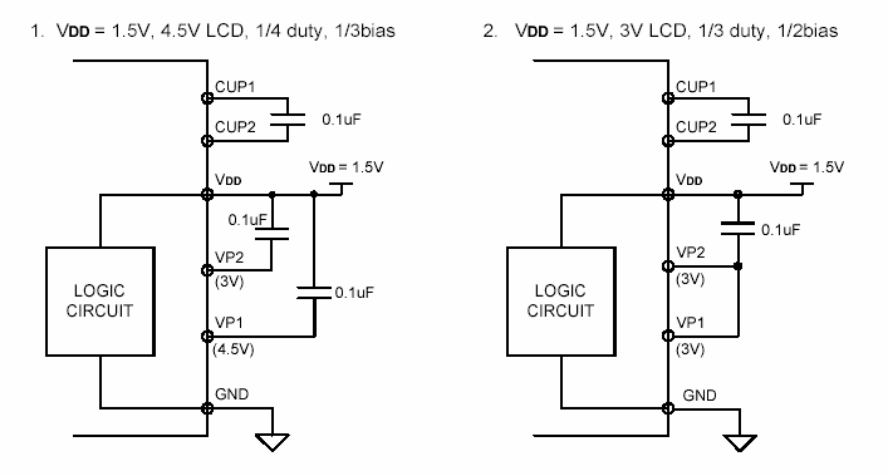


图 3-6-8 电容分压结构的偏压电路示意图

COM/SEG 驱动波形产生部分(COM/SEG driver)：

此部分的结构示意如图 3-6-9：

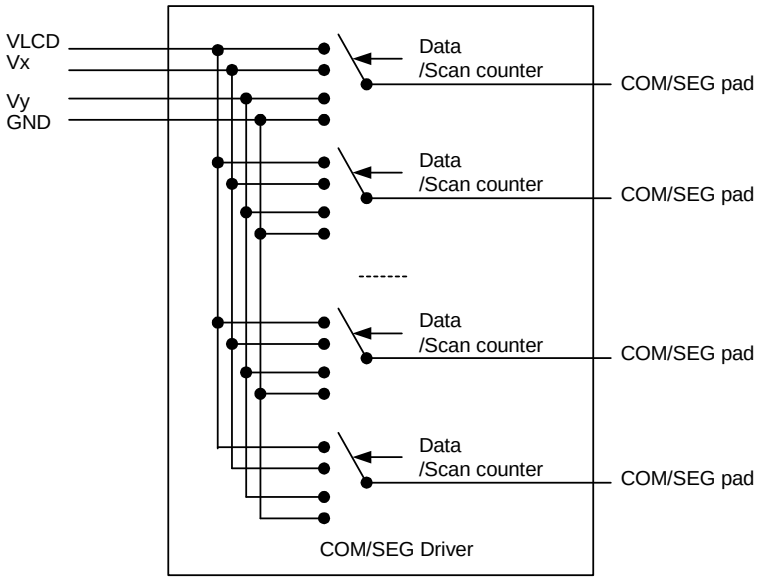


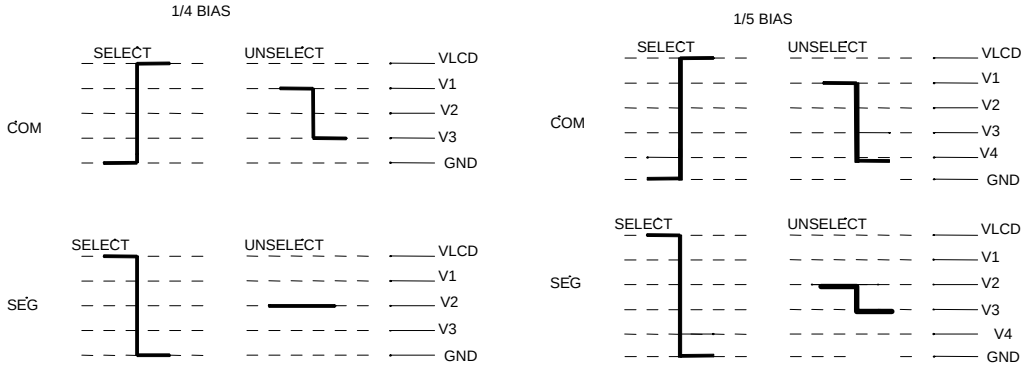
图 3-6-9 COM/SEG 驱动波形产生示意图

COM/SEG driver 可以看作一组多路选择开关，COM driver 依据扫描计数器的值，SEG driver 依据显示数据 RAM 对应的值，从输入的直流分压电平中进行选择并从相应的 COM/SEG 引脚加以输出。这样从整个 LCD 扫描周期来讲，从 COM/SEG 引脚上就输出了驱动 LCD Panel 所需要的模拟电压波形。

直流分压电平的选择关系如表 3-6-1 及图 3-6-10：

项目			1/3Bias	1/4Bias	1/5Bias	1/7Bias
直流分压电平			VLCD V1=2/3VLCD V2=1/3VLCD GND	VLCD V1=3/4VLCD V2=2/4VLCD V3=1/4VLCD GND	VLCD V1=4/5VLCD V2=3/5VLCD V3=2/5VLCD V4=1/5VLCD GND	VLCD V1=6/7VLCD V2=5/7VLCD V3=2/7VLCD V4=1/7VLCD GND
前半扫描周期	COM driver	选电平	VLCD	VLCD	VLCD	VLCD
		非选电平	V2	V3	V4	V4
	SEG driver	选电平	GND	GND	GND	GND
		非选电平	V1	V2	V2	V2
后半扫描周期	COM driver	选电平	GND	GND	GND	GND
		非选电平	V1	V1	V1	V1
	SEG driver	选电平	VLCD	VLCD	VLCD	VLCD
		非选电平	V2	V2	V3	V3

表 3-6-1 直流分压电平的选择关系表



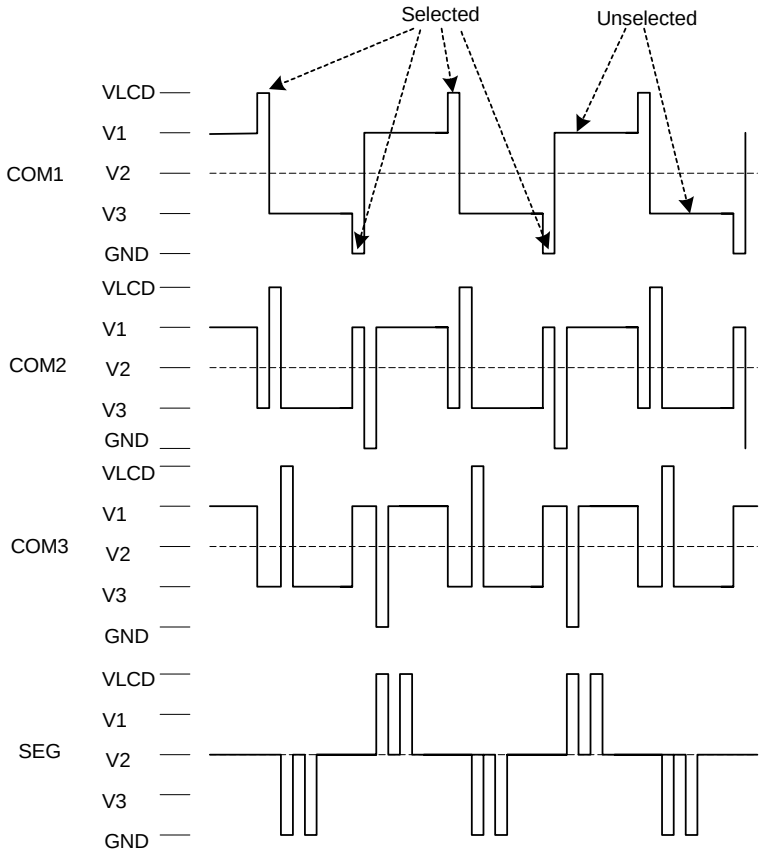


图 3-6-10 SEG/COM 波形图

3.6.3 LCD显示RAM(LCD Display RAM)

用户对 LCD Driver 的操作一方面是通过操作 LCD driver 的控制寄存器，来设置 LCD driver 的工作模式(包括 Duty/Bias/Contrast/扫描频率/LCD 开关等的设置)，另一方面 LCD Panel 上显示所需的内容是通过读写 LCD 显示 RAM 来实现。LCD RAM 的结构不同于其它 Data RAM，它是一个双口 RAM(Dual Port)的结构，一边为 CPU 的读写接口，另外一边是与 LCD driver 的读接口。

LCD RAM 的字节排列顺序是与 LCD 输出的 COM/SEG 阵列相对应的。

如表 3-6-2：

Address	Bit3	Bit2	Bit1	Bit0
	COM4	COM3	COM2	COM1
$300	SEG1	SEG1	SEG1	SEG1
$301	SEG2	SEG2	SEG2	SEG2
$302	SEG3	SEG3	SEG3	SEG3
$303	SEG4	SEG4	SEG4	SEG4
$304	SEG5	SEG5	SEG5	SEG5
$305	SEG6	SEG6	SEG6	SEG6
$306	SEG7	SEG7	SEG7	SEG7
$307	SEG8	SEG8	SEG8	SEG8
$308	SEG9	SEG9	SEG9	SEG9
$309	SEG10	SEG10	SEG10	SEG10
$30A	SEG11	SEG11	SEG11	SEG11
$30B	SEG12	SEG12	SEG12	SEG12
$30C	SEG13	SEG13	SEG13	SEG13
$30D	SEG14	SEG14	SEG14	SEG14
$30E	SEG15	SEG15	SEG15	SEG15
…	…	…	…	…

表 3-6-2 LCD RAM 分配表

举例，如果要将 COM2 与 SEG3 交点处的点点亮，只需将 LCD RAM $302 的 Bit1 置 1 即可，其余的工作由 LCD driver 的硬体自动完成。

3.6.4 LCD COM/SEG的复用功能

SH6xxx 产品线中，有些产品的 LCD COM/SEG 正常情况下用作 LCD driver 的 COM/SEG 信号的输出线，但依据实际应用情况，如 LCD 扫描线有空余，输入输出端口有有所不够，在这个时候，就可以讲这些扫瞄线设置为输入输出端口，当作通用的输入输出端口来使

用，当然有些可能设置为输出端口，具体的功能和设置请参考相应的产品说明书。

例如：在 SH69P54 中，SEG1～8 的第二功能为输入输出端口 PORTC 和 PORTD，

SH67K(P)93 中，SEG36～50 的第二功能为 PORTF，PORTE，PORTD 和 PORTC，SEG24～35 的第二功能为输出端口。

LCD COM/SEG 复用为输入输出端口(I/O)，输出端口(Output)

LCD 扫描与键盘扫描复用

LCD COM/SEG 的第二功能设置为输入输出端口时，其可以作为通用的端口使用，设置为输出端口时，其仅可以作为输出端口使用。在后一种情况下，LCD 与键盘扫描扫描线复用是一种最常见的应用，下面就这一应用进行讨论。

下图为 LCD 扫描与键盘扫描复用时的示意图(以 3x3 键盘为例)：

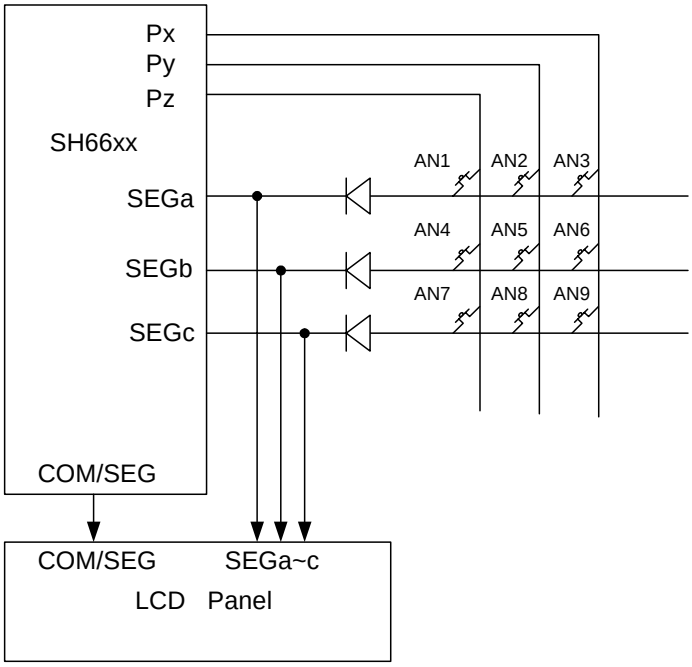


图 3-6-11 LCD 扫描与键盘扫描复用时的示意图

如图 3-6-11 中，SEGa～c 的第二功能为输出扫描口，同时接到键盘矩阵和 LCD Panel 上，Px，Py，Pz 为三根输入输出端口，用作键盘扫瞄输入线，其内部上拉电阻打开，或外接上拉电阻，图中二极管的作用为防止双键或多键按下时，按键对 LCD 显示产生影响。SEG/COM 用作输出口时，对应的数据位一般为一独立的数据寄存器。如 SH67P90 中 SEG24～35：

Addr	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$15	-	O/S2	O/S1	O/S0	R/W	LCD control register1
$3C8	SCAN35	SCAN34	SCAN33	SCAN32	R/W	Data Register of LCD SEG35 - 32 when SEG35 - 32 shared as output port.

$3C9	SCAN31	SCAN30	SCAN29	SCAN28	R/W	Data Register of LCD SEG31 - 28 when SEG20 - 17 shared as output port.
$3CA	SCAN27	SCAN26	SCAN25	SCAN24	R/W	Data Register of LCD SEG27 - 24 when SEG16 - 13 shared as output port.

$15 为 SEG24～35 第一/第二功能选择控制寄存器，$3C8～3CA 为 SEG24～35 用作输出端口时的数据寄存器。

程序操作流程如图 3-6-12：

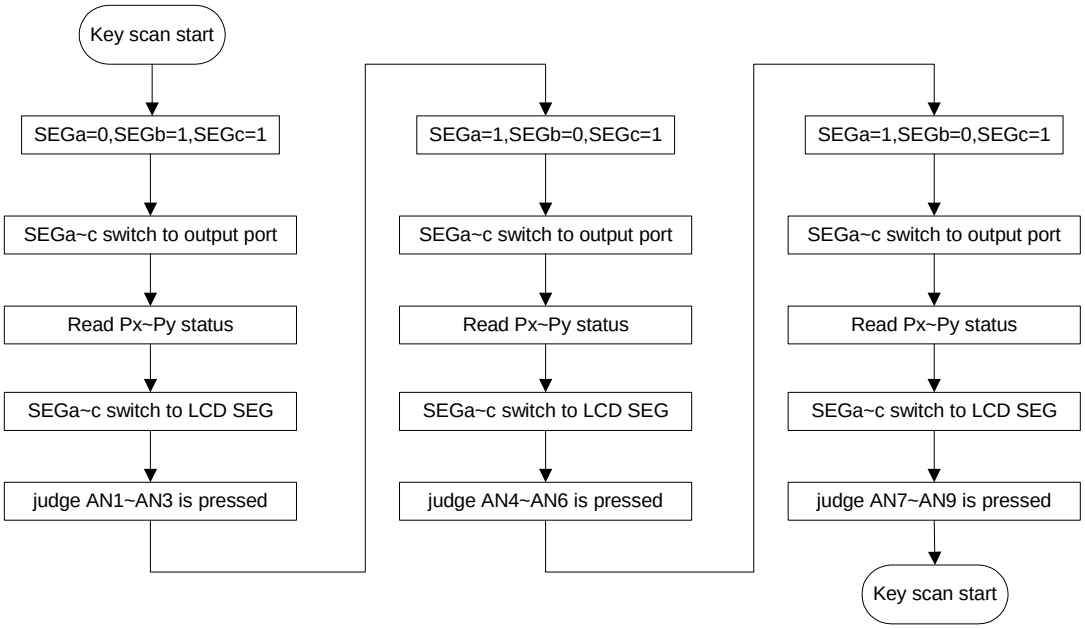


图 3-6-12 LCD 扫描与键盘扫描复用程序流程图

在上述流程中有几点非常重要：

- COM/SEG 用作键盘扫描线的时间越快越好，这样才能最大程度的避免键盘扫瞄对 LCD 显示带来的影响。
    1) 切换到键盘扫描线之前，先将扫描输出端口数据寄存器数据设置好，设置好后再行切换；
    2) 键盘扫描线输出后，立即读取键盘输入端口的状态，并将 COM/SEG 切换回 LCD COM/SEG 输出，再行处理键盘扫瞄数据；
    3) 键盘扫瞄的频率不宜太快，否则即使每次扫描的时间很短，但同一时间内扫描次数太多，结果是对 LCD 显示仍然产生较大的影响，同键盘扫瞄频率较慢，但每次扫描时间较长的产生的影响一样。一般键盘扫瞄的频率限制在 2～3Hz 以下，由于存在按键数量差异，液晶扫描频率不同，液晶面板的大小/电压/段码差异，此值依据具体的案例会有所不同。
- 上述流程只是提供一种思路，并未涉及如消抖等细节问题，具体应用中仅供参考。

3.6.5 LCD应用实例

在使用带 LCD 驱动模块的中颖单片机来驱动 LCD 的时候,在电路接口设计上是非常简单的。

首先要先查看所要驱动的 LCD 共有多少个 COMMON 口和 SEGMENT 口，根据 COMMON 口和 SEGMENT 口的数目来挑选所要匹配的单片机上的 LCD 驱动口。由于有些单片机的 LCD 驱动口是与 I/O 口复用的,所以在考虑这一点的时候还要考虑一下 I/O 口是否够用。比如作用 SH66P51 来驱动一片有 4 个 COMMON 口、12 个 SEGMENT 口的 LCD，且 I/O 口的使用上使用到了 PORTC 和 PORTD，这样先根据 LCD 有 4 个 COMMON 口，所以 SEG29/COM4 的引脚必须选择作为 COM4 使用，而 SEG27/COM6 和 SEG28/COM5 就可以选择为 SEG27 和 SEG28 使用（由于这块 LCD 只有 12 个 SEGMENT 口，所以最终单片机的 SEG27 和 SEG28 也是没用到)。单片机由于被使用到了 PORTC 和 PORTD 的 I/O 口，所以 SEG1～SEG8 就不能再使用了，LCD 上的 SEGMENT1～SEGMENT12 就只好依次接到单片机的 SEG9～SEG20，剩下的 SEG21～SEG28 不使用，可以悬空。

■ LCD 显示软件模块

现在以 SH66P51 为例讲述一下一个 4 个 COMMON 口，8 个 SEGMENT 口的显示时间的 LCD 的驱动。

LCD 平面显示图如图 3-6-13 所示：

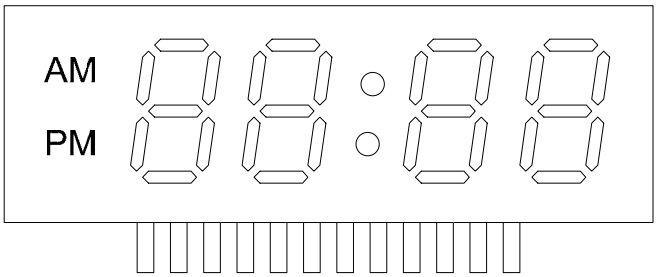


图 3-6-13 LCM 平面显示图

这块 LCD 共有 12 个引脚，包括 4 个 COMMON 口和 8 个 SEGMENT 口，显示 12 小时制的时钟和分钟。

这里，我们可以看一下这块 LCD 的 SEGMENT 图与 COMMON 图：

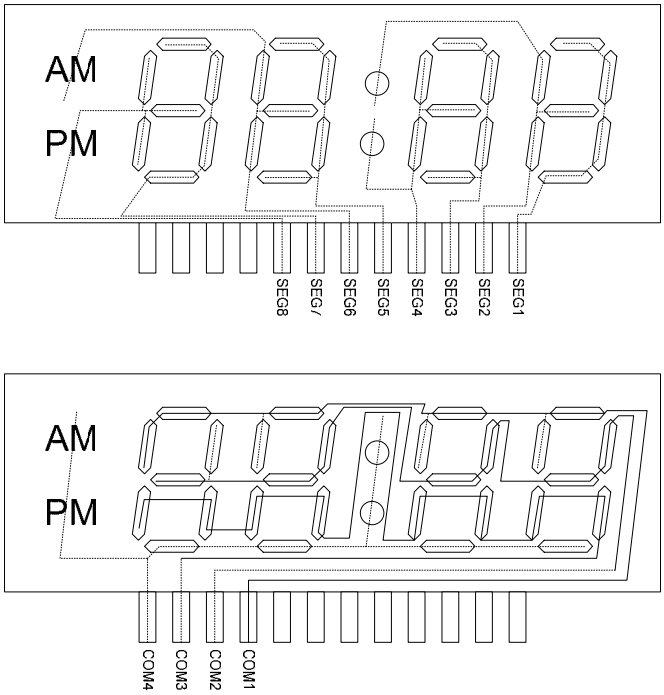


图 3-6-14 LCD SEG/COM 走线图

■ 电路原理图

由于系统设计中并没有要求要使用到 PORTC 和 PORTD 的 I/O 口，所以将 PORTC 和 PORTD 与 SEG1～8 这八个复用引脚设置为 SEG1～8 来使用。LCD 有 4 个 COMMON 口，SEG29/COM4 设置为 COM4 使用。

SH66P51 内部有自带的 $\overline{\text{RESET}}$ 引脚上拉电阻,所以在搭建电路时可以省去外部 RESET 电路中的上拉电阻，只需在烧写 IC 的时候打开 $\overline{\text{RESET}}$ 引脚上拉电阻即可。使用 32.768KHz 晶振作为主振荡器，LCD 驱动的时锺来源于晶振的振荡时锺。

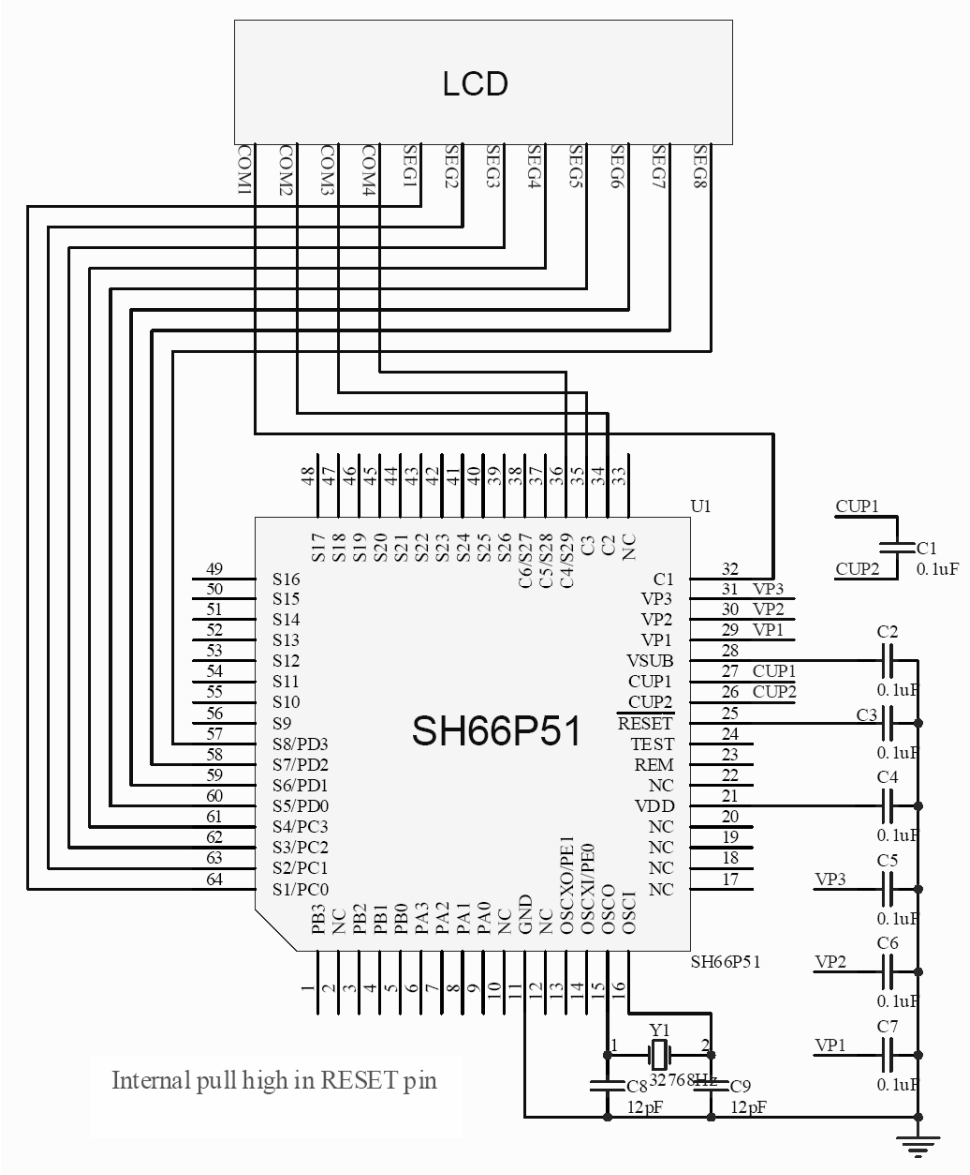


图 3-6-15 LCD 显示实例原理图

■ 程序设计

软件将设计一个 LCD 显示模块作为例子供写 LCD 显示程序的程序员参考一下而已。

为了方便说明 LCD 显示，所以挑选了一个时间的显示作为例子，在程序中也就写进去一个时间跳动的小程序，目的只是作为 LCD 显示模块调试用而已。时间从 AM12：00 开始，循环运行。

当上电时，由于 LCD 驱动寄存器的值是不定的，若在这个时候打开 LCD 的话，就会导致 LCD 显示一些乱的点,所以在上电未打开 LCD 的时候要先对用于 LCD 显示的寄存器进行清零，以避免出现上电乱显示。

在写 LCD 显示程序之前，要先对 LCD 进行分析，由显示的段和 SEGMENT 线与 COMMON

线的关系列出 LCD 的矩阵表，得到如下表：

REGISTER	*LCD*	COM4	COM3	COM2	COM1
$300	SEG1	MIN_L_d	MIN_L_c	MIN_L_b	MIN_L_a
$301	SEG2	CIRH	MIN_L_e	MIN_L_g	MIN_L_f
$302	SEG3	MIN_H_d	MIN_H_c	MIN_H_b	MIN_H_a
$303	SEG4	CIRL	MIN_H_e	MIN_H_g	MIN_H_f
$304	SEG5	HOU_L_d	HOU_L_c	HOU_L_b	HOU_L_a
$305	SEG6	AM	HOU_L_e	HOU_L_g	HOU_L_f
$306	SEG7	HOU_H_d	HOU_H_c	HOU_H_b	HOU_H_a
$307	SEG8	PM	HOU_H_e	HOU_H_g	HOU_H_f

表 3-6-3 LCD 显示矩阵表

得到矩阵表之后，可对矩阵表进行分析，找出最方便写出显示程序的规律。比如上面这张表，我们可以看出 LCD 中的每个 7 段码的排列是有规律的：都是从高到低为 d 段、c 段、b 段和 a 段依次存在同一个寄存器中，e 段、g 段和 f 段又是从高到低放在同一个寄存器的低三位。这样就方便了建表，具体的建表方法请参看程序。

例 3-6-1 以时间 LCD 显示为例带有 LCD 显示模块的程序

```
LIST P=66P51
ROMSIZE=2048
;************************************************
; 系统寄存器(BANK0)
;************************************************
IE          EQU   00H          ;中断使能标志
IRQ         EQU   01H          ;中断请求标志
BTM         EQU   03H          ;基本定时器模式寄存器
LCDON       EQU   07H          ;位 2:设置打开 LCD
TBR         EQU   0EH          ;查表寄存器
INX         EQU   0FH          ;间接寻址伪索引寄存器
DPL         EQU   10H          ;INX 数据指针低四位
DPM         EQU   11H          ;INX 数据指针中三位
DPH         EQU   12H          ;INX 数据指针高三位
OSDUTY      EQU   15H          ;位 1-0:选择 LCD 占空比, 位 2:设置 PORTC 作为 SEG1-4, 位 3:设置
                               ;PORTD 作为 SEG5-8
WDT         EQU   1EH          ;位 2-0:看门狗定时控制寄存器, 位 3:看门狗溢出标志
;************************************************
; 系统寄存器 BANK6(LCD)
```

```
;***********************************************
SEG1        EQU   00H         ;SEG1 寄存器
SEG2        EQU   01H         ;SEG2 寄存器
SEG3        EQU   02H         ;SEG3 寄存器
SEG4        EQU   03H         ;SEG4 寄存器
SEG5        EQU   04H         ;SEG5 寄存器
SEG6        EQU   05H         ;SEG6 寄存器
SEG7        EQU   06H         ;SEG7 寄存器
SEG8        EQU   07H         ;SEG8 寄存器
;***********************************************
; 用户定义寄存器 (BANK0)
;***********************************************
AC_BAK      EQU   28H         ;AC 值备份寄存器
TMP         EQU   29H         ;临时寄存器
FLAG1       EQU   2AH         ;位 0=1, 已计一次 0.5 秒
                              ;位 1=1 为 PM, 位 1=0 为 AM
SEC_L       EQU   2BH         ;秒钟低位寄存器
SEC_H       EQU   2CH         ;秒钟高位寄存器
MIN_L       EQU   2DH         ;分钟低位寄存器
MIN_H       EQU   2EH         ;分钟高位寄存器
HOU_L       EQU   2FH         ;时钟低位寄存器
HOU_H       EQU   30H         ;时钟高位寄存器
;***********************************************
; 程序
;***********************************************
            ORG         0000H
            JMP         RESET
            RTNI
            RTNI
            JMP         BASETIMER       ;基准定时器中断服务程序入口
            RTNI
;*****************************************
; 子程序: BASETIEMR 中断服务程序
;*****************************************
BASETIMER:
            STA         AC_BAK,00H      ;备份 AC 值
            ANDIM IRQ,1101B             ;清基准定时器中断请求标志
BASETIMER_END:
```

```
            LDI         IE,0010B                ;打开基准定时器中断
            LDA         AC_BAK,00H              ;取出 AC 值
            RTNI
;****************************************
; 上电程序
;****************************************
RESET:
            NOP
            LDA         WDT,00H                 ;重置 WDT
;--------------------------------------
;先清用户寄存器
POWER_RESET:
            LDI         DPL,08H
            LDI         DPM,02H
            LDI         DPH,00H
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TBR,00H
            ADCM  DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
            ANDIM DPM,0111B
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,02H
            BNZ         POWER_RESET_1
            SBI         DPL,08H
            BNZ         POWER_RESET_1
;--------------------------------------
;上电时，先对驱动 LCD 的寄存器进行清零，以免打开 LCD 时出现乱点
CLR_LCD:
            LDI         DPL,00H
            LDI         DPM,00H
            LDI         DPH,06H
```

```
CLR_LCD_1:
          LDI       INX,00H
          ADIM      DPL,01H
CLR_LCD_2:
          SBI       DPL,08H
          BNZ       CLR_LCD_1
;--------------------------------------
;初始化系统寄存器
SYSTEM_INITIAL:
                                    ;初始化基准定时器
          LDI       BTM,1010B       ;设置基准定时器预分频为/8, 中断定时为 0.5s
                                    ;初始化 LCD 驱动模块
          LDI       OSDUTY,1100B    ;设置 PORTC 作为 SEG1-4, 设置 PORTD 作为 SEG5-8,
                                    ;1/4 占空比, 1/3 偏压, SEG29/COM4 作为 COM4 使用
          LDI       LCDON,0100B     ;打开 LCD
                                    ;初始化 WDT
          LDI       WDT,0001B       ;设置看门狗定时为 1s
;--------------------------------------
;初始化用户寄存器
USER_INITIAL:
          LDI       HOU_H,01H       ;设置初始时间为 AM12:00
          LDI       HOU_L,02H
          LDI       MIN_H,00H
          LDI       MIN_L,00H
;--------------------------------------
MAIN_PRE:
          LDI       IRQ,00H
          LDI       IE,0010B        ;打开基准定时器中断
;**********************************
MAIN:
          NOP
          HALT                      ;进入 HALT 模式，等待中断唤醒，不影响 LCD 显示
          NOP
          NOP
;**********************************
TIME:
          LDA       WDT,00H         ;重置 WDT
          LDA       FLAG1,00H
```

```
        BAO     TIME_ADDS           ;之前已计一次 0.5s，应该加上 1s
        ORIM    FLAG1,0001B         ;置 "已计一次 0.5s"标志
        JMP     TIME_END
TIME_ADDS:
        ANDIMFLAG1,1110B            ;清"已计一次 0.5s"
        ADIM    SEC_L,01H           ;秒数加一
        DAA     SEC_L
        LDI     TBR,00H
        ADCM  SEC_H,00H
        SBI     SEC_H,06H
        BAZ     TIME_ADDM           ;满 60 秒，分钟应该加一
        JMP     TIME_END
TIME_ADDM:
        LDI     SEC_H,00H           ;清秒钟为 00
        ADIM    MIN_L,01H           ;分钟加一
        DAA     MIN_L
        LDI     TBR,00H
        ADCM  MIN_H,00H
        SBI     MIN_H,06H
        BAZ     TIME_ADDH           ;满 60 分，时钟应该加一
        JMP     TIME_END
TIME_ADDH:
        LDI     MIN_H,00H           ;清分钟为 00
        SBI     HOU_H,01H
        BNZ     TIME_ADDH_2
        SBI     HOU_L,02H
        BNZ     TIME_ADDH_2         ;时钟不是 12 点，跳转
                                    ;时钟为 12 点，下一个小时应该是 1 点，设置时钟为 01
TIME_ADDH_1:
        LDI     HOU_H,00H
        LDI     HOU_L,01H
        JMP     TIME_END
TIME_ADDH_2:
        ADIM    HOU_L,01H       ;时钟加一
        DAA     HOU_L
        LDI     TBR,00H
        ADCM  HOU_H,00H
        SBI     HOU_H,01H
```

```
            BNZ         TIME_END
            SBI         HOU_L,02H
            BNZ         IME_END             ;加完时钟后不是 12 点，跳转
                                            ;加完之后时钟为 12 点，该转换 AM 和 PM
            EORIM FLAG1,0010B               ;转换 AM/PM 标志
TIME_END:
;*************************************
; 模块:    LCD 显示模块
; 输入变量:      FLAG1,MIN_L,MIN_H,HOU_L,HOU_H
; 使用变量:      TBR,TMP
; 输出变量:      SEG1,SEG2,SEG3,SEG4,SEG5,SEG6,SEG7,SEG8
;*************************************
DISP:
            LDI         TBR,1000B
            EORM  SEG2,06H
            LDI         TBR,1000B
            EORM  SEG4,06H                  ;每 0.5 秒，改变":"的亮/灭，达到闪烁 ":"，周期为 1 秒
DISP_AMPM:
            LDA         FLAG1,00H
            BA1         DISP_PM             ;PM，跳转
DISP_AM:
            LDI         TBR,0111B
            ANDM  SEG8,06H                  ;清"PM"显示
            LDI         TBR,1000B
            ORM         SEG6,06H            ;显示"AM"
            JMP         DISP_M
DISP_PM:
            LDI         TBR,0111B
            ANDM  SEG6,06H                  ;清"AM"显示
            LDI         TBR,1000B
            ORM         SEG8,06H            ;显示"PM"
DISP_M:
            LDI         TBR,0FH
            LDA         MIN_L,00H
            CALL        07FAH               ;查表得显示数据
            STA         SEG1,06H            ;显示分钟低位的 dcba 四段
            LDI         TMP,1000B
            ANDM  SEG2,06H
```

```
        LDA     TBR,00H
        ORM     SEG2,06H        ;显示分钟低位的 egf 三段

        LDI     TBR,0FH
        LDA     MIN_H,00H
        CALL    07FAH           ;查表得显示数据
        STA     SEG3,06H        ;显示分钟高位的 dcba 四段
        LDI     TMP,1000B
        ANDM  SEG4,06H
        LDA     TBR,00H
        ORM     SEG4,06H        ;显示分钟高位的 egf 三段
DISP_H:
        LDI     TBR,0FH
        LDA     HOU_L,00H
        CALL    07FAH           ;查表得到显示数据
        STA     SEG5,06H        ;显示时钟低位的 dcba 四段
        LDI     TMP,1000B
        ANDM  SEG6,06H
        LDA     TBR,00H
        ORM     SEG6,06H        ;显示时钟低位的 egf 三段

        LDA     HOU_H,00H
        BAZ     DISP_H_1        ;时钟高位若为 0, 不显示高位，跳转
        LDI     TBR,0FH
        LDA     HOU_H,00H
        CALL    07FAH           ;查表得到显示数据
        STA     SEG7,06H        ;显示时钟高位的 dcba 四段
        LDI     TMP,1000B
        ANDM  SEG8,06H
        LDA     TBR,00H
        ORM     SEG8,06H        ;显示时钟高位的 egf 三段
        JMP     DISP_END
DISP_H_1:
        LDI     TBR,00H
        STA     SEG7,06H
        LDI     TBR,1000B
        ANDM  SEG8,06H          ;清时钟高位显示
DISP_END:
```

```
;************************************
        JMP     MAIN
;********************************************
;LCD 显示数据表:根据 LCD 的矩阵表定出数据表是按 egf,dcba 来建立的
        ORG     07F0H
        ;0egf, dcba
        RTNW    0101B,1111B             ;0
        RTNW    0000B,0110B             ;1
        RTNW    0110B,1011B             ;2
        RTNW    0010B,1111B             ;3
        RTNW    0011B,0110B             ;4
        RTNW    0011B,1101B             ;5
        RTNW    0111B,1101B             ;6
        RTNW    0000B,0111B             ;7
        RTNW    0111B,1111B             ;8
        RTNW    0011B,1111B             ;9
        ORG     07FAH
        TJMP

        END
```

3.7 模数转换器(Analog-to-Digital Converter)

很多的系统设计中要求提供模拟信号的输入和监测功能,这就需要模/数转换电路(ADC)把输入的连续变化的模拟电压信号转换成单片机能够识别的数字信号。中颖公司SH6xxx产品线中很多产品已经集成了多种特性的ADC模块,其转换输出分辨率从普通的8位到10位和12位,基本可以满足不同系统设计的需求。在这一章中我们将以最普通的8位分辨率ADC模块为基础,从应用角度讲解在一个控制系统中如何合理的使用片上ADC功能。

3.7.1 SH6xxx 单片机片上ADC模块综述

对于8位的ADC模块,其输出的数字量在00H～FFH(即十进制的0～255)之间,00H对应于允许输入的最低电压(一般为地电平0V),FFH对应于允许输入的最高电压(输入电压等于基准参考电压时)。SH6xxx系列单片机的片上ADC模块最多可以有10个模拟信号输入通道(不同的芯片上提供的模拟信号通道数量不同),程序在运行过程中经过内部的一个多路开关可以选择任何一个通道进行A/D 转换。转换所需的基准电压也可以

通过软件配置选择。其中 8 位分辨率 A/D 转换模块的基本工作原理如图 3-7-1 所示。内部 A/D 转换过程采用的是“逐次逼近”法,一次转换所需的时间最短为几十微秒,速度相当快。

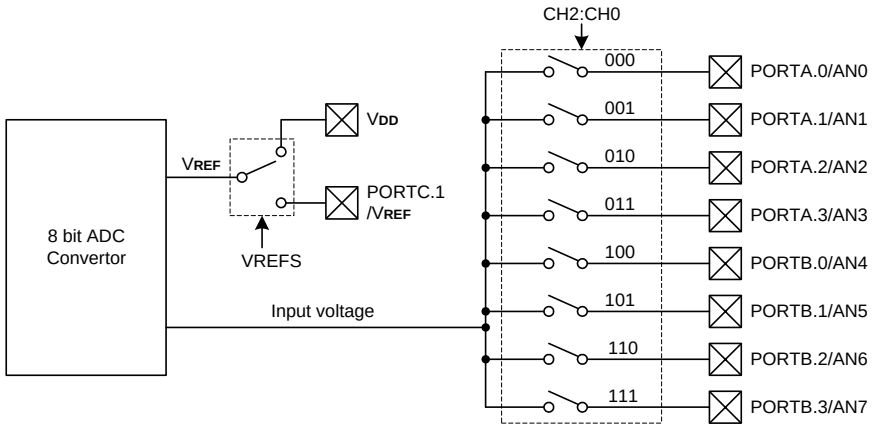


图 3-7-1 8 位分辨率 A/D 转换模块工作原理图

3.7.2 ADC相关控制寄存器介绍

ADC 模块最重要的寄存器是控制寄存器(ADC Control Register 和 ADC Channel Control Register) ,还有就是转换结果寄存器(ADC Data Register)。对于模拟信号输入管脚的配置,需要相应的端口输入/输出与模拟信号通道控制寄存器(ADC Port Configuration Control Register)的正确设定。另外,如果需要 A/D 转换结束时产生中断响应,则有相关的寄存器(IE Control Register 和 IE Request Flag Register)需要考虑。

ADC 控制寄存器

ADC 控制寄存器位于内部 RAM 零页的地址 17H,其中的各数据位定义如下图所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$17	GO/$\overline{DONE}$	TADC1	TADC0	ADCS	R/W	

ADC 控制寄存器的数据位定义：

TADC1～0: A/D 转换时钟周期选择

00 = t_{AD} = t_{OSC},即时钟周期源自芯片的 1 倍主振荡周期

01 = t_{AD} = 4 x t_{OSC},即时钟周期源自芯片的 4 倍主振荡周期

10 = t_{AD} = 8 x t_{OSC},即时钟周期源自芯片的 8 倍主振荡周期

11 = t_{AD} = 16 x t_{OSC},即时钟周期源自芯片的 16 倍主振荡周期

ADCS: A/D 转换周期选择

0 = 50 t_{AD}

1 = 330 t_{AD}

GO/$\overline{DONE}$: A/D 转换启动控制位和转换状态标志位

这位既是 A/D 转换控制位,通过软件将其置 1 后开始一个 A/D 转换过程,同时又是一个标志位

1 = A/D 转换正在进行中

0 = A/D 转换过程结束

ADC 通道选择控制寄存器

ADC 通道选择控制寄存器位于内部 RAM 零页的地址 14H,其中的各数据位定义如下图所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$14	ADCON	CH2	CH1	CH0	R/W	

ADC Channel Control Register 寄存器的数据位定义：

CH2:CH0: A/D 转换输入模拟信号通道选择

000 = 选择通道 0,AN0

001 = 选择通道 1,AN0

010 = 选择通道 2,AN0

011 = 选择通道 3,AN0

100 = 选择通道 4,AN0

101 = 选择通道 5,AN0

110 = 选择通道 6,AN0

111 = 选择通道 7,AN0

ADCON: A/D 转换模块启用控制位

1 = A/D 转换模块开始工作

0 = A/D 转换模块被禁止,该部分电路没有任何耗电

ADC 与端口复用控制寄存器

ADC 与端口复用控制寄存器位于内部 RAM 零页的地址 13H,其中的各数据位定义如下图所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$13	VREFS	ACR2	ACR1	ACR0	R/W	

ADC Port Configuration Control Register 寄存器的数据位定义：

ACR2～0: A/D 转换模块管脚功能配置位

这 3 位决定了功能复用的引脚哪些作为普通数字 I/O,哪些作为数/模转换时的电压信号输入。其组合控制模式如下表所示。

ACR2:ACR0	PORTB.3	PORTB.2	PORTB.1	PORTB.0	PORTA.3	PORTA.2	PORTA.1	PORTA.0
000	D	D	D	D	D	D	D	D
001	D	D	D	D	D	D	D	AN0
010	D	D	D	D	D	D	AN1	AN0
011	D	D	D	D	D	AN2	AN1	AN0
100	D	D	D	D	AN3	AN2	AN1	AN0
101	D	D	D	AN4	AN3	AN2	AN1	AN0
110	D	D	AN5	AN4	AN3	AN2	AN1	AN0
111	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

注: ANx 表示对应的引脚为模拟信号输入,D 表示引脚为数字输入输出。

VREFS: A/D 转换模块的基准参考电压选择位

1: V_{REF}=PORTC.1,即 A/D 转换时的基准电压为从 PORTC.1 端口输入的模拟电压

0: V_{REF} = V_{DD}, 即 A/D 转换时的基准电压为 V_{DD}

ADC 数据寄存器

ADC 数据寄存器位于内部 RAM 零页的地址 15H 和 16H,其中的各数据位定义如下所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$15	A3	A2	A1	A0	R	
$16	A7	A6	A5	A4	R	

这 2 个寄存器只可进行读操作,上电复位时其中的内容不确定。

A/D 转换中断相关的寄存器

A/D 转换作为一个标准的外围功能模块,当 A/D 转换过程结束时将产生中断标志位 IRQAD。如果在软件中欲响应 IRQAD 中断,则 IEAD 必须置 1。在中断服务程序中软件必须将 IRQAD 标志位清除。相关的使能位和标志位的指示如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$00	IEAD	IET0	IET1	IEP	R/W	
$01	IRQAD	IRQT0	IRQT1	IRQP	R/W	

ADC IE Control and IE Request Flag Register 寄存器的数据位定义:

IEAD: A/D 转换模块中断使能控制位

1: A/D 转换模块中断使能

0: A/D 转换模块中断被禁止

IRQAD: A/D 转换模块中断请求标志位

1: A/D 转换模块中断请求标志有效

0: A/D 转换模块中断请求标志无效

3.7.3 A/D转换过程说明

一个完整的 A/D 转换可以按如下步骤实现:

(1) 设定管脚工作模式和模拟信号通道个数,选择基准电压输入方式。

(2) 设定 A/D 转换的时钟周期(确保 1s ≤ t_{AD} ≤ 33.4 s),选择模拟信号的输入通道。(注意 ADCON = 1,但 GO/$\overline{DONE}$ 不能置 1。)

(3) 若需要中断响应,则按3.7.2中的指示设定相关中断控制寄存器(即 IRQAD清0和 IEAD 置1) 。

(4) 等待足够长的采样延时。

(5) 将 GO/$\overline{DONE}$ 置 1 启动一次 A/D 转换过程。

(6) 查询 A/D 转换结束标志: GO/$\overline{DONE}$ 位在 A/D 转换结束时会自动清 0, IRQAD 标志位在A/D 转换结束时会自动置1,这两个位都可以作为软件查询A/D 转换是否结束的标志。使用 IRQAD 标志位时记得要用软件及时将其清除。

(7) 若使用中断响应 A/D 转换的结束,则(6)将不再适用。A/D 转换结束时 IRQAD 标志位的置 1 将使单片机进入中断服务程序,在处理中断服务程序时记得将 IRQAD 标志位清 0。

(8) A/D 转换结束,直接从ADC Data Register(016H和015H)中读取 8位转换结果,存入其它缓冲单元或直接进行运算处理。

(9) 若要选择其它通道输入的模拟信号进行 A/D 转换,则修改(2)中的 CH2:CH0,重复(3) ～(8)的循环。

例 3-7-1 (SH69P43) 8 路仿真信号信道循环进行模/数转换

```
ADCON1      EQU         13H                 ;ADC 参考电压选择及仿真信道数设置寄存器
```

```
ADCON2    EQU     14H          ;ADC 模块状态及转换信道选择寄存器
AL        EQU     15H          ;ADC 数据低 4 位
AH        EQU     16H          ;ADC 数据高 4 位
ADCON3    EQU     17H          ;ADC 状态标志、转换时间及时钟周期选择寄存器
          :
DATA_L    EQU     31H          ;转换结果缓冲区低位
DATA_H    EQU     32H          ;转换结果缓冲区高位
          :
          :
          ORG     00H
          JMP     RESET
          NOP
          NOP
          NOP
          NOP
RESET:
          :
          LDI     ADCON1,07H   ;使用内部参考电压，使用所有的 8 个 A/D 转换口
          :
          LDI     ADCON2,00H   ;启动 A/D 转换模块，选择 ADC 信道 AN0
          LDI     ADCON3,06H   ;选择 tAD=8tOSC，A/D 转换时间 50tAD
          :
          :
MAIN:
          CALL    ADC_PROC     ;进行 A/D 转换
          CALL    NEXT_CHN     ;选择下一个通道
          CALL    DATA_PROC    ;根据采到的数据进行处理
          JMP     MAIN              ;重复采样
;*************************************************************************************
;对当前所选择的通道进行一次模/数转换，并将转换结果保存于缓冲区 DATA_L 和 DATA_H 中
;*************************************************************************************
ADC_PROC:
          ORIM    ADCON3,1000B      ;启动 A/D 转换
          LDA     ADCON3,00H
          BA3     $-1               ;A/D 转换未完成，继续检测
          LDA     AL,00H            ;读取 ADC 结果低 4 位
          STA     DATA_L,00H        ;保存数据低 4 位，以备后用
          LDA     AH,00H            ;读取 ADC 结果高 4 位
```

```
        STA     DATA_H,00H          ;保存数据高 4 位，以备后用
        RTNI
;*****************************************************************************************
;选择下一个通道，准备下一次模/数转换
;*****************************************************************************************
NEXT_CHN:
        SBI     ADCON2,0FH
        BAZ     NEXT_CHN8           ;第 8 个通道 AN7，将设定下一个通道为 AN0
        ADIM    ADCON2,01H          ;选择下一个通道
        JMP     NEXT_CHN_END
NEXT_CHN8:
        LDI     ADCON2,08H          ;设定为第 1 个通道 AN0
NEXT_CHN_END:
        RTNI
```

3.7.4 电容型ADC与电阻型ADC

在中颖公司的 SH6xxx 系列单片机中，有采用电容网络的模/数转换模块，也有采用电阻网络的模/数转换模块，但是就目前来讲，只有 SH69P55 使用的是电阻型的 SAR 模/数转换模块。

电容型和电阻型的模/数转换模块在原理上的区别在于 SAR 模/数转换电路中的重要组成电路 DA（模数转换）电路的构造不同，电容型是基于电容网络的，电阻型则是基于电阻网络的。

电容型的模/数转换模块的优点是精度比较高，缺点是由于有输入电容比较大的原因而导致了外部信号源内阻不能太大；电阻型的模/数转换模块的优点是输入电容小，缺点就是限于电阻的匹配，转换的精度不高。

在使用上，电容型要注意因为输入电容较大，由于采样时间是跟外部信号源的内阻有关，所以使用上要注意信号源的内阻不要太高以保证采样时信号建立完全，或者给予足够的采样时间。一般信号源内阻要控制在 10KΩ以内。 电阻型则要注意信号的稳定以及保证足够干净的电源和地。

由于 SH69P55 使用的是电阻型的模/数转换模块，可以在 SH69P55 的规格书中看到 SH69P55 是没有 ADCS 这个寄存器的，它的模/数转换时间固定为 $13t_{AD}$，实际应用中可根据模/数转换时间为 $13t_{AD}$ 来进一步选择模/数转换时钟源 t_{AD}。，因为要求模/数转换时间不超过 25μs。

3.7.5 输入的仿真信号的参数要求

模/数转换模块检测的模拟量是电压信号，所以所要检测的电压必须符合要求才能正确地得到模/数转换结果。检测的信号如果不是电压信号则需要先把信号转换成电压信号才能

进行检测，而且电压的信号要符合相关要求。

为了使模拟输入通道的电压信号能够被正确地进行模/数转换，则输入的电压信号的电压范围必须在 0V 与基准电压之间，超出这个范围模/数转换不能得到正确的转换结果。低于 0V 的电压信号转换结果将得到 0x00，高于基准参考电压的电压信号转换结果将得到 0xFF。

为了提高模/数转换的精确度和分辨率，在设计硬件电路的时候要注意让模拟输入通道的电压信号的动态变化范围尽可能地覆盖整个有效的电压区间，也就是 0V～基准电压。当然一般较少出现模拟输入通道的电压信号刚好覆盖了 0V～基准电压,但是在设计电路时可以使模拟输入通道的电压信号所占有的 0V～基准电压的区间比例越大越好。比如选择内部参考电压 V_{DD}＝5V，若要进行采样的信号电压为 2V～3V，这样采样的电压范围只占有效电压区间的 1/5，这个时候就有必要对要进行采样的信号通过电路进行调整，把微小变化的信号进行放大，调整到接近有效区间电压再送给单片机进行模/数转换，这样就能很好的提高精度。

注意，模拟输入通道的电压不要超过 V_{DD}，因为有可能因为电压太高而烧坏芯片。

3.7.6 设置用于模/数转换采样通道的I/O口

由于 SH6xxx 单片机用于 ADC 采样通道的管脚是与普通 I/O 口功能管脚复用的，所以在使用 ADC 之前要先对管脚进行正确的设定才能够正常地使用模/数转换。

在 3.7.3 小节中我们已经对模/数转换控制寄存器 ADCON1 的每一位的定义都作了说明，其中位 2～位 0 的 ACR2～ACR0 是用来控制 I/O 口的配置,即选择该管脚是作为普通 I/O 口功能使用，还是用来做 ADC 采样的通道。具体的配置可参考表 7-1，这里我们要讲的是详细的配置。

由表 7-1 中可以看出，管脚在作为普通 I/O 口和作为 ADC 模拟输入通道端口的切换要从 PA0/AN0 开始，直到 PB2/AN6 和 PB3/AN7。而且 PB2/AN6 和 PB3/AN7 是同时设定的，也就由于 PB2/AN6 和 PB3/AN7 是同时设定的，才不能单独选择只使用 7 个 ADC 模拟输入通道，只能选择 6 个或 8 个 ADC 模拟输入通道。

首先要通过实际应用电路中所需要使用的 ADC 模拟输入通道的个数来设定 ACR2～ACR0 的值。例如，实际应用电路中所需要使用的 ADC 模拟输入通道的个数为 4 个，由从 PA0/AN0 开始的原则可以设定使用 PA0/AN0～PA3/AN3 四个管脚来为 ADC 模拟输入通道的端口，也就是要往 ACR2～ACR0 写入“100B”的数值。

对于用作 ADC 模拟输入通道的管脚，在其作为普通 I/O 口的 I/O 口输入/输出状态寄存器中最好将其设置为输入状态，以避免写程序时处理不小心而导致出现问题。因为 ADC 模块的打开和关闭是作为选择管脚为普通 I/O 口还是 ADC 模拟输入通道端口的总开关。在硬件上管脚要成为一个真正的 ADC 模拟输入通道端口必须满足两个条件：一是 ADC 模块打开，二是 ACR2～ACR0 的设置中将其设置为 ADC 模拟输入通道端口。所以当一个用作 ADC 模拟输入通道端口的管脚在其作为普通 I/O 口时所要设置的输入/输出状态寄存器中没有设置为输入的话，万一在程序中把 ADC 模块关闭，这个时候虽然 ACR2～ACR0 的值虽设置其为 ADC 模拟输入通道端口，但实际由于 ADC 模块的关闭而使其成为一个普通 I/O 口，这个时候这个 I/O 口所输出的电平可能会对外部的电路造成一定的影响而出问题或导致之后的ADC采样数据不对。所以，

如果不是很特殊的应用，最好把用来做 ADC 采样的管脚的 I/O 口状态设定为输入状态。

另外，还要注意 SH69P56 这颗芯片的 AN0 管脚是和外部参考电压输入口复用的，当其用作外部参考电压时，就不能用来做 ADC 采样了。

3.7.7 ADC时钟和时间的选择

在进行模/数转换之前，必须先设置正确的转换时锺和适当的转换时间。在 3.7.2 小节里边模/数转换控制寄存器ADCON3中已经对TADC1～TADC0和ADCS进行了说明。TADC1～TADC0是用来控制模/数转换时锺的，而 ADCS 是用来控制模/数转换时间。模/数转换的时锺来源于单片机的主振荡器（包括 RC 振荡）的振荡频率，通过对 TADC1～TADC0 写入不同的值来设置模/数转换时锺为主振荡器的振荡频率还是 2 分频、4 分频还是 8 分频。通过对 ADCS 写 0 或写 1 可以将模/数转换的时间设置为 $50t_{AD}$或是 $330t_{AD}$，具体应该设置的模/数转换的时间按实际应用来选择。

模/数转换时锺的设定必须满足一个条件：1μs≤t_{AD}≤33.4μs。根据这个要求，我们就可以知道如何设定 TADC1～TADC0 的值。例如，单片机的主振荡器的振荡频率为 4MHz，要使得 t_{AD}满足条件的话，就是说 t_{AD}≥1μs，要求至少要进行 4 分频；t_{AD}≤33.4μs 理论上最多只能进行 133.6 分频。不过，由于通过设定 TADC1～TADC0 只能设置 1 分频、2 分频、4 分频和 8 分频，所以在此就可以选择 4 分频和 8 分频。

通过设定 ADCS 的值可以设定模/数转换的时间长短，时间可为 $50t_{AD}$或 $330t_{AD}$，也就是说，整个模/数转换时间的长短是由 ADCS 和 TADC1～TADC0 共同控制的，可以按实际应用来选择模/数转换的时间。

3.7.8 参考电压的选择

在模/数转换的应用中，参考电压的选择是一个很重要的环节。ADC 模块所使用的参考电压的精度和稳定度将直接决定了模/数转换结果的准确性。

SH6xxx 单片机中的 ADC 模块可以直接使用芯片的电源电压 V_{DD}作为参考电压（即内部参考电压），也可以使用在外部参考电压管脚上外接一个基准电压 V_{REF}作为参考电压（即外部参考电压）。需要说明的是 SH69P46 只能选择内部参考电压。用作外部参考电压的管脚一般是与一个普通 I/O 口共享，大部分型号的芯片都是共享了 PC1 口，也就是说当选择内部参考电压时，这个管脚就作为普通的 I/O 口来用，而选择外部参考电压时这个管脚就作为外部基准电压的输入管脚。有个特例，是 SH69P56 的外部参考电压共享的是 PA0/AN0 管脚，当作为外部参考电压使用时，则就不能做普通 I/O 口使用和模/数转换模拟输入通道使用了。

选择内部参考电压还是外部参考电压由模/数转换控制寄存器 ADCON1 位 3 的 VREFS 来控制，具体参考 3.7.2 小节。

在一般的设计中，通常是使用内部参考电压，也就是使用芯片的 V_{DD} 作为参考电压，所以要求 V_{DD}在系统运行中要保持稳定，主要是在 ADC 进行采样的时候一定要保持稳定。在设计 PCB 的时候，就要对芯片的 V_{DD}的布线进行考虑，尽量对单片机的电源和地线进行单独布

线，以避免受到其它负载电路的干扰，保证其稳定性。由于 I/O 口的输出状态改变可能会带来外部负载的电流的变化，这个时候在电源中可能就产生毛刺或波动，也会影响到 V_{DD}，所以在进行 ADC 采样时最好不要对 I/O 口的输出状态进行改变。使用内部参考电压的好处是可以留出一个 I/O 口可以作其它用途，缺点是 V_{DD} 没处理好的话参考电压可能容易波动导致模/数转换结果的误差会稍稍大一点。

对于模/数转换结果要求比较高的应用中，最好是选择外部参考电压。使用外部参考电压时，可将模拟地与数字地分开布线，模拟电源与数字电源分开布线，最后才将模拟地与数字地连接，将模拟电源与数字电源连接稳压电源。这样可以很好的隔离数字电路对模拟电路的串扰，大大提高了参考电压的稳定性，从而保证了模/数转换结果的精度。使用外部参考电压时，参考电压最大不能超过 V_{DD}，最小不能小于 2.4V。

进行 A/D 转换的模拟输入信号的电压范围为 0～V_{REF}，超出这个范围 A/D 转换结果就与电压没有关系了。

3.7.9 HALT模式下的模/数转换

SH6xxx 系列单片机在 HALT 模式下保持主振荡器还在继续振荡，所以模/数转换模块能够工作在 HALT 模式下，并且可以通过模/数转换中断来将单片机从 HALT 模式下唤醒。而在 STOP 模式下主振荡器是停止工作的，模/数转换模块没有了时钟来源，所以不能在 STOP 模式下工作。一旦进入了 STOP 模式，模/数转换模块就停止工作。

要使用模/数转换中断来唤醒 HALT 模式，只需在打开模/数转换中断及将 GO/$\overline{\text{DONE}}$ 置 1 启动模/数转换后就可以进入 HALT 模式了。由于主振荡器还在振荡，所以模/数转换模块还在工作，进行模/数转换。当转换结束时，GO/$\overline{\text{DONE}}$ 被清 0 的同时，模/数转换中断请求标志 IRQAD 也被置 1，由于模/数转换中断被允许（IEAD＝1），所以这个时候将产生中断，并将单片机从 HALT 模式下唤醒。唤醒后，进入模/数转换中断服务程序执行指令。

若是在模/数转换过程中，进行了 STOP 模式，则由于 STOP 模式下主振荡器停止，造成模/数转换也相应停止。所以，模/数转换不能用来唤醒 STOP 模式。

现在举个例子来说明一下使用模/数转换中断来将单片机从 HALT 模式中唤醒。例 3-9 是以 SH69P43 为控制 IC，4MHz 的主振荡器的一段代码来说明 HALT 模式下的模/数转换过程。

例 3-7-2 （SH69P43）HALT 模式下进行模/数转换

```
ADCON1      EQU     13H     ;ADC 参考电压选择及仿真信道数设置寄存器
ADCON2      EQU     14H     ;ADC 模块状态及转换信道选择寄存器
AL          EQU     15H     ;ADC 数据低 4 位
AH          EQU     16H     ;ADC 数据高 4 位
ADCON3      EQU     17H     ;ADC 状态标志、转换时间及时钟周期选择寄存器
            :
DATA_L      EQU     31H     ;转换结果缓冲区低位
DATA_H      EQU     32H     ;转换结果缓冲区高位
```

```
    AC_BAK          EQU     33H         ;备份 AC 值
                    :
                    :
            ORG     00H
            JMP     RESET
            JMP     ADC_ISP
            NOP
            NOP
            NOP
    RESET:
            :
            LDI     ADCON1,07H          ;使用内部参考电压，使用所有的 8 个 A/D 转换口
            :
            LDI     ADCON2,00H          ;启动 A/D 转换模块，选择 ADC 信道 AN0
            LDI     ADCON3,06H          ;选择 tAD=8tOSC，A/D 转换时间 50tAD
            :
            :
MAIN:
            LDI     IE,1000B            ;打开 A/D 转换中断
            ORIM    ADCON3,1000B        ;启动 A/D 转换
            NOP
            HALT                        ;进入 HALT 模式
            NOP
            NOP
            CALL    NEXT_CHN            ;选择下一个通道
            CALL    DATA_PROC           ;根据采到的数据进行处理
            JMP     MAIN                ;重复采样
;*********************************************************************************************
;选择下一个通道，准备下一次模/数转换
;*********************************************************************************************
NEXT_CHN:
            SBI     ADCON2,0FH
            BAZ     NEXT_CHN8           ;第 8 个通道 AN7，将设定下一个通道为 AN0
            ADIM    ADCON2,01H          ;选择下一个通道
            JMP     NEXT_CHN_END
NEXT_CHN8:
            LDI     ADCON2,08H          ;设定为第 1 个通道 AN0
NEXT_CHN_END:
```

```
            RTNI
;*********************************************************************************************
;模/数转换中断服务程序
;*********************************************************************************************
ADC_ISP:
            STA         AC_BAK,00H              ;先备份 AC 值
            ANDIM IRQ,0111B                     ;清模/数转换中断请求标志
            LDA         AL,00H                  ;读取 ADC 结果低 4 位
            STA         DATA_L,00H              ;保存数据低 4 位，以备后用
            LDA         AH,00H                  ;读取 ADC 结果高 4 位
            STA         DATA_H,00H              ;保存数据高 4 位，以备后用
ADC_ISP_END:
            LDA         AC_BAK,00H              ;恢复 AC 值
            RTNI
```

3.7.10 模/数转换模块的应用

在这一节我们将以 8 位分辨率的模/数转换模块应用电压检测和温度检测进行举例说明，以方便大家参考。

■ **电路工作原理**

以 SH69P43 为例，电压及温度检测电路原理图如图 3-7-2 所示。

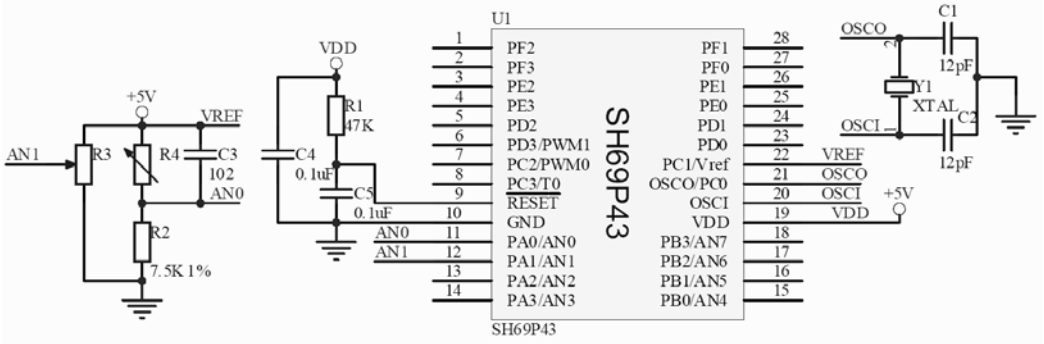


图 3-7-2　　电压及温度检测电路原理图

从电路原理图中可以看到，使用两路模/数转换通道进行 ADC 转换：一路是 AN0，电压采样是为了判断热敏电阻 R4 的阻值，从而得到相应的温度；另一路是 AN1，是直接用来检测这个仿真输入信道的输入电压。

电压检测电路很简单，就是一个变阻器来仿真，当滑动位置发生变化时，电压也随之变化，用 AN1 来检测这个电压。

温度检测电路中，使用参考电阻 R2 与热敏电阻 R4 串联来分压，当热敏电阻 R4 发

生变化时，AN0 检测到的电压也发生变化。现在假设 AN0 检测的电压为 U，采样等到的数值为 X（十进制），则有：

$$\frac{X}{256}=\frac{U}{VREF}=\frac{U}{5}=\frac{R2}{R2+R4}$$

由于 R2 已知，所以 X 和 R4 则成为一一对应关系。一个采样得到的数值对应于一个电阻，由这个电阻也就知道了当前的温度值。在程序建表时可直接由采样值将温度值建成温度表即可。为了提高检测的精确度，参考电阻 R2 必须使用精确电阻，若是在环境温度变化较大的地方，还要使用金属膜电阻。

■ 数字滤波器的软件设计方法

在一个实际系统中，一定会存在各种各样的干扰。要有效的防止这些干扰对整个系统的影响，首先就要从硬件入手，从硬件上利用各种各样的滤波电路来防止干扰的影响。

在后期的软件设计上，还可以使用程序来搭建数字滤波器来防止干扰的影响。由于各种参数的干扰成分不同，所以所要搭建的数字滤波器的方法也不一样。

目前数字滤波方法有很多种，比如限幅滤波法、中位值滤波法、算术平均滤波法、递推平均滤波法、防脉冲平均滤波法等等。各个数字滤波方法各有各的优点和缺点。这里讲述应用较广效果也较好的递推平均滤波法及防脉冲平均滤波法。

1）递推平均滤波法

递推平均滤波法是一种只需采样一次数值就能马上得到平均值的数字滤波方法，主要应用于测量速度较慢或要求数据计算速度较快的控制系统。递推平均滤波法其实就是一个队列，这个队列中可以放 N 个数据，当队列放满后，把队列里边所有数据进行平均，就可以得到需要的采样数据；当再进行一次新的测量时，就把新的测量数据放入队尾，也就把最早测量到的数据也就是队首的数据挤了出去，这个时候还是有 N 个数据，又可以再求平均值，又得到需要的采样数据。

递推平均滤波法对周期性干扰良好的抑制作用，平滑度高，灵敏度低；但由于寄存器及其它方面的限制，N 值一般取 1～4，所以对偶尔出现的脉冲干扰的抑制作用较差，不适用于脉冲干扰比较严重的系统，而是适用于高频振荡系统。但是在实际单片机的应用中这个问题的影响就不是很大了。

在实际的单片机应用中，如果要开一个队列的寄存器出来，那将会消耗大量的寄存器，所以在单片机应用中，方法是这样的：设定一个数据累加缓冲区的寄存器，要求可以放得下 N 个数据的总和，先采集 N 个数据，每次采到数据就把数据加到数据累加缓冲区，直到采集到 N 个数据，这个时候求 N 个数据的平均值可得第一个处理后的数据；之后，每采集一个新的数据，则将之前数据累加缓冲区的数据减去之前所得到的平均值，再加上这次采集到的新数据，数据累加缓冲区得到的又是一个新的 N 个数据的总和，又可以再求平均值作为处理后的数据，就这样不停地循环下去，每采一个数据就能得到一个数据处理后得到的数据，计算速度就比较快了。由于可以将 A/D 采样的速度放得很快，这个时候 N 的数值就可设置的比较大，比如 256 也是可以的。由于求平均的数据个数很

多，这样就能比较有效的去除偶尔脉冲的干扰了。

现在以 N＝256 来说明一个递推平均滤波法在 8 位模/数转换中的使用方法。

先在单片机中的寄存器中定义出 4 个半字节作为数据累加缓冲区 DATA，从高到低分别定义为 DATA_HH、DATA_HL、DATA_LH 和 DATA_LL。然后进行 256 次的模/数转换，每一次转换得到的 8 位采样数据都直接加到数据累加缓冲区里边。256 次采样完毕，求平均值就是把数据累加缓冲区里边的数据除以 256。由于除以 256 就相当于把数据右移 8 位后所得到的数据，所以不用除就能知道平均值了，平均值就是由 DATA_HH 和 DATA_HL 所组成的 8 位数据。以后每进行模/数转换一次，得到转换结果 ADCDATA，数据累加缓冲区的数据要更新为减去上次得到的平均值再加上这次的转换结果 ADCDATA，用公式表示就是：

DATA＝DATA－（DATA_HH、DATA_HL）＋ADCDATA

得到新的 DATA 后，平均值又是新的 DATA 里边的 DATA_HH 和 DATA_HL 所组成的 8 位数据。当然，这个方法的缺点还是灵敏度太低，对于突变的信号反应较慢。N 的值越大，灵敏度越低，但是精度越高；反之，N 的值越小，灵敏度越高，精度越低。实际应用中要根据系统的实际需要来设定 N 的值。

2）防脉冲平均滤波法

在脉冲干扰比较严重的场合，算术平均滤波法就很容易把干扰脉冲平均到结果中去造成错误，所以一般的平均滤波法是不能用在脉冲干扰比较严重的场合。对于脉冲干扰比较严重的场合，可以对算术平均滤波法进行改进，使其能够很好的抑制脉冲的干扰，这就是防脉冲平均滤波法。防脉冲平均滤波法与算术平均滤波法的区别在于算术平均滤波法是采样 N 个数据后就对 N 个数据进行求平均值，而防脉冲平均滤波法是采集到 N 个数据后先对 N 个数据进行排序，去除最大的几个值和最小的几个值，剩下的值再进行求平均值，这样就可以滤去正脉冲和负脉冲，很好地提高了采样的准确性。

防脉冲平均滤波法的优点就是抗干扰强，准确度高，缺点就是速度较慢，因为对数据进行排序会耗去很多时间。N 的取值越大，去除的值相应增多，抗干扰就越强，速度就越慢；反之，N 的取值越大，去除的值相应减少，抗干扰就越弱，速度越快。由于单片机内部 RAM 和 ROM 较小，也不能耗太多时间在排序上，所以，为了提高测量速率，一般取 N＝4。

■ 程序实例

电路图如图 3-7-2 所示，以 SH69P43 为控制 IC，VDD＝＋5V，4M 晶振为主振荡器，选择内部参考电压 $V_{REF}=V_{DD}=5V$，AN0 用来做温度检测，AN1 用来做电压检测。

在程序中，1ms 进行一次 A/D 转换。对于温度检测，采用递推平均滤波法来进行数据处理，因为温度的变化不会突变，对灵敏度要求不是太高；而电压检测则使用防脉冲均值滤波法来进行数据处理，能够有效地去除脉冲的干扰。

电压检测中，由于参考电压为 5V，8 位的 ADC 可将电压分为 256 个等分，这样每一个计数值的电压为近 0.02V，理论上会有 0.02V 的误差，实际中的误差可能还要再大些，

最好取两个有效数字，为了方便说明，程序中保留了四个有效数字。

温度检测中，假设热敏电阻阻值和温度的对应如下表 3-7-1：

溫度（℃）	阻值（KΩ）	溫度（℃）	阻值（KΩ）	溫度（℃）	阻值（KΩ）
0	167.570	17	72.121	34	33.798
1	159.010	18	68.831	35	32.401
2	151.000	19	65.703	36	31.068
3	143.420	20	62.735	37	29.798
4	136.260	21	59.918	38	28.586
5	129.510	22	57.244	39	27.471
6	123.130	23	54.705	40	26.328
7	117.100	24	52.292	41	25.275
8	111.400	25	50.000	42	24.471
9	106.020	26	47.821	43	23.311
10	100.930	27	45.749	44	22.395
11	96.112	28	43.780	45	21.520
12	91.553	29	41.901	46	20.683
13	87.236	30	40.120	47	19.883
14	83.148	31	38.123	48	19.119
15	79.276	32	36.805	49	18.388
16	75.607	33	35.265	50	17.688

表 3-7-1 热敏电阻阻值和温度的对应表

例 3-7-3 （SH69P43）ADC 温度检测与电压检测的应用

```
LIST P=69P43
ROMSIZE=3072
;***********************************************
; 系统寄存器
;***********************************************
IE              EQU   00H         ;中断使能标志
IRQ             EQU   01H         ;中断请求标志
TM0             EQU   02H         ;Timer0 模式寄存器
TL0             EQU   04H         ;Timer0 装入/记数寄存器低四位
TH0             EQU   05H         ;Timer0 装入/记数寄存器高四位
TBR             EQU   0EH         ;查表寄存器
INX             EQU   0FH         ;间接寻址伪索引寄存器
DPL             EQU   10H         ;INX 数据指针低四位
DPM             EQU   11H         ;INX 数据指针中三位
DPH             EQU   12H         ;INX 数据指针高三位
VACR            EQU   13H         ;位 2-0:A/D 口配置控制寄存器, 位 3:内部/外部参考电压选择寄存器
ADCONCH         EQU   14H         ;位 2-0:ADC 通道选择寄存器, 位 3:ADC 模块状态寄存器
AD_DTL          EQU   15H         ;ADC 转换结果低四位
```

```
AD_DTH          EQU   16H         ;ADC 转换结果高四位
GOTADC          EQU   17H         ;位 0:设置 A/D 转换时间, 位 2-1:选择 A/D 时钟, Bit3:ADC 状态标志
PACR            EQU   18H         ;PortA 转入/输出状态控制寄存器
;***********************************************
; 用户定义寄存器(Bank0)
;***********************************************
AC_BAK          EQU   30H         ;AC 值备份寄存器
TMP1            EQU   31H         ;临时寄存器
TMP2            EQU   32H
TMP3            EQU   33H
TMP4            EQU   34H
;---------------------------------------
; Used for timer
T1MS_CT         EQU   35H         ;1ms 定时计数器=04H
;---------------------------------------
F_TIMER         EQU   36H         ;bit0=1, 1ms 到
FLAG1           EQU   37H         ;bit0=1, 检测温度, bit2=0, 检测电压
                                  ;bit1=1, 已经采样了前 256 个数据在温度检测中
DET_CT          EQU   38H
;---------------------------------------
VOL_1           EQU   39H         ;电压寄存器
VOL_2           EQU   3AH
VOL_3           EQU   3BH
VOL_4           EQU   3CH
TEMPER_1        EQU   3DH         ;温度寄存器
TEMPER_2        EQU   3EH
;---------------------------------------
TEM_HH          EQU   3FH         ;递推平均滤波法中的数据累加缓冲区
TEM_HL          EQU   40H
TEM_LH          EQU   41H
TEM_LL          EQU   42H
;---------------------------------------
TEM_FCT1        EQU   43H         ;温度检测中，前 256 次采样计数器
TEM_FCT2        EQU   44H
;***********************************************
; 用户定义寄存器(Bank1)
;***********************************************
ADC_DL1         EQU   00H         ;电压检测第一次采样数据
```

```
ADC_DH1         EQU   01H
ADC_DL2         EQU   02H                 ;电压检测第二次采样数据
ADC_DH2         EQU   03H
ADC_DL3         EQU   04H                 ;电压检测第三次采样数据
ADC_DH3         EQU   05H
ADC_DL4         EQU   06H                 ;电压检测第四次采样数据
ADC_DH4         EQU   07H
;***********************************************
; 程序
;***********************************************
          ORG       0000H
          JMP       RESET
          RTNI
          JMP       TIMER0_ISP
          RTNI
          RTNI
;*****************************************
; 子程序:  TIMER0_ISP
; 描述:    TIMER0 中断服务程序
;*****************************************
TIMER0_ISP:
          STA       AC_BAK,00H          ;备份 AC 值
          ANDIM IRQ,1011B               ;清 TIMER0 中断请求标志
J1MS:
          SBIM      T1MS_CT,01H
          BNZ       TIMER0_ISP_END
          LDI       T1MS_CT,04H         ;重置 1ms 计数器
          ORIM      F_TIMER,0001B       ;设置 "1ms 到"标志
TIMER0_ISP_END:
          LDI       IE,0100B            ;打开 TIMER0 中断
          LDA       AC_BAK,00H          ;取出 AC 值
          RTNI
;*****************************************
; 上电程序
;*****************************************
RESET:
          NOP
;-------------------------------------
```

```
;清用户寄存器
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
            LDI         DPH,00H
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TBR,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,04H
            BNZ         POWER_RESET_1
;--------------------------------------
;初始化系统寄存器
SYSTEM_INITIAL:
            ;TIMER0 初始化
            LDI         TM0,07H             ;设置 TIMER0 预分频为/1
            LDI         TL0,06H
            LDI         TH0,00H             ;设置中断时间为 250us
            LDI         T1MS_CT,04H         ;定时 1ms

            ;ADC 初始化
            LDI         PACR,00H            ;设置 PortA 作为输入口
            LDI         VACR,0010B          ;选择内部参考电压 VDD5 伏, PA0 配置为 AN0, PA1 配置为
                                            ;AN1
            LDI         ADCONCH,1001B       ;A/D 转换模块打开, ADC 通道 AN1
            LDI         GOTADC,0110B        ;A/D 时钟 tAD=8tOSC, A/D 转换时间= 50tAD
;--------------------------------------
MAIN_PRE:
            LDI         IRQ,00H
            LDI         IE,0100B            ;打开 Timer0 中断
```

```
;*********************************************
MAIN:
          ADI       F_TIMER,0001B
          BAO       HALTMODE            ;未到 1ms,跳转
          ;1ms 已到
          ANDIMF_TIMER,1110B            ;清 "1ms 到"标志
;*************************************
; 模块: ADC 检测
;*************************************
ADC_PROC:
          LDA       FLAG1,00H
          BAO       ADC_TEMPER          ;检测温度，跳转
;---------------------------------
;检测电压
ADC_VOLTAGE:
          LDI       ADCONCH,1001B       ;ADC 通道 AN1
          ORIM      GOTADC,1000B        ;进行 A/D 转换
          LDA       GOTADC,00H
          BA3       $-1                 ;转换未结束，继续检测
;转换结束
ADC_VOL_1:
          ADIM      DET_CT,01H          ;次数加一
          SBI       DET_CT,04H
          BAZ       ADC_VOL_14
          SBI       DET_CT,03H
          BAZ       ADC_VOL_13
          SBI       DET_CT,02H
          BAZ       ADC_VOL_12
ADC_VOL_11:
          LDA       AD_DTL,00H          ;保存第一次 A/D 转换结果
          STA       ADC_DL1,01H
          LDA       AD_DTH,00H
          STA       ADC_DH1,01H
          JMP       ADC_PROC_END
ADC_VOL_12:
          LDA       AD_DTL,00H          ;保存第二次 A/D 转换结果
          STA       ADC_DL2,01H
          LDA       AD_DTH,00H
```

```
        STA     ADC_DH2,01H
        JMP     ADC_PROC_END
ADC_VOL_13:
        LDA     AD_DTL,00H          ;保存第三次 A/D 转换结果
        STA     ADC_DL3,01H
        LDA     AD_DTH,00H
        STA     ADC_DH3,01H
        JMP     ADC_PROC_END
ADC_VOL_14:
        LDA     AD_DTL,00H          ;保存第四次 A/D 转换结果
        STA     ADC_DL4,01H
        LDA     AD_DTH,00H
        STA     ADC_DH4,01H
        LDI     DET_CT,00H
        ORIM    FLAG1,0001B         ;设置"检测温度"标志
        CALL    CAL_ADCDATA         ;调用防脉冲平均滤波法子程序处理 ADC 数据
ADC_VOL_2:
        ;根据处理后的数据高四位进行查表，得到相应的电压值
        LDI     TBR,0FH
        LDA     ADC_DH4,01H
        CALL    07BFH
        STA     VOL_3,00H
        LDA     TBR,00H
        STA     VOL_4,00H
        LDI     TBR,0EH
        LDA     ADC_DH4,01H
        CALL    07BFH
        STA     VOL_1,00H
        LDA     TBR,00H
        STA     VOL_2,00H
        ;根据处理后的数据低四位进行查表，得到相应的电压值
        LDI     TBR,0DH
        LDA     ADC_DL4,01H
        CALL    07BFH
        STA     TMP3,00H
        LDA     TBR,00H
        STA     TMP4,00H
        LDI     TBR,0CH
```

```
            LDA         ADC_DL4,01H
            CALL        07BFH
            STA         TMP1,00H
            LDA         TBR,00H
            STA         TMP2,00H
            ;将两个电压值相加，得到最终得到的电压值存于 VOL_4,VOL_3,VOL_2,VOL_1
            LDA         TMP1,00H
            ADDM  VOL_1,00H
            DAA         VOL_1
            LDA         TMP2,00H
            ADCM  VOL_2,00H
            DAA         VOL_2
            LDA         TMP3,00H
            ADCM  VOL_3,00H
            DAA         VOL_3
            LDA         TMP4,00H
            ADCM  VOL_4,00H
            JMP         ADC_PROC_END
;--------------------------------
;检测温度
ADC_TEMPER:
            LDI         ADCONCH,1000B          ;ADC 通道 AN0
            ORIM        GOTADC,1000B           ;进行 A/D 转换
            LDA         GOTADC,00H
            BA3         $-1                    ;转换未结束，继续检测
;转换结束
ADC_TEM_1:
            LDA         FLAG1,00H
            BA1         ADC_TEM_3              ;已经采样了前 256 个数据
;还在前 256 个数据采样中，将当前采样数据加入数据累加缓冲区
ADC_TEM_2:
            LDA         AD_DTL,00H
            ADDM  TEM_LL,00H
            LDA         AD_DTH,00H
            ADCM  TEM_LH,00H
            LDI         TBR,00H
            ADCM  TEM_HL,00H
            LDI         TBR,00H
```

```
        ADCM  TEM_HH,00H

        ADIM        TEM_FCT1,01H
        LDI         TBR,00H
        ADCM  TEM_FCT2,00H
        BNC         ADC_PROC_END            ;没到 256 次,跳转
        ;前 256 个数据采样结束
        ORIM        FLAG1,0010B             ;设置"已经采样了前 256 个数据在温度检测中"
        JMP         ADC_TEM_4
ADC_TEM_3:
        ;数据累加缓冲区的数据减去一个平均值
        LDA         TEM_HL,00H
        SUBM  TEM_LL,00H
        LDA         TEM_HH,00H
        SBCM  TEM_LH,00H
        LDI         TBR,00H
        SBCM  TEM_HL,00H
        LDI         TBR,00H
        SBCM  TEM_HH,00H
        ;数据累加缓冲区再加上当前 A/D 转换的数据，得到一个新的 256 采样数据的总和
        LDA         AD_DTL,00H
        ADDM  TEM_LL,00H
        LDA         AD_DTH,00H
        ADCM  TEM_LH,00H
        LDI         TBR,00H
        ADCM  TEM_HL,00H
        LDI         TBR,00H
        ADCM  TEM_HH,00H
ADC_TEM_4:
        ;累加缓冲区除 256，得到 TEM_HH 和 TEM_HL 组成的 8 位数据刚好为平均值
        ANDIMFLAG1,1110B                    ;设置"检测电压"标志
        LDI         TBR,0DH
        SUB         TEM_HL,00H
        LDI         TBR,04H
        SBC         TEM_HH,00H
        BNC         ADC_TEM_ERROR           ;低于 0 摄氏度，错误，跳转

        LDI         TBR,01H
```

```
            SUB         TEM_HL,00H
            LDI         TBR,0DH
            SBC         TEM_HH,00H
            BC          ADC_TEM_ERROR       ;超过 50 摄氏度，错误，跳转
ADC_TEM_5:
            LDI         TBR,05H
            SUB         TEM_HH,00H
            BNC         ADC_TEM_D00         ;平均值<50H,为 00 度

            LDI         TBR,0DH
            SUB         TEM_HH,00H
            BC          ADC_TEM_D50         ;平均值>=D0H,为 50 度
            ;其它由平均值查表得到温度值
            SBI         TEM_HH,02H
            STA         TBR,00H
            LDA         TEM_HL,00H
            CALL        07BFH
            STA         TEMPER_1,00H
            LDA         TBR,00H
            STA         TEMPER_2,00H
            JMP         ADC_PROC_END
ADC_TEM_D00:
            LDI         TEMPER_1,00H
            LDI         TEMPER_2,00H
            JMP         ADC_PROC_END
ADC_TEM_D50:
            LDI         TEMPER_1,00H
            LDI         TEMPER_2,05H
            JMP         ADC_PROC_END
ADC_TEM_ERROR:
            LDI         TEMPER_1,0EH
            LDI         TEMPER_2,0EH
            JMP         ADC_PROC_END        ;出错，温度寄存器中放入"EE"
ADC_PROC_END:
;*****************************************
HALTMODE:
            NOP
            HALT
```

```
            NOP
            NOP
            NOP
            JMP         MAIN
;***********************************************
;*********************************************
; 子程序:  CAL_ADCDATA
; 描述:     防脉冲平均滤波法（N=4，去一个最大值和一个最小值，剩下两个求平均值）
;*********************************************
CAL_ADCDATA:
;----------------------------
;寻找最小值
CAL_AD_MIN12:
            LDA         ADC_DL1,01H
            SUB         ADC_DL2,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH2,01H
            BC          CAL_AD_MIN13          ;D1<D2
;D2<D1
CAL_AD_MIN23:
            LDA         ADC_DL2,01H
            SUB         ADC_DL3,01H
            LDA         ADC_DH2,01H
            SBC         ADC_DH3,01H
            BC          CAL_AD_MIN24          ;D2<D3
;D3<D2
CAL_AD_MIN34:
            LDA         ADC_DL3,01H
            SUB         ADC_DL4,01H
            LDA         ADC_DH3,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MIN3           ;D3<D4
            ;D4<D3
            JMP         CAL_AD_MIN4
;D2<D3
CAL_AD_MIN24:
            LDA         ADC_DL2,01H
            SBI         ADC_DL4,01H
```

```
            LDA         ADC_DH2,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MIN2             ;D2<D4
            ;D4<D2
            JMP         CAL_AD_MIN4
;D1<D2
CAL_AD_MIN13:
            LDA         ADC_DL1,01H
            SUB         ADC_DL3,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH3,01H
            BC          CAL_AD_MIN14            ;D1<D3
            ;D3<D1
            JMP         CAL_AD_MIN34
;D1<D3
CAL_AD_MIN14:
            LDA         ADC_DL1,01H
            SUB         ADC_DL4,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MIN1             ;D1<D4
            ;D4<D1
            JMP         CAL_AD_MIN4
;-----------------------
;将最小值清零
CAL_AD_MIN1:
            LDI         TBR,00H
            STA         ADC_DL1,01H
            STA         ADC_DH1,01H
            JMP         CAL_AD_MAX12
CAL_AD_MIN2:
            LDI         TBR,00H
            STA         ADC_DL2,01H
            STA         ADC_DH2,01H
            JMP         CAL_AD_MAX12
CAL_AD_MIN3:
            LDI         TBR,00H
            STA         ADC_DL3,01H
```

```
            STA         ADC_DH3,01H
            JMP         CAL_AD_MAX12
CAL_AD_MIN4:
            LDI         TBR,00H
            STA         ADC_DL4,01H
            STA         ADC_DH4,01H
            JMP         CAL_AD_MAX12
;----------------------------
;寻找最大值
CAL_AD_MAX12:
            LDA         ADC_DL1,01H
            SUB         ADC_DL2,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH2,01H
            BC          CAL_AD_MAX23          ;D2>D1
;D1>D2
CAL_AD_MAX13:
            LDA         ADC_DL1,01H
            SUB         ADC_DL3,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH3,01H
            BC          CAL_AD_MAX34          ;D3>D1
;D1>D3
CAL_AD_MAX14:
            LDA         ADC_DL1,01H
            SUB         ADC_DL4,01H
            LDA         ADC_DH1,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MAX4           ;D4>D1
            ;D1>D4
            JMP         CAL_AD_MAX1
;D3>D1
CAL_AD_MAX34:
            LDA         ADC_DL3,01H
            SUB         ADC_DL4,01H
            LDA         ADC_DH3,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MAX4           ;D4>D3
```

```
            ;D3>D4
            JMP         CAL_AD_MAX3
;D2>D1
CAL_AD_MAX23:
            LDA         ADC_DL2,01H
            SUB         ADC_DL3,01H
            LDA         ADC_DH2,01H
            SBC         ADC_DH3,01H
            BC          CAL_AD_MAX34        ;D3>D2
;D2>D3
CAL_AD_MAX24:
            LDA         ADC_DL2,01H
            SUB         ADC_DL4,01H
            LDA         ADC_DH2,01H
            SBC         ADC_DH4,01H
            BC          CAL_AD_MAX4         ;D4>D2
            ;D2>D4
            JMP         CAL_AD_MAX2
;-----------------------
;将最大值清零
CAL_AD_MAX1:
            LDI         TBR,00H
            STA         ADC_DL1,01H
            STA         ADC_DH1,01H
            JMP         CAL_AD_ADD
CAL_AD_MAX2:
            LDI         TBR,00H
            STA         ADC_DL2,01H
            STA         ADC_DH2,01H
            JMP         CAL_AD_ADD
CAL_AD_MAX3:
            LDI         TBR,00H
            STA         ADC_DL3,01H
            STA         ADC_DH3,01H
            JMP         CAL_AD_ADD
CAL_AD_MAX4:
            LDI         TBR,00H
            STA         ADC_DL4,01H
```

```
            STA         ADC_DH4,01H
            JMP         CAL_AD_ADD
;-----------------------------
;计算总和并存放在 ADC_DH4T 和 ADC_DL4（包括两个被清零的）
CAL_AD_ADD:
            LDI         TMP4,00H
            LDA         ADC_DL1,01H
            ADDM  ADC_DL2,01H
            LDA         ADC_DH1,01H
            ADCM  ADC_DH2,01H
            LDI         TBR,00H
            ADCM  TMP4,00H

            LDA         ADC_DL2,01H
            ADDM  ADC_DL3,01H
            LDA         ADC_DH2,01H
            ADCM  ADC_DH3,01H
            LDI         TBR,00H
            ADCM  TMP4,00H

            LDA         ADC_DL3,01H
            ADDM  ADC_DL4,01H
            LDA         ADC_DH3,01H
            ADCM  ADC_DH4,01H
            LDI         TBR,00H
            ADCM  TMP4,00H
;-----------------------------
;总和除以 2，得到平均值，存放在 ADC_DH4T 和 ADC_DL4
CAL_AD_DIV:
            LDA         ADC_DL4,01H
            SHR
            STA         ADC_DL4,01H
            LDA         ADC_DH4,01H
            SHR
            STA         ADC_DH4,01H
            BNC         $+3
            LDI         TBR,1000B
            ORM         ADC_DL4,01H
```

```
            LDA         TMP4,00H
            SHR
            BNC         $+3
            LDI         TBR,1000B
            ORM         ADC_DH4,01H
;---------------------------
CAL_ADCDATA_END:
            RTNI
;************************************************
;温度数据表
;************************************************
            ORG         0730H
            RTNW  00H,00H    ;50-00
            RTNW  00H,01H    ;51-01
            RTNW  00H,01H    ;52-01
            RTNW  00H,01H    ;53-01
            RTNW  00H,02H    ;54-02
            RTNW  00H,02H    ;55-02
            RTNW  00H,02H    ;56-02
            RTNW  00H,03H    ;57-03
            RTNW  00H,03H    ;58-03
            RTNW  00H,03H    ;59-03
            RTNW  00H,04H    ;5A-04
            RTNW  00H,04H    ;5B-04
            RTNW  00H,04H    ;5C-04
            RTNW  00H,05H    ;5D-05
            RTNW  00H,05H    ;5E-05
            RTNW  00H,05H    ;5F-05
            ORG         0740H
            RTNW  00H,06H    ;60-06
            RTNW  00H,06H    ;61-06
            RTNW  00H,06H    ;62-06
            RTNW  00H,07H    ;63-07
            RTNW  00H,07H    ;64-07
            RTNW  00H,07H    ;65-07
            RTNW  00H,08H    ;66-08
            RTNW  00H,08H    ;67-08
            RTNW  00H,08H    ;68-08
```

```
RTNW  00H,09H     ;69-09
RTNW  00H,09H     ;6A-09
RTNW  00H,09H     ;6B-09
RTNW  01H,00H     ;6C-10
RTNW  01H,00H     ;6D-10
RTNW  01H,00H     ;6E-10
RTNW  01H,01H     ;6F-11
ORG             0750H
RTNW  01H,01H     ;70-11
RTNW  01H,01H     ;71-11
RTNW  01H,02H     ;72-12
RTNW  01H,02H     ;73-12
RTNW  01H,02H     ;74-12
RTNW  01H,03H     ;75-13
RTNW  01H,03H     ;76-13
RTNW  01H,03H     ;77-13
RTNW  01H,04H     ;78-14
RTNW  01H,04H     ;79-14
RTNW  01H,04H     ;7A-14
RTNW  01H,05H     ;7B-15
RTNW  01H,05H     ;7C-15
RTNW  01H,05H     ;7D-15
RTNW  01H,06H     ;7E-16
RTNW  01H,06H     ;7F-16
ORG             0760H
RTNW  01H,06H     ;80-16
RTNW  01H,07H     ;81-17
RTNW  01H,07H     ;82-17
RTNW  01H,07H     ;83-17
RTNW  01H,08H     ;84-18
RTNW  01H,08H     ;85-18
RTNW  01H,08H     ;86-18
RTNW  01H,09H     ;87-19
RTNW  01H,09H     ;88-19
RTNW  01H,09H     ;89-19
RTNW  02H,00H     ;8A-20
RTNW  02H,00H     ;8B-20
RTNW  02H,00H     ;8C-20
```

```
RTNW  02H,01H     ;8D-21
RTNW  02H,01H     ;8E-21
RTNW  02H,01H     ;8F-21
ORG           0770H
RTNW  02H,02H     ;90-22
RTNW  02H,02H     ;91-22
RTNW  02H,02H     ;92-22
RTNW  02H,03H     ;93-23
RTNW  02H,03H     ;94-23
RTNW  02H,03H     ;95-23
RTNW  02H,04H     ;96-24
RTNW  02H,04H     ;97-24
RTNW  02H,04H     ;98-24
RTNW  02H,05H     ;99-25
RTNW  02H,05H     ;9A-25
RTNW  02H,05H     ;9B-25
RTNW  02H,06H     ;9C-26
RTNW  02H,06H     ;9D-26
RTNW  02H,07H     ;9E-27
RTNW  02H,07H     ;9F-27
ORG           0780H
RTNW  02H,07H     ;A0-27
RTNW  02H,08H     ;A1-28
RTNW  02H,08H     ;A2-28
RTNW  02H,09H     ;A3-29
RTNW  02H,09H     ;A4-29
RTNW  02H,09H     ;A5-29
RTNW  03H,00H     ;A6-30
RTNW  03H,00H     ;A7-30
RTNW  03H,00H     ;A8-30
RTNW  03H,01H     ;A9-31
RTNW  03H,01H     ;AA-31
RTNW  03H,02H     ;AB-32
RTNW  03H,02H     ;AC-32
RTNW  03H,03H     ;AD-33
RTNW  03H,03H     ;AE-33
RTNW  03H,04H     ;AF-34
ORG           0790H
```

```
        RTNW  03H,04H     ;B0-34
        RTNW  03H,04H     ;B1-34
        RTNW  03H,05H     ;B2-35
        RTNW  03H,05H     ;B3-35
        RTNW  03H,06H     ;B4-36
        RTNW  03H,06H     ;B5-36
        RTNW  03H,07H     ;B6-37
        RTNW  03H,07H     ;B7-37
        RTNW  03H,08H     ;B8-38
        RTNW  03H,08H     ;B9-38
        RTNW  03H,09H     ;BA-39
        RTNW  03H,09H     ;BB-39
        RTNW  04H,00H     ;BC-40
        RTNW  04H,00H     ;BD-40
        RTNW  04H,01H     ;BE-41
        RTNW  04H,01H     ;BF-41
        ORG           07A0H
        RTNW  04H,02H     ;C0-42
        RTNW  04H,02H     ;C1-42
        RTNW  04H,03H     ;C2-43
        RTNW  04H,03H     ;C3-43
        RTNW  04H,04H     ;C4-44
        RTNW  04H,04H     ;C5-44
        RTNW  04H,05H     ;C6-45
        RTNW  04H,05H     ;C7-45
        RTNW  04H,06H     ;C8-46
        RTNW  04H,06H     ;C9-46
        RTNW  04H,07H     ;CA-47
        RTNW  04H,08H     ;CB-48
        RTNW  04H,08H     ;CC-48
        RTNW  04H,09H     ;CD-49
        RTNW  04H,09H     ;CE-49
        RTNW  05H,00H     ;CF-50
;************************************************
        ORG           07BFH
        TJMP
;************************************************
; 电压数据表
```

```
;***********************************************
        ORG         07C0H
        RTNW  01H,00H    ;00-0.010
        RTNW  02H,09H    ;01-0.029
        RTNW  04H,09H    ;02-0.049
        RTNW  06H,08H    ;03-0.068
        RTNW  08H,08H    ;04-0.088
        RTNW  00H,07H    ;05-0.107
        RTNW  02H,07H    ;06-0.127
        RTNW  04H,06H    ;07-0.146
        RTNW  06H,06H    ;08-0.166
        RTNW  08H,05H    ;09-0.185
        RTNW  00H,05H    ;0A-0.205
        RTNW  02H,04H    ;0B-0.224
        RTNW  04H,04H    ;0C-0.244
        RTNW  06H,03H    ;0E-0.263
        RTNW  08H,03H    ;0E-0.283
        RTNW  00H,03H    ;0F-0.303
        ORG         07D0H
        RTNW  00H,00H    ;00-0.010
        RTNW  00H,00H    ;01-0.029
        RTNW  00H,00H    ;02-0.049
        RTNW  00H,00H    ;03-0.068
        RTNW  00H,00H    ;04-0.088
        RTNW  00H,01H    ;05-0.107
        RTNW  00H,01H    ;06-0.127
        RTNW  00H,01H    ;07-0.146
        RTNW  00H,01H    ;08-0.166
        RTNW  00H,01H    ;09-0.185
        RTNW  00H,02H    ;0A-0.205
        RTNW  00H,02H    ;0B-0.224
        RTNW  00H,02H    ;0C-0.244
        RTNW  00H,02H    ;0D-0.263
        RTNW  00H,02H    ;0E-0.283
        RTNW  00H,03H    ;0F-0.303
        ORG         07E0H
        RTNW  00H,00H    ;00-0.000
        RTNW  01H,03H    ;10-0.313
```

```
RTNW  02H,05H     ;20-0.625
RTNW  03H,08H     ;30-0.938
RTNW  05H,00H     ;40-1.250
RTNW  06H,03H     ;50-1.563
RTNW  07H,05H     ;60-1.875
RTNW  08H,08H     ;70-2.188
RTNW  00H,00H     ;80-2.500
RTNW  01H,03H     ;90-2.813
RTNW  02H,05H     ;A0-3.125
RTNW  03H,08H     ;B0-3.438
RTNW  05H,00H     ;C0-3.750
RTNW  06H,03H     ;D0-4.063
RTNW  07H,05H     ;E0-4.375
RTNW  08H,08H     ;F0-4.688
ORG           07F0H
RTNW  00H,00H     ;00-0.000
RTNW  00H,03H     ;10-0.313
RTNW  00H,06H     ;20-0.625
RTNW  00H,09H     ;30-0.938
RTNW  01H,02H     ;40-1.250
RTNW  01H,05H     ;50-1.563
RTNW  01H,08H     ;60-1.875
RTNW  02H,01H     ;70-2.188
RTNW  02H,05H     ;80-2.500
RTNW  02H,08H     ;90-2.813
RTNW  03H,01H     ;A0-3.125
RTNW  03H,04H     ;B0-3.438
RTNW  03H,07H     ;C0-3.750
RTNW  04H,00H     ;D0-4.063
RTNW  04H,03H     ;E0-4.375
RTNW  04H,06H     ;F0-4.688

END
```

3.8 脉冲宽度调制器(Pulse Width Modulator)

PWM(Pulse Width Modulator)即为脉冲宽度调制。中颖公司 SH6xxx 产品线中很多产品已经集成了多种特性的 PWM 模块,其 PWMx 管脚可以输出占空比分辨率从 6 位到 8 位和 10 位分别可调的方波,满足不同系统设计的需求。所谓占空比是指在一个方波周期内有效电平(高电平或低电平)的宽度(脉宽)。在这一章中我们将以最普通的 8+2 位分辨率(即 8 位分辨率加 2 位微调) PWM 模块为基础,从应用角度讲解在一个控制系统中如何合理的使用片上 PWM 功能。

3.8.1 SH6xxx单片机片上PWM模块综述

SH6xxx 系列单片机的片上 PWM 模块通常由三个部分组成，即输出方波的周期控制，方波占空比的调整控制以及 PWM 的系统时钟和占空比有效电平的选择控制。不同的芯片上提供的 PWM 输出通道数量不同。PWM 输出的高低电平并非来自于普通端口寄存器的锁存值，而是直接由 PWM 模块输出。一般 PWMx 管脚和某个端口管脚复用的,可以通过软件配置选择。软件一旦选择 PWM 输出管脚,不论端口输入/输出寄存器设置如何,系统将自动切换至 PWM 模块输出。

3.8.2 PWM相关控制寄存器介绍

PWM 模块最重要的寄存器是控制寄存器(PWM Control Register，以下接称 PWMC) ,还有就是 PWM 周期控制寄存器(PWMP)和 PWM 占空比控制寄存器(PWMD)。

PWM 控制寄存器(PWMC)

PWM 控制寄存器位于内部 RAM 零页的地址 20H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$20	PWMS	TCK1	TCK0	PWM	R/W	

PWM 控制寄存器的数据位定义

TCK1～0: PWM 时钟周期选择。

00: PWM clock = t_{osc},即时钟周期源自芯片的 1 倍主振荡周期。

01: PWM clock = $2t_{osc}$,即时钟周期源自芯片的 2 倍主振荡周期。

10: PWM clock = $4t_{osc}$,即时钟周期源自芯片的 4 倍主振荡周期。

11: PWM clock = $8t_{osc}$,即时钟周期源自芯片的 8 倍主振荡周期。

PWM: PWM 模块使能选择和 PWM 管脚/普通端口管脚配置选择控制。

0 = PWM 模块禁止，对应的管脚作为普通端口管脚使用。

1 = PWM 模块使能，对应的管脚作为 PWM 输出管脚使用。

PWMS: 占空比有效电平选择

0 = 高电平为有效电平

1 = 低电平为有效电平

PWM 周期控制寄存器(PWMP)

PWM 周期控制寄存器位于内部 RAM 零页的地址 021H ～ 022H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$21	PP.3	PP.2	PP.1	PP.0	R/W	PWM period low nibble
$22	PP.7	PP.6	PP.5	PP.4	R/W	PWM period high nibble

PWM 输出方波的周期 = [PP.7, PP.0] x PWM clock

如果 PWMS 选择高电平为有效电平，当 [PP.7, PP.0]= 00H, PWM 模块输出低电平。

如果 PWMS 选择低电平为有效电平，当 [PP.7, PP.0]= 00H, PWM 模块输出高电平。

PWM 占空比控制寄存器(PWMD)

PWM 占空比控制寄存器位于内部 RAM 零页的地址 024H ～ 026H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$24	-	-	PDF.1	PDF.0	R/W	PWM duty fine tune nibble
$25	PD.3	PD.2	PD.1	PD.0	R/W	PWM duty low nibble
$26	PD.7	PD.6	PD.5	PD.4	R/W	PWM duty high nibble

PWM 输出占空比的宽度 = ([PD.7, PD.0] + [PDF.1, PDF.0] / 4) x PWM clock

如果 PWMS 选择高电平为有效电平，当 [PP.7, PP.0] ≤ [PD.7, PD.0], PWM 模块输出高电平。

如果 PWMS 选择低电平为有效电平，当 [PP.7, PP.0] ≤ [PD.7, PD.0], PWM 模块输出低电平。

关于占空比的宽度微调控制，模块内部利用两位寄存器控制每 4 个 PWM 输出方波为一组信号，00B ～ 11B 分别决定一组信号中出现占空比为[PD.7, PD.0] +1 的方波依次有 0，1，2，3 个。从而实现一组信号中占空比平均值的微调。

PWM 占空比的宽度微调寄存器(Duty Fine Tune Register)位于内部 RAM 零页的地址 024H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$24	-	-	PDF.1	PDF.0	R/W	

PWM Duty Fine Tune Register 寄存器的数据位定义

PDF1～0: PWM 占空比微调选择

00: 即在一组信号（4 个 PWM 输出方波）中，占空比为[PD.7, PD.0]的方波位于第 0 个/第 1 个/第 2 个/第 3 个周期位置。

01： 即在一组信号（4 个 PWM 输出方波）中，占空比为[PD.7, PD.0]的方波位于第 1 个/第 2 个/第 3 个周期位置，占空比为[PD.7, PD.0]+1 的方波位于第 0 个周期位置。

10： 即在一组信号（4 个 PWM 输出方波）中，占空比为[PD.7, PD.0]的方波位于第 2 个/第 3 个周期位置，占空比为[PD.7, PD.0]+1 的方波位于第 0 个/第 1 个周期位置。

11： 即在一组信号（4 个 PWM 输出方波）中，占空比为[PD.7, PD.0]的方波位于第 3 个周期位置，占空比为[PD.7, PD.0]+1 的方波位于第 0 个/第 1 个/第 2 个周期位置。

占空比的宽度微调具体变化如下图 3-8-1 所示。

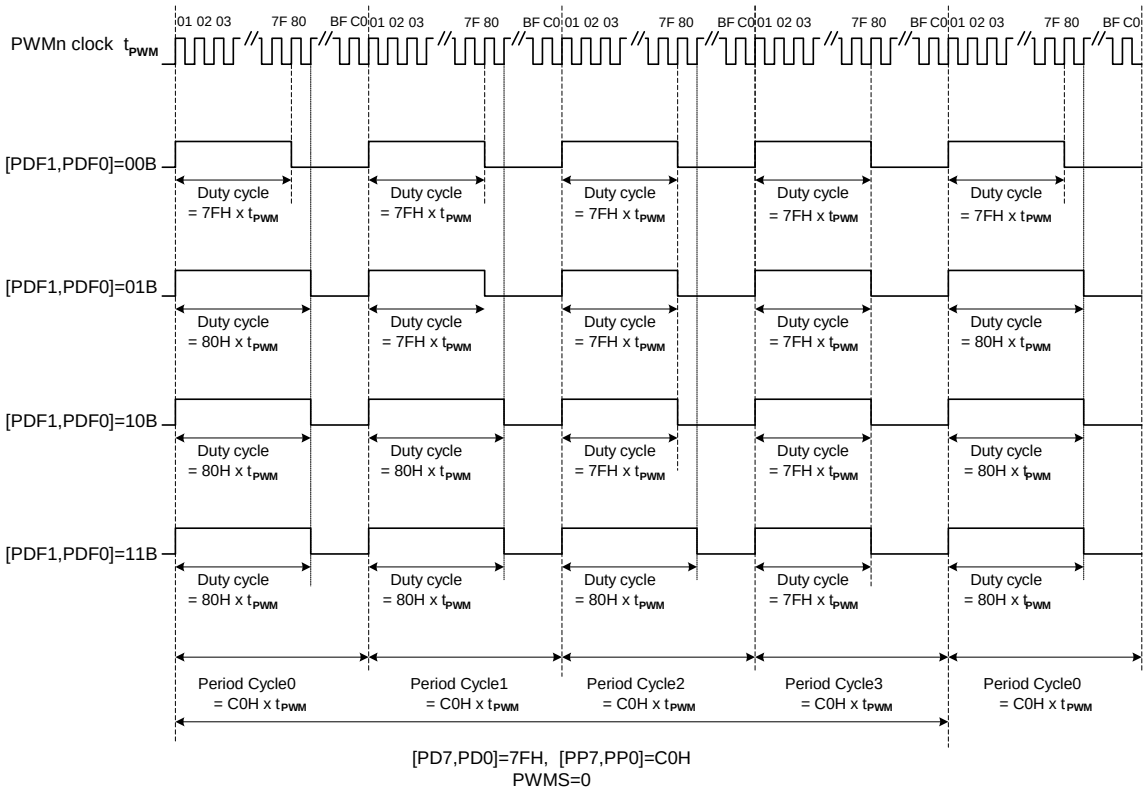


图 3-8-1 PWM 调节波形示意图

3.8.3 PWM模块工作模式设定说明

按照下面所属的操作步骤，一般都能实现所需的 PWM 波形输出。

(1) 首先设定 PWM 周期控制寄存器(PWMP),决定 PWM 方波的周期。

(2) 根据 PWM 方波的周期选择合适的 PWM clock。

(3) 根据选择的 PWM clock 设定所需的占空比（即有效脉宽时间），即设置 PWM 占空比控制寄存器(PWMD)。

(4) 视实际情况决定占空比微调控制寄存器设定，（默认值为 00B）。

(5) 最后选择 PWM 输出有效电平并使能模块工作，即设置 PWM 控制寄存器(PWMC)。

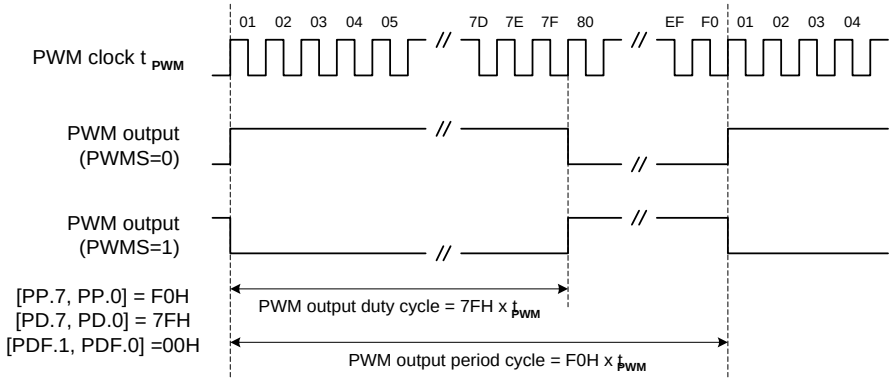


图 3-8-2 PWM 输出有效电平示意图

(6) 程序在运行过程中的任何时候都可以修改 PWM 的周期或者占空比。PWM 的周期改写顺序是先写低位寄存器，再写高位寄存器。只有高位寄存器写操作才能真正刷新高/低寄存器的值,而且必须等到当前 PWM 周期结束,才可开始一个新的 PWM 周期。同理，PWM 的占空比改写顺序也是先写低位寄存器（如果微调寄存器需要修改，也可先改），最后改写高位寄存器。而且必须等到当前 PWM 周期结束,才可开始一个新占空比的 PWM 周期。（详情请参考下图所示）

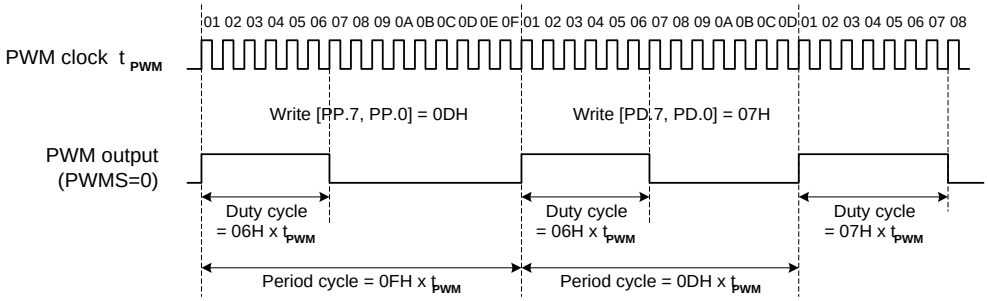


图 3-8-3 PWM 输出周期和占空比示意图

(7) 程序在运行过程中的任何时候都可以读取 PWM 的周期或者占空比。PWM 周期控制寄存器(PWMP)的读取顺序是先读高位寄存器，再读低位寄存器。同理，PWM 占空比控制寄存器(PWMD) 的读取顺序也是先读高位寄存器,再读低位寄存器。

(8) PWM模块可以在单片机进入HALT模式下依然工作,但如果单片机系统进入STOP模式，因其振荡器被关闭而使 PWM 模块无法工作。

3.8.4 PWM实现模/数转换（DAC）应用实例

PWM 即为脉冲宽度调制，从本质上来讲说是把数字信号转换为模拟信号，其一个周

期内输出高电平的比例决定其输出平均电压的高低。PWM 通常用来做 DAC（数/模转换），对电机、风扇、蜂鸣器等的控制。

■ 实现原理

单片机输出的 PWM 信号通过积分（即低通滤波）后，即可得到与 PWM 占空比成正比的模拟电压，这样，通过修改 PWM 的占空比就可以很容易的改变外部的模拟电压，从面实现从数字量到模拟量的转换（DAC）。

PWM 输出实现 D/A 具有成本低及实现方便的优点，输出的模拟电压噪声比专用 DAC 芯片大的缺点，因此可以适用于一些不需要精密电压输出的场合。

下面是用 SH69P42 设计的用 PWM0 的信号实现 DAC 的简单电路。

■ 电路图(如图 3-8-4)

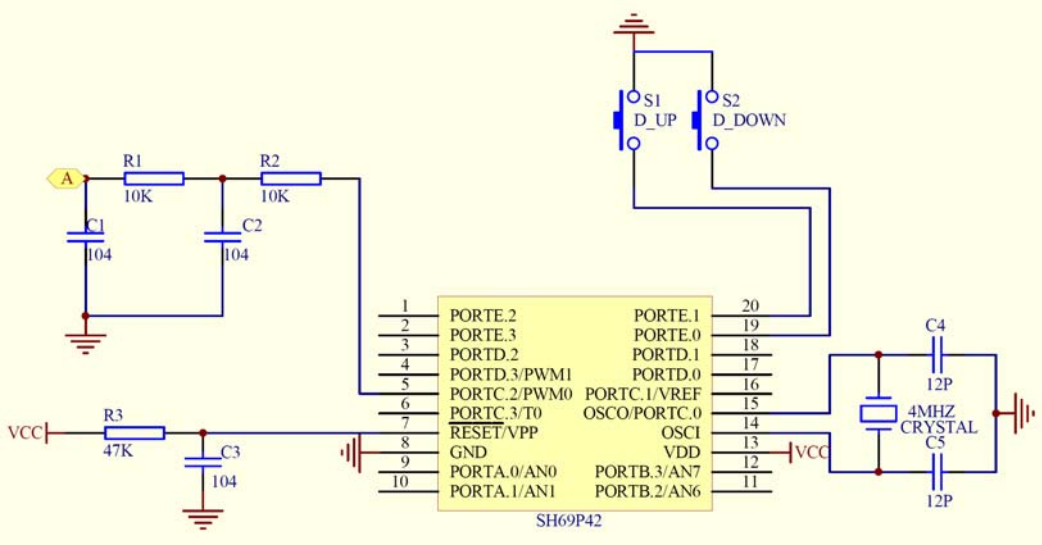


图 3-8-4 PWM 应用实例原理图

电路采用 5V 直流电源，4M 晶振，2 个按键分别为：

D_UP：增大 PWM 占空比按键

D_DOWN：减小 PWM 占空比按键

PWM0 的周期固定为 3F0H，PWM0 的占空比初始为 000H，每次按键调整 010H，当达到最大值时，再按 D_UP 键则无效；当达到最小值时，再按 D_DOWN 键则无效。

PWM0 输出经过两阶 RC 低通滤波，然后接到 A 点，通过按键调整 PWM 的占空比，可以调整 A 点的电压，使 A 点的电压从 0V 到 5V。

当 PWM0 输出高电平时，通过电阻 R1 和 R2 对 C1 和 C2 两个电容进行充电，使 A 点的电压升高；当 PWM0 输出低电平时，两电容 C1 和 C2 通过电阻 R1 和 R2 进行放电，使 A 点的电压降低；这样当 PWM0 输出时，A 点的电压就表现为在一个电压值上下波动，波动的幅度与两电阻及两电容的值有关。

下图 3-8-5 是当 PWM 的周期为 3FOH，占空比为 0FOH 时的波型图，图中蓝色线为原理图中 A 点的波型，绿色线为 PWM 口输出的波型，此时 A 点的电压约为 1.20V，与理论值 1.19V（0FOH×5÷3FOH）基本一致；此时 PWM 的占空比值为 480 微秒，与理论值（0FOH×8Tosc=240×2us=480us）一致；周期为 2.016 毫秒，与理论值（3FOH×8Tosc=1008×2us=2016us）一致。

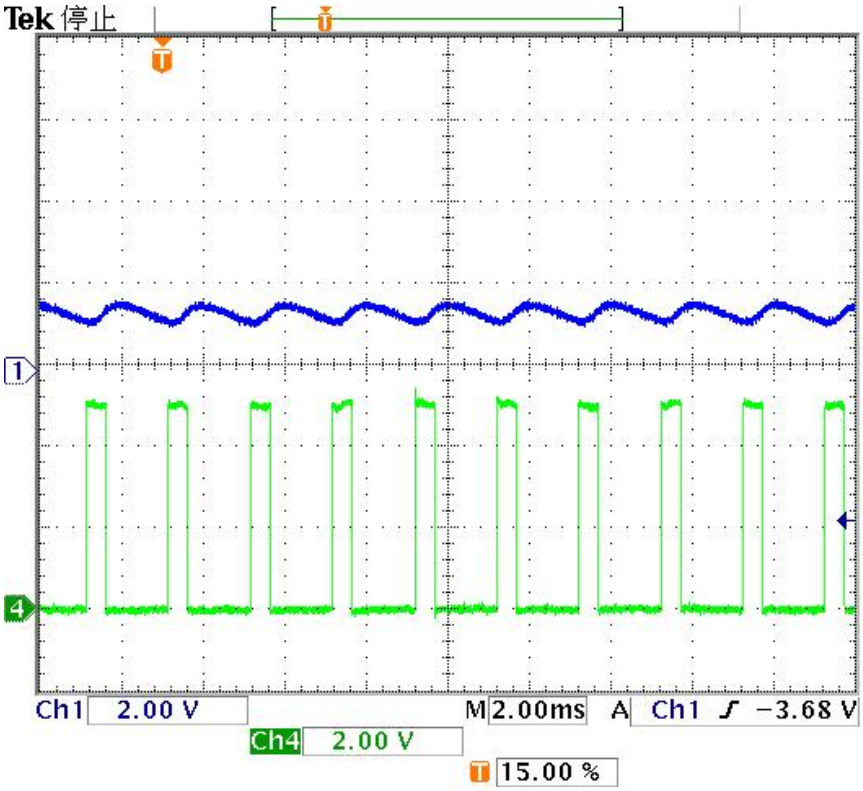


图 3-8-5 PWM 输出波形图

例 3-8-1 PWM 实现模/数转换（DAC）

```
;*************************************************************
;               系统寄存器的定义
;*************************************************************
IE              EQU     00H         ;中断使能标志
IRQ             EQU     01H         ;中断请求标志
PORTE           EQU     0CH         ;PORT E 数据寄存器
PEOUT           EQU     1CH         ;PORT E 输入/输出状态控制寄存器
WDT             EQU     1FH         ;看门狗定时器
PWMOSET         EQU     20H
                                    ;BIT3:PWMO 占空模式输出设置
                                    ;BIT2～1:PWMO 时钟设置
                                    ;BIT0:PWMO 输出选择
```

```
P0PDL           EQU         22H         ;PWM0 周期低四位
P0PDM           EQU         23H         ;PWM0 周期中四位
P0PDH           EQU         24H         ;PWM0 周期高二位
P0DDL           EQU         25H         ;PWM0 占空比低四位
P0DDM           EQU         26H         ;PWM0 占空比中四位
P0DDH           EQU         27H         ;PWM0 占空比高二位
;************************************************************
;               用户寄存器的定义
;************************************************************
TEMP                EQU         30H         ;临时寄存器
FLAG                EQU         31H         ;按键已处理标志
DELAY_TIMER0        EQU         32H         ;延时用寄存器 0
DELAY_TIMER1        EQU         33H         ;延时用寄存器 1
DELAY_TIMER2        EQU         34H         ;延时用寄存器 2
CLEAR_AC            EQU         35H         ;清除累加器 AC 值用寄存器
;************************************************************
;                   向量地址区域
;************************************************************
    ORG         00H
    JMP         RESET           ;跳转到复位服务子程序的入口地址
    RTNI                        ;跳转到 ADC 中断服务程序
    RTNI                        ;定时器 0 服务子程序的入口地址
    RTNI                        ;定时器 1 服务子程序的入口地址
    RTNI                        ;端口中断服务子程序的入口地址
;************************************************************
;                   程序初始化部分
;************************************************************
RESET:
    LDI         IE,00H                  ;清除中断使能标志
    LDI         PEOUT,00H               ;设置 PE 口作为输入口
    CALL        PWM0_OUTPUT_SET         ;设置 PWM0 的输出方式,时钟,输出周期及占空比
    LDI         WDT,0100B               ;设置看门狗定时器的定时时间为 64 毫;秒
                                        ;客户可根据需要自行设定看门狗定;时器的定时时间
    LDI         IRQ,00H                 ;清除中断请求标志
    LDI         TEMP,00H                ;初始化 TEMP 寄存器
    LDI         FLAG,00H                ;初始化 FLAG 寄存器
    LDI         DELAY_TIMER0,00H        ;初始化 DELAY_TIMER0 寄存器
    LDI         DELAY_TIMER1,00H        ;初始化 DELAY_TIMER1 寄存器
```

```
    LDI     DELAY_TIMER2,00H        ;初始化 DELAY_TIMER2 寄存器
    LDI     CLEAR_AC,00H            ;初始化 CLEAR_AC 寄存器

    JMP     MAIN_LOOP_PART          ;跳转到主程序部分
;************************************************************
;               主程序部分
;************************************************************
MAIN_LOOP_PART:
    LDI     WDT,0100B               ;复位看门狗定时器
    CALL    KEY_CHECK_PROCESS       ;按键扫描及处理子程序
    JMP     MAIN_LOOP_PART
;************************************************************
;             按键扫描及处理部分
;************************************************************
KEY_CHECK_PROCESS:
    LDI     PEOUT,00H               ;设置 PE 口为输入口
KEY_CHECK:
    LDI     PORTE,0FH               ;打开 PE 口内部上拉电阻
    CALL    DELAY_5MS               ;消除按键抖动
    LDA     PORTE,00H               ;读取 PE 口状态
    STA     TEMP,00H                ;把 PE 口状态存到 TEMP 寄存器中
    LDI     PEOUT,00H               ;设置 PE 口为输入口
    LDI     PORTE,0FH               ;打开 PE 口内部上拉电阻
    CALL    DELAY_5MS               ;消除按键抖动
    LDA     PORTE,00H               ;读取 PE 口状态
    SUB     TEMP,00H                ;比较读取 PE 口状态值，不相等则错误
    BNZ     KEY_ERROR
    LDI     PEOUT,00H               ;设置 PE 口为输入口
    LDI     PORTE,0FH               ;打开 PE 口内部上拉电阻
    CALL    DELAY_5MS               ;消除按键抖动
    LDA     PORTE,00H               ;读取 PE 口状态
    SUB     TEMP,00H                ;比较读取 PE 口状态值，不相等则错误
    BNZ     KEY_ERROR
    SBI     TEMP,1111B              ;没有按键按下检测
    BNZ     DETAIL_KEY_CHECK
    LDI     FLAG ,00H               ;清除按键已处理标志
    JMP     KEY_CHECK_PROCESS_OVER
DETAIL_KEY_CHECK:                   ;根据键值检查具体的按键
```

```
        SBI         FLAG ,0FH                   ;检查按键已处理标志
        BAZ         KEY_CHECK_PROCESS_OVER
        LDI         FLAG ,0FH                   ;设置按键已处理标志
        SBI         TEMP,1110B
        BAZ         DUTY_DOWN_KEY               ;减小占空比键
        SBI         TEMP,1101B
        BAZ         DUTY_UP_KEY                 ;增大占空比键
        JMP         KEY_ERROR
DUTY_DOWN_KEY:                                  ;减小 PWM 占空比 10H
        SBIM        PODDM,01H
        LDI         CLEAR_AC,00H                ;清除累加器 AC 的值
        SBCM        PODDH,00H                       ;带进位减
        BC          KEY_CHECK_PROCESS_OVER
        CALL        RESET_PWM_DUTY              ;不够减，则重设 PWM 占空比的值
        JMP         KEY_CHECK_PROCESS_OVER
DUTY_UP_KEY:                                    ;增大 PWM 占空比 10H
        SBI         PODDH,03H                   ;判断是否到达最大值
        BNZ         DUTY_UP_KEY1
        SBI         PODDM,0FH
        BNZ         DUTY_UP_KEY1
        JMP         KEY_CHECK_PROCESS_OVER      ;到达最大值，则不增加 PWM 占空比的值
DUTY_UP_KEY1:
        ADIM        PODDM,01H
        LDI         CLEAR_AC,00H                ;清除累加器 AC 的值
        ADCM        PODDH,00H                   ;带进位加
        JMP         KEY_CHECK_PROCESS_OVER
KEY_ERROR:                                      ;错误键值处理
        JMP         KEY_CHECK_PROCESS_OVER
KEY_CHECK_PROCESS_OVER:                         ;按键扫描及处理结束，返回
        LDI         POPDL,00H                   ;设置 PWM0 周期低 4 位(更新周期数据)
        LDI         PODDL,00H                   ;设置 PWM0 占空比低 4 位(更新占空比数据)
        RTNI
;************************************************************
;           PWM0 周期与占空比设置子程序
;************************************************************
PWM0_OUTPUT_SET:
        LDI         PWM0SET,0110B               ;设置 PWM0 为正向占空比,PWM0 时
                                                ;钟为 8Tosc,关闭 PWM0 输出,置为 I/O
```

```
        LDI         P0PDH,03H                   ;设置 PWM0 周期高 2 位
        LDI         P0PDM,0FH                   ;设置 PWM0 周期中 4 位
        LDI         P0PDL,00H                   ;设置 PWM0 周期低 4 位
        CALL        RESET_PWM_DUTY              ;设置 PWM0 占空比
        ORIM        PWM0SET,0001B               ;选择 PWM0 输出
        RTNI
;************************************************************
;           PWM0 占空比重新设置子程序
;************************************************************
RESET_PWM_DUTY:
        LDI         P0DDH,00H                   ;设置 PWM0 占空比高 2 位
        LDI         P0DDM,00H                   ;设置 PWM0 占空比中 4 位
        LDI         P0DDL,00H                   ;设置 PWM0 占空比低 4 位
        RTNI
;************************************************************
;           延时 5 毫秒子程序
;************************************************************
DELAY_5MS:
        LDI         DELAY_TIMER2,03H            ;设置初始值
        LDI         DELAY_TIMER1,03H
        LDI         DELAY_TIMER0,0CH
DELAY_5MS_LOOP:
        SBIM        DELAY_TIMER0,01H            ;每次减 1
        LDI         CLEAR_AC    ,00H
        SBCM        DELAY_TIMER1,00H
        LDI         CLEAR_AC    ,00H
        SBCM        DELAY_TIMER2,00H
        BC          DELAY_5MS_LOOP
        RTNI
        END                                     ;程序结束
```

PWM 驱动直流电机

a. 实现原理

单片机输出的 PWM 信号通过低通滤波后，即可得到与 PWM 占空比成正比的模拟电压，再推动三极管 8050，这样，通过修改 PWM 的占空比就可以很容易的改变外部的模拟电压，从面改变直流电机的转速。

下面是用 SH69P42 设计的用 PWM0 的推动直流电机的电路。

b. 电路图如图 3-8-6:

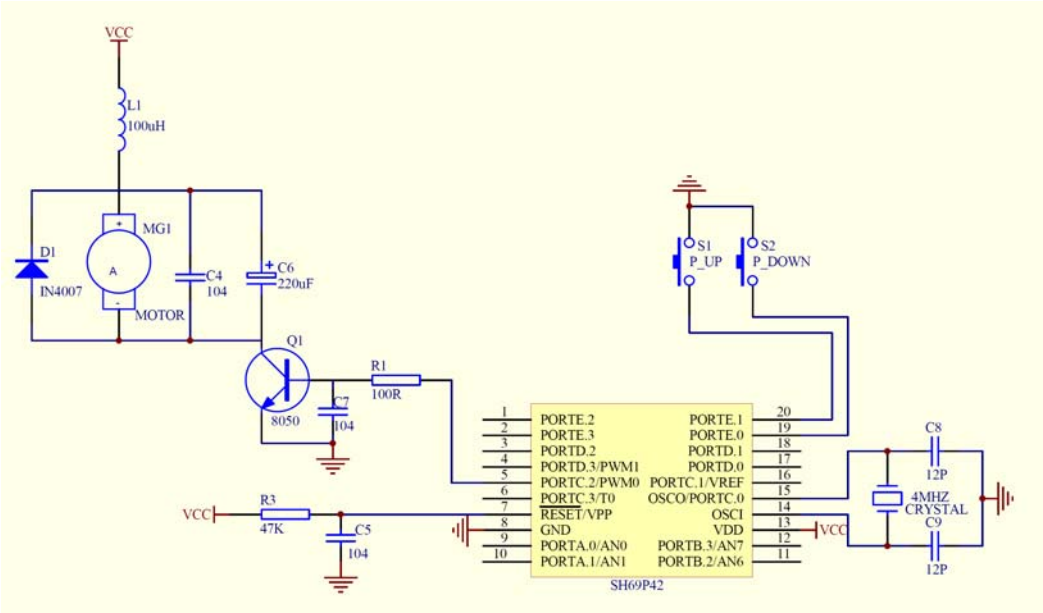


图 3-8-6 PWM 驱动直流电机实例原理图

电路采用 5V 直流电源，4M 晶振，2 个按键分别为：

P_UP：增大 PWM 周期按键

P_DOWN：减小 PWM 周期按键

PWM0 的占空比固定为 050H，PWM0 的周期初始为 03FH，每次按键调整 010H，当达到最大值 3F0H 时，再按 P_UP 键则无效；当达到 000H 时，再按 P_DOWN 键则无效。

当 PWM0 输出高电平时，通过 R1 对电容 C7 进行充电，当电压达到三极管 8050 的开启电压时， 8050 导通，此时 C4 和 C6 进行充电，电机开始工作；当 PWM0 输出低电平时，电容 C4 通过电阻 R1 进行放电，当电压低于三极管 8050 的开启电压时，8050 截止，此时 C4 和 C6 通过电机进行放电，当电压低于电机的工作电压时，电机停止工作。

因为电机在工作时对电源的干扰比较强，所以此电路中增加电感 L1 来抑制其对电源的干扰；电路中的二极管 D1 起保护作用。

下图 3-8-7 是当 PWM 的周期为 3F0H，占空比为 1F0H 时的波型图，图中蓝色线为直流电机正极的波型，绿色线为 PWM 口输出的波型，此时 PWM 的占空比值为 992 微秒，与理论值（1F0H×8Tosc=496×2us=992us）一致；周期为 2.016 毫秒，与理论值（3F0H×8Tosc=1008×2us=2016us）一致。

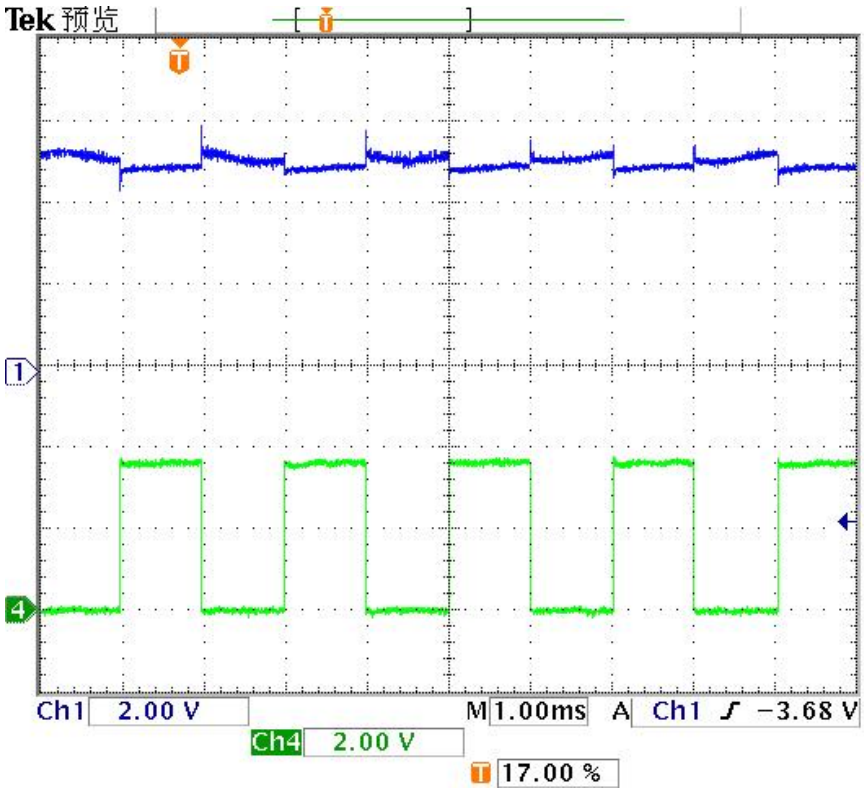


图 3-8-7 PWM 驱动直流电机实例输出波形图

程序设计

例 3-8-2：

```
;************************************************************
;                系统寄存器的定义
;************************************************************
IE              EQU     00H         ;中断使能标志
IRQ             EQU     01H         ;中断请求标志
PORTE           EQU     0CH         ;PORT E 数据寄存器
PEOUT           EQU     1CH         ;PORT E 输入/输出状态控制寄存器
WDT             EQU     1FH         ;看门狗定时器
PWM0SET         EQU     20H         ;BIT3:PWM0 占空模式输出设置
                                    ;BIT2～1:PWM0 时钟设置
                                    ;BIT0:PWM0 输出选择
P0PDL           EQU     22H         ;PWM0 周期低四位
P0PDM           EQU     23H         ;PWM0 周期中四位
P0PDH           EQU     24H         ;PWM0 周期高二位
P0DDL           EQU     25H         ;PWM0 占空比低四位
P0DDM           EQU     26H         ;PWM0 占空比中四位
P0DDH           EQU     27H         ;PWM0 占空比高二位
```

```
;************************************************************
;                用户寄存器的定义
;************************************************************
TEMP                EQU         30H   ;临时寄存器
FLAG                EQU         31H   ;按键已处理标志
DELAY_TIMER0        EQU         32H   ;延时用寄存器 0
DELAY_TIMER1        EQU         33H   ;延时用寄存器 1
DELAY_TIMER2        EQU         34H   ;延时用寄存器 2
CLEAR_AC            EQU         35H   ;清除累加器 AC 值用寄存器
;************************************************************
;                   向量地址区域
;************************************************************
    ORG         00H
    JMP         RESET                 ;跳转到复位服务子程序的入口地址
    RTNI                              ;跳转到 ADC 中断服务程序
    RTNI                              ;定时器 0 服务子程序的入口地址
    RTNI                              ;定时器 1 服务子程序的入口地址
    RTNI                              ;端口中断服务子程序的入口地址
;************************************************************
;                   程序初始化部分
;************************************************************
RESET:
    LDI         IE,00H                ;清除中断使能标志
    LDI         PEOUT,00H             ;设置 PE 口作为输入口
    CALL        PWM0_OUTPUT_SET       ;设置 PWM0 的输出方式,时钟,输出周期及占空比
    LDI         WDT,0100B             ;设置看门狗定时器的定时时间为 64 毫秒,客户可根据需要
                                      ;自行设定看门狗定时器的定时时间
    LDI         IRQ,00H                ;清除中断请求标志
    LDI         TEMP,00H               ;初始化 TEMP 寄存器
    LDI         FLAG ,00H              ;初始化 FLAG 寄存器
    LDI         DELAY_TIMER0,00H       ;初始化 DELAY_TIMER0 寄存器
    LDI         DELAY_TIMER1,00H       ;初始化 DELAY_TIMER1 寄存器
    LDI         DELAY_TIMER2,00H      ;初始化 DELAY_TIMER2 寄存器
    LDI         CLEAR_AC,00H          ;初始化 CLEAR_AC 寄存器

    JMP         MAIN_LOOP_PART        ;跳转到主程序部分
;************************************************************
;                主程序部分
```

```
;*************************************************************
MAIN_LOOP_PART:
      LDI         WDT,0100B              ;复位看门狗定时器
      CALL        KEY_CHECK_PROCESS      ;按键扫描及处理子程序
      JMP         MAIN_LOOP_PART
;*************************************************************
;                 按键扫描及处理部分
;*************************************************************
KEY_CHECK_PROCESS:
      LDI         PEOUT,00H              ;设置 PE 口为输入口
KEY_CHECK:
      LDI         PORTE,0FH              ;打开 PE 口内部上拉电阻
      CALL        DELAY_5MS              ;消除按键抖动
      LDA         PORTE,00H              ;读取 PE 口状态
      STA         TEMP,00H               ;把 PE 口状态存到 TEMP 寄存器中
      LDI         PEOUT,00H              ;设置 PE 口为输入口
      LDI         PORTE,0FH              ;打开 PE 口内部上拉电阻
      CALL        DELAY_5MS              ;消除按键抖动
      LDA         PORTE,00H              ;读取 PE 口状态
      SUB         TEMP,00H               ;比较读取 PE 口状态值，不相等则错误
      BNZ         KEY_ERROR
      LDI         PEOUT,00H              ;设置 PE 口为输入口
      LDI         PORTE,0FH              ;打开 PE 口内部上拉电阻
      CALL        DELAY_5MS              ;消除按键抖动
      LDA         PORTE,00H              ;读取 PE 口状态
      SUB         TEMP,00H               ;比较读取 PE 口状态值，不相等则错误
      BNZ         KEY_ERROR
      SBI         TEMP,1111B             ;没有按键按下检测
      BNZ         DETAIL_KEY_CHECK
      LDI         FLAG ,00H              ;清除按键已处理标志
      JMP         KEY_CHECK_PROCESS_OVER
DETAIL_KEY_CHECK:                        ;根据键值检查具体的按键
      SBI         FLAG ,0FH              ;检查按键已处理标志
      BAZ         KEY_CHECK_PROCESS_OVER
      LDI         FLAG ,0FH              ;设置按键已处理标志
      SBI         TEMP,1110B
      BAZ         PERIOD_DOWN_KEY        ;减小周期键
      SBI         TEMP,1101B
```

```
    BAZ         PERIOD_UP_KEY          ;增大周期键
    JMP         KEY_ERROR
PERIOD_DOWN_KEY:                       ;减小 PWM 周期 10H
    SBIM        POPDM,01H
    LDI         CLEAR_AC,00H           ;清除累加器 AC 的值
    SBCM  POPDH,00H                    ;带进位减
    BC          KEY_CHECK_PROCESS_OVER
    CALL        RESET_PWM_PERIOD       ;不够减，则重设 PWM 周期的值
    JMP         KEY_CHECK_PROCESS_OVER
PERIOD_UP_KEY:                         ;增大 PWM 周期 10H
    SBI         POPDH,03H              ;判断是否到达最大值
    BNZ         PERIOD_UP_KEY1
    SBI         POPDM,0FH
    BNZ         PERIOD_UP_KEY1
    JMP         KEY_CHECK_PROCESS_OVER     ;到达最大值，则不增加 PWM 周期的值
PERIOD_UP_KEY1:
    ADIM        POPDM,01H
    LDI         CLEAR_AC   ,00H        ;清除累加器 AC 的值
    ADCM  POPDH,00H                    ;带进位加
    JMP         KEY_CHECK_PROCESS_OVER
KEY_ERROR:                             ;错误键值处理
    JMP         KEY_CHECK_PROCESS_OVER
KEY_CHECK_PROCESS_OVER:                ;按键扫描及处理结束，返回
    LDI         POPDL,00H              ;设置 PWM0 周期低 4 位(更新周期数据)
    LDI         PODDL,00H              ;设置 PWM0 占空比低 4 位(更新占空比数据)
    RTNI
;************************************************************
;          PWM0 周期与占空比设置子程序
;************************************************************
PWM0_OUTPUT_SET:
    LDI         PWM0SET,0110B          ;设置 PWM0 为正向占空比,PWM0 时钟为 8Tosc
                                       ; 关闭 PWM0 输出,置为 I/O
    LDI         POPDH,03H              ;设置 PWM0 周期高 2 位
    LDI         POPDM,0FH              ;设置 PWM0 周期中 4 位
    LDI         POPDL,00H              ;设置 PWM0 周期低 4 位
    LDI         PODDH,00H              ;设置 PWM0 占空比高 2 位
    LDI         PODDM,05H              ;设置 PWM0 占空比中 4 位
    LDI         PODDL,00H              ;设置 PWM0 占空比低 4 位
```

```
    ORIM        PWM0SET,0001B          ;选择 PWM0 输出
    RTNI
;************************************************************
;           PWM0 占空比重新设置子程序
;************************************************************
RESET_PWM_PERIOD:
    LDI         P0PDH,00H              ;设置 PWM0 周期高 2 位
    LDI         P0PDM,00H              ;设置 PWM0 周期中 4 位
    LDI         P0PDL,00H              ;设置 PWM0 周期低 4 位
    RTNI
;************************************************************
;           延时 5 毫秒子程序
;************************************************************
DELAY_5MS:
    LDI         DELAY_TIMER2,03H       ;设置初始值
    LDI         DELAY_TIMER1,03H
    LDI         DELAY_TIMER0,0CH
DELAY_5MS_LOOP:
    SBIM        DELAY_TIMER0,01H       ;每次减 1
    LDI         CLEAR_AC   ,00H
    SBCM  DELAY_TIMER1,00H
    LDI         CLEAR_AC   ,00H
    SBCM  DELAY_TIMER2,00H
    BC          DELAY_5MS_LOOP
    RTNI
    END                                ;程序结束
```

3.9 模拟比较器(Comparator)

中颖公司SH6xxx产品线中很多产品已经集成了多种特性的模拟比较器(Comparator,CMP)模块,一般有三个管脚,CMPxP 管脚为正端输入,CMPxN 管脚为负端输入,CMPxC 管脚为输出端。不同的单片机有不同的比较器(CMP)管脚配置，有的可以开漏输出（Open-Drain)，有的可以内接 1/2VDD 位比较电平,有的甚至可以选择多个内建比较电平。这样是用户满足不同系统设计的需求。在这一章中我们将以最普通的模拟比较器(CMP)模块为基础,从应用角度讲解在一个控制系统中如何合理的使用片上 CMP 功能。

3.9.1 CMP模块综述

CMP 模块通常由三个部分组成，即 CMP 的工作模式控制，CMP 的输入选择和输出状态控制以及 CMP 的中断选择控制。不同的芯片上提供的 CMP 输入通道数量不同。CMP 输出的高低电平并非来自与普通端口寄存器的锁存值，而是直接由 CMP 模块输出。一般 CMPxC 管脚和某个端口管脚复用的，可以通过软件配置选择。软件一旦选择 CMP 输出管脚,不论端口输入/输出寄存器设置如何,系统将自动切换至 CMP 模块输出。

3.9.2 CMP相关控制寄存器介绍

CMP 模块最重要的寄存器是控制寄存器(Comparator Control Register,以下接称CMPC)，还有 CMP 比较输出和配置控制寄存器(CMPG)以及 CMP 状态控制寄存器(CMPS)。

CMP 控制寄存器(CMPC)

CMP 控制寄存器位于内部 RAM 零页的地址 013H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$13	CMPE	CMPSO	CMPSN	CMPEN	R/W	

CMP 控制寄存器数据位定义

CMPEN: CMP 模块使能选择

0：CMP 模块禁止。

1：CMP 模块使能。

CMPSN: CMP 模块 CMPxN 选择（CMPxN 管脚/普通端口管脚配置选择)。

0：CMP 模块选用内部 1/2VDD 为负端输入（选择普通端口管脚)。

1：CMP 模块选用外部端口为负端输入（选择 CMPxN 管脚)。

CMPSO: CMP 模块 CMPxC 选择（CMPxC 管脚/普通端口管脚配置选择)。

0： CMP 模块选用内部输出（选择普通端口管脚)。

1： CMP 模块选用外部端口为输出（选择 CMPxC 管脚)。

CMPE: CMP 模块输出产生中断的有效沿选择。

0 = 选择输出下降沿。

1 =选择输出上升沿。

CMP 比较输出和配置控制寄存器(CMPG)

CMP 比较输出和配置控制寄存器(Comparator output and Configuration Control Register)位于内部 RAM 第七页的地址 0FH,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$38F	CNF2	CNF1	CNF0	CMPOD	R/W	

CMP 比较输出和配置控制寄存器数据位定义:

CMPOD: CMP 模块

0: 当 CMP 模块使能且 VCMPxP < VCMPxN。或者，如果 CMP 模块禁止。

1: 当 CMP 模块使能且 VCMPxP > VCMPxN。

CNF2～0:　　CMP 模块 CMPxP 选择（CMPxP 管脚/普通端口管脚配置选择）。

0 = CMP 模块禁止 CMPxP 管脚(正端)输入（选择普通端口管脚）。

1 = CMP 模块使能 CMPxP 管脚(正端)输入（选择 CMPxP 管脚）。

CNF2	CNF1	CNF0	5	4	3	2	1	0
0	0	0	PORTG.0	PORTG.1	PORTG.2	PORTG.3	PORTE.3	PORTE.2
0	0	1	PORTG.0	PORTG.1	PORTG.2	PORTG.3	PORTE.3	CMPP1
0	1	0	PORTG.0	PORTG.1	PORTG.2	PORTG.3	CMPP2	CMPP1
0	1	1	PORTG.0	PORTG.1	PORTG.2	CMPP3	CMPP2	CMPP1
1	0	0	PORTG.0	PORTG.1	CMPP4	CMPP3	CMPP2	CMPP1
1	0	1	PORTG.0	CMPP5	CMPP4	CMPP3	CMPP2	CMPP1
1	1	0	CMPP6	CMPP5	CMPP4	CMPP3	CMPP2	CMPP1
1	1	1	CMPP6	CMPP5	CMPP4	CMPP3	CMPP2	CMPP1

CMP 状态控制寄存器(CMPS)

CMP 状态控制寄存器位于内部 RAM 零页的地址 014H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$14	CMPGO	CMPSP2	CMPSP1	CMPSP0	R/W	

Comparator Status Control Register 寄存器的数据位定义

CMPGO:　CMP 模块工作状态选择。

0:　CMP 模块工作停止, CMP 模块比较输出始终为 0。

1:　CMP 模块正常工作, CMP 模块比较输出为有效状态。

CMPSP2～0:　CMP 模块当前 CMPxP 管脚位置选择。

000:　CMP 正端输入选择 CMPP1 管脚

001:　CMP 正端输入选择 CMPP2 管脚

010:　CMP 正端输入选择 CMPP3 管脚

011:　CMP 正端输入选择 CMPP4 管脚

100:　CMP 正端输入选择 CMPP5 管脚

101～111:　CMP 正端输入选择 CMPP6 管脚

CMP 中断控制寄存器(CMPS)

CMP 中断控制寄存器(Comparator IE Control Register)位于内部 RAM 第七页的地址 014H,其中的各数据位定义如下表所示。

Address	Bit 3	Bit 2	Bit 1	Bit 0	R/W	Remarks
$394	-	-	CMPIF	CMPIE	R/W	

Comparator IE Control Register 寄存器的数据位定义

CMPIE: CMP 模块中断使能标记

0: CMP 模块中断禁止。

1: CMP 模块中断使能。

CMPIF: CMP 模块中断请求标记

0: CMP 模块输出有效沿产生中断请求无效。

1: CMP 模块输出有效沿产生中断请求有效。

3.9.3 CMP模块工作模式设定说明

按照下面所属的操作步骤，一般都能实现 CMP 模块正常工作。

(1) 首先在设定 CMP 模块使能(CMPEN=1)之前,必须完成上述所有寄存器的正确设置。

(2) 由于 CMP 模块在使能(CMPEN=1)后需等待 3 微秒才能稳定工作,其中包括 1/2VDD 比较电平的建立。因此，使能(CMPEN=1)后等待 5 微秒才可打开 CMP 模块(CMPGO=1)。

(3) 当 CMP 模块正常工作(CMPGO=1)，不能随意切换 CMPxP 管脚(正端)输入位置和 CMPxN 管脚(负端)输入状态。只有在 CMP 模块工作停止(CMPGO=0)后方可进行 CMPxP 管脚(正端)输入位置和 CMPxN 管脚(负端)输入状态。若需 CMP 模块再次工作，必须再次打开 CMP 模块(CMPGO=1)。

(4) CMP 模块可以在单片机进入 HALT 模式下工作, 也可以在单片机进入 STOP 模式下依然工作。因此，只要 CMP 模块中断使能标记（CMPIE=1），CMP 模块输出有效沿产生中断就能将 CPU 从 HALT/STOP 模式下唤醒。

3.10 运算放大器(Operaional Amplifier)

3.10.1 单片机片上运算放大器（OP）模块综述

中颖公司 SH6xxx 产品线中很多产品已经集成了内建的运算放大器(OP)模块,一般有三个管脚,OAP_P 管脚为正端输入，OAP_N 管脚为负端输入，OAP_O 管脚为输出端。由于单片机通常为单电源供电，为适应普通应用，OP 模块采用特殊工艺和补偿电路，使其具有稳定的高增益，较宽的共模电压范围（接近 0V)。考虑运算放大器往往仅参与模

拟信号工作，通常很少直接与单片机相连，因此，OP 模块的控制采取代码选项而非一般的寄存器设置。

OP 模块的输入/输出管脚通常和普通端口管脚复用，SH6xxx 系列单片机提供 OTP 代码选项（OP_OAP）控制 OP 模块的工作。当 OP_OAP = 1,即 OP 模块使能，那些复用的端口立即切换至 OAP_P（正端）输入管脚，OAP_N（负端）输入管脚和 OAP_O 输出管脚。当 OP_OAP = 0,即 OP 模块被禁止工作，那些复用的端口仅能作为普通端口。

OP 模块可以在单片机进入 STOP 模式下依然工作。

3.11 电阻/频率转换功能（RFC）

电阻/频率转换功能（RFC）是利用电阻电容的充放电原理，在相应端口上产生 RC 振荡，用计数输入口读得一定时间内的脉冲数量，计算出振荡频率后，通过和标准频率的对比查表获得参于振荡的电阻值。此设计可用于温度和湿度传感器的阻值的测量。

RFC 相关系统寄存器如下：

第 3 位	第 2 位	第 1 位	第 0 位	说明
ENX	RX3EN	RX2EN	RX1EN	第 2-0 位： RX1, 2, 3 端口的 RC 振荡允许控制（初始值 = 0 禁止） 第 3 位： RFC 计数器启用控制
RFL.3	RFL.2	RFL.1	RFL.0	RFC 16位计数寄存器数据低位
RFML.3	RFM.2	RFML.1	RFML.0	RFC 16 位计数寄存器数据中低位
RFMH.3	RFMH.2	RFMH.1	RFMH.0	RFC 16 位计数寄存器数据中高位
RFH.3	RFH.2	RFH.1	RFH.0	RFC 16 位计数寄存器数据高位

在中颖的 RFC 设计中，一般有 3 路和 I/O 端口共享的 RFC 测量端口 RX1～3 和一个 RXB 端口。RXB 端口为电容电平抬升端口，用于提高 RC 翻转的电压。

当一路 RFC 测量端口被允许时,该端口的 RC 振荡电路和一个 16 位的计数器可以用来测量其振荡频率。其它的 RFC 端口处于高阻状态，不影响振荡。若其中有一路是标准电阻（参考电阻），则通过不同 RFC 端口振荡频率的比较，可以通过计算或查表得到目标电阻值。示意图如下图 3-11-1：

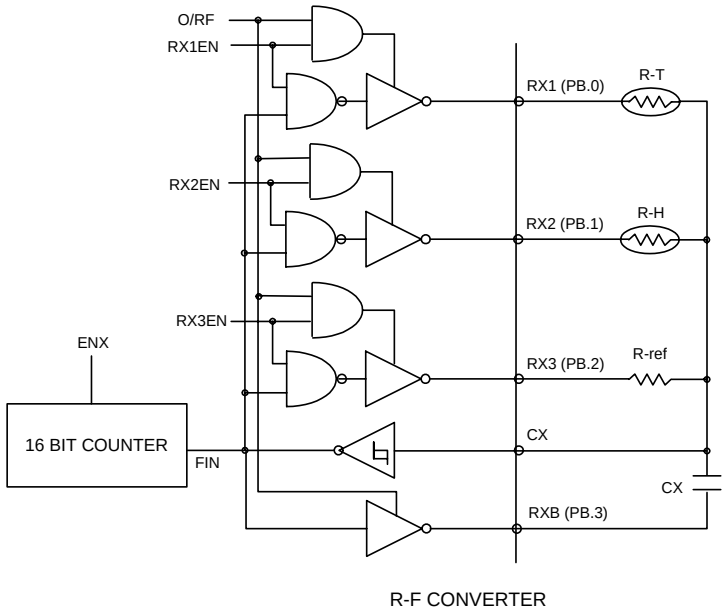


图 3-11-1 RFC 工作原理示意图

说明： R－T 电阻为温敏电阻；

R－H 电阻为湿敏电阻；

R－ref 电阻为参考电阻。（精密电阻）

CX 端口为 RC 频率测试端，翻转次数将记入 16 位长度的计数器。

RXB 端口为电容电平抬升端口，用于提高 RC 翻转的电压。

RFC 电阻测量原理

RFC 功能提供 3 路电阻测量端口“RX1～3”，一个频率测量端口“CX”以及电容电平抬升端口“RXB”。

利用 RC 振荡原理：

a.. 不同的电阻 R 对应同一电容 C 会得到不同的频率 f。

即：$f = 1/T = 1/KRC$ （K 为充放电系数）

频率 f 即由 CX 端口记入 16 位的计数器。

所以在一定时间内的计数值 N：

$N = K_X RC$ （K_X 为充放电系数）

b. 在同样的时间内对不同的电阻（温度电阻和参考电阻）会得到不同的计数值 N。

$$N_T = K_X R_T C \quad ; \quad N_{REF} = K_X R_{REF} C$$

$$N_T / N_{REF} = R_T / R_{REF}$$

所以，不同的计数比即等于电阻比率。由于参考电阻 R_{REF} 的电阻值是预先知道的，所以按照计数比率即可换算出 R_T 的电阻值。

c. 但是由于电阻和频率之间的关系并非线性及升压的影响，直接使用公式获得的

R_T 阻值会有不同大小的误差。可用用如下工程方法获得比较近似的数据：

按照上述公式，确定 R_{REF} 的电阻值后，直接使用电阻箱摹拟温度电阻（厂家会给出不同温度对应的电阻值），获得不同的温度下的计数比值，并建立温度直接对应表表格（使用对应的仿真板进行编程及验证）。

在测温时，可以按照计数比值直接查表，获得温度数值。表格可以设置为每1℃的变化对应一计数比值。小于 1℃的温度值可以通过插值计算获得。

使用上述方式可以比较精确的获得温度值。

应用注意点：

a) 测量温度范围的不同应选择不同的温度电阻：如为了能检测-20℃～100℃的范围。温敏电阻取值 15～50Kohm@25℃比较合适。为了在 100℃时有相对较高的精度，最好取 50K ohm@25℃的温度电阻。如果只测量-20℃～70℃摄氏度的范围，则取 10Kohm@25℃亦可。同理，取阻值大一点的温度电阻亦可提高在 70℃的测量精度。

例如：可以采用一温度电阻在 25℃时为 30Kohm，其在－20℃时为 300Kohm，在 100℃时为 2Kohm。

b) RFC 的最大工作频率必须小于 2MHz。

c) 当 16 位计数器不作为 RFC 使用时，可以作为一个通用计数器。

d) RFC 可以在 HALT 模式下保持工作，当执行“STOP”指令后自动停止。（保持 RX1-3 端口最后的状态和停止 R-F 计数器）

e) 由于R-F测温的充放电脉冲会引起的额外的干扰，容易干扰port口的输入，建议使用如下方法解决：

方法 1：软件解决方法：在测温期间关闭 Port 口中断；

方法 2：硬件解决方法：输入端口加一 2200pF 的电容。

附图 3-11-2：RFC 中电阻和频率关系(仅供参考)

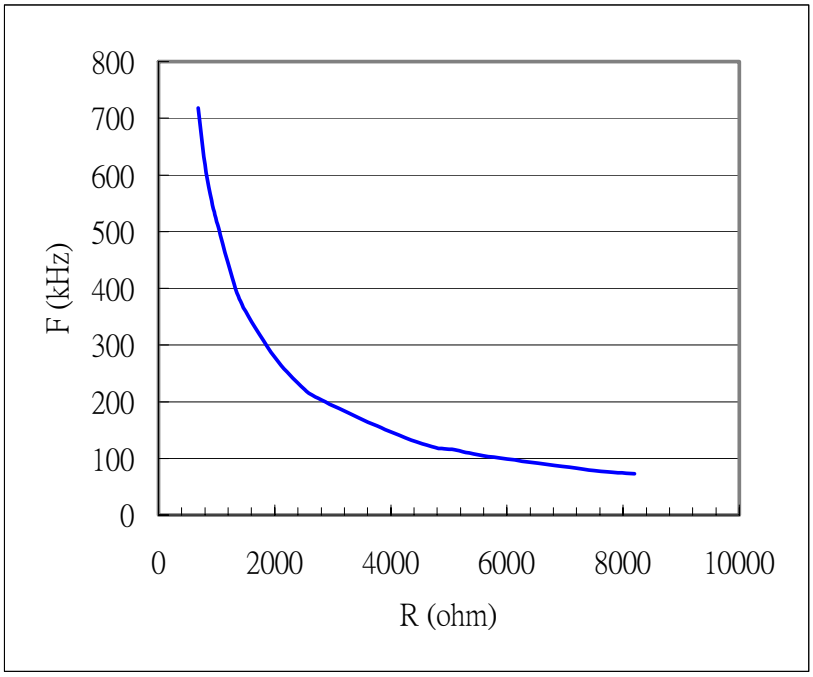


图 3-11-2 电阻和频率关系对应图

3.12 冷光驱动器（EL driver）

冷光（EL）背光系统是由EL灯片和EL驱动电路组成，多用于为LCD提供背光。EL灯片是由绝缘基底上喷涂了场致发光材料并夹在两层电极之间组成。EL场致发光灯的供应商可以通过使用不同的发光材料，比如硫化锌、硫化钙或硫化锶，再掺杂其它成份如镁、钐、铕或添加萤光染色剂等，来调整光的亮度和颜色。改变激励频率同样能引起光的颜色变化。

中颖公司SH6xxx产品线中有些MCU内建了冷光(EL)驱动器。

以下是针对此功能的典型应用电路(如下图)及注意事项。

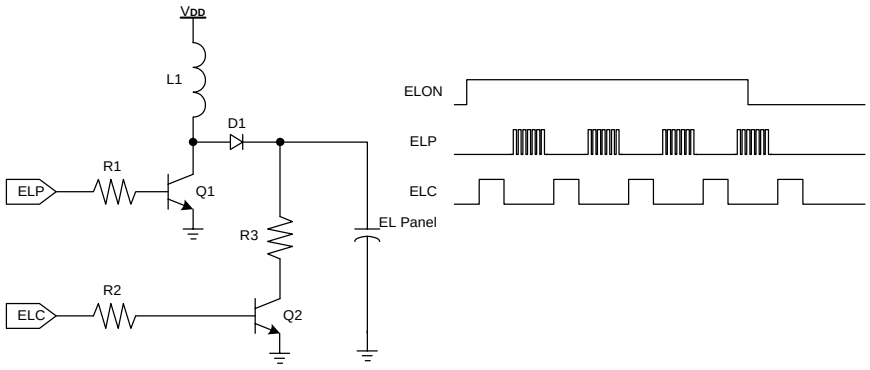


图3-12-1 EL driver典型应用电路

ELC和ELP的输出波形如上图所示。ELP用于对EL面板充电，通过外接三极管， 二极管，电感，电阻，我们能够把EL 面板上的电压升压到100－250V。ELC用于释放EL面板上的电压。通过循环充放电的动作，在EL面板上产生了一个交流的电压，不同的电压值和频率可以获得各种亮度和色彩。

EL驱动控制寄存器如下：

第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
－	ELF	ELPF	ELON	读/写	第 0 位： EL 打开/关闭控制（初始值 0 = 关闭） 第 1 位： EL 驱动器充电频率选择 第 2 位： EL 驱动器放电频率选择
－	X	X	0	读/写	EL 驱动器关闭（初始值）
－	X	X	1	读/写	EL 驱动器打开
－	X	0	X	读/写	ELP 引脚输出频率 = ELCLK （初始值）
－	X	1	X	读/写	ELP 引脚输出频率 = ELCLK/2
－	0	X	X	读/写	ELC 引脚输出频率 = ELCLK/64 （初始值）
－	1	X	X	读/写	ELC 引脚输出频率 = ELCLK/32

EL 驱动电路的工作时钟 ELCLK 为 32kHz。

当 EL 驱动器关闭时，ELP 和 ELC 引脚输出低电平，确保 EL 面板上不会残留电压。当开启 EL 驱动器时，需先设置系统寄存器以选择 EL 驱动器波形。设置 ELON = 1 可以打开 EL 驱动器。

当启动 EL 以后，ELC 将比 ELP 提前打开。当关闭后，ELP 将先关闭。为了保证 EL 上没有残留电压，ELC 将继续工作一个周期。

在 HALT 模式下 EL 驱动电路会继续工作。当执行了 “STOP” 指令后 EL 驱动电路会被关闭(ELC 和 ELP 为低电平)。

典型应用电路如图3-12-2 (以SH67L19为例)：

1.5V电池供电，振荡器为 32.768KHz 晶振。

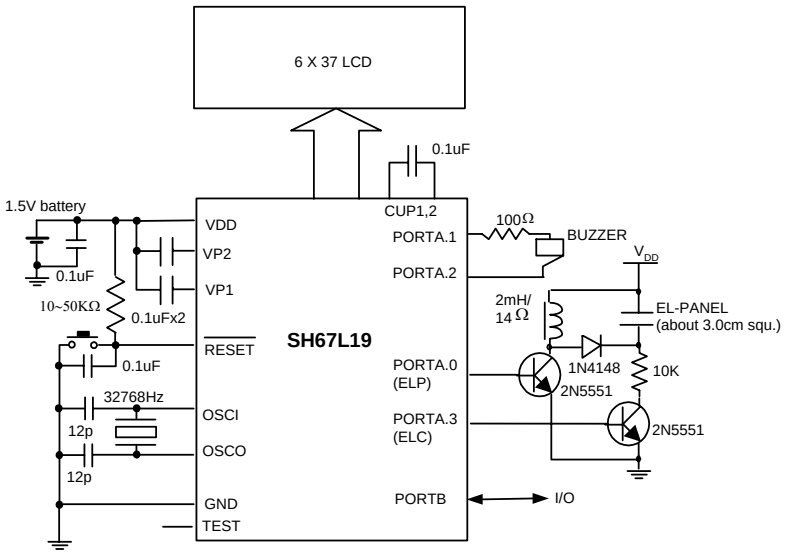


图3-12-2 SH67L19典型应用电路实例

应用注意事项

a) EL电路的驱动管脚ELC及 ELP所接的NPN三极管的耐压要达到150V以上(如2N5551)。

b) 1N4148在某些高电压的应用中的耐压不够(100V以上)， 可以使用2个1N4148串联的方法或选用更高耐压的二极管。

c) 电感值和使用的冷光片的面积有关，其关系可大致参考如下：

电感		EL 面积建议	EL 电压建议
mH	ohm	cm2	V
2	14	<3.5	<120
1	11	<6.0	<100
1	11	<3.5	<120
1	11	<2.5	<160

d) EL驱动电路工作时，其电流达10mA左右，所以需要采用输出电流可以达到10mA的氧化银电池。

e) 应用中，请在执行“STOP”指令前关闭EL驱动功能。

f) 在印制板布局方面, 需要注意如下：

- 在靠近MCU的位置要布置一0.1μF的旁路电容,以减小高压部分对MCU的干扰。
- MCU的晶振必须远离高压部分。
- 高压部分尽量远离MCU及其它电路, 并用地平面隔离。

3.13 上电复位/低电压检测和复位/看门狗定时器

低电压复位（Low Voltage Reset, LVR）

低电压复位（LVR）用于监控芯片工作电压并产生芯片内部复位信号。它一般用于交流供电电路或有大负载的电路，这些电路工作时负载的启动会引起器件工作电压暂时低于电路的最低允许工作电压。当芯片工作电压 V_{DD} 小于检测电压 V_{LVR} 时，系统会产生芯片复位，并一直持续到工作电压 V_{DD} 高于检测电压 V_{LVR} 后再结束复位状态，恢复运行。

LVR 功能的开启和不同 LVR 电压的选择都通过代码选项进行设定。

低电压检测（Low Power Detect,LPD）

低电压检测（LPD）功能用来监测当前工作电压。它一般用于电池供电应用，提供电池欠压标志。当芯片工作电压 V_{DD} 小于检测电压 V_{LPD} 时，系统会在系统寄存器中设置 LPD 标志，但不会产生芯片复位。当芯片工作电压 V_{DD} 回复高于检测电压 V_{LPD} 时，LPD 标志被清除。

LPD 控制寄存器：

第3位	第2位	第1位	第0位	读/写	说明
-	-	-	LPD	只读	第0位： LPD 标志

看门狗定时器 （WatchDog Timer,WDT）

当由于某些原因,如电源干扰或软件自身的缺陷导致程序意外的进入了死循环或出错运行状态时，这时我们可以利用看门狗定时器(WDT)来强制系统产生硬件复位，使系统及时退出死循环或出错状态。当然，当系统发生了硬件死锁现象时，WDT 也无法发挥作用。用户程序无需关心 WDT 具体的计数值，而只要在 WDT 的溢出周期内(或者说在溢出发生前)不断的复位 WDT，防止产生溢出复位信号即可。

看门狗定时器是一个递减计数器，一般拥有独立的内建 RC 振荡器作为时钟源，因此在 STOP 模式下仍会持续运行(也有某些型号的 MCU 的 WDT 时钟源来自系统时钟,因此在 STOP 模式下不会运行）。当定时器溢出时，WDT 将复位 MCU。该功能通过代码选项可以允许或禁止该功能。

WDT 控制寄存器（第 2～0 位）用来选择不同的溢出时间。定时器溢出后，WDT 溢出标记（第 3 位）将由硬件自动设置为“1”。 通过读或者写 WDT 控制寄存器，WDT 会被复位，重新开始计数。

WDT 系统寄存器：

第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
WDT	WDT.2	WDT.1	WDT.0	读/写 只读	第 2-0 位： 看门狗定时器控制位。 第 3 位： 看门狗定时器溢出标记 （只读）。
X	0	0	0	读/写	WDT溢出周期为4096ms。
X	0	0	1	读/写	WDT 溢出周期为 1024ms。
X	0	1	0	读/写	WDT 溢出周期为 256ms。
X	0	1	1	读/写	WDT 溢出周期为 128ms。
X	1	0	0	读/写	WDT 溢出周期为 64ms。
X	1	0	1	读/写	WDT 溢出周期为 16ms。
X	1	1	0	读/写	WDT 溢出周期为 4ms。
X	1	1	1	读/写	WDT 溢出周期为 1ms。
0	X	X	X	R	未发生 WDT 溢出复位。
1	X	X	X	R	WDT 溢出，发生 WDT 复位。

注意：规格书中的看门狗定时器溢出周期是在工作电压 V_{DD} = 5V 时的参考值，如果工作电压不同，将会有差异。

在软件应用中，在合理的程序位置清 WDT 才能发挥其有效的程序监控作用。一般清 WDT 指令只能放置在程序主循环中，且只能放置一个。在子程序、循环程序以及中断服务程序中不能放置清 WDT 指令，否则有机会在发生错误的情况下永远无法退出。

3.14 红外遥控发射

利用红外遥控实现人机界面的应用，在日常生活中非常广泛。其原理大致如下，在发射端，通过将遥控信息(二进制脉冲码)调制在 38kHz 载波上，经放大后送至红外发光二极管，转化为红外信号发射出去；在接收端，通过红外接收二极管接收红外信号，再经过解调模块，得到有用遥控信息信息。

以下结合中颖电子的红外遥控单片机(SH66K(P51),SH67P33(A)等)对发射端的设计(红外遥控波形发生器)进行说明。

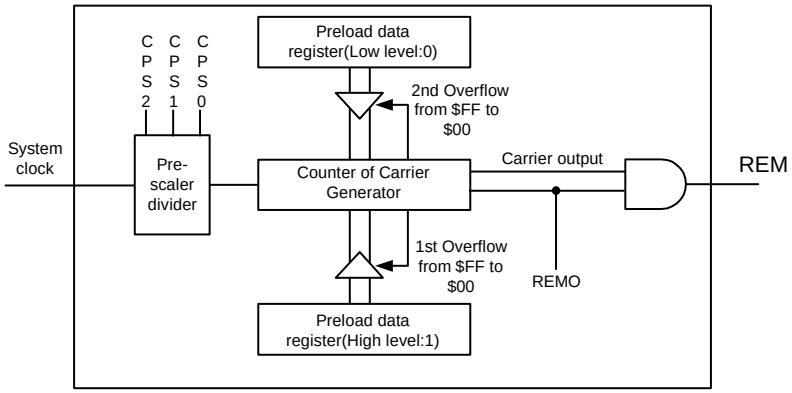


图 3-14-1　红外遥控波形发生器框图

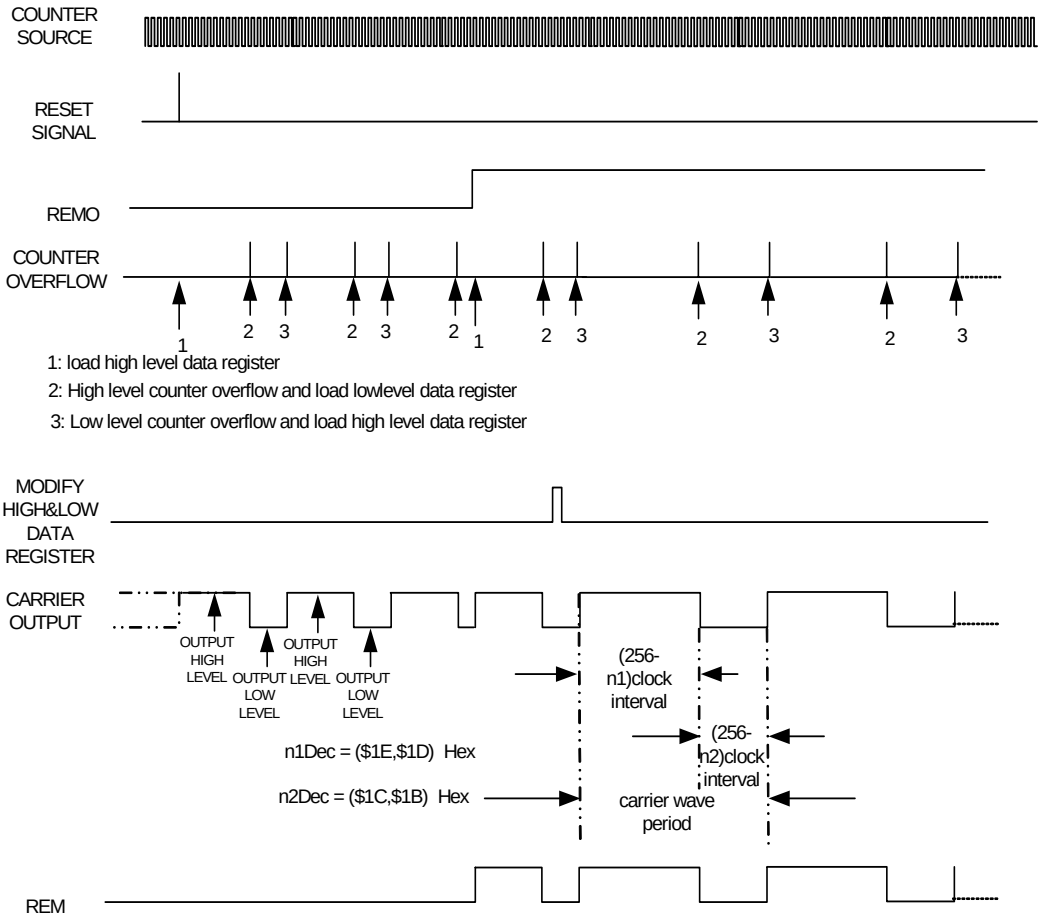


图 3-14-2　红外遥控合成载波波形

3.14.1 工作原理

载波发生计数器由一个 8 位向上计数器和两个 8 位数据重载寄存器(高电平数据寄存器和低电平数据寄存器）组成。 写数据到数据重载寄存器可以初始化载波发生计数器。系统复位后，载波发生计数器自动重载高电平数据寄存器的数据，同时输出高电平。接着当计数器计数从$FF 到 $00 溢出时，计数器自动重载低电平数据寄存器的数据同时输出低电平。当计数器计数从$FF 到$00 再次溢出，计数器自动重载高电平数据寄存器

的数据，同时输出高电平。上述操作组成一个完整的循环，如图 2 所示。 因此载波合成器可以输出具有特定占空比和周期的连续载波波形。

从图 1 可以看到载波发生器时钟源为系统时钟。系统时钟经过一个分频器进行分频，该分频器可以选择 8 种不同的分频比，将分频之后的信号输入到一个 8 位计数器作为计数器的触发信号，触发信号的每一个周期计数器将自加一。通过选择不同分频比，通过修改低电平数据重载寄存器来改变低电平的宽度或者修改高电平数据重载寄存器来改变高电平的宽度，将得到不同周期和占空比的载波波型。修改的高电平数据寄存器或低电平数据寄存器的值，不影响当前计数周期，新的数值只有在发生溢出后被重新加载。

通过 REMO 数据位控制 REM 输出，当写“1”到 REMO 时，计数器将重新加载高电平数据寄存器的数值，且 REM 将输出 Carrier output；当写“0”到 REMO 时，REM 将输出低电平。读$0D 寄存器，将会读入 REM 口输出状态。

在 HALT 模式下 REM 保持输出载波波形，但在 STOP 模式下将输出 GND。

3.14.2 相关的寄存器（以SH66P51为例）

系统寄存器

地址	第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
$13		CPS2	CPS1	CPS0	读/写	载波计数器预分频比控制
$0D	-	-	-	REMO REM	写 只读	REMO 输出数据控制 REM 引脚输出状态

载波加载数据寄存器

地址	第 3 位	第 2 位	第 1 位	第 0 位	读/写	说明
$20	CFL3	CFL2	CFL1	CFL0	读/写	载波低电平数据寄存器 （低位）
$21	CFL7	CFL6	CFL5	CFL4	读/写	载波低电平数据寄存器 （低位）
$22	CFH3	CFH2	CFH1	CFH0	读/写	载波高电平数据寄存器 （低位）
$23	CFH7	CFH6	CFH5	CFH4	读/写	载波高电平数据寄存器 （高位）

下表为 CPS2-0 控制位对应的预分频比:

CPS2	CPS1	CPS0	预分频比	比例 N
0	0	0	系统时钟/2^{11}	2048
0	0	1	系统时钟/2^{9}	512
0	1	0	系统时钟/2^{7}	128
0	1	1	系统时钟/2^{5}	32
1	0	0	系统时钟/2^{3}	8
1	0	1	系统时钟/2^{2}	4
1	1	0	系统时钟/2^{1}	2
1	1	1	系统时钟/2^{0}	1

例如下表，在不同的系统时钟下，设置不同的分频比和载波高/低电平数据寄存器可以得到预定频率的载波。

系统时钟	CPS2～0	CFL	CFH	载波占空比	载波频率
4M/4	1, 1, 1	$EF	$F8	8/25 ≈ 1/3	40.00kHz
4M/4	1, 1, 1	$EF	$F7	9/26 ≈ 1/3	38.46kHz
4M/4	1, 1, 1	$F2	$F3	13/27 ≈ 1/2	37.04kHz
4M/4	1, 1, 1	$EB	$F9	7/28 = 1/4	35.71kHz
480k/4	1, 1, 1	$FE	$FF	1/3 = 1/3	40.00kHz
455k/4	1, 1, 1	$FE	$FF	1/3 = 1/3	37.92kHz
432k/4	1, 1, 1	$FE	$FF	1/3 = 1/3	36.00kHz

3.14.3 应用电路图

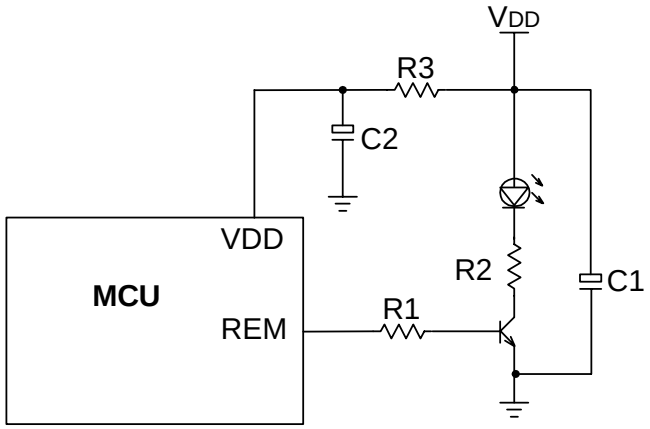


图 3-14-3 红外发射应用电路

图 3-14-3 为红外遥控发射部分的应用电路，其中 R1=510 ，R2=2.2 ，C1=47uF。其它还包括按键，LCD 显示等，这里省略该部分电路。

红外遥控器一般是采用电池来进行供电的。当载波发射时，系统需要输出较大功率，而由于电池的瞬时功率输出的能力有限，电池的输出电压将会在发射载波时暂时下降。这种情况在电池电量属于正常时，不会对应用产生影响。而在电量不足时，可能出现在发射载波时输出电压降低很多而导致 MCU 复位或死机的现象(MCU 内部一般都内嵌一个低电压复位保护电路，当电源电压很低时，IC 内部很多功能模块可能会工作不正常，为了避免产生此类情况，低电压复位保护电路在 IC 可能会出现误动作之前，将强制芯片产生复位信号，并一直保持到电源电压恢复)，为了避免这一现象，在应用图中加入了 R3/C2，即可有效避免此现象，R3 的阻值尽量选择小一些，C2 的容值尽量大一些。

3.15 OTP(One Time Program)产品的编程

在 SH6xxx 产品线中，OTP 产品占据了重要角色，特别是家电(SH69XX)系列，几乎全部为 OTP 产品。OTP 产品，顾名思义，为一次性可编程器件，从厂家购买到的产品，程序 ROM/代码选项区为全空，客户需要将自己的应用程序代码烧录到芯片中，芯片才

能正常工作。所以对于 OTP 产品来讲，程序的编程是一个重要的步骤。

目前中颖公司，对于 OTP 产品的编程问题，主要提供 2 种方式进行处理。

1. 对于是单一程序代码，而且单次出货量很大的客户，提供在产品出货时就将客户程序写入芯片的服务。在这种方式下，客户的操作流程同投掩膜片(MASK type)一样处理，只需提供程序和代码选项表即可。
2. 购买空的 OTP 芯片，然后使用编程工具，将程序写入芯片。一般客户均采取这种方法。

下面主要就第 2 种方式的一些问题进行介绍。

目前中颖公司提供的 OTP 编程工具，其名称/适用 IC 型号等见下表 3-15-1：

编程器名称	Pro-01	Pro-03	USB Rice66
说明	量产编程器	量产编程器	SH6xxx 产品的仿真器，可以提供调试阶段的少量 OTP 编程功能。
支持的 IC 型号	SH66N12 SH66P12 SH66P13A SH66P14A SH66P20A SH66P22A SH66P31A SH69P20	除 Pro-01 支持的产品以外的所有 OTP 产品	绝大多数产品，具体见 USB Rice66 的产品说明书
接口	存有客户程序的 FLASH 芯片	USB PC 接口或存有客户程序的 FLASH 芯片	USB PC 接口
输入电源	DC 14～16V，1A	DC 14～16V，1A	USB power 或 DC 14～16V，1A
升级操作	不能升级	网上下载最新程序，自行升级 (www.sinowealth.com)	

表 3-15-1 OTP 编程工具分类

编程操作注意点：

1） 对于使用 Pro-01 进行编程的 OTP 产品

由于芯片的开发时间较早，此类 IC 并未提供在系统编程功能，所以推荐的操作步骤见下表 3-15-1：

步骤	封装片		裸片
	方式 1	方式 2	

1	使用对应的 Socket 对芯片进行编程	应用电路板预留出 OTP 编程接口，将 OTP 芯片插入/焊接到应用电路板，其它所有原器件不能焊接	应用电路板预留出 OTP 编程接口,将 OTP 邦定到应用电路板上,其它所有原器件不能焊接
2	插入/焊接到应用电路板	通过编程接口编程	通过编程接口编程
3		焊接其它原器件	焊接其它原器件

表 3-15-2 OTP 推荐编程步骤

2） 对于使用 Pro-03 进行编程的 OTP 产品

此类产品均提供在系统编程功能。

当用户采用 COB(Chip on Board)组装方式或封装片在编程前就已经插入或焊接到应用电路板时，OTP 芯片可以使用在系统编程 (In System Programming)方式编程。

使用在系统编程方式编程时，用户必须在印制板(PCB)上预留出 OTP 芯片的编程接口，以便连接 OTP 编程器进行编程。

在此模式下，用户可在 OTP 芯片编程前将包括 OTP 芯片在内的所有器件组装在 PCB 上后，再对 OTP 芯片进行编程。当然也可以可先将 OTP 芯片组装到 PCB 上，对 OTP 芯片编程完成后再组装其它器件。

为了提高 OTP 编程的可靠性，在编程操作时 OTP 编程信号线必须直接连接到 OTP 编程器上，不允许有其它器件或外加电路与之并联.所以在 PCB 上必须预留 4 组跳线或分割焊盘，将 OTP 编程接口(VDD, VPP, SDA, SCK)与应用电路分隔开，如下图 3-15-1 所示：

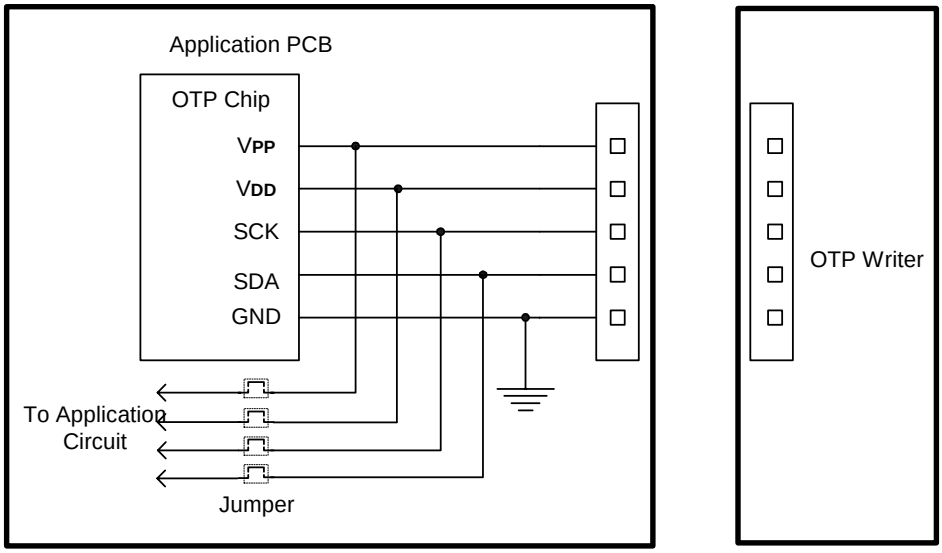


图 3-15-1 在系统编程连接示意图

具体操作步骤如下：

1） 在 OTP 芯片编程前将 4 组跳线断开；

2） 将 OTP 芯片的编程接口连接到 OTP 编程器，完成代码编程；

3） 将用户板与 OTP 烧写器编程器断开，将 4 组跳线短接；

有关 OTP 编程的更多详细资料，请参见后续第四章中对相关工具的详细介绍。

第四章

Sino Wealth 4-bit 单片机开发工具介绍

任何单片机的设计应用都有一个开发过程，整个开发过程需要开发环境和开发工具。针对 Sinowealth SH6xxx 单片机的开发工作，应用最广泛的开发环境是 Sinowealth 自主推出的集成开发平台 Rice66 V4.x。此平台整合了源程序的编写、机器码的编译、各种开发、调试功能的支持。另外，中颖公司还开发了 OTP 量产编程器 Pro03，用于 OTP 产品的批量编程。

Rice66 是 Sino Wealth 单片机开发应用必备的软件，它是完全免费的，可以从 Sino Wealth 公司官方网站 www.sinowealth.com 上下载，其本身还在不断的升级更新，以支持不断推出的新器件和新工具。目前的版本是 Rice66 V4.3。在这里我们要详细介绍 Rice66 开发环境及 Pro-03 的应用。

4.1 Rice66的功能组成

4.1.1 Rice66综述

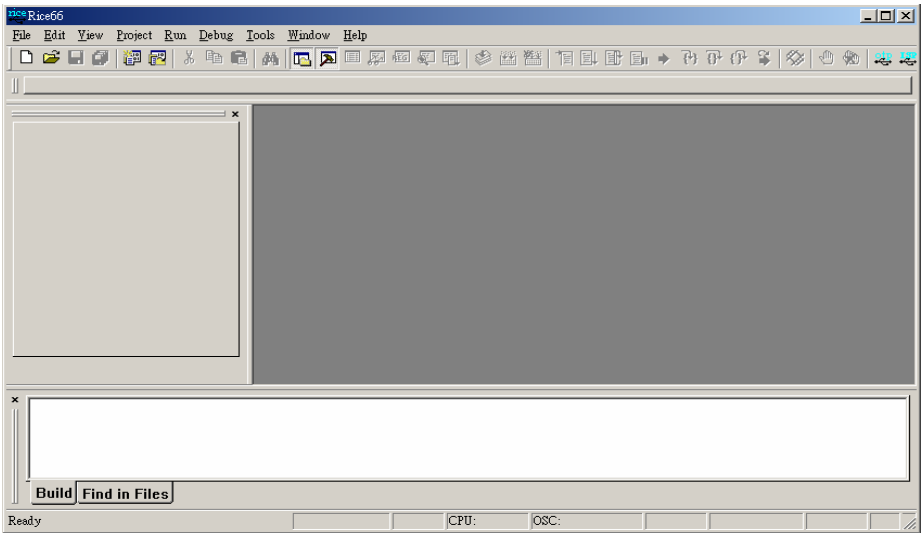
从 1997 年起，Sinowealth 公司就已推出 SH6xxx 单片机开发环境 Rice66。历经 8 年的改进工作和扩充，现在的 Rice66 V4.3 是基于 Win32 操作系统的标准 32 位 Windows 应用软件，完全由 Sinowealth 公司自主开发。它把开发过程中用到的各个不同且独立的工具集合为一体，实现了 SH6xxx 单片机的一站式开发。它集成了源程序编辑器、汇编语言编译器，直接支持硬件仿真器对目标系统进行源程序级调试，直接驱动 OTP 的 OTP 编程器，实现芯片的编程、读取、校验等标准功能。

4.1.2 Rice66的安装

安装时先从网上下载一个压缩文件 RICE66-USB-WIN-V4X.zip，解压后，运行安装文件 Setup.exe，就进入自主引导的安装流程。如果都选择缺省安装选项，则安装结束后所有软件文件都放在“C:\Program Files\SinoWealth”目录下。在以下章节中凡涉及 Rice66 安装路径时，如果没有特殊说明就是指此缺省安装路径。

为了支持硬件仿真器和 OTP 编程器，在 Rice66 安装完毕后还需要另外安装 USB 驱动程序。硬件仿真器或 OTP 编程器在第一次接入时 Windows 98 或 Windows 2000 或 XP 操作系统会自动发现新硬件并自动安装驱动。

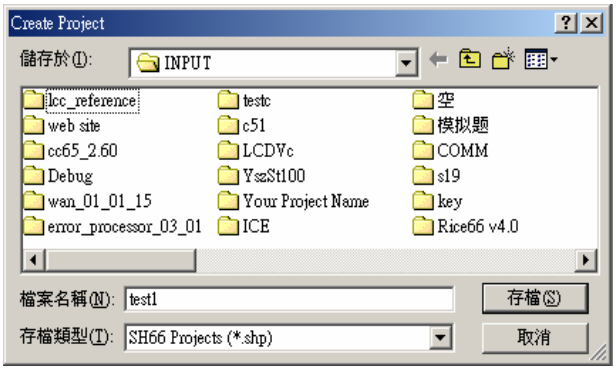
安装成功后执行 Rice66 文件，即可看到类似图所示的窗口界面。



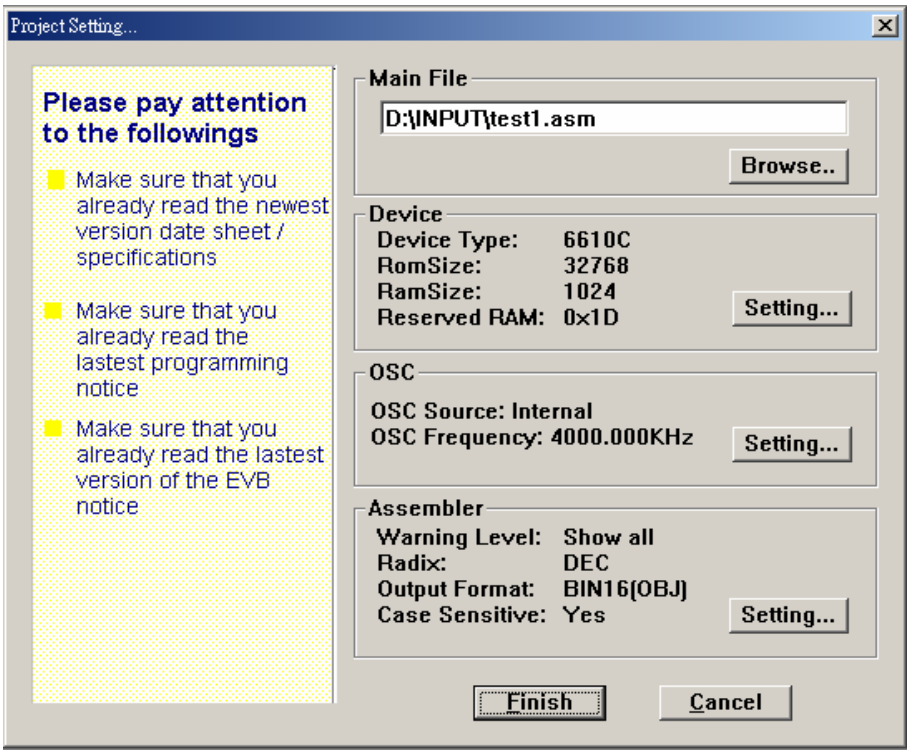
4.1.3 项目管理

开发 SH6xxx 单片机的一个良好习惯，是在 Rice66 平台下进行项目管理。Rice66 提供了方便的项目创建机制，通过菜单项 Project→New Project 进入，或工具条中的  。

(1) 第一步：弹出 Create Project 文件存档对话框，选择项目路径,输入项目名 test1。项目文件后缀名为 shp。

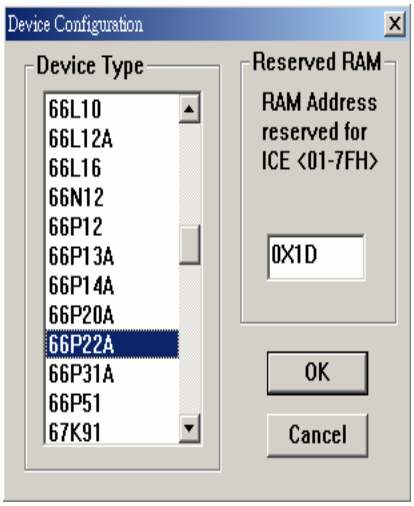


(2) 第二步：弹出 Project Setting 对话框。包含项目主文件添加，仿真器件设置，仿真时钟选择和编译参数设置四项设置。



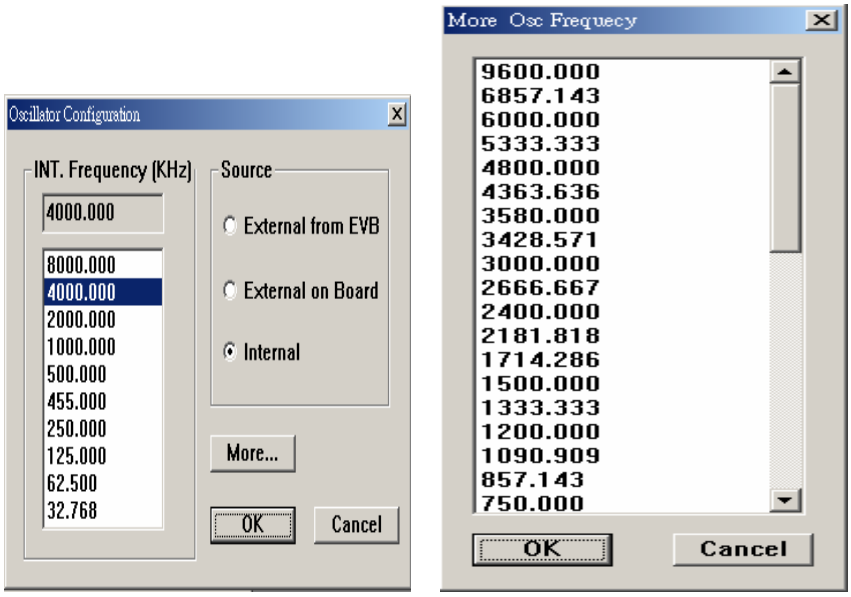
添加项目文件：在 Main File 区直接输入要添加至项目中的源文件，也可以通过 Browse 键弹出 Select the Main File 文件开启对话框，选择需添加的源文件。由于汇编工具只有编译，没有连接功能，编译参数只能是一个源文件名。

设置器件:通过 Device 区 Setting 键弹出 Device Configuration 对话框,在 Device Type 区选择需要调试的单片机类型，再在 Reserved RAM 输入十六进制的数值，数值必须在 00H 至 7FH 之间。注:Reserved RAM 是仿真过程中必须使用的一个 RAM byte，客户程序在仿真过程中避免使用此 RAM，防止出现不可预料的问题。

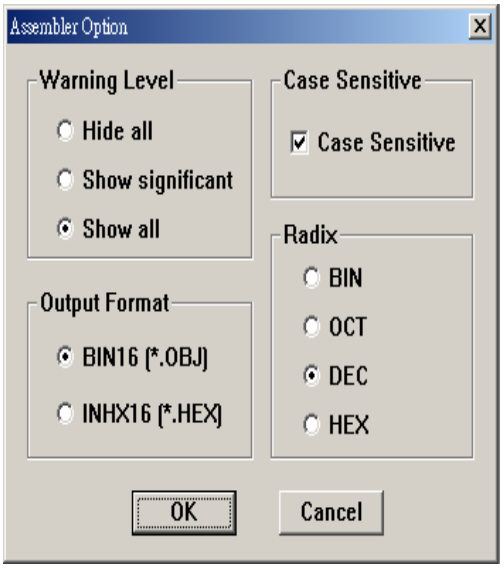


设置时钟:通过 Setting 键弹出 Oscillator Configuration 对话框，在 Source 区选择时钟源，时钟源有三种选项,选择 External from EVB，时钟由 EV 板提供；选择 External

On Board，时钟来自 RICE66 电路板上”EXT XTAL1”插座中的晶振；选择 Internal，时钟由 ICE66 提供，有多种频率供选择。”INT OSC frequency”区列出常用频率选择，若在”INT OSC frequency”区找不到所需频率，单击 More 键，会弹出更多频率供选择。

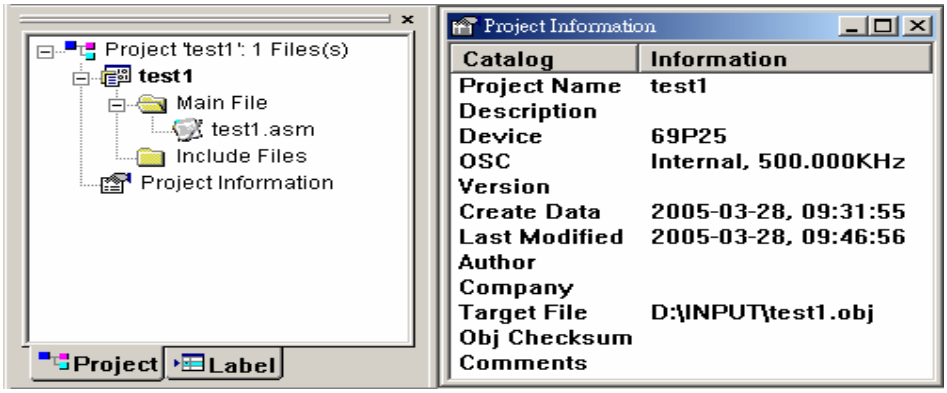


设置编译器参数:通过 Setting 键弹出”Assembler Option”对话框，Warning Level 区有对编译产生的警告信息显示的三个选项，设置 Hide all 所有警告信息都不显示，设置 Show significant 在 Output 窗口 build 页面显示重大的警告信息，设置 Show All 在 Output 窗口 build 页面显示所有警告信息。Case Sensitive 区选择编译是否对字符大小敏感。Output Format 区是选择编译生成文件格式是 16 位二进制 OBJ 格式还是 16 位 HEX 格式。Radix 区是选择源文件中数值的缺省进制，是 BIN(二进制)，或 OCT(八进制)，或 DEC(十进制)，还是 HEX(十六进制)。



最后单击 Finish 键完成项目创建。在 Project 窗口 Project 页面显示项目名，项目主源

文件和项目主源文件中 include 的源文件。Project 窗口中 Include Files 中所列的源文件不能添加,只能编译后自动添加。



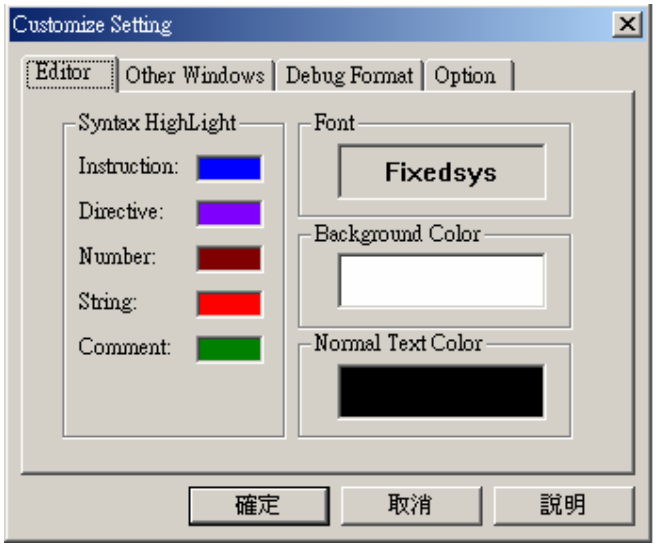
双击 Project 窗口 Project 页面中 Project information 项,会弹出 Project information 窗口,显示项目信息。你可以在此添加些个人信息,如项目版本等。

对已创建项目的开启,选择 Project 菜单中 Open Project 或快捷图标  ,会弹出 Select the Project 文件开启对话框,选择要开启的项目名后确认,即开启选中的项目。若已有项目开启,则先关闭已开启项目。最快捷的项目开启方法是选择 Project 菜单中最近项目列表中的项目名。

4.1.4 Rice66源程序编辑

源程序的编写需要文本编辑工具。Rice66 已经内建一个类似 UltraEdit 风格的文本编辑环境,可以方便实现源程序编写。通过主菜单 File→New 建立一个新文件,开始用程序编辑器编写你的程序代码。也可以用 File→Open 打开现有的程序文件进行编辑修改。

Rice66 的程序编辑器提供了源程序语法颜色区分的功能,不考虑文件格式。对于汇编语言源程序的语法颜色区分,Rice66 在安装时提供了默认的颜色选择,如汇编指令或其它的保留关键词用蓝色显示,注释用淡绿色显示等等。用户可以按照自己的习惯选择所喜欢的颜色来区分源程序中的不同部分,方法是在主菜单 Tool 中选择 Customize,弹出对话框 Customize Settings,在 Editor 一栏内单击要设置项的颜色框,会弹出色彩选择对话框,选择自己喜欢的颜色。还有字体和背景设置。Customize Setting 对话框是在源程序编写和其后调试过程中非常重要的一个设定工具,一定要善加利用。



当用户新建了一个文件开始编写自己的源代码时,所有的语法颜色马上显示在屏幕中,使用户编写代码和诊断语法错误的效率大大提高。

Rice66 编辑环境目前不支持挂第三方的文本编辑器,但完全可以在 Rice66 环境外用自己最熟悉的文本编辑器编写源程序,存盘后可以再回到 Rice66 环境下打开。若文件原本在Rice66下已经打开,则在外面修改后回到Rice66时会提醒文件被修改,是否重新装入,选择重新装入即可。

4.2 汇编编译器UASM66

用户的汇编语言源程序里用的全部是些助记符号,包括指令的助记符和其它数据、地址的符号变量等等,而这些符号需要转变成单片机的机器码,并通过编程工具烧入到单片机内,单片机才能按照用户设计的指令执行。把源程序中助记符号转换成机器码,这就是汇编编译器要做的工作。Rice66 安装后就已经内含汇编编译工具,无需用户另外单独安装。

关于汇编程序结构、汇编语句的格式、常量/符号/表达式的定义、伪指令、宏定义等请参见 2.7 部分。

4.3 硬件仿真器ICE66

ICE66 是 Sino Wealth 自主开发的高性能硬件在线仿真器。

ICE66 的结构组成主要分成两大部分:仿真主机和仿真板(EV Board)。其它还有一些附属的配件,包括 USB 连接电缆,仿真接口板和连接电缆。

SH6xxx 系列单片机型号众多,不同芯片间结构和功能可能差异很大,一种仿真板基本支持一种芯片。ICE66 的设计方针是采用积木搭配的方式:仿真主机不变, 同一种的

芯片使用同一种仿真板,使用时将仿真板插入仿真主机的仿真接口板中。针对每一颗不同的 SH6xxx 单片机,Sino Wealth 都提供了所对应的仿真板。

4.3.1 ICE66的基本功能

- 可以实现汇编语言的源程序级调试。
- 可以实现芯片最高运行速度的实时运行仿真(8 MHz) 。
- 全地址,全空间仿真,基本不占用芯片任何资源(除 Reserved ram),最大仿真程序空间 32K。
- 实时代码运行跟踪深度为 8K 单元,每个跟踪单元有案可查 24 位宽的数据信息。
- 可以进行低电压仿真,最低可到 1.5V。
- 无限制的断点设定。
- 可以输入或输出外部触发信号,具备简单的逻辑分析仪功能,可以同步触发外接的示波器以捕捉观察特定的信号序列。
- 提供 53 种可选择时钟频率,频率范围 32K～9.6M。仿真时目标板无需提供振荡电路。
- 与计算机通过 USB 口相连,数据传输速度快。
- 可以不需要电源适配器,而由 USB 供电。

4.3.2 ICE66与计算机的连接

ICE66 与计算机在线路连接上非常简单,只要用配属的 USB 电缆把两者相连即可。对于 USB 驱动程序,在安装完 Rice66 程序后,系统会自动安装。

4.3.3 ICE66与目标板的连接

在与目标板连接前,ICE66 的各个模块必须先连接好。再通过排线与目标板相连。

仿真器完成与目标板的连接后,注意上下电的顺序。上电时先打开仿真器的电源,下电(关电)时先关断目标板的电源。若不按此顺序,仿真工具可能运行不正常。

仿真器的电源可以是 USB 电源,也可是外接电源适配器,在 ICE66 中有跳线供选择。ICE66 出厂时,缺省选择是 USB 电源。USB 电源能提供 5V/200mA 电流给用户。

4.3.4 启动ICE66仿真器

仿真器时钟频率设定在前面的项目管理中已介绍。

选择 Project 菜单中 Download 命令或工具条中 Download 命令,代码由 USB 口下载至 ICE66 中,之后能够进行硬件调试。

4.3.5 程序运行控制方式

1.程序“复位”

在调试程序的停止运行的状态下可以让仿真板上的 EV 复位,复位程序指针(PC 值)至地址 0x0000 处,同时 EV 内部的一些寄存器数值变为系统初始值。“复位”的实现是通过单击快捷图标  ,或通过菜单 Run→Reset Process,或键盘热键 F5 启用。

2.启动程序“全速运行”

“全速运行”可以通过快捷图标  ,或通过菜单 Run→Go,或用键盘热键 F7 启动。启动后程序从当前指令处开始全速运行。在全速运行时,有些调试控制的快捷图标或菜单选项将被禁止(用灰色表示),在 Rice66 窗口的状态条左中部将显示信息“Running”,并且整个状态条为黄色。

3.程序“复位且运行”

在调试程序过程中经常要复位后再全速运行,可以选择复位且运行。“复位且运行”方式可以通过单击快捷图标  ,或通过菜单 Run→Go From Reset,或键盘热键 F6 启用。

4.程序“段运行”

程序段运行可以通过菜单 Run→From,或用键盘热键 ALT+F7 弹出的 Run From...对话框中输入起始运行地址和终止运行地址,单击 Run 键即可启动程序段全速运行。若程序运行过终止地址或遇断点停止运行,否则一直全速运行。

5.程序“运行至光标”

”运行至光标”可以通过菜单 Run→Run to Cursor,或用键盘热键 CTRL+F7 启动程序从当前地址全速运行,当程序运行过光标所在的指令行后停止。Rice66 先在光标所在行设置断点,程序停止后再取消此断点。此命令可以让用户省去先设定又取消断点的麻烦,非常适用于那些偶尔需要停一次看看结果的地方。要注意光标不是鼠标箭头,在 Rice66 的编辑窗口中光标一般是一个闪动的黑竖线。若光标所在行非指令行,则无法执行“运行至光标”。

6.程序“终止运行”

在程序全速运行或单步连续运行时,可以通过单击快捷图标  ,或通过菜单 Run→Stop,或用键盘热键 F5 终止当前程序的运行。程序运行停止后,在 Rice66 的源程序窗口内会显示出当前停留处的指令,同时按程序运行后的结果更新打开着的寄存器窗口,内存窗口和变量观察窗口的数据内容。在程序被终止后,你可以从当前停留处的指令开始单步或连续运行。

7.程序“单步运行”

“单步运行”,顾名思义即为程序在你的控制下一次只执行一条指令。单步运行可以仔细观察每条指令运行后的结果。可以通过单击快捷图标  ,或通过菜单 Run→Step

Into,或用键盘热键 F8 控制程序的单步运行。当然,由于单步运行在调试过程中频繁用到,所以用快捷图标或热键是最有效的方法。程序每执行一步后,程序指针就会指向下一条要执行的指令,并更新打开着的各类寄存器显示窗口和变量观察窗口中的数据内容,以分析这条指令的运行结果是否正确。在单步运行停下来后,你可以从当前停留处开始连续运行。

8.程序“单步跨过运行”

对于一般的程序指令,“ 单步运行”和“单步跨过运行”的效果是一模一样的。唯一不同的是针对子程序调用指令call的处理。如果当前程序指针正指向一条call指令,单步运行 call 指令后,程序将停留在子程序内的第一条指令处,即单步运行可以跟踪进入子程序内部,一条一条地执行指令直到遇到子程序的返回指令才退出,停留到原 call 指令的下一条指令处。用单步跨过方式运行 call 指令时,它把所调用的子程序内全部指令一次执行完毕,然后直接停留在 call 的下一条指令处,就好像把 call 看成一条“大指令” 。这一条功能主要是为了提高调试效率,如果确信子程序没有问题,那就没有必要再去一条条地跟踪进去,只需看结果就行。”单步跨过运行”的控制方式是通过单击快捷图标  ,或通过菜单 Run→Step Over,或键盘热键 F9 启用。

9.程序”单步跨出运行”

“单步跨出运行”只能作用于子程序代码。不管因何种原因使指令运行停止于子程序内某处时,用”单步跨出运行”方式可以让子程序内剩下的指令一次连续运行直到执行完任何形式的返回指令后再停下来,即停在原 call 指令的下一条指令处。这也是提高调试效率的一种方法。“单步跨出运行”的控制方式是通过单击快捷图标  , 或通过菜单 Run→Step Out,或键盘热键 Shift+F9 启用。

10.程序“单步连续运行”

如果有一组代码需要单步运行,并观察运行后的结果,可以用上面介绍的“单步运行”方式连续不断地单击相关的快捷图标或连续按 F8 热键。但也可以选择单步连续运行。单步连续运行时,每执行完一条指令,就更新相关观察数据的内容,稍停片刻和自动继续下一条指令的单步运行,如此一直运行直到使用程序“终止运行”的操作。“单步连续运行”方式可以通过单击快捷图标  ,或通过菜单 Run→Animate,或键盘热键 Alt+F8 启用。”单步连续运行”的速度是可以通过菜单 Run→Animate Speed 设置。

Rice66 提供简便设置的程序指针的方法,通过快捷图标  可以直接让程序指针强行设到当前光标所在的指令行上,而不是通过运行后停在此指令处。这一直接设定程序指针的功能在做一些局部代码调试时比较有用。

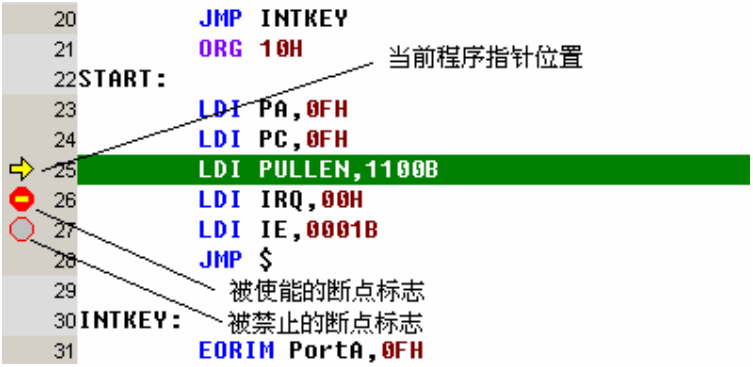
4.3.6 断点的设定和取消

在程序调试时,设定断点是一个非常重要的调试手段。当调试一个复杂的程序时,用户不可能用前面介绍的各种单步把整个程序调试一遍。最经常用到的调试方法是让程序全速运行,然后让其在所关心的地方自动停下来,再配合单步运行等调试手段对局部程序进行细调。程序能自动停下来的地方就是所谓的断点。

Rice66 可以允许设定任意多个断点。当程序处于全速运行时,如果碰到已被设定了断点的指令,在该条指令运行完后停下来,程序指针显示指向断点处的指令的下一条运行指令,等待你的下一步指示。

Rice66 提供了多种断点设定方法。第一种方法是通过菜单 Debug→Toggle Breakpoint,或快捷图标  ,或快捷键F2在光标所在行设置断点,若已经有断点存在,则取消已存在断点。第二种方法是通过菜单 Debug→Set Breakpoint...,或快捷键 CTRL+B 弹出 Set Breakpoint 对话框中,选择项目中源文件及直接输入行号,再单击 Add 键添加设置断点,已添加或设置的所有断点显示在 Breakpoint 框中。对已添加或设置的断点在 Breakpoint 框中选中,再单击 Remove 键取消此断点,或单击 Disable 键禁止此断点。对于被禁止的断点,可以在选中的情况下,单击 Enable 键使能此断点。被使能的断点在程序行前有一红色圆点标记“  ”。已设置的断点被禁止后在程序行前有一个红色的圆圈标志“”,断点被禁止后程序程序就不会在此停住。取消所以使能或禁止的断点可以通过菜单 Debug→Clear All Breakpoint 或快捷图标  实现,此命令还能取消多次断点。

设定断点最方便的方法是用鼠标在程序行前的页边双击。双击鼠标为该程序行设定断点,若已经有断点存在,则双击后取消该断点。



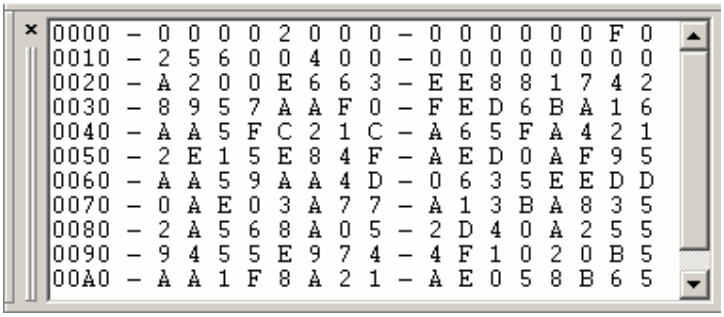
Rice66 还提供了一个多次断点方便用户程序调试。多次断点是程序运行经过此断点的次数等于设置的数值时,则停止程序运行。通过菜单 Debug→Set MultiBreakpoint at Cursor,或快捷键 CTRL+F2 弹出对话框 MultiBreakpoint Setting,输入计数值(数值范围 1～255),单击 OK 键,在光标所在程序行设置多次断点,若已设置过多次断点,则取消已设置的多次断点;若单击 Clear 键则取消已设置的多次断点。多次断点在程序行前有一红底黄字的标记“  ” 。

4.3.7 运行结果的观察窗口

一条程序指令运行的结果总是会改变一些东西,或者是寄存器的值变化,或者是某个内存单元的值被修改等等。我们开发调试的一个重要任务就是要观察这些是不是和设计预想相符合,如果不符,就表明程序或硬件有问题,就需要想办法去解决。Rice66 提供了多种观察窗口,可以让你观察芯片中几乎所有的数据资源,包括寄存器的观察窗口、内存显示窗口、特定自选变量的观察(Watch)窗口等。在此介绍几种最常用的观察窗口。

1.内存观察窗口

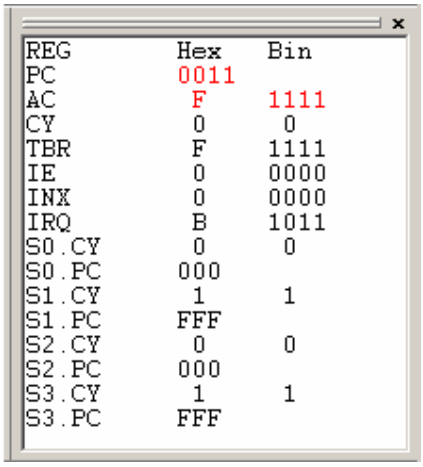
打开内存观察窗口的方式是通过主菜单 View→Debug Windows/Memory,或快捷图标  ,或快捷键 CTRL+3。若内存观察窗口已打开,则关闭内存观察窗口。内存包含用以控制硬件资源的寄存器和一般功能性的 RAM。当程序单步或碰到断点停止运行时,Rice66 会读取 EV 中的内存中的数据,并更新观察窗口内的数据内容,有内容变化的单元将以红色显示以提醒注意。观察窗口不仅仅让用户直接“看”内容,还可以让用户直接“修改”数据内容。用鼠标单击相关的数据项,选中后通过键盘直接输入新的数据值,即可修改内存数据处理。Rice66 将修改的数据写入 EV 内对应地址的 RAM,然后再读取 EV 内存数据更新显示。用鼠标双击相关的数据项,会弹出对话框 Inspect/Modify,可以整块的修改数据。内存观察窗口一行能显示的内存数量取决于窗口的宽度,但一行最多显示 16 个内存单元。



2.寄存器观察窗口

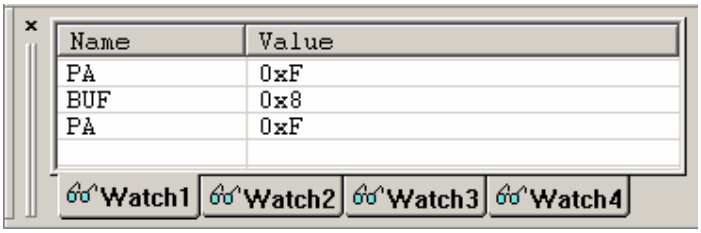
打开寄存器观察窗口的方式是通过主菜单 View→Debug Windows/Register,或快捷图标  ,或快捷键 CTRL+5。若寄存器观察窗口已打开,则关闭寄存器观察窗口。打开后寄存器数值有十六进制和二进制两种格式显示。

此观察窗口的作用与内存观察窗口相同,你可以看寄存器的内容,寄存器内容改变后也将以醒目的红色突显。观察窗口中堆栈数值不能修改,其它寄存器数值都可修改,通过窗口内寄存器所在行鼠标双击,弹出对话框 Inspect/Modify,直接输入数值在 Data 框中,之后单击 Write 键后,Rice66 会将数值写入 EV 中,再从 EV 中读出寄存器并更新显示。寄存器观察窗口中 TBR、IE、INX、IRQ 实际是内存单元。



3.特定自选变量观察(Watch)窗口

在大多数情况下,在调试程序时只需关心少量且特定的数据信息。这时可以通过主菜单 View→Debug/Watch,或快捷图标,或快捷键 CTRL+4 打开一个自选变量的观察窗口,它可以让用户自由添加任何自己程序定义的符号变量。若 Watch 观察窗口已打开,则关闭 Watch 观察窗口。



当需要观察自定义的变量时,可以用鼠标单击观察窗口内 Name 栏下紧临的空白格,然后直接输入变量符号,回车完成输入,在 Value 栏下会显示对应变量的值。也可以通过主菜单 Debug→Add Watch,或快捷键 CTRL+W 弹出 Set Watch Variables 对话框,其中 Variable Labels 列出程序中定义所有变量,Watch Variables 列出对应观察页中添加的所有变量,在 Variable Labels 中选中变量,单击 Add >>键,此变量添加至 Watch Variables 栏中;若在 Watch Variables 栏中选中变量,单击 Remove 键,此变量便移出,最后单击 Close 键完成观察窗口变量添加设置。

在 Watch 窗口的底部有四个观察页选择按钮,可以在不同的观察页内添加不同的变量组合以针对程序不同代码段调试信息的需要。通过 Watch 观察窗口不但能观察变量数据内容,还能修改数据值,方法是用鼠标单击数据所在栏,选中数据,之后直接修改数据,回车后完成数据修改。同样数据先写入 EV,再读出更新显示。

4.反汇编窗口

通过主菜单 View→Debug/Disassembly,或快捷图标,或快捷键 CTRL+6 可以打开反汇编窗口,此窗口显示了与源程序一一对应的反汇编文件。

4.3.8 代码执行的跟踪功能

能真正体现 Rice66 强大调试功能是其代码跟踪能力,利用此调试手段可以轻松诊断埋藏很深的软件故障。

1.跟踪窗口打开方式

Rice66 提供了 8K 深度的跟踪记录,每一个记录数据宽度为 24 位,能在程序暂停时追踪查看前面执行过的 8192 条指令运行的结果。用户可以用菜单命令 View→Debug/Trace,或快捷图标  ,或快捷键 CTRL+7 打开代码跟踪窗口。若代码跟踪窗口已打开,关闭代码跟踪窗口。无论代码跟踪窗口是否打开,用户通过主菜单 Debug→Read Trace Buffer,或快捷键 CTRL+T 都可打开代码跟踪窗口,并显示在最前面。

2.跟踪窗口显示信息详解

跟踪行号(Line)

显示从 1 至 8192

取指地址(Addr)

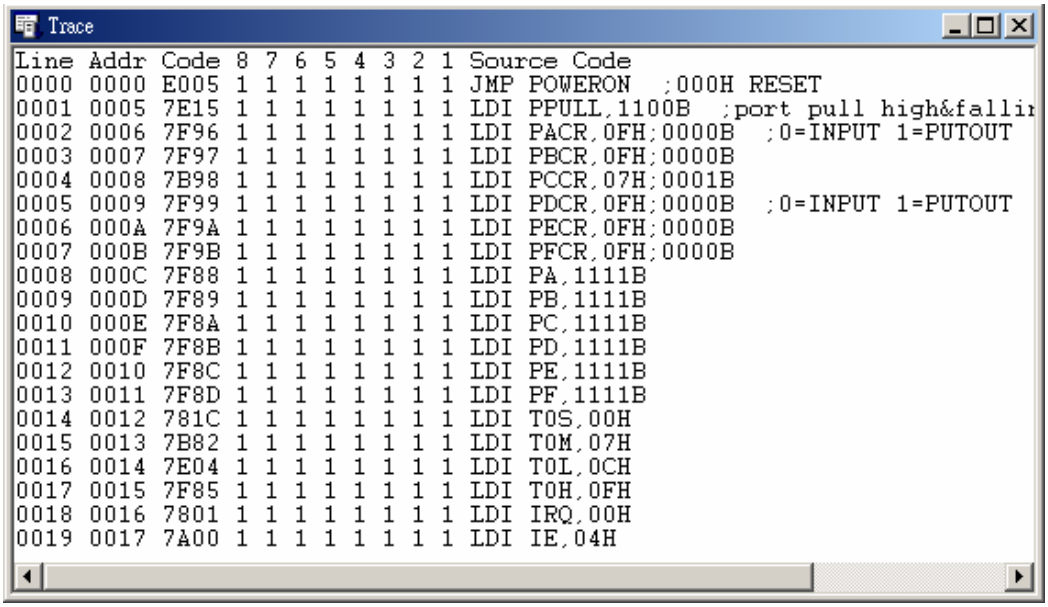
它指示的是每一条指令的存放地址。

指令操作码(Code)

它代表的是每一条指令十六进制机器码。

外部信号跟踪记录(8～1)

这 8 栏内显示的是,在指令运行的每一个指令周期时,8 路外部信号输入的电平值。这 8 路数字信号可以用排线把目标板上的实际信号连接到主机的逻辑分析探头处进行跟踪。用户可以通过指令运行和外部电平实际变化的跟踪记录来诊断一些软硬件配合的故障。而这种类型的故障光靠单步运行或简单的设定几个断点是很难发现的。



源程序代码

显示指令对应的源程序行内容。

3.跟踪窗口的控制命令

键盘的 Home 键使跟踪窗口内最顶行可见。键盘的 End 键使跟踪窗口内最底行可见。鼠标双击跟踪窗口内某行,将激活源程序窗口直接定位到对应的源程序代码位置。在跟踪窗口内鼠标右键单击,右键菜单有保存跟踪窗口内容至文件和查找对话框。查找对话框输入要查找的字符串,单击 Search 键,在跟踪窗口内查找符合要求的特定字符串,若找到,则特定字符串所在行可见。

ICE66 可以跟踪记录每一条指令的运行过程,也可以记录某段指令的运行。通过主菜单 Debug→Set Trace Range 弹出 Trace Setting...对话框,可以输入跟踪指令的起始地址和结束地址,若选择 Disable 项,则跟踪每一条运行指令,不考虑起始和结束地址;若选择 Enable Halt Trace 项,记录从起始地址开始至结束地址之间的运行指令,其它运行指令不记录,若运行从此指令区跳出,就停止程序运行;若选择 Enable Real-Time Trace 项,则记录从起始地址开始至结束地址之间的运行指令,其它运行指令不记录。

通过主菜单 Debug→Cear Trace Buffer,或复位操作都会清除跟踪窗口内的显示内容。

4.4 OTP芯片编程烧写工具Pgm66和Pro03(以下部分只针对OTP产品)

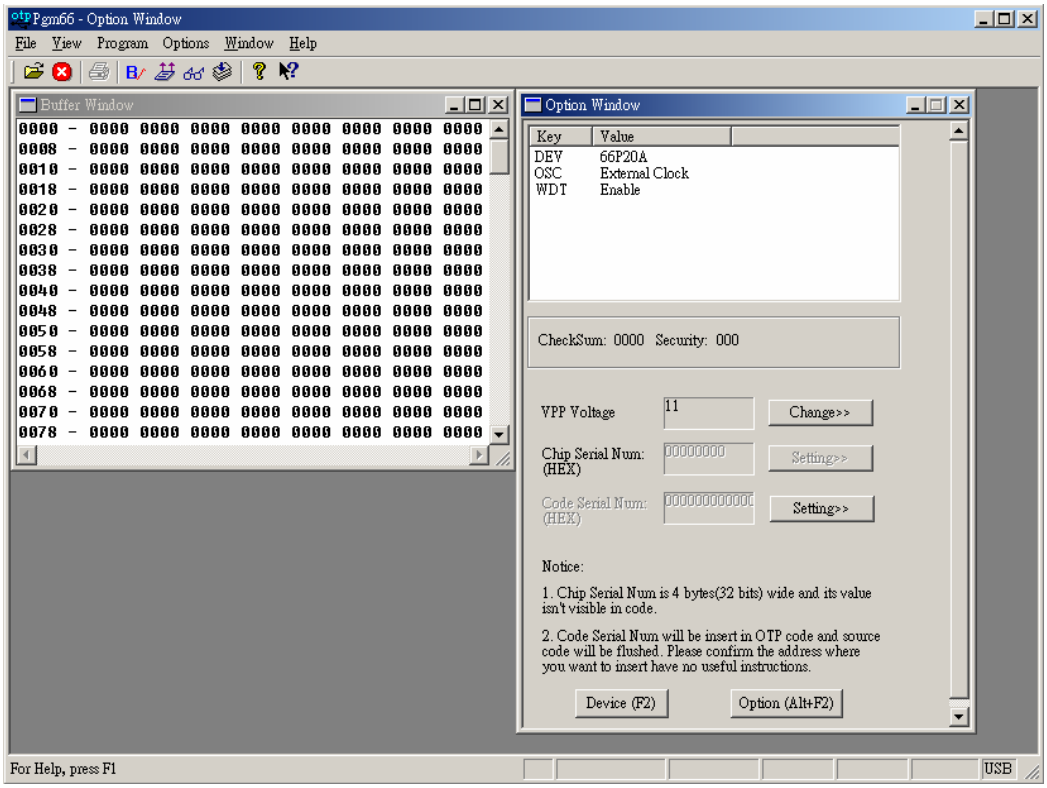
程序开发调试的最后一步工作就是要代码烧入到 OTP 单片机,然后在目标板上让其独立运行,在各种实验条件下验证整个系统设计的可靠性。实现代码烧写就需要用到编程工具。Sino Wealth 提供了两种通用的烧写工具,分别是实验开发型编程工具 Pgm66 和专业生产型编程工具 Pro03。

4.4.1 Pgm66

Pgm66 是为实验开发目的而设计的编程器,其编程接口在 ICE66 的左下部。ICE66 上的 DIP(OTP Port1)插座支持对 66P22A,66P20A,66P31A 和 69P20 的烧写,其它 OTP 芯片的烧写引脚在侧面(OTP Port2) 。

4.4.1.1 Pgm66的启用

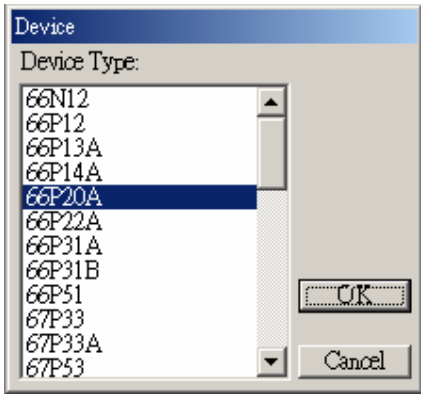
在 Rice66 中 Tool 菜单中选择 Pgm66,会出现 Pgm66 运行程序界面



4.4.1.2 Pgm66的操作和使用

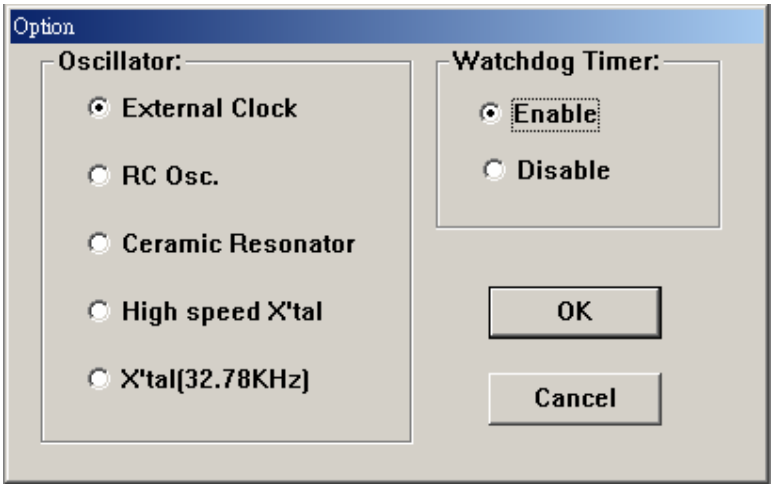
芯片类型选择(Program→Device)

每种 OTP 芯片 ROM 大小可能不同,选项基本不同,所以必须选择要编程芯片的类型。选择 Program→Device 命令后,弹出设备对话框,列出所有现在支持的芯片类型,选中要编程芯片的类型,再单击 OK 键完成选择。



芯片选项选择(Program→Option)

由于每种 OTP 芯片选项基本不同,所以要单独设置。选择 Program→Option 命令后,弹出选项对话框,这种芯片类型的所有选项及每个选项的内容,选中要编程选项,再单击 OK 键完成选择。



芯片查空命令(Program→Blank Check 或快捷图标)

如果想知道一颗芯片是否是“空”的,即没有写入任何东西,可以用此查空命令。查空命令会检查 Option 和 Memory 是否已烧写。

芯片编程 Option 命令(Program→Program Option 或快捷键 ALT+F5)

使用此命令可以把设置好的 Option 烧写到芯片内 Option 区。

芯片编程 Memory 命令(Program→Program Memory 或快捷键 ALT+F7)

使用此命令可以把现成的程序代码烧写到芯片内。程序代码可以用菜单命令 File→Open File 从外部装入的 BIN 文件,也可以从芯片中读取的代码。

芯片检验命令(Program→Verify 或快捷图标)

用此命令可以把已经烧写的芯片和 Pgm66 内的数据做对比校验,当然芯片没有烧写保密位,不然你无法读出片内的数据,也就无从校验。

芯片编程 Security 命令(Program→Program Security)

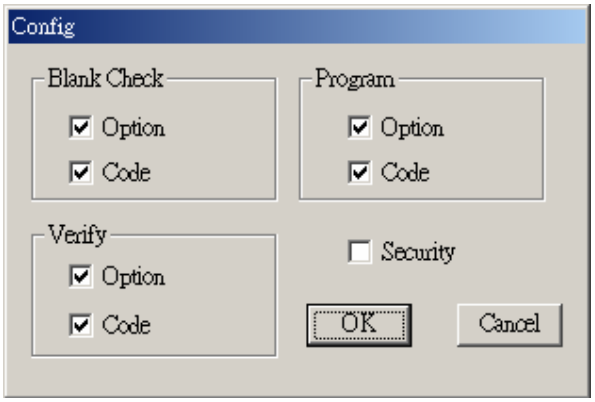
用此命令可以实现对芯片内程序代码保密,之后无法读取芯片内程序代码,但芯片内 Option 可以读取。

芯片读取命令(Program→Read 或快捷图标)

此命令可以把没有代码保护的芯片内的所有数据读回到 Pgm66 内相关的数据窗口。可以用读取一颗”母片”的方式实现多个芯片的复制。

芯片自动编程(Program→Auto Program 或快捷图标)

此命令弹出 Config 对话框,可以选择需要的编程选项,之后单击 OK 键,开始执行。此命令包含上述的查空,编程 Memory,编程 Option,校验和编程 Security 等命令组合。



4.4.1.3 Pgm66观察窗口

Pgm66 提供多个窗口显示信息,有 Buffer Window, Option Window, Verify Window, Disassembly Window 等多个窗口。

Buffer Window(View→Buffer Window)显示芯片读取命令读到的指令码,或通过开启文件,读取 obj 文件中的指令码。

Option Window(View→Option Window)显示芯片读取命令读到的 Option 设置,或要烧写的 Option 设置。

Disassembly Window(View→Option Window)显示对应 Buffer Window 中指令码的反汇编指令。

4.4.1.4 Pgm66特性

Pgm66 提供 32 比特的 Chip Serial Num,不占用 ROM,供客户使用。

Pgm66 采用了一种库驱动形式,方便了对新 OTP 芯片的烧写支持。客户从中颖网站下载专区中开发工具相关下载中选择 OTP 库更新文件下载,下载完成后,解压并运行 OTP_LibUpdate_Vx.x.exe,选择Pgm66或Pro-03运行文件所在路径,完成安装后,即可支持对最新 OTP 芯片的烧写支持。

4.4.2 Pro-03

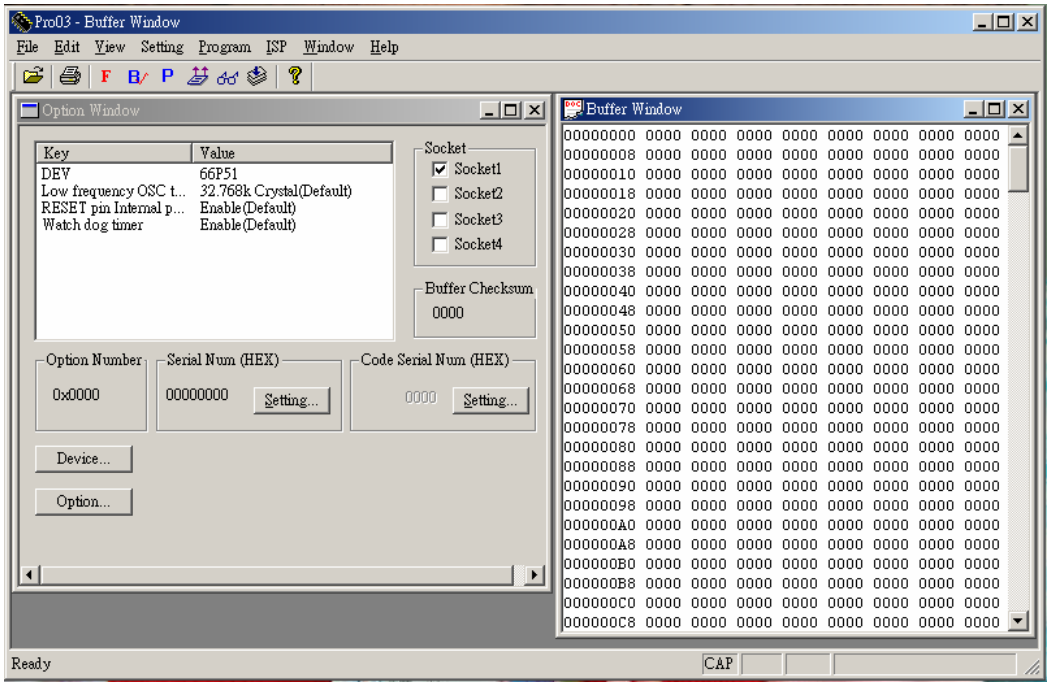
Pro-03 是 SinoWealth 公司为批量生产而设计制造的高性能编程工具。它的设计思路也是采用积木组合的方式,有一个主机,配以不同的芯片适配器,与计算机也是通过 USB 口相连。

4.4.2.1 Pro-03的安装

安装时先从网上下载一个压缩文件 Pro-03-WIN-V0.XX.zip,解压后,运行安装文件

Setup.exe,就进入自主引导的安装流程。如果都选择缺省安装选项,则安装结束后所有软件文件都放在“C:\Program Files\SinoWealth”目录下。在以下章节中凡涉及 Pro-03 安装路径时,如果没有特殊说明就是指此缺省安装路径。

为了支持硬件仿真器和烧写编程器,在 Pro-03 安装完毕后还需要另外安装 USB 驱动程序。硬件仿真器或烧写编程器在第一次接入时 Windows 98 或 Windows 2000 或 XP 操作系统会自动发现新硬件并自动安装驱动。安装成功后执行 Pro-03 文件,即可看到类似图所示的窗口界面。



4.4.2.2 Pro-03的运行/控制介绍

Pro-03 有单独的 PC 端运行程序 Pro-03.exe。Pro-03 的菜单与 Pgm66 基本相似。但不提供对 66P22A,66P20A,66P31A,69P20,66P12,66N12,66P13A 和 66P14A 等芯片支持。与 Pgm66 相比,它具备如下的优点:

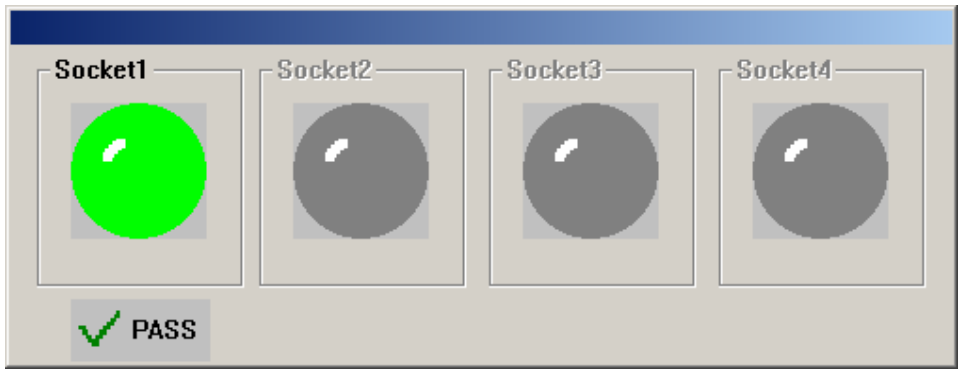
1. 可以脱离计算机独立工作. Pro-03 主机面板上有一个八段数码管,三个发光二极管,一个按键和两组拨动开关,有自己独立的一套操作环境. 生产时通过 PC 将烧写内容下载至主机内的 FLASH 中.

2. 主机提供四个烧写接口,能顺序烧写。

3. 可以开启多种文件格式, 后缀为.obj 的文件只包含指令代码,不包含 Option 设置;后缀为.opf 或.wrt 的文件既包含指令代码又包含 Option 设置。也可将指令代码和 Option 设置保存至.opf 的文件。

4. 更加清晰明显的状态信息显示,见下图,绿色代表操作通过(PASS);红色代表操作失败(FAIL);黄色代表操作中,如读(Read…),烧写(Program…),校验(Verify…),查空

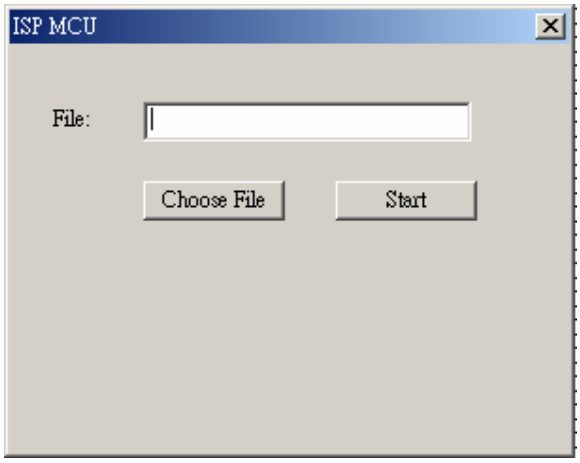
(Blank Check…)。



4.4.2.3 软件自动升级功能

在选择了主菜单 Setting→Configuration 对话框中 Auto Update Detected 后,Pro-03 启动时会查找中颖网站中最新版本信息,若发现 Pro-03 版本低于网上版本,会弹出提示。

PC 端软件升级参照 Pro-03 安装,对于 Pro-03 硬件编程器的控制软件,选择主菜单 ISP→ISP MCU,弹出对话框



通过 Choose File 选择要升级的 MCU 文件,之后单击 Start,启动系统控制软件在线更新。按提示完成控制软件升级。

第五章

Sino Wealth 4 bit 单片机一些应用实例

5.1 4位7段LED显示

在单片机应用系统中经常使用发光二极管来显示，发光二极管简称 LED（Light Emitting Diode)。LED 的价格便宜，而且配置比较灵活，与单片机的接口也比较方便。在这里将讲解如何使用中颖的单片机进行 4 位 7 段 LED 显示的方法。

5.1.1 7段LED的结构原理

单片机中经常使用 7 段 LED 来显示数字，也就是用 7 个 LED 构成字型“8”，并另外用一个圆点 LED 来显示小数点，也就是说一共有 8 个 LED，构成了“8.”的字型。

7 段 LED 分共阴级和共阳极两种，共阴级 7 段 LED 的原理图和管脚配置图如图 5-1 所示，共阳级 7 段 LED 的原理图和管脚配置图如图 5-2 所示。实际中，各个型号的 7 段 LED 的管脚配置可能不会是一样的，在实际应用中要先测试一下各个管脚的配置，再进行电路原理图的设计。

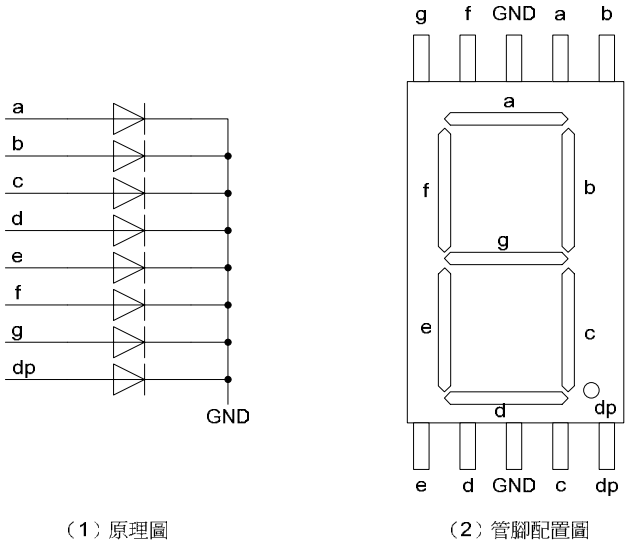


图5-1 共阴极7段LED

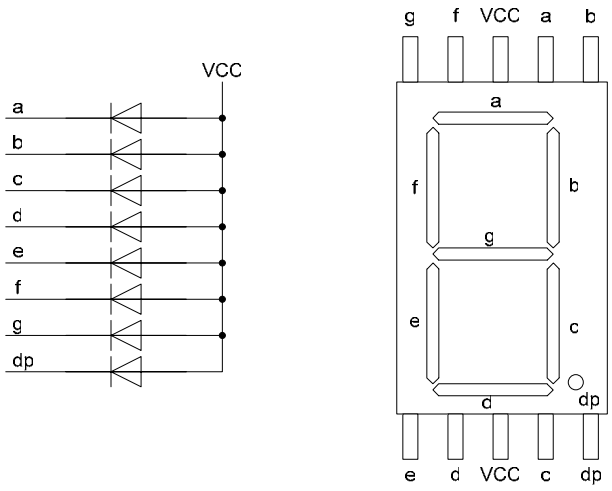


图 5-2 共阳极 7 段 LED

共阳极 7 段 LED 是指发光二极管的阳极连接在一起为公共端的 7 段 LED，而共阴极 7 段 LED 是指发光二极管的阴极连接在一起为公共端的 7 段 LED。一个 7 段 LED 由 8 个发光二极管组成，其中 7 个发光二极管构成字型“8”的各个笔划（a～g），另一个发光二极管为小数点（dp）。

当在某一段发光二极管上施加一定的正向电压时，该段 LED 即被点亮；不加电压则为暗。以共阳极 7 段 LED 为例，若是要显示“5.”，则需要在 VCC 上加上电压，向 dp、g、f、e…、a 送出 00010010 的信号，就能显示出来。

为了保护各段 LED 不因电流过大而损坏，需在各个段上外加限流电阻保护。

共阳极 7 段 LED 显示 0～F 的编码表如表 5-1 所示（以 dp 为最高位，a 为最低位）。

表 5-1

显示字符	dp	g	f	e	d	c	b	a	段选码
0	1	1	0	0	0	0	0	0	C0H
1	1	1	1	1	1	0	0	1	F9H
2	1	0	1	0	0	1	0	0	A4H
3	1	0	1	1	0	0	0	0	B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H
6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	F8H
8	1	0	0	0	0	0	0	0	80H

9	1	0	0	1	0	0	0	0	90H
A	1	0	0	0	1	0	0	0	88H
B	1	0	0	0	0	0	1	1	83H
C	1	1	0	0	0	1	1	0	C6H
D	1	0	1	0	0	0	0	1	A1H
E	1	0	0	0	0	1	1	0	86H
F	1	0	0	0	1	1	1	0	8EH

5.1.2 7段LED动态显示原理

LED 的静态显示虽然有编程容易、管理简单等优点，但是静态显示所要占的 I/O 口资源很多，所以在显示的 LED 点较多的情况下，一般都采用动态显示方式。

在多位 7 段 LED 显示中，为了简化电路，降低成本，则将所有位的段选线并联在一起，刚好由 8 个 I/O 口来控制 8 个段。而公共端（共阳极/共阴极）则分别由相应的 I/O 口控制，以实现各个位的分时选通。

原理图如图 5-3 所示。

由于所有的段选线并联到同一个 I/O，由这个 I/O 口来控制，因此，若是所有的 4 位 7 段 LED 都选通的话，4 位 7 段 LED 将会显示相同的字符。要使各个位的 7 段 LED 显示不同的字符，就必须采用动态扫描方法来轮流点亮每一位 7 段 LED，即在每一瞬间只选通一位 7 段 LED 进行显示单独的字符。在此段点亮时间内，段选控制 I/O 口输出要显示的相应字符的段选码，而位选控制 I/O 口则输出位选信号，向要显示的位送出选通电平（共阴极则送出低电平，共阳极则送出高电平），使得该位显示相应字符。这样将四位 7 段 LED 轮流去点亮，使得每位分时显示该位应显示的字符。由于人眼的视觉暂留时间为 0.1 秒，当每位显示的间隔未超过 33ms 时，并在显示时保持直到下一位显示，则由于人眼的视觉暂留效果眼睛看上去就像是 4 位 7 段 LED 都在点亮。设计时，要注意每位显示的间隔时间，由于一位 7 段 LED 的熄灭时间不能超过 100ms，也就是说点亮其它位所用的时间不能超过 100ms，这样当有 N 位的 7 段 LED 用来显示时，每一位间隔的时间 t 就必须符合下面的式子：

$$t \leqq 100ms/(N-1)$$

比如，现在使用 4 位，也就是 N＝4，则由式子可以算出 t≦33ms，就是每一位的间隔时间不能超过 33ms。当然时间可以也设得短一些，比如 5ms 或 1ms 也可以。

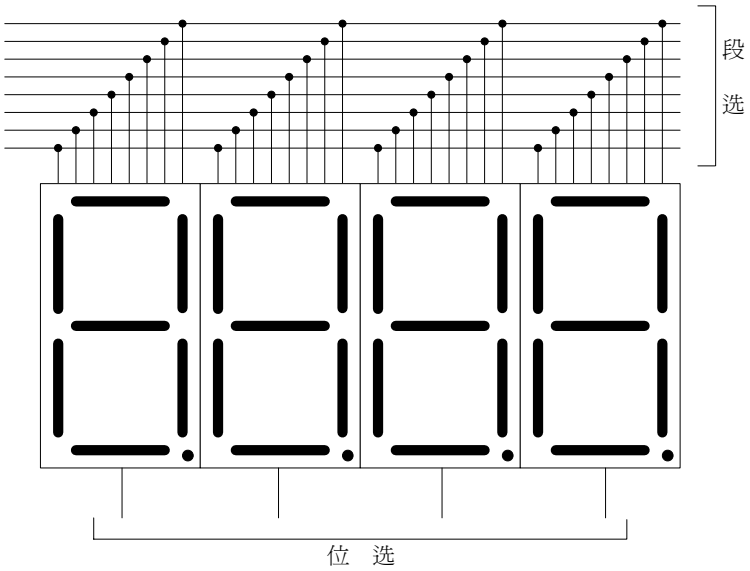


图 5-3　7 段 LED 动态显示原理图

5.1.3　7段LED与中颖单片机的接口及应用程序

■　电路原理图

以 SH69P43 控制芯片为例，4 位 7 段 LED 动态显示的电路原理图如图 5-4 所示。使用 4MHz 晶振作为主振荡器，PE 口和 PF 口控制 4 位 7 段 LED 的段选，PB 口控制 4 位 7 段 LED 的位选。将 4 个共阳极的 7 段 LED 的段选线并联起来接到 PE 口和 PF 口，dp、g、f 和 e 段选线接到 PF3～PF0 上，d、c、b 和 a 段选线接到 PE3～PE0 上；将 4 个公共端 VCC 分别接到 PB 口上。

由于 SH69P43 的 I/O 口电流较小,所以在位选端要使用三极管来进行 7 段 LED 的推动，向 7 段 LED 提供足够大的电流来点亮 LED。在中颖单片机中，有些芯片的 I/O 口是可以提供至少 200mA 的反向电流，比如 SH69P26 和 SH69P55，SH69P26 有 6 个大电流 I/O 口（PORTA 和 PORTD1～PORTD0），SH69P55 也是有 6 个大电流 I/O 口（PORTD 和 PORTE1～PORTE0）。当使用这些有大驱动电流的 I/O 口的芯片来做 7 段 LED 显示时，当显示电路的位数不超过芯片所带有的大驱动电流 I/O 数时，比如在 SH69P26 和 SH69P55 中做 6 位以内的 7 段 LED 显示时，就无需使用三极管，而可以直接去驱动 LED 了。要注意的是，由于这些 I/O 口是提供大的反向电流，也就是说，电流是流向 I/O 的，所以只能使用共阴极的 7 段 LED，而不能使用共阳极的 7 段 LED。

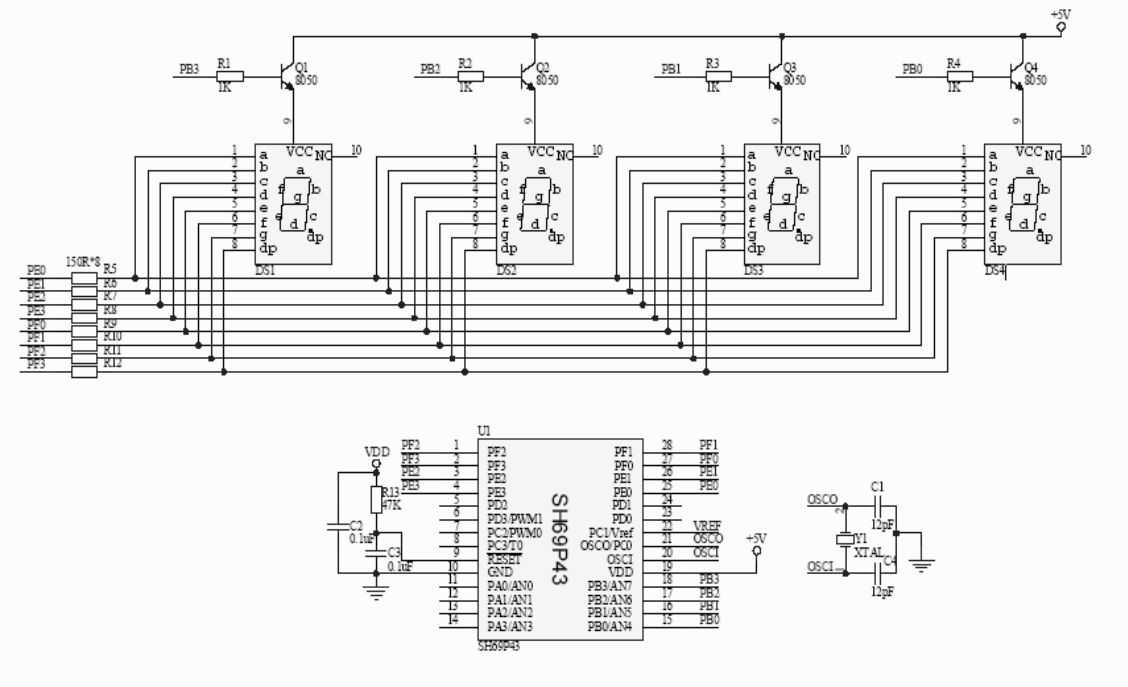


图 5-4　4 位 7 段 LED 显示电路原理图

■　程序

如图 5-4 的电路原理图，现以程序来举个例子。以 SH69P43 为控制芯片，4M 晶振为主振荡器，以动态扫描方式驱动 4 位 7 段 LED，1ms 扫描一个位，4 位循环扫描。程序中有个加载数据的地方只是为了测试显示设定的,实际应用中可在那里更新所要显示的数据。当全速运行程序时，就能从 4 位 7 段 LED 中看到所要显示的字符。

例[5-1] 4 位 7 段 LED 显示

```
LIST P=69P43
ROMSIZE=3072
;************************************************
; 系统寄存器
;************************************************
IE          EQU         00H         ;中断使能标志
IRQ         EQU         01H         ;中断请求标志
TM0         EQU         02H         ;Timer0 模式寄存器
TL0         EQU         04H         ;Timer0 装入/记数寄存器低四位
TH0         EQU         05H         ;Timer0 装入/记数寄存器高四位
```

```
TBR         EQU         0EH         ;查表寄存器
PORTB       EQU         09H         ;Port B 数据寄存器
PORTE       EQU         0CH         ;Port E 数据寄存器
PORTF       EQU         0DH         ;Port F 数据寄存器
INX         EQU         0FH         ;间接寻址伪索引寄存器
DPL         EQU         10H         ;INX 数据指针低四位
DPM         EQU         11H         ;INX 数据指针中三位
DPH         EQU         12H         ;INX 数据指针高三位
PBCR        EQU         19H         ;PortB 输入/输出控制寄存器
PECR        EQU         1CH         ;PortE 输入/输出控制寄存器
PFCR        EQU         1DH         ;PortF 输入/输出控制寄存器
;***********************************************
; 用户定义寄存器
;***********************************************
AC_BAK      EQU         30H         ;AC 值备份寄存器
PB_BAK      EQU         32H         ;PortB 数据备份寄存器
PE_BAK      EQU         35H         ;PortE 数据备份寄存器
PF_BAK      EQU         36H         ;PortF 数据备份寄存器
;-------------------------------------
; 用于 TIMER 定时
T1MS_CT     EQU         37H         ;计数值=04H，定时 1ms
;-------------------------------------
F_TIMER     EQU         39H         ;bit0=1, 1ms 到
FLAG1       EQU         3AH         ;bit0=1, 按键未松开
;-------------------------------------
; Used for display
DISP_R1     EQU         3BH         ;第一位 7 段 LED 显示的字符
DISP_R2     EQU         3CH         ;第二位 7 段 LED 显示的字符
DISP_R3     EQU         3DH         ;第三位 7 段 LED 显示的字符
DISP_R4     EQU         3EH         ;第四位 7 段 LED 显示的字符
DISP_PT     EQU         3FH         ;位选指针
;***********************************************
; 程序
;***********************************************
            ORG         0000H
            JMP         RESET
            RTNI
            JMP         TIMER0_ISP      ;TIMER0 中断程序入口地址
```

```
            RTNI
            RTNI
;*******************************************
; TIMER0 中断服务程序
;*******************************************
TIMER0_ISP:
            STA         AC_BAK,00H              ;备份 AC 值
            ANDIM       IRQ,1011B               ;清 TIMER0 中断请求标志
J1MS:
            SBIM        T1MS_CT,01H
            BNZ         TIMER0_ISP_END          ;未到 1ms, 跳转
            LDI         T1MS_CT,04H             ;重置 1ms 计数器
            ORIM        F_TIMER,0001B           ;设置 "1ms 到"标志
TIMER0_ISP_END:
            LDI         IE,0100B                ;打开 TIMER0 中断
            LDA         AC_BAK,00H              ;取出 AC 值
            RTNI
;*****************************************
; 主程序
;*****************************************
RESET:
            NOP
;-----------------------------------------------
;清用户寄存器
;-----------------------------------------------
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
            LDI         DPH,00H
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TBR,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
```

```
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,04H
            BNZ         POWER_RESET_1
;-------------------------------------------
;初始化系统寄存器
;-------------------------------------------
SYSTEM_INITIAL:
            ;TIMER0 初始化
            LDI         TM0,07H             ;设置 TIMER0 预分频为/1
            LDI         TL0,06H
            LDI         TH0,00H             ;设置中断时间为 250us
            LDI         T1MS_CT,04H         ;定时 1ms
            ;I/O 口初始化
            LDI         PORTB,00H
            LDI         PBCR,0FH            ;设置 PortB 作为输出口
            LDI         PORTE,0FH
            LDI         PECR,0FH            ;设置 PortE 作为输出口
            LDI         PORTF,0FH
            LDI         PFCR,0FH            ;设置 PortF 作为输出口
;-------------------------------------
MAIN_PRE:
            LDI         IRQ,00H
            LDI         IE,0100B            ;打开 Timer0 中断
MAIN:
            ADI         F_TIMER,0001B
            BA0         HALTMODE            ;未到 1ms,跳转
            ANDIM       F_TIMER,1110B       ;清 "1ms 到"标志
;-----------------------------------------------------------
; 加载显示数据(用于测试 7 段 LED 显示模块)
;-----------------------------------------------------------
            LDI         DISP_R1,03H
            LDI         DISP_R2,07H
            LDI         DISP_R3,09H
            LDI         DISP_R4,0FH
*****************************************************************************
* 模块:    4 位 7 段 LED 显示模块                                  *
```

```
* 输入变量:      DISP_R1,DISP_R2,DISP_R3,DISP_R4            *
* 使用变量:      DISP_PT,TBR,PB_BAK,PE_BAK,PF_BAK           *
* 输出变量:      PORTB,PORTE,PORTF                          *
;*****************************************************************************
DISPLAY:
DISP_1:
          ADIM      DISP_PT,01H          ;指针加一
          SBI       DISP_PT,01H
          BAZ       DISP_11              ;显示位 1 数码管, 跳转
          SBI       DISP_PT,02H
          BAZ       DISP_12              ;显示位 2 数码管, 跳转
          SBI       DISP_PT,03H
          BAZ       DISP_13              ;显示位 3 数码管, 跳转
;-------------------------------------------------------------
;显示位 4 数码管
;-------------------------------------------------------------
DISP_14:
          LDI       DISP_PT,00H          ;指针清零
          LDI       PB_BAK,0001B         ;预设位选码
          LDI       TBR,0FH
          LDA       DISP_R4,00H
          CALL      07EFH
          STA       PE_BAK,00H
          LDA       TBR,00H
          STA       PF_BAK,00H           ;由字符查表得预设段选码的值
          JMP       DISPLAY_END
;-------------------------------------------------------------
;显示位 3 数码管
;-------------------------------------------------------------
DISP_13:
          LDI       PB_BAK,0010B         ;预设位选码
          LDI       TBR,0FH
          LDA       DISP_R3,00H
          CALL      07EFH
          STA       PE_BAK,00H
          LDA       TBR,00H
          STA       PF_BAK,00H           ;由字符查表得预设段选码的值
          JMP       DISPLAY_END
```

```
;-------------------------------------------------------
;显示位 2 数码管
;-------------------------------------------------------
DISP_12:
          LDI       PB_BAK,0100B        ;预设位选码
          LDI       TBR,0FH
          LDA       DISP_R2,00H
          CALL      07EFH
          STA       PE_BAK,00H
          LDA       TBR,00H
          STA       PF_BAK,00H          ;由字符查表得预设段选码的值
          JMP       DISPLAY_END

;-------------------------------------------------------------------
;显示位 1 数码管
;-------------------------------------------------------------------
DISP_11:
          LDI       PB_BAK,1000B        ;预设位选码
          LDI       TBR,0FH
          LDA       DISP_R1,00H
          CALL      07EFH
          STA       PE_BAK,00H
          LDA       TBR,00H
          STA       PF_BAK,00H          ;由字符查表得预设段选码的值
          JMP       DISPLAY_END
DISPLAY_END:
          LDI       PORTB,00H           ;关闭显示
          LDA       PE_BAK,00H
          STA       PORTE,00H
          LDA       PF_BAK,00H
          STA       PORTF,00H           ;送出预设的段选码到 I/O 口
          LDA       PB_BAK,00H
          STA       PORTB,00H           ;送出预设的位选码到 I/O 口，显示该位
;*****************************************
HALTMODE:
          NOP
          HALT
          NOP
```

```
            NOP
            NOP
            JMP         MAIN
;*********************************************
            ORG         07EFH
            TJMP
;-------------------------------------------
;显示段选码数据表（共阳极）
            ORG         07F0H
            ;dp g f e, d c b a
            RTNW  1100B,0000B           ;0
            RTNW  1111B,1001B           ;1
            RTNW  1010B,0100B           ;2
            RTNW  1011B,0000B           ;3
            RTNW  1001B,1001B           ;4
            RTNW  1001B,0010B           ;5
            RTNW  1000B,0010B           ;6
            RTNW  1111B,1000B           ;7
            RTNW  1000B,0000B           ;8
            RTNW  1001B,0000B           ;9
            RTNW  1000B,1000B           ;A
            RTNW  1000B,0011B           ;B
            RTNW  1100B,0110B           ;C
            RTNW  1010B,0001B           ;D
            RTNW  1000B,0110B           ;E
            RTNW   1000B,1110B          ;F

            END
```

5.2 蜂鸣器驱动模块

在单片机应用的设计上，很多方案都会用到蜂鸣器，大部分都是使用蜂鸣器来做提示或报警，比如按键按下、开始工作、工作结束或是故障等等。这里对中颖电子的单片机在蜂鸣器驱动上的应用作一下描述。

5.2.1 驱动方式

由于自激蜂鸣器是直流电压驱动的，不需要利用交流信号进行驱动，只需对驱动口输出驱动电平并通过三极管放大驱动电流就能使蜂鸣器发出声音，很简单，这里就不对自激蜂鸣器进行说明了。这里只对必须用 1/2duty 的方波信号进行驱动的他激蜂鸣器进行说明。

单片机驱动他激蜂鸣器的方式有两种：一种是 PWM 输出口直接驱动，另一种是利用 I/O 定时翻转电平产生驱动波形对蜂鸣器进行驱动。

PWM 输出口直接驱动是利用 PWM 输出口本身可以输出一定的方波来直接驱动蜂鸣器。在单片机的软件设置中有几个系统寄存器是用来设置 PWM 口的输出的，可以设置占空比、周期等等，通过设置这些寄存器产生符合蜂鸣器要求的频率的波形之后，只要打开 PWM 输出，PWM 输出口就能输出该频率的方波，这个时候利用这个波形就可以驱动蜂鸣器了。比如频率为 2000Hz 的蜂鸣器的驱动，可以知道周期为 500μs，这样只需要把 PWM 的周期设置为 500μs，占空比电平设置为 250μs，就能产生一个频率为 2000Hz 的方波，通过这个方波再利用三极管就可以去驱动这个蜂鸣器了。

而利用 I/O 定时翻转电平来产生驱动波形的方式会比较麻烦一点，必须利用定时器来做定时，通过定时翻转电平产生符合蜂鸣器要求的频率的波形，这个波形就可以用来驱动蜂鸣器了。比如为 2500Hz 的蜂鸣器的驱动，可以知道周期为 400μs，这样只需要驱动蜂鸣器的 I/O 口每 200μs 翻转一次电平就可以产生一个频率为 2500Hz，占空比为 1/2duty 的方波，再通过三极管放大就可以驱动这个蜂鸣器了。

5.2.2 蜂鸣器驱动电路

由于蜂鸣器的工作电流一般比较大，以致于单片机的 I/O 口是无法直接驱动的，所以要利用放大电路来驱动，一般使用三极管来放大电流就可以了。

蜂鸣器的驱动电路有很多种，这里举两个常用的例子，也是建议使用的驱动电路：

1. 无源压电式蜂鸣器、无源电磁式蜂鸣器（他激）

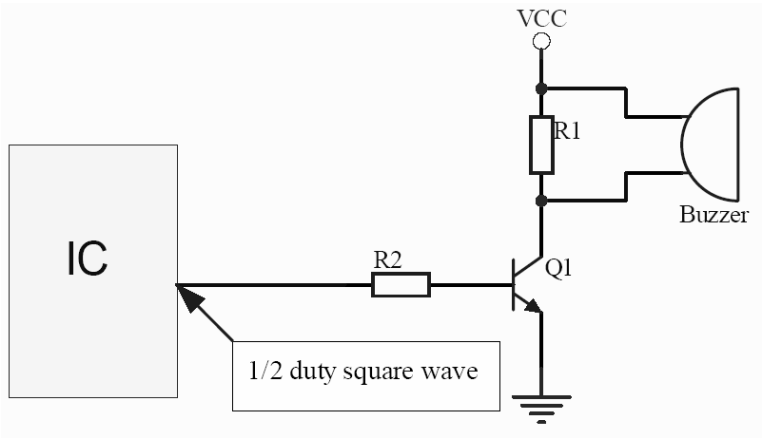


图 5-5　无源压电式蜂鸣器、无源电磁式蜂鸣器驱动电路

2. 有源压电式蜂鸣器、有源电磁式蜂鸣器（自激）

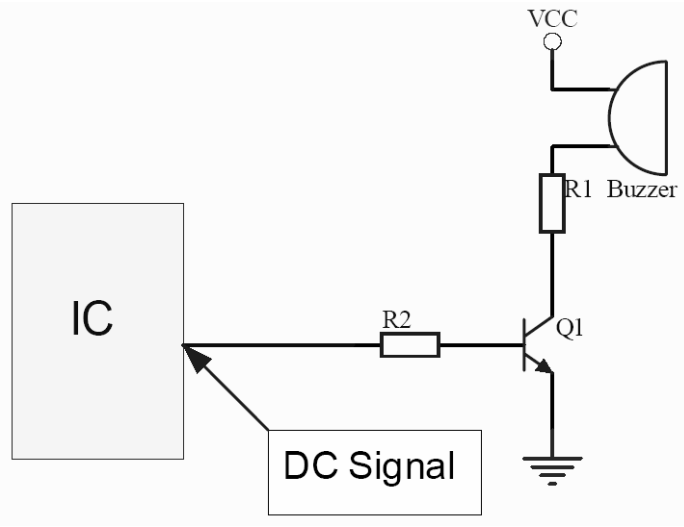


图 5-6　有源压电式蜂鸣器、有源电磁式蜂鸣器驱动电路

5.2.3　蜂鸣器驱动设计

由于这里要介绍两种驱动方式的方法,所以在设计模块系统中将两种驱动方式做到一块，即程序里边不仅介绍了 PWM 输出口驱动蜂鸣器的方法，还要介绍 I/O 口驱动蜂鸣器的方法。所以，我们将设计如下的一个系统来说明单片机对蜂鸣器的驱动：系统有两个他激蜂鸣器，频率都为 2000Hz，一个由 I/O 口进行控制，另一个由 PWM 输出口进行控制；系统还有两个按键，一个按键为 PORT 按键，I/O 口控制的蜂鸣器不鸣叫时按一次按键 I/O 口控制的蜂鸣器鸣叫，再按一次停止鸣叫，另一个按键为 PWM 按键，PWM 口控制的蜂鸣器不鸣叫时按一次按键 PWM 输出口控制的蜂鸣器鸣叫，再按一次停止鸣叫。

■　电路原理图

如图 5-7 所示,使用 SH69P43 为控制芯片,使用 4MHz 晶振作为主振荡器。PORTC.3/T0 作为 I/O 口通过三极管 Q2 来驱动蜂鸣器 LS1，而 PORTC.2/PWM0 则作为 PWM 输出口通过三极管 Q1 来驱动蜂鸣器 LS2。另外在 PORTA.3 和 PORTA.2 分别接了两个按键，一个是 PWM 按键，是用来控制 PWM 输出口驱动蜂鸣器使用的；另一个是 PORT 按键，是用来控制 I/O 口驱动蜂鸣器使用的。连接按键的 I/O 口开内部上拉电阻。

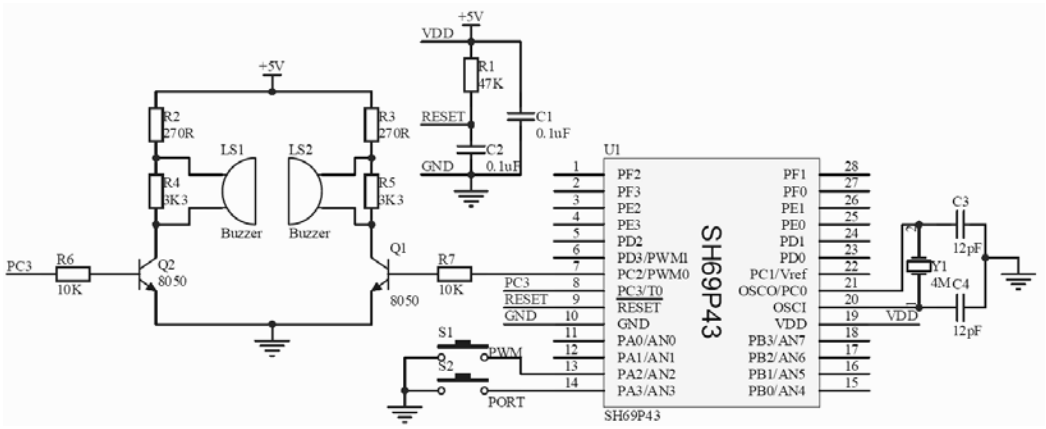


图 5-7　SH69P43 驱动蜂鸣器原理图

■　软件设计方法

先分析一下蜂鸣器。所使用的蜂鸣器的工作频率是 2000Hz，也就是说蜂鸣器的驱动信号波形周期是 500μs，由于是 1/2duty 的信号，所以一个周期内的高电平和低电平的时间宽度都为 250μs。

软件设计上，我们将根据两种驱动方式来进行说明。

a）PWM 输出口直接驱动蜂鸣器方式

由于 PWM 只控制固定频率的蜂鸣器,所以可以在程序的系统初始化时就对 PWM 的输出波形进行设置。

首先根据 SH69P43 的 PWM 输出的周期宽度是 10 位数据来选择 PWM 时钟。系统使用 4MHz 的晶振作为主振荡器，一个 t_{osc} 的时间就是 0.25μs，若是将 PWM 的时钟设置为 t_{osc} 的话，则蜂鸣器要求的波形周期 500μs 的计数值为 500μs/0.25μs=（2000）$_{10}$=（7D0）$_{16}$，7D0H 为 11 位的数据，而 SH69P43 的 PWM 输出周期宽度只是 10 位数据，所以选择 PWM 的时钟为 t_{osc} 是不能实现蜂鸣器所要的驱动波形的。

这里我们将 PWM 的时钟设置为 $4t_{osc}$，这样一个 PWM 的时钟周期就是 1μs 了，由此可以算出 500μs 对应的计数值为 500μs/1μs=（500）$_{10}$=（1F4）$_{16}$，即分别在周期寄存器的高 2 位、中 4 位和低 4 位三个寄存器中填入 1、F 和 4，就完成了对输出周期的设置。再来设置占空比寄存器，在 PWM 输出中占空比的实现是通过设定一个周期内电平的宽度来实现的。当输出模式选择为普通模式时，占空比寄存器是用来设置高电平的宽度。250μs 的宽度计数值为 250μs/1μs=（250）$_{10}$=（0FA）$_{16}$。只需要在占空比寄存器的高 2 位、中 4 位和低 4 位中分别填入 0、F 和 A 就可以完成对占空比的设置了，设置占空比为 1/2duty。

以后只需要打开 PWM 输出，PWM 输出口自然就能输出频率为 2000Hz、占空比为 1/2duty 的方波。

b）I/O 口定时翻转电平驱动蜂鸣器方式

使用 I/O 口定时翻转电平驱动蜂鸣器方式的设置比较简单，只需要对波形分析一下。由于驱动的信号刚好为周期 500μs，占空比为 1/2duty 的方波，只需要每 250μs 进行一次电平翻转，就可以得到驱动蜂鸣器的方波信号。

在程序上，可以使用 TIMER0 来定时，将 TIMER0 的预分频设置为/1，选择 TIMER0 为系统时钟(主振荡器时钟/4),在 TIMER0 的载入/计数寄存器的高 4 位和低 4 位分别写入 00H 和 06H，就能将 TIMER0 的中断设置为 250μs。

当需要 I/O 口驱动的蜂鸣器鸣叫时，只需要在进入 TIMER0 中断的时候对该 I/O 口的电平进行翻转一次，直到蜂鸣器不需要鸣叫的时候，将 I/O 口的电平设置为低电平即可。不鸣叫时将 I/O 口的输出电平设置为低电平是为了防止漏电。

■ 程序

以下是带有两种驱动方式的蜂鸣器驱动模块程序，其中黄色背景的内容是关于 I/O 口定时翻转电平驱动方式的，绿色背景的内容则是关于 PWM 输出口直接驱动方式的：

例[5-2]

```
LIST P=69P43
ROMSIZE=3072
;***********************************************
; 系统寄存器
;***********************************************
IE          EQU   00H          ;中断使能标志
IRQ         EQU   01H          ;中断请求标志
TM0         EQU   02H          ;Timer0 模式寄存器
TL0         EQU   04H          ;Timer0 装入/记数寄存器低四位
TH0         EQU   05H          ;Timer0 装入/记数寄存器高四位
PORTA       EQU   08H          ;Port A 数据寄存器
PORTC       EQU   0AH          ;Port C 数据寄存器
TBR         EQU   0EH          ;查表寄存器
INX         EQU   0FH          ;间接寻址伪索引寄存器
DPL         EQU   10H          ;INX 数据指针低四位
DPM         EQU   11H          ;INX 数据指针中三位
DPH         EQU   12H          ;INX 数据指针高三位
PACR        EQU   18H          ;PortA 输入/输出控制寄存器
PCCR        EQU   1AH          ;PortC 输入/输出控制寄存器
PWM0        EQU   20H          ;位 0:选择 PWM 输出, 位 2-1:设置 PWM0 时钟, 位 3:设置 PWM0 占空比输出模式
PP0L        EQU   22H          ;PWM0 周期低四位
PP0M        EQU   23H          ;PWM0 周期中四位
```

```
PP0H        EQU   24H           ;PWM0 周期高二位
PD0L        EQU   25H           ;PWM0 占空比低四位
PD0M        EQU   26H           ;PWM0 占空比中四位
PD0H        EQU   27H           ;PWM0 占空比高二位
;************************************************
; 用户定义寄存器(Bank0)
;************************************************
AC_BAK      EQU   30H           ;AC 值备份寄存器
PA_BAK      EQU   31H           ;PortA 缓冲寄存器
PC_BAK      EQU   32H           ;PortC 缓冲寄存器
TMP_T0      EQU   33H           ;临时寄存器用于 TIMER0
;--------------------------------------
F_TIMER     EQU   34H
                                ;位 0=1, 5ms 到
FLAG1       EQU   35H
                                ;位 0=1, 按键未松开
                                ;位 3=1, 需使用 I/O 口驱动蜂鸣器
;--------------------------------------
KCODE_NEW   EQU   36H           ;旧的按键码
KCODE_OLD   EQU   37H           ;新的按键码
KEYS_CT     EQU   38H           ;按键扫描次数
;--------------------------------------
T5MS_CT1    EQU   39H           ;5ms 计数器低位
T5MS_CT2    EQU   3AH           ;5ms 计数器高位
;************************************************
; 程序
;************************************************
            ORG         0000H
            JMP         RESET
            RTNI
            JMP         TIMER0_ISP
            RTNI
            RTNI
;*****************************************
; 子程序: TIMER0 中断服务程序（250us 中断）
;*****************************************
TIMER0_ISP:
            STA         AC_BAK,00H      ;备份 AC 值
```

```
            ANDIM IRQ,1011B                 ;清 TIMER0 中断请求标志
;*****************************
; 模块:     I/O 口驱动蜂鸣器模块
;*****************************
PORT_BUZ:
            ADI         FLAG1,1000B
            BA3         PORT_BUZ_NO         ;不需要 I/O 口驱动蜂鸣器，跳转
            EORIM       PC_BAK,1000B
            STA         PORTC,00H           ;PORTC.3 翻转电平
            JMP         PORT_BUZ_END
PORT_BUZ_NO:
            ANDIM       PC_BAK,0111B
            STA         PORTC,00H           ;PORTC.3 口驱动的蜂鸣器未鸣叫时，输出低电平以保证不漏电
PORT_BUZ_END:
;*****************************
J5MS:                                       ;判断 5ms 到
            SBIM        T5MS_CT1,01H
            LDI         TMP_T0,00H
            SBCM        T5MS_CT2,00H
            OR          T5MS_CT1,00H
            BNZ         TIMER0_ISP_END      ;5ms 未到，跳转
            LDI         T5MS_CT1,04H
            LDI         T5MS_CT2,01H        ;重新加载 5ms 计数值
            ORIM        F_TIMER,0001B       ;设置"5ms 到"标志
TIMER0_ISP_END:
            LDI         IE,0100B            ;开 TIMER0 中断
            LDA         AC_BAK,00H          ;恢复 AC 值
            RTNI                            ;返回
;*****************************************
; 上电程序
;*****************************************
RESET:
            NOP
;-------------------------------------
; 清用户寄存器
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
```

```
            LDI         DPH,00H
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TBR,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,04H
            BNZ         POWER_RESET_1
;--------------------------------------
; 初始化系统寄存器
SYSTEM_INITIAL:
            ;TIMER0 初始化
            LDI         TM0,07H             ;设置 TIMER0 预分频为/1
            LDI         TL0,06H
            LDI         TH0,00H             ;设置中断时间为 250us
            LDI         T5MS_CT1,04H
            LDI         T5MS_CT2,01H
            ;I/O 口初始化
            LDI         PORTC,0011B
            LDI         PCCR,1100B          ;设置 PORTC.2 和 PORTC.3 为输出口
            LDI         PORTA,0FH           ;打开 PORTA 上拉电阻
            ;PWM 初始化
            LDI         PWM0,04H            ;设置 PWM0 时钟为 4tosc，占空比输出模式为正常模式
            LDI         PP0H,01H
            LDI         PP0M,0FH
            LDI         PP0L,04H            ;设置周期为 500us
            LDI         PD0H,00H
            LDI         PD0M,0FH
            LDI         PD0L,0AH            ;设置占空比电平为 250us，得频率为 2KHz，占空比为 1/2duty 的
                                            ;信号
;--------------------------------------
```

```
MAIN_PRE:
          LDI       IRQ,00H
          LDI       IE,0100B            ;打开 Timer0 中断
;***********************************************
MAIN:
          ADI       F_TIMER,0001B
          BA0       HALTMODE            ;未到 5ms, 跳转
          ANDIM     F_TIMER,1110B       ;清"5ms 到"标志
;*************************************
; 按键扫描
;*************************************
KEYSCAN:
          LDA       PORTA,00H
          STA       TBR,00H
          SBI       TBR,0FH
          BAZ       NO_KEY              ;没有扫描到按键按下，跳转

          LDA       FLAG1,00H
          BA0       KEYSCAN_END         ;之前的按键未松开，不再扫描按键，跳转
          SBI       TBR,0111B
          BAZ       KS_PORT             ;PORT 键按下，跳转
          SBI       TBR,1011B
          BAZ       KS_PWM              ;PWM 键按下，跳转
          JMP       KEYSCAN_END
KS_PORT:
          LDI       KCODE_NEW,0001B     ;设置 PORT 键的键码为 01H
          JMP       KEYSCAN_1
KS_PWM:
          LDI       KCODE_NEW,0010B     ;设置 PWM 键的键码为 02H
KEYSCAN_1:
          ADIM      KEYS_CT,01H         ;扫描次数加一
          SBI       KEYS_CT,01H
          BAZ       KEYSCAN_2           ;第一次扫描到该按键，无需与旧按键进行比较，跳转
          LDA       KCODE_OLD,00H
          SUB       KCODE_NEW,00H       ;新扫描的按键码与上次扫描的按键码比较
          BAZ       KEYSCAN_2           ;同一个按键，跳转
          LDI       KEYS_CT,01H         ;不是同一按键，将扫描次数置为 1，为新按键
KEYSCAN_2:
```

```
            LDA         KCODE_NEW,00H
            STA         KCODE_OLD,00H       ;将新的按键码赋值给旧的按键码
            SBI         KEYS_CT,0AH
            BNC         KEYSCAN_END         ;扫描未到 10 次（50ms），跳转
            ;已扫描到该按键 10 次
            LDI         KEYS_CT,00H         ;清按键扫描次数
            ORIM        FLAG1,0001B         ;设置"按键未松开"标志
            LDA         KCODE_OLD,00H
            BAO         KEY_PORT
KEY_PWM:                                    ;PWM 键
            EORIM       PWM0,0001B          ;PWM 输出开启或关闭，PWM 口驱动的蜂鸣器鸣叫或停止
            JMP         KEYSCAN_END
KEY_PORT:                                   ;PORT 键
            EORIM       FLAG1,1000B         ;I/O 定时翻转电平开启或停止翻转
                                            ;I/O 口驱动的蜂鸣器鸣叫或停止
            JMP         KEYSCAN_END
NO_KEY:
            ANDIM       FLAG1,1110B         ;清"按键未松开"标志
            LDI         KEYS_CT,00H         ;清按键扫描次数
KEYSCAN_END:
;************************************
HALTMODE:
            NOP
            HALT                            ;进入 HALT 模式
            NOP
            NOP
            JMP         MAIN

            END
```

5.3 键盘扫描

键盘是由按键构成，是单片机系统里最常用的输入设备。我们可以通过键盘输入数据或命令来实现简单的人-机通信。

5.3.1 按键及键抖动

按键是一种常开型按钮开关。平时，按键的两个触点处于断开状态，按下按键时

两个触点才闭合（短路）。如图 5-8 所示，平常状态下，当按键 K 未被按下时，按键断开，PA0 输入口的电平为高电平；当按键 K 被按下时，按键闭合，PA0 输入口的电平为低电平。

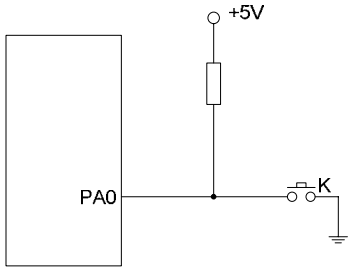


图 5-8　按键电路

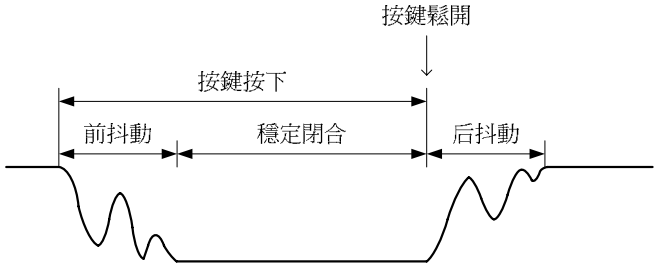


图 5-9　按键抖动

一般的按键所用开关都是机械弹性开关，由于机械触点的弹性作用，按键开关在闭合时不会马上稳定地连接，在断开进也不会马上完全的断开，在闭合和断开的瞬间均有一连串的抖动。按键按下的电压信号波形图如图 5-9 所示，从图中可以看出按键按下和松开的时候都存在着抖动。抖动时间的长短因按键的机械特性不同而有所不同，一般为 5ms～10ms。

如果不处理键抖动，则有可能引起一次按键被误读成多次，所以为了确保能够正确地读到按键，必须去除键抖动，确保在按键的稳定闭合和稳定断开的时候来判断按键状态，判断后再做处理。按键在去抖动，可用硬件或软件两种方法消除。由于使用硬件方法消除键抖动，一般会给系统的成本带来提高，所以通常情况下都是使用软件方法去除键抖动。

常用的去除键抖动的软件方法有很多种，但是都离不开基本的原则：就是要么避开抖动

的时候检测按键或是在抖动的时候检测到的按键不做处理。这里说明一下常用的两种方法：

第一种方法是检测到按键闭合电平后先执行一个延时程序，做一个 12ms～24ms 的延时，让前抖动消失后再一次检测按键的状态，如果仍是闭合状态的电平，则认为真的有按键按下；若不是闭合状态电平，则认为没有键按下。若是要判断按键松开的话，也是要在检测到按键释放电平之后再给出 12ms～24ms 的延时，等后抖动消失后再一次检测按键的状态，如果仍为断开状态电平，则确认按键松开。这种方法的优点是程序比较简单，缺点是由于延时一般采用跑空指令延时，造成程序执行效率低。

第二种方法是每隔一个时间周期检测一次按键，比如每 5ms 扫描一次按键，要连续几次都扫描到同一按键才确认这个按键被按下。一般确认按键的扫描次数由实际情况决定，扫描次数的累积时间一般为 50ms～60ms。比如，以 5ms 为基本时间单位去扫描按键的话，前后要连续扫描到同一个按键 11 次而达到 50ms 来确认这个按键。按键松开的检测方法也是一样要连续多次检测到按键状态为断开电平才能确认按键松开。这种方法的优点是程序执行效率高，不用刻意加延时指令，而且这种方法的判断按键抗干扰能力要更好；缺点是程序结构较复杂。

在以下的介绍中，我们将使用第二种方法来去除键抖动。

5.3.2 键盘结构及工作原理

键盘一般有独立式和行列式（矩阵式）两种。当然还有其它的结构，比如交互式结构等等，不过其它的结构比较少用，在这里就不介绍了。在中颖的单片机中，有些单片机的 LCD 驱动引脚的 SEGMENT 口可以共享按键扫描口，当选择为按键扫描口时，可以使用这些口来扫描按键，所以在外部电路可以连接 LCD 和按键矩阵，采用分时扫描进行处理，下面也将介绍这个特殊应用的方法和注意的地方。

■ 独立式键盘结构

独立式键盘是指各个按键相互独立地连接到各自的单片机的 I/O 口，I/O 口只需要做输入口就能读到所有的按键。

独立式键盘可以使用上拉电阻也可以使用下拉电阻，基本原理是一样的。使用上拉电阻的独立式键盘结构如图 5-10 所示。

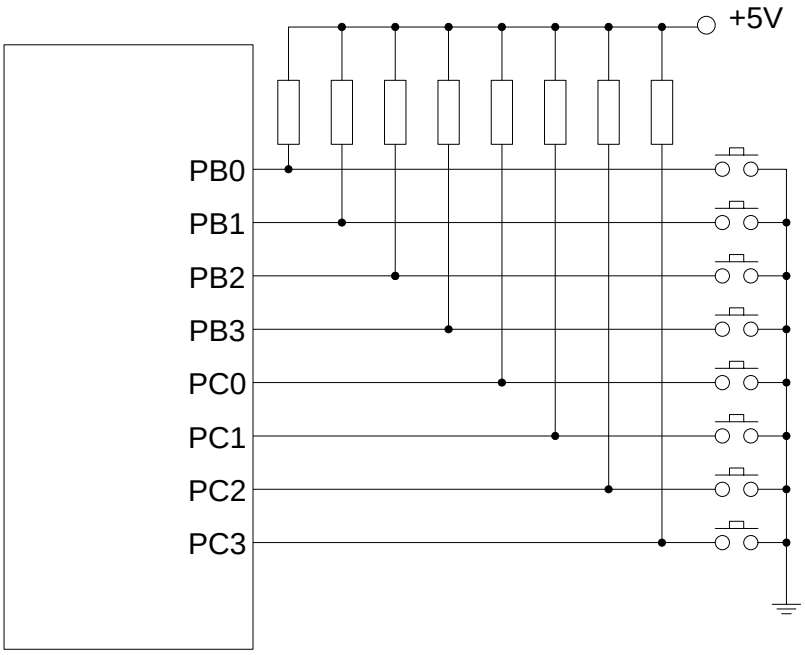


图 5-10　独立式键盘结构

图 5-10 所示的是利用 PB 口和 PC 口共 8 个 I/O 口独自连接 8 个按键，使用外部上拉电阻构成的独立式键盘。在中颖的单片机中，有很多型号的单片机有 I/O 内部上拉电阻或内部下拉电阻，所以在实际应用，若是使用到这样的单片机，是不需要接外部上拉电阻或下拉电阻了，只需在程序中把内部上拉电阻或内部下拉电阻打开即可。

从图 5-10 可以看出，当按键没有被按下的时候，连接到该按键的 I/O 口输入电平为高电平，当按键按下去之后，输入电平则变为低电平。所以要判别有无按键按下，只需判断输入口的电平即可，程序写起来十分方便。

这种键盘虽然有电路简单、程序容易写的优点，但是也有缺点：当按键个数较多的时候，要占用较多的 I/O 口资源。所以当按键个数比较多的时候，比较少用这样的按键结构，而是使用下面我们要讲的行列式结构。

■　行列式键盘结构

为了减少键盘占用太多的单片机 I/O 口资源，当按键个数较多的时候，通常都使用行列式键盘。

行列式键盘同样可以使用上拉电阻或是下拉电阻，使用上拉电阻的行列式键盘结构如图 5-11 所示。

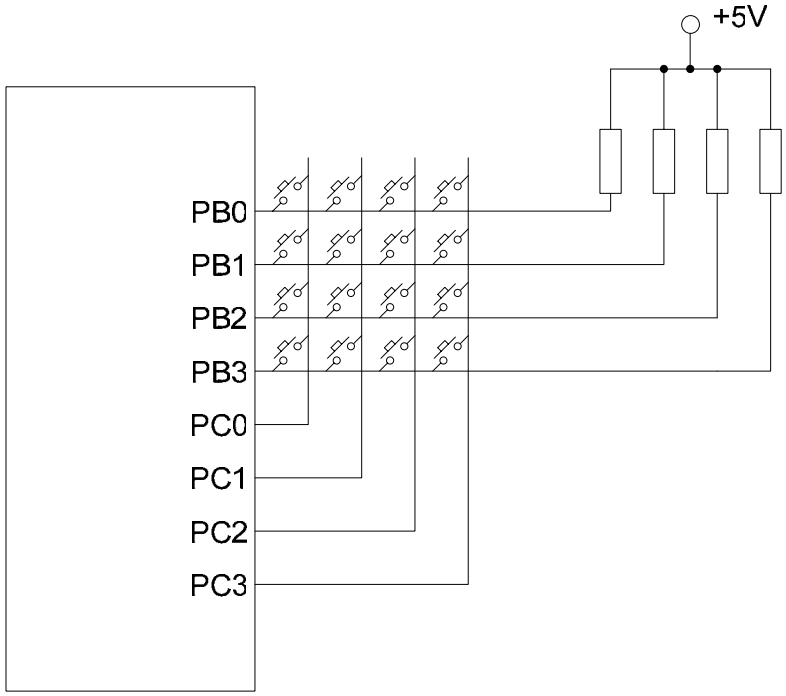


图 5-11　行列式键盘结构

跟独立式键盘一样，若是使用有内部上拉电阻或是下拉电阻的单片机时，外面不需连接上拉电阻或是下拉电阻，只需在程序内打开内部上拉电阻或是内部下拉电阻即可。

行列式键盘的原理就是每一行线与每一列线的交叉地方不相通，而是接上一个按键，通过按键来接通。所以利用这种结构，a 个 I/O 口可以接 a 个行线，另外的 b 个 I/O 可以接 b 个列线，总共可以组成 a×b 个按键的键盘。如图 5-11 所示，共有 4 个行线，4 个列线，可以组成 4×4＝16 个按键的键盘。

可以看出，行列式的键盘结构可以省出不少的 I/O 口资源。

对行列式的键盘进行扫描的时候，要先判断整个键盘是否有按键按下，有按键按下才对哪一个按键按下进行判别扫描。对按键的识别扫描通常有两种方法：一种是比较常用的逐行（或逐列）扫描法，另一种是线反转法。

现以图 5-11 的 4×4 键盘为例，讲解一下这两种扫描方法的工作原理。

a)　逐行（或逐列）扫描法的工作原理

首先要先判别整个键盘中是否有按键按下，由单片机连接到列线的 PC 口输出低电平，然后读取连接到行线的 PB 口的电平状态。若是没有按键按下，则 PB 口读进来的数据为 0FH；若读进来的数据不是 0FH，那就是有按键按下，因为只要有按键按下，该按键连接到的行线电平就会被拉至低电平。

若是有判断到按键按下之后就要进行对按键的识别扫描。扫描的方法是将列线

逐列置低电平，并检测行线的电平状态来实现的。依次向 PC 口的每个列线送低电平，然后检测所有行线 PC 口的状态，若是全为 1，则所按下的按键不在此列，进入下一列的扫描；若是不全为 1，则所按下的按键必在此列，并且按键正是此列与读取到为低电平的行线的交点上。

b) 线反转法

线反转法的优点是扫描速度比较快，但是程序处理起来却是比较不方便。

线反转法最好是将行、列线按二进制顺序排列。线反转法同样也要先判别整个键盘有无按键按下，有按键按下才对键盘进行扫描。

当有某一按键按下时，键盘扫描扫到给该列置低电平时，读到了行状态为非全 1，这个时候就可以将行数据和列数据组合成一个键值。如图 5-11 的键盘从左到右、从上到下的键值依次是 EE，ED，EB，E7；DE，DD，DB，D7；…；7E，7D，7B，77。这是负逻辑的排列，可以通过软件的取反指令把这些数据变成正逻辑：11，12，14，18；21，22，24，28；…；81，82，84，88。不过不管是正逻辑还是负逻辑的数据，可以看出这样的数据是很难使用散转指令的。所以一般都要想办法把这样的键值数据再修正一下成为等距能用于散转指令的键值数据。

若是所使用的单片机内部具有上拉电阻的话，外部无须接上拉电阻。先使用 PB 口作为输入口，打开 PB 口上拉电阻，而 PC 口作为输出口输出低电平，读 PB 口得到列数据；再使用 PC 口作为输入口，打开 PC 口上拉电阻，而 PB 口作为输出口输出低电平，读 PC 口得到行数据。这样就可直接得到行数据和列数据，而得组合的键值。

线反转法一般用于 4 的倍数的键盘，比如 4×4 键盘、4×8 键盘、8×8 键盘。

■ 共享 LCD SEGMENT 输出口的键盘结构

中颖的一些单片机通过共享 LCD SEGMENT 输出口作为键盘扫描的输出控制口来节省 IO 口资源，例如 SH6613/SH6614,SH6790/SH6791 等等。

当采用共享 LCD SEGMENT 驱动输出口键盘结构时，键盘扫描的输出控制口与 SEGMENT 口共享。在这种键盘结构中，为了减小在按键按下时电流对 LCD 的影响，需要在扫描输入口接一个约 1M 的上拉电阻；同时为了防止同一行两个按键按下时，两个 SEGMENT 口之间不会出现大电流，需要在每一个 SEGMENT 扫描输出口接一个几十 KΩ 的电阻或二极管。如图 5-12 所示：

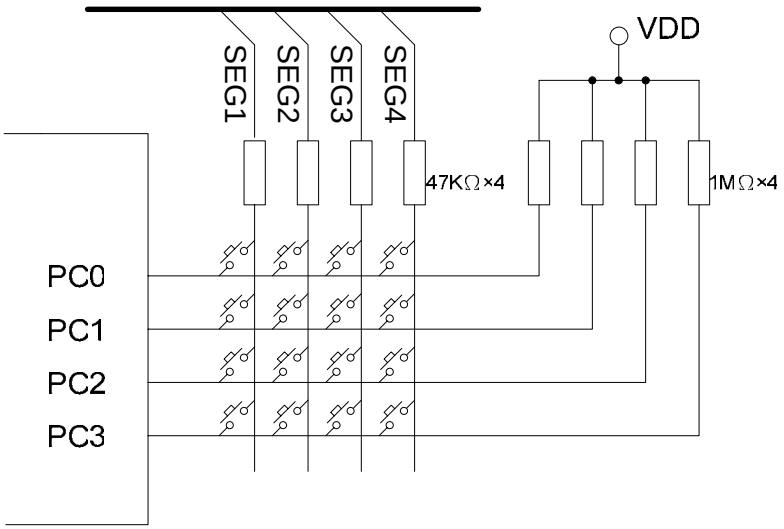


图 5-12　共享 LCD SEGMENT 输出口的键盘矩阵结构

对于共享 LCD SEGMENT 口的键盘结构，扫描原理跟行列式键盘矩阵基本是一样的，只是在扫描的时候要将共享在键盘的 SEGMENT 口切换为按键扫描输出口，输出扫描电平再通过 I/O 口来读取状态，读取状态后将 SEGMENT 口切换回 LCD 显示就可以了，需要注意的是：

1. 这种结构中上拉电阻较大，因此当扫描输出口输出低电平后，需等待一段时间，待放电完毕后再读输入口状态，一般约几十 us。
2. 为了减少键盘扫描对 LCD 显示的影响，必须让 SEGMENT 切换为按键扫描输出口的时间尽可能的短。一般为当 SEGMENT 切换为按键扫描输出口后读到扫描输入口电平状态，将电平状态备份至临时寄存器后马上将扫描输出口切换回 SEGMENT 输出状态，然后再对备份的电平状态寄存器进行判断。
3. 由于带有 LCD SEGMENT 驱动口可以共享按键扫描口的这些单片机都是有低频时钟源和高频时钟源，当系统时间选择为低频时钟时，指令周期时间比较长，共享口在作为按键扫描输出口时执行尽可能少的指令显得尤为重要。

5.3.3　行列式键盘扫描模块原理及应用程序

使用 SH69P25 做 4×4 键盘的键盘扫描程序，分别用逐行（或逐列）扫描法和线反转法进行按键扫描。

■　电路原理图

使用 SH69P25 为控制芯片，利用 PB 口和 PC 口来扫描一个 4×4＝16 个按键的行列式键盘的电路原理图如图 5-13 所示。

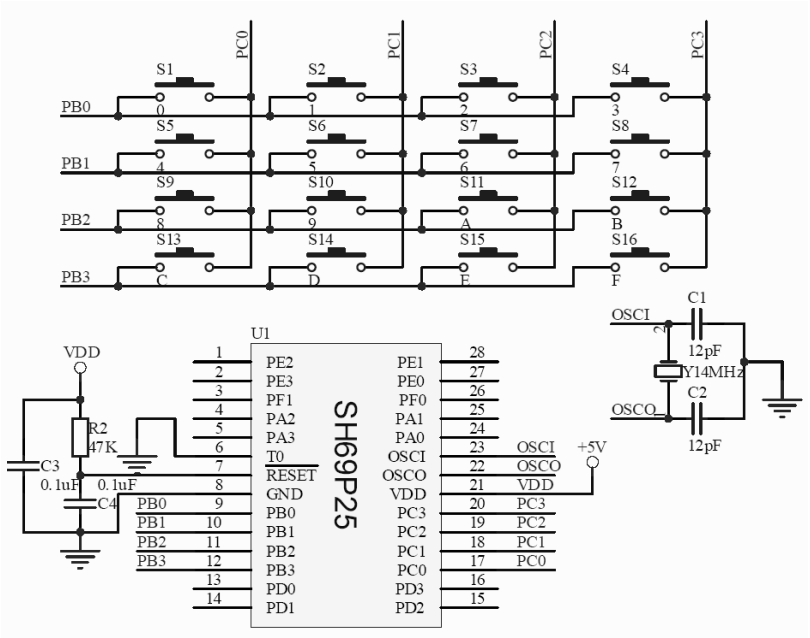


图 5-13　行列式键盘扫描模块电路原理图

电路中单片机 SH69P25 作用 4MHz 的晶振 Y1 作为主振荡器，使用 PB 口连接到行列式键盘的行线，使用 PC 口连接到行列式键盘的列线，由于 SH69P25 有内部上拉电阻和内部下拉电阻，所以外部不需再接上拉电阻了。

16 个按键的键值从左到右、从上到下，分别为 0，1，2，3，4，5，6，…，A，B，C，D，E，F。

■　程序

根据上面的电路图，这里分别有逐行（或逐列）扫描法和线反转法的两个程序对这两种方法进行按键扫描作出描述。

例[5-3] 逐行（或逐列）扫描法

```
LIST  P=69P25
ROMSIZE=4096
;************************************************
; 系统寄存器
;************************************************
IE          EQU   00H          ;中断使能标志
IRQ         EQU   01H          ;中断请求标志
TM0         EQU   02H          ;Timer0 模式寄存器
TL0         EQU   04H          ;Timer0 装入/记数寄存器低四位
TH0         EQU   05H          ;Timer0 装入/记数寄存器高四位
PORTB       EQU   09H          ;PORTB 数据寄存器
```

```
PORTC       EQU   0AH         ;PORTC 数据寄存器
TBR         EQU   0EH         ;查表寄存器
INX         EQU   0FH         ;间接寻址伪索引寄存器
DPL         EQU   10H         ;INX 数据指针低四位
DPM         EQU   11H         ;INX 数据指针中三位
DPH         EQU   12H         ;INX 数据指针高三位
SETTING     EQU   15H         ;位 1: PBC 中断设置为上升沿/下降沿
                              ;位 2: 设置 I/O 口上拉电阻/下拉电阻
                              ;位 3: I/O 口上拉电阻/下拉电阻使能控制
PBOUT       EQU   17H         ;PORTB 输入/输出控制寄存器
PCOUT       EQU   18H         ;PORTC 输入/输出控制寄存器
;************************************************
; 用户定义寄存器(Bank0)
;************************************************
AC_BAK      EQU   20H         ;AC 值备份寄存器
TMP         EQU   21H         ;临时寄存器
TMP_T0      EQU   22H         ;用在 Timer0 中断服务程序的临时寄存器
PB_BAK      EQU   23H         ;PortB 数据备份寄存器
PC_BAK      EQU   24H         ;PortC 数据备份寄存器
F_TIMER     EQU   25H         ;位 0=1, 5ms 到
FLAG1       EQU   26H
                              ;位 0=1, 按键未松开标志
                              ;位 1=1, 有按键按下标志
KCODE_OLD   EQU   27H         ;上次扫描到的键码(用于按键扫描)
KCODE_NEW   EQU   28H         ;本次扫描到的键码(用于按键扫描)
KEYCODE     EQU   29H         ;按下的按键键码
T5MS_CT1    EQU   2AH         ;5ms 定时计数器(14H)
T5MS_CT2    EQU   2BH
KEYS_CT     EQU   2CH         ;按键扫描次数
;************************************************
; 程序
;************************************************
            ORG         0000H
            JMP         RESET
            RTNI
            JMP         TIMER0_ISP
            RTNI
            RTNI
```

```
;*****************************************
; 子程序: TIMER0 中断服务程序
;*****************************************
TIMER0_ISP:
          STA       AC_BAK,00H          ;备份 AC 值
          ANDIM     IRQ,1011B           ;清 TIMER0 中断请求标志
J5MS:
          SBIM      T5MS_CT1,01H
          LDI       TMP_T0,00H
          SBCM      T5MS_CT2,00H
          OR        T5MS_CT1,00H
          BNZ       TIMER0_ISP_END      ;没到 5ms, 跳转
          LDI       F_TIMER,0001B       ;设置“5ms 到”标志
          LDI       T5MS_CT2,01H        ;重置 5ms 计数器
          LDI       T5MS_CT1,04H
TIMER0_ISP_END:
          LDI       IE,0100B            ;打开 TIMER0 中断
          LDA       AC_BAK,00H          ;取出 AC 值
          RTNI                          ;返回
;*****************************************
; 上电程序
;*****************************************
RESET:
          NOP
;--------------------------------------
;清用户寄存器
POWER_RESET:
          LDI       DPL,00H
          LDI       DPM,02H
          LDI       DPH,00H
POWER_RESET_1:
          LDI       INX,00H
          ADIM      DPL,01H
          LDI       TMP,00H
          ADCM      DPM,00H
          BA3       POWER_RESET_2
          JMP       POWER_RESET_3
POWER_RESET_2:
```

```
            ADIM        DPH,01H
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,04H
            BNZ         POWER_RESET_1
;-------------------------------------
;初始化系统寄存器
SYSTEM_INITIAL:
            ;TIMER0 初始化
            LDI         TM0,07H             ;设置 TIMER0 预分频为/1
            LDI         TL0,06H
            LDI         TH0,00H             ;设置中断时间为 250us
            LDI         T5MS_CT2,01H
            LDI         T5MS_CT1,04H        ;计数器=14H=20, 计到 5ms
            ;初始化 I/O 口
            LDI         SETTING,1100B       ;设置上拉电阻使能
;-------------------------------------
MAIN_PRE:
            LDI         IRQ,00H
            LDI         IE,0100B            ;打开 Timer0 中断
;*********************************************
MAIN:
            ADI         F_TIMER,0001B
            BA0         HALTMODE            ;未到 5ms, 跳转
            ANDIM       F_TIMER,1110B       ;清"5ms 到"标志
;************************************
; 模块:    逐行（逐列）扫描法按键扫描
; 输入变量:      FLAG1
; 使用变量:      PB_BAK, PC_BAK, PORTB, PORTC, PBOUT, PCOUT, TMP
;           KCODE_OLD, KCODE_NEW, KEYS_CT
; 输出变量:      FLAG1
;************************************
KEYSCAN:
            LDI         PC_BAK,0FH
            STA         PORTC,00H       ;打开 PORTC 电阻
            LDI         PB_BAK,00H
            STA         PORTB,00H
```

```
        LDI     PBOUT,0FH       ;PORTB 输出低电平
        NOP                     ;等待稳定
        NOP
        NOP
        LDA     PORTC,00H       ;读 PC 口
        STA     TMP,00H         ;暂存于临时寄存器
        SBI     TMP,0FH         ;若是没有按键按下，则读入的数据全为 1
        BAZ     NO_KEY          ;没有按键按下，跳转

        LDI     KCODE_NEW,00H   ;先将本次扫描的键值清零
        LDI     PB_BAK,1111B    ;PB 口输出高电平，准备从 PB0 开始逐行扫描，直到 PB3
KS_LINE:
        LDA     PB_BAK,00H
        ADDM    PB_BAK,00H
        STA     PORTB,00H       ;左移，扫描到下一行
        NOP                     ;等待稳定
        NOP
        NOP
        LDA     PORTC,00H       ;读 PC 口
        STA     TMP,00H         ;暂存于临时寄存器
        SBI     TMP,0FH         ;若是按键不在此行，则读入的数据全为 1
        BNZ     KS_ROW          ;按键在此行，准备寻找在哪一列
        ;不在此行
        LDA     PB_BAK,00H
        BAZ     NO_KEY          ;四行已都扫描完毕，仍未发现按键，认为没有按键，跳转
        ADIM    KCODE_NEW,04H   ;键值加 4
        JMP     KS_LINE         ;准备扫描下一行
KS_ROW:
        EORIM TMP,0FH           ;由于是读进来的 PC 口数据是负逻辑的，取反变成正逻辑
KS_ROW_1:
        LDA     TMP,00H
        SHR
        STA     TMP,00H         ;右移，下一列（从 PC0 开始）
        BAZ     KS_CHECK        ;右移之后数据变为 0，则在此列，已经得到键值
        ADIM    KCODE_NEW,01H   ;不在此列，键值加 1
        JMP     KS_ROW_1        ;准备查找下一列
KS_CHECK:
        LDI     PBOUT,00H       ;设置 PORTB 口为输入口
```

```
            LDA         FLAG1,00H
            BA0         NO_KEY_1            ;之前的按键未松开，不做处理，跳转

            ADIM        KEYS_CT,01H         ;按键扫描次数加一
            SBI         KEYS_CT,01H
            BAZ         KEYSCAN_2           ;第一扫到该按键，直接把新扫到的键值赋给旧的键值，跳转
            LDA         KCODE_OLD,00H
            SUB         KCODE_NEW,00H       ;比较本次扫的键值与上次扫的键值是否相等
            BAZ         KEYSCAN_2           ;相等
            LDI         KEYS_CT,01H         ;不相等，认为是新的，将扫描次数设置为 1
KEYSCAN_2:
            LDA         KCODE_NEW,00H
            STA         KCODE_OLD,00H       ;把新扫到的键值赋给旧的键值
            SBI         KEYS_CT,0BH         ;前后连续扫描同一按键 11 次，中间间隔达到 50ms
            BNC         KEYSCAN_END         ;不到 50ms，跳走
            ;到 50ms
            LDI         KEYS_CT,00H         ;将扫描次数清零
            ORIM        FLAG1,0011B         ;置"按键未松开”标志和"有按键按下”标志
            JMP         KEYSCAN_END
NO_KEY:
            ADI         FLAG1,0001B
            BA0         NO_KEY_1            ;按键已松开，不需再判断按键松开，跳走
            ADIM        KEYS_CT,01H         ;扫描次数加一
            SBI         KEYS_CT,0BH         ;前后连续扫描到没有按键 11 次，中间间隔达到 50ms
            BNC         KEYSCAN_END         ;不到 50ms，跳走
            ANDIM       FLAG1,1110B         ;清"按键未松开”标志
NO_KEY_1:
            LDI         KEYS_CT,00H         ;扫描次数清零
KEYSCAN_END:
;*************************************
KEY_PROC:
            ADI         FLAG1,0010B
            BA1         HALTMODE            ;没有按键按下，跳转
KEY_PROC_1:
            ANDIM       FLAG1,1101B         ;清"有按键按下”标志
            LDA         KCODE_OLD,00H
            STA         KEYCODE,00H         ;把扫描到的键值送给真正的键码
            ;----------------------
```

```
            ;这里可以对按键按下后的程序状态进行处理
            ;----------------------
KEY_PROC_END:
;*************************************
HALTMODE:
            NOP
            HALT
            NOP
            NOP
            JMP         MAIN
;*************************************************
            END
```

例[5-4]线反转法

```
LIST  P=69P25
ROMSIZE=4096
;*************************************************
; 系统寄存器
;*************************************************
IE          EQU   00H         ;中断使能标志
IRQ         EQU   01H         ;中断请求标志
TM0         EQU   02H         ;Timer0 模式寄存器
TL0         EQU   04H         ;Timer0 装入/记数寄存器低四位
TH0         EQU   05H         ;Timer0 装入/记数寄存器高四位
PORTB       EQU   09H         ;PORTB 数据寄存器
PORTC       EQU   0AH         ;PORTC 数据寄存器
TBR         EQU   0EH         ;查表寄存器
INX         EQU   0FH         ;间接寻址伪索引寄存器
DPL         EQU   10H         ;INX 数据指针低四位
DPM         EQU   11H         ;INX 数据指针中三位
DPH         EQU   12H         ;INX 数据指针高三位
SETTING     EQU   15H         ;位 1: PBC 中断设置为上升沿/下降沿
                              ;位 2: 设置 I/O 口上拉电阻/下拉电阻
                              ;位 3: I/O 口上拉电阻/下拉电阻使能控制
PBOUT       EQU   17H         ;PORTB 输入/输出控制寄存器
PCOUT       EQU   18H         ;PORTC 输入/输出控制寄存器
;*************************************************
; 用户定义寄存器(Bank0)
```

```
;***********************************************
AC_BAK          EQU   20H         ;AC 值备份寄存器
TMP             EQU   21H         ;临时寄存器
TMP_T0          EQU   22H         ;用在 Timer0 中断服务程序的临时寄存器
ROWDATA         EQU   23H         ;列数据（用于按键扫描）
LINEDATA        EQU   24H         ;行数据（用于按键扫描）
;--------------------------------------
F_TIMER         EQU   25H         ;位 0=1, 5ms 到
FLAG1           EQU   26H         ;位 0=1, 按键未松开标志
                                  ;位 1=1, 有按键按下标志
;--------------------------------------
KCODE_OLD       EQU   27H         ;上次扫描到的键码(用于按键扫描)
KCODE_NEW       EQU   28H         ;本次扫描到的键码(用于按键扫描)
KEYCODE         EQU   29H         ;按下的按键键码
;--------------------------------------
T5MS_CT1        EQU   2AH         ;5ms 定时计数器(14H)
T5MS_CT2        EQU   2BH
KEYS_CT         EQU   2CH         ;按键扫描次数
;***********************************************
; 程序
;***********************************************
          ORG       0000H
          JMP       RESET
          RTNI
          JMP       TIMER0_ISP
          RTNI
          RTNI
;*****************************************
; 子程序:  TIMER0 中断服务程序
;*****************************************
TIMER0_ISP:
          STA       AC_BAK,00H          ;备份 AC 值
          ANDIM     IRQ,1011B           ;清 TIMER0 中断请求标志
J5MS:
          SBIM      T5MS_CT1,01H
          LDI       TMP_T0,00H
          SBCM      T5MS_CT2,00H
          OR        T5MS_CT1,00H
```

```
            BNZ         TIMER0_ISP_END
            ;5ms 到
            LDI         F_TIMER,0001B           ;设置“5ms 到”标志
            LDI         T5MS_CT2,01H            ;重置 5ms 计数器
            LDI         T5MS_CT1,04H
TIMER0_ISP_END:
            LDI         IE,0100B                ;打开 TIMER0 中断
            LDA         AC_BAK,00H              ;取出 AC 值
            RTNI                                ;返回
;*****************************************
; 上电程序
;*****************************************
RESET:
            NOP
;-------------------------------------
;清用户寄存器
POWER_RESET:
            LDI         DPL,00H
            LDI         DPM,02H
            LDI         DPH,00H
POWER_RESET_1:
            LDI         INX,00H
            ADIM        DPL,01H
            LDI         TMP,00H
            ADCM        DPM,00H
            BA3         POWER_RESET_2
            JMP         POWER_RESET_3
POWER_RESET_2:
            ADIM        DPH,01H
POWER_RESET_3:
            SBI         DPH,01H
            BNZ         POWER_RESET_1
            SBI         DPM,04H
            BNZ         POWER_RESET_1
;-------------------------------------
;初始化系统寄存器
SYSTEM_INITIAL:
            ;初始化 TIMER0
```

```
        LDI     TM0,07H             ;设置 TIMER0 预分频为/1
        LDI     TL0,06H
        LDI     TH0,00H             ;设置中断时间为 250us
        LDI     T5MS_CT2,01H
        LDI     T5MS_CT1,04H        ;计数器=14H=20, 计到 5ms
        ;初始化 I/O 口
        LDI     SETTING,1100B       ;设置上拉电阻使能
;-------------------------------------
MAIN_PRE:
        LDI     IRQ,00H
        LDI     IE,0100B            ;打开 Timer0 中断
;*********************************************
MAIN:
        ADI     F_TIMER,0001B
        BA0     HALTMODE            ;未到 5ms, 跳转
        ANDIM   F_TIMER,1110B       ;清"5ms 到" 标志
;************************************
; 模块:    线反转法按键扫描
; 输入变量:     FLAG1
; 使用变量:     PORTB, PORTC, PBOUT, PCOUT, ROWDATA, LINEDATA, TMP
;          KCODE_NEW, KCODE_OLD, KEYS_CT
; 输出变量:     FLAG1
;************************************
KEYSCAN:
        LDI     PORTC,0FH           ;打开 PORTC 上拉电阻
        LDI     PORTB,00H
        LDI     PBOUT,0FH           ;PORTB 输出低电平
        NOP                         ;等待稳定
        NOP
        NOP
        LDA     PORTC,00H           ;读取 PORTC 口电平
        STA     TMP,00H             ;暂存于 TMP
        SBI     TMP,0FH             ;若是没有按键按下, 则读入的数据全为 1
        BAZ     NO_KEY              ;数据全为 1, 没有按键按下, 跳转
        ;有按键按下
        LDA     TMP,00H             ;刚才从 PORTC 口读到的数据就是列数据
        STA     ROWDATA,00H         ;保存列数据到 ROWDATA
```

```
        LDI     PBOUT,00H           ;设置 PORTB 为输入口
        LDI     PORTB,0FH           ;打开 PORTB 上拉电阻
        LDI     PORTC,00H
        LDI     PCOUT,0FH           ;PORTC 输出低电平
        NOP                         ;等待稳定
        NOP
        NOP
        LDA     PORTB,00H           ;读取 PORTB 口电平，得到行数据
        STA     LINEDATA,00H        ;保存行数据到 LINEDATA
        LDI     PCOUT,00H           ;设置 PORTC 为输入口
;本可以使用行数据和列数据组成键值，但为方便使用，修改键值为 0～F
GET_CODE:
        SBI     LINEDATA,0EH
        BAZ     GET_LINE1           ;第一行，跳转
        SBI     LINEDATA,0DH
        BAZ     GET_LINE2           ;第二行，跳转
        SBI     LINEDATA,0BH
        BAZ     GET_LINE3           ;第三行，跳转
        SBI     LINEDATA,07H
        BAZ     GET_LINE4           ;第四行，跳转
        JMP     KEYSCAN_END         ;多个按键按下，无效不响应，跳转
GET_LINE1:
        LDI     KCODE_NEW,00H       ;第一行首键值为 0
        JMP     GET_CODE_1
GET_LINE2:
        LDI     KCODE_NEW,04H       ;第二行首键值为 4
        JMP     GET_CODE_1
GET_LINE3:
        LDI     KCODE_NEW,08H       ;第三行首键值为 8
        JMP     GET_CODE_1
GET_LINE4:
        LDI     KCODE_NEW,0CH       ;第四行首键值为 C
GET_CODE_1:
        SBI     ROWDATA,0EH
        BAZ     GET_ROW1            ;第一列，跳转
        SBI     ROWDATA,0DH
        BAZ     GET_ROW2            ;第二列，跳转
        SBI     ROWDATA,0BH
```

```
            BAZ         GET_ROW3            ;第三列，跳转
            SBI         ROWDATA,07H
            BAZ         GET_ROW4            ;第四列，跳转
            JMP         KEYSCAN_END         ;多个按键按下，无效不响应，跳转
GET_ROW1:
            ADIM        KCODE_NEW,00H       ;第一列，直接为行首键值
            JMP         KS_CHECK
GET_ROW2:
            ADIM        KCODE_NEW,01H       ;第二列，为行首键值加一
            JMP         KS_CHECK
GET_ROW3:
            ADIM        KCODE_NEW,02H       ;第三列，为行首键值加二
            JMP         KS_CHECK
GET_ROW4:
            ADIM        KCODE_NEW,03H       ;第四列，为行首键值加三
KS_CHECK:
            LDA         FLAG1,00H
            BA0         NO_KEY_1            ;之前的按键未松开，不做处理，跳转

            ADIM        KEYS_CT,01H         ;按键扫描次数加一
            SBI         KEYS_CT,01H
            BAZ         KEYSCAN_2           ;第一扫到该按键，直接把新扫到的键值赋给旧的键值，跳转
            LDA         KCODE_OLD,00H
            SUB         KCODE_NEW,00H       ;比较本次扫的键值与上次扫的键值是否相等
            BAZ         KEYSCAN_2           ;相等，跳转
            LDI         KEYS_CT,01H         ;不相等，认为是新的，将扫描次数设置为 1
KEYSCAN_2:
            LDA         KCODE_NEW,00H
            STA         KCODE_OLD,00H       ;把新扫到的键值赋给旧的键值
            SBI         KEYS_CT,0BH         ;前后连续扫描同一按键 11 次，中间间隔达到 50ms
            BNC         KEYSCAN_END         ;不到 50ms，跳走
            ;到 50ms
            LDI         KEYS_CT,00H         ;将扫描次数清零
            ORIM        FLAG1,0011B         ;置"按键未松开”标志和"有按键按下”标志
            JMP         KEYSCAN_END
NO_KEY:
            ADI         FLAG1,0001B
            BA0         NO_KEY_1            ;按键已松开，不需再判断按键松开，跳走
```

```
            ADIM        KEYS_CT,01H             ;扫描次数加一
            SBI         KEYS_CT,0BH             ;前后连续扫描到没有按键 11 次, 中间间隔达到 50ms
            BNC         KEYSCAN_END             ;不到 50ms, 跳走
            ANDIM       FLAG1,1110B             ;清"按键未松开"标志
NO_KEY_1:
            LDI         KEYS_CT,00H             ;扫描次数清零
KEYSCAN_END:
;*************************************
KEY_PROC:
            ADI         FLAG1,0010B
            BA1         HALTMODE                ;没有按键按下, 跳转
KEY_PROC_1:
            ANDIM       FLAG1,1101B             ;清"有按键按下"标志
            LDA         KCODE_OLD,00H
            STA         KEYCODE,00H             ;把扫描到的键值送到键码
            ;----------------------
            ;这里可以对按键按下后的程序状态进行处理
            ;----------------------
KEY_PROC_END:
;*************************************
HALTMODE:
            NOP
            HALT
            NOP
            NOP
            JMP         MAIN
;*********************************************
            END
```

5.4 用I/O 驱动LCD

在现今的电子产品中，LCD 显示被广泛的应用。LCD 显示驱动有内建于 MCU 中的亦有独立于 MCU 单一的驱动 IC。这些 IC 能驱动的点数一般比较多，在一些需要显示点数不多的应用中这些 IC 就显得浪费。所以在这里我们介绍一种用 I/O 来驱动 LCD 做显示的方法来满足一些点数不多的显示应用。

5.4.1 I/O 驱动LCD 原理

LCD 的显示原理请参考 3.6 节，这里不多叙述。大家都知道 I/O 口最多只能有三种状态: 高电平，低电平及悬空状态(Floating)如何造出 LCD 驱动波形呢? 下面我们以 2X8(2 个 COM 和 8 个 SEG)的例子来做一介绍。观察图 5-14，这是一个 1/2bias，1/2duty 的 LCD 驱动波形，观察 COM0 和 COM1 可以发现同一时间内只有一个 COM 口有输出，它的平均电压为零。节点信号 SEGMENT 则表示要显示的数据，如果节点信号与 COM 口之间有出现脉冲，代表对应的这个点是亮的，反之则是暗的。在知道要显示的内容的情况下，如果是要显示，那么 SEGMENT 的信号就和 COM 相反而产生脉冲，如果是不显示，那么 SEGMENT 的信号就和 COM 的信号一样而两端之间没有脉冲，没有显示。

有了 LCD 驱动波形的概念之后，接着就可以着手设计电路了，观察 SEGMENT 上的波形，对微控制器来讲不是问题，VLCD=1，VSS=0 就能造出漂亮的方波。那 COMMON 上面的阶梯波怎么解决呢? 其实只要利用 I/O Port 三种状态就可实现，如图 5-15 所示，PD1、PD2 为 COM0、COM1 的输出波形，利用两个分压电阻以及 I/O Port 的特性，表 5-1 是它的真值表，由此表不难看出，要产生 VDD 或者 0 电压，只要利用端口输出 1 或者 0 就可以产生，可是没有直接输出 1/2 VDD(V1)。其实要产生 1/2VDD 是很简单的，因为当 I/O Port 设成输入时形成一个高阻抗的状态，对 COM 而言所看到的只是 R1 和 R2 分压而已。所以我们 将 R1=R1=100K 欧姆，那么在 COM 端来说就是 1/2VDD 的电压了。

PD1	COM0 电压
0	VSS
1	VLCD
Floating	V1

表 5-1

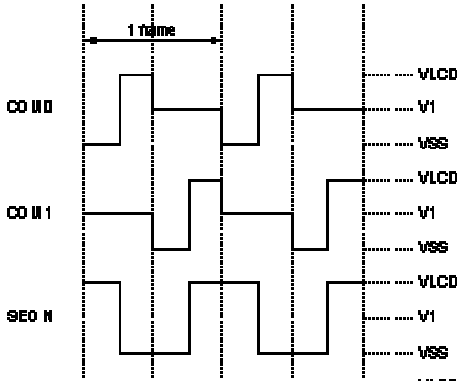


图 5-14　1/2 bias 与 1/2 duty

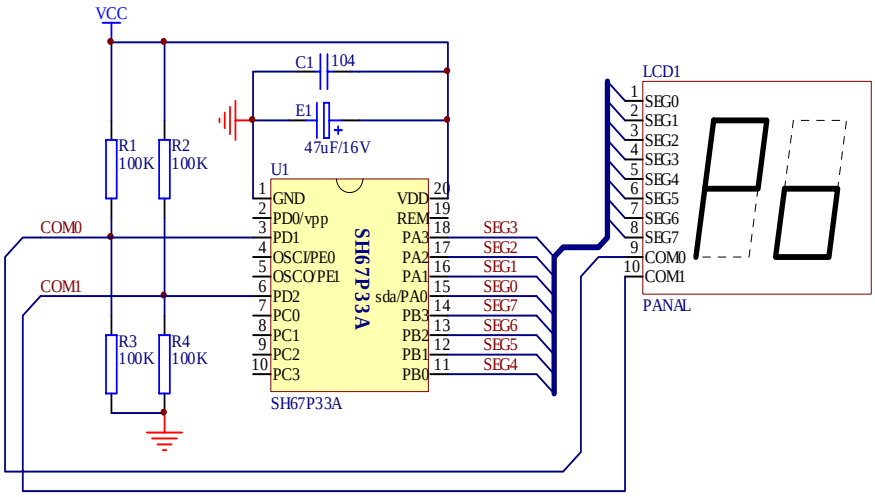


图 5-15　2X8 板面显示

同理如果您需要 1/3 bias 的场合，如图 5-16 所示再变化一下 I/O 的状态，但是 SEGMENT 与 COMMON 都需要电阻分压，而且驱动一条 LCD Pin 需要两个 I/O，不管对多点数或者少点数来说，都十分不经济，相对地程序也会更复杂。

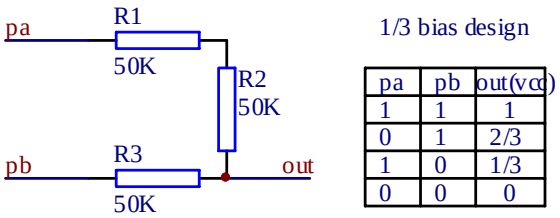


图 5-16　1/3bias 原理及真值表

5.4.2 电路设计

利用 SH67P33 的 PORTD1,2 来做为 COM 口，PORTA,B 共 8 个 I/O 做为 SEG 来驱动 2 个 8 段 LCD 显示“Po”，原理图如图 5-15。其真值表如 5-2 所示。

表 5-2

	SEG0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7
COM0	1A	1B	1C		2A	2B	2C	
COM1	1F	1G	1E	1D	2F	2G	2E	2D

程序设计

例[5-5]

```
;;;;;;;;;;;;;;;;system define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IE              EQU   00H        ;中断允许标志
IRQ             EQU   01H        ;中断请求标志
PA              EQU   08H        ;PORT A 数据寄存器
PB              EQU   09H        ; PORTB 数据寄存器
PC              EQU   0AH        ; PORT C 数据寄存器
PD              EQU   0BH        ; PORT D 数据寄存器
PE              EQU   0CH        ; PORT E 数据寄存器
PACR            EQU   16H        ; PORT A 输入输出控制寄存器
PBCR            EQU   17H        ; PORT B 输入输出控制寄存器
PCCR            EQU   18H        ; PORT C 输入输出控制寄存器
PDCR            EQU   19H        ; PORT D 输入输出控制寄存器
PECR            EQU   1AH        ; PORT E 输入输出控制寄存器
PMOD            EQU   12H        ;BIT0-BIT2：载波输出时钟预分频设定
                                 ;BIT3：下拉允许控制位
DPL             EQU   10H        ;数据指针低位
DPM             EQU   11H        ;数据指针中间位
DPH             EQU   12H        ;数据指针高位
INX             EQU   0FH        ;间接索引寄存器
;;;;;;;;;;;;;;;; ram define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ACCBK           EQU   20H        ;备份 ACC 寄存器

SCANFLAG        EQU   2CH        ;信号输出扫描寄存器
COMP            EQU   1H         ;0=扫描 COM0 端  1=扫描 COM1 端
CYCLE           EQU   2H         ;0=扫描端口的前半周期,      1=扫描端口的后半周期

TEMP            EQU   30H        ;4 位临时寄存器
TEMPL           EQU   31H        ;8 位临时寄存器低位
TEMPH           EQU   32H        ;8 位临时寄存器高位

PABUF           EQU   38H        ;PORTA 输出缓冲
PBBUF           EQU   39H        ;PORTB 输出缓冲
PCBUF           EQU   3AH        ;PORTC 输出缓冲

SEG0            EQU   40H        ;LCD 显示区      ;PA0 端输出
SEG1            EQU   SEG0+1     ;LCD 显示区;PA1 端输出
SEG2            EQU   SEG0+2     ;LCD 显示区;PA2 端输出
```

```
SEG3            EQU   SEG0+3          ;LCD 显示区;PA3 端输出
SEG4            EQU   SEG0+4          ;LCD 显示区;PB0 端输出
SEG5            EQU   SEG0+5          ;LCD 显示区;PB1 端输出
SEG6            EQU   SEG0+6          ;LCD 显示区;PB2 端输出
SEG7            EQU   SEG0+7          ;LCD 显示区;PB3 端输出

;;;;;;;;;;;;;;;; CONSTANT define ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
COMCR           EQU   PDCR            ;定义 COM 端输入输出控制寄存器
COM             EQU   PD              ;定义 COM 输出端口
IET0            EQU   04H             ;
SCALE8          EQU   0100B;

          ORG       0000H
          JMP       RESET             ;复位入口地址
          JMP       ONLYRET
          JMP       T0ISR             ;定时器 T0 中断入口地址
          JMP       ONLYRET
          JMP       PBCISR            ;PBC 中断入口地址

PBCISR:

ONLYRET:
          RTNI                        ;推出 PBC 中断

;LCD 刷新 32hz，FRAME=1000/32HZ=31.25ms，所以 FRAME/4=7.8125ms 约等于 8ms。
;定时器 8ms 溢出，振荡器采用内部 4M RC，系统时钟采用 16 分频。
;定时器设置为计数 250，所以初值为 6。
;所以 8000/250=32，算出 timer0 预分频 32/4=8

T0ISR:
          STA       ACCBK,00H         ;备份 ACC 到寄存器在 ACCBK
          LDI       WDT,08H           ;清除 watchdog 计数

          EORIM     SCANFLAG,CYCLE    ;切换扫描端口输出周期
          LDA       SCANFLAG,00H
          STA       TEMP,00H
          ANDIM     TEMP,03H
          SBI       TEMP,03H
```

```
            BAZ     CANCOM11        ;扫描 com1 的后半周期
            SBI     TEMP,02H
            BAZ     SCANCOM10       ;扫描 com0 的后半周期
            SBI     TEMP,01H
            BAZ     SCANCOM01       ;扫描 com1 的前半周期
SCANCOM00:                          ;扫描 com0 的前半周期
            LDI     COMCR,0010B     ;COM0 输出，COM1 输入
            LDI     COM,0000B       ;COM0 输出低电平

            LDI     PABUF,0FH
            LDA     SEG0,00H        ;输出 SEG0.0 到 PA0
            BA0     $+2
            ANDIM   PABUF,1110B
            LDA     SEG1,00H        ;输出 SEG1.0 到 PA1
            BA0     $+2
            ANDIM   PABUF,1101B
            LDA     SEG2,00H        ;输出 SEG2.0 到 PA2
            BA0     $+2
            ANDIM   PABUF,1011B
            LDA     SEG3,00H        ;输出 SEG3.0 到 PA3
            BA0     $+2
            ANDIM   PABUF,0111B
            LDA     PABUF,00H
            STA     PA,00H
            LDI     PBBUF,0FH
            LDA     SEG4,00H        ;输出 SEG4.0 到 PB0
            BA0     $+2
            ANDIM   PBBUF,1110B
            LDA     SEG5,00H        ;输出 SEG5.0 到 PB1
            BA0     $+20
            ANDIM   PBBUF,1101B
            LDA     SEG6,00H        ;输出 SEG6.0 到 PB2
            BA0     $+2
            ANDIM   PBBUF,1011B
            LDA     SEG7,00H        ;输出 SEG7.0 到 PB3
            BA0     $+2
            ANDIM   PBBUF,0111B
            LDA     PBBUF,00H
```

```
            STA     PB,00H
            JMP     TORET

SCANCOM10:                          ;扫描 COM0 后半周期
            LDI     COMCR,0010B     ;COM0 输出，COM1 输入
            LDI     COM,0010B       ;COM0 输出高电平

            LDI     PABUF,00H
            LDA     SEG0,00H        ;输出 SEG0.0 到 PA0
            BA0     $+2
            ORIM    PABUF,0001B
            LDA     SEG1,00H        ;输出 SEG1.0 到 PA1
            BA0     $+2,00H
            ORIM    PABUF,0010B
            LDA     SEG2,00H        ;输出 SEG2.0 到 PA2
            BA0     $+2
            ORIM    PABUF,0100B
            LDA     SEG3,00H        ;输出 SEG3.0 到 PA3
            BA0     $+2
            ORIM    PABUF,1000B
            LDA     PABUF,00H
            STA     PA,00H
            LDI     PBBUF,00H
            LDA     SEG4,00H        ;输出 SEG4.0 到 PB0
            BA0     $+2
            ORIM    PBBUF,0001B
            LDA     SEG5,00H        ;输出 SEG5.0 到 PB1
            BA0     $+2
            ORIM    PBBUF,0010B
            LDA     SEG6,00H        ;输出 SEG6.0 到 PB2
            BA0     $+2
            ORIM    PBBUF,0100B
            LDA     SEG7,00H        ;输出 SEG7.0 到 PB3
            BA0     $+2
            ORIM    PBBUF,1000B
            LDA     PBBUF,00H
            STA     PB,00H
            JMP     TORET
```

```
SCANCOM01:                          ;扫描 COM1 前半周期
          LDI       COMCR,0100B     ;COM0 输入，COM1 输出
          LDI       COM,0000B       ;COM1 输出低电平

          LDI       PABUF,0FH
          LDA       SEG0,00H        ; 输出 SEG0.1 到 PA0
          BA1       $+2
          ANDIM     PABUF,1110B
          LDA       SEG1,00H        ; 输出 SEG1.1 到 PA1
          BA1       $+2
          ANDIM     PABUF,1101B
          LDA       SEG2,00H        ; 输出 SEG2.1 到 PA2
          BA1       $+2
          ANDIM     PABUF,1011B
          LDA       SEG3,00H        ; 输出 SEG3.1 到 PA3
          BA1       $+2
          ANDIM     PABUF,0111B
          LDA       PABUF,00H
          STA       PA,00H
          LDI       PBBUF,0FH
          LDA       SEG4,00H        ; 输出 SEG4.1 到 PB0
          BA1       $+2
          ANDIM     PBBUF,1110B
          LDA       SEG5,00H        ; 输出 SEG5.1 到 PB1
          BA1       $+2
          ANDIM     PBBUF,1101B
          LDA       SEG6,00H        ; 输出 SEG6.1 到 PB2
          BA1       $+2
          ANDIM     PBBUF,1011B
          LDA       SEG7,00H        ; 输出 SEG7.1 到 PB3
          BA1       $+2
          ANDIM     PBBUF,0111B
          LDA       PBBUF,00H
          STA       PB,00H
          JMP       TORET

SCANCOM11:                          ;扫描 COM1 后半周期
```

```
        LDI     COMCR,0100B     ;COM0 输入，COM1 输出
        LDI     COM,0100B       ;COM1 输出高电平

        LDI     PABUF,0H0
        LDA     SEG0,00H        ; 输出 SEG0.1 到 PA0
        BA1     $+2
        ORIM    PABUF,0001B
        LDA     SEG1,00H        ; 输出 SEG1.1 到 PA1
        BA1     $+2
        ORIM    PABUF,0010B
        LDA     SEG2,00H        ; 输出 SEG2.1 到 PA2
        BA1     $+2
        ORIM    PABUF,0100B
        LDA     SEG3,00H        ; 输出 SEG3.1 到 PA3
        BA1     $+2
        ORIM    PABUF,1000B
        LDA     PABUF,00H
        STA     PA,00H
        LDI     PBBUF,00H
        LDA     SEG4,00H        ; 输出 SEG4.1
        BA1     $+2
        ORIM    PBBUF,0001B
        LDA     SEG5,00H        ; 输出 SEG5.1
        BA1     $+2
        ORIM    PBBUF,0010B
        LDA     SEG6,00H        ; 输出 SEG6.1
        BA1     $+2
        ORIM    PBBUF,0100B
        LDA     SEG7,00H        ; 输出 SEG7.1
        BA1     $+2
        ORIM    PBBUF,1000B
        LDA     PBBUF,00H
        STA     PB,00H
        JMP     TORET

TORET:
        LDA     SCANFLAG,00H    ;检测是否是后半周期
        BA1     $+2
```

```
        JMP     $+2
        EORIM   SCANFLAG,COMP   ;如果是后半周期的话就切换 COM 端口
        LDI     IRQ,00H         ;清除中断请求标志
        LDI     IE,IET0         ;允许定时器 T0 中断
        LDA     ACCBK,00H       ;恢复 ACC 的值
        RTNI

RESET:
        NOP                     ;上电稳定空跑 4 条指令
        NOP
        NOP
        NOP
        LDI     WDT,08H         ;清除 watchdog 计数
        LDI     IRQ,00H         ;清除中断请求标志
        LDI     IE,00H          ;不允许中断

        LDI     TEMPH,0FH       ;上电延时
        LDI     TEMPL,0FH
        SBIM    TEMPL,01H
        BC      $-1
        LDI     WDT,08H
        SBIM    TEMPH,01H
        BC      $-4

        LDI     REM,00H         ;关闭红外输出
        LDI     PACR,0FH        ;PA 设置为输出
        LDI     PBCR,0FH        ;PB 设置为输出
        LDI     PCCR,0FH        ;PC 设置为输出
        LDI     PDCR,06H        ;PD1，PD2 设置为输出，PD0 为输入
        LDI     PECR,0FH        ;PE 设置为输出
        LDI     PA,00H          ;PA 输出低电平
        LDI     PD,00H          ;PD 输出低电平
        LDI     PE,00H          ;PE 输出低电平
        LDI     PB,00H          ;PB 输出低电平
        LDI     PC,00H          ;PC 输出低电平
        LDI     PMOD,0101B      ;不允许内部上下拉电阻

CLEARRAM:                       ;清 0 RAM 020H-4FH 区
```

```
        LDI     DPL,00H
        LDI     DPM,02H
        LDI     DPH,00H
?CLRINX:
        LDI     INX,00H
        ADIM    DPL,01H
        BNC     ?CLRINX
        ADIM    DPM,01H
        ANDIM   DPM,07H
        SBI     DPM,05H
        BNC     ?CLRINX

        LDI     T0M,SCALE8      ;设置定时器 T0 预分频为 8 分频
        LDI     T0L,06H         ;设置定时期初值为 6
        LDI     T0H,00H
        LDI     IE,IET0         ;允许定时器中断

        LDI     SEG0,03H        ;设置显示的“Po”显示区数据
        LDI     SEG1,03H
        LDI     SEG2,02H
        LDI     SEG3,00H
        LDI     SEG4,00H
        LDI     SEG5,02H
        LDI     SEG6,03H
        LDI     SEG7,02H

TOHALT:
        LDI     WDT,08H         ;清除 watchdog 计数
        LDI     IRQ,00H         ;清除中断请求标志
        LDI     IE,IET0         ;允许定时器 T0 中断
        NOP
        HALT                    ;halt
        NOP
        NOP
        NOP
        JMP     TOHALT

        END
```