7.3　综合程序

7.3.A　LED/LCE/键盘扫描综合程序源程序

源程序：SLAVR73A.ASM
综合程序功能如下：
(1)．LED 及 LCD 显示程序,有自动识别 LED 或 LCD 功能,设 LCD 优先级高;
(2)．键盘扫描输入程序,0-F 为 16 个数字键；还有上档命令键,EXEC--执行键;
E2PROM--读键；SRAM--读写键；MON--返回初始状态键;
LAST--上一单元地址键；NEXT--下一单元地址键;
SHIFT--转换上档命令键,先按 SHIFT 键,再按命令键,就执行上档键的命令;
/RST--复位键,执行程序后,要机器回到初始化状态,必须按复位键;
(3) 按数字键显示对应数字,并有小数点作为光标,提示下一步工作位置,
按命令键(先按 SHIFT)执行相应命令;
(4) 对应功能入口地址(地址数字后零可省)
0070H-01FFH 读、写内部 SRAM(监控规定 SRAM 读写范围)
0000H-01FFH 读片内 E2PROM 数据
0200H-歌曲-祝你生日快乐,万水千山总是情
0300H-LED 上 8 字循环显示
0320H-LED 上 0-F 字符循环显示
0400H-逐次逼近法 A/D 转换(需接网络电阻,另见说明)
0500H-LCD 初始化程序
0700H-LCD 上尖头字符左右移位程序
0740H-LCD 上 0-F 字符循环显示
0800H-LCD 显示 LCD 所有字符
程序清单见光盘文件 SLAVR73A.ASM

7.3. B　LED 键盘扫描综合程序

源程序：SLAVR73B.ASM
综合程序功能如下：
(1)．键盘扫描输入程序,0-F 为 16 个数字键；还有上档命令键,EXEC--执行键;
E2PROM--读键；SRAM--读写键；MON--返回初始状态键;
LAST--上一单元地址键；NEXT--下一单元地址键;
SHIFT--转换上档命令键,先按 SHIFT 键,再按命令键,就执行上档键的命令;
/RST--复位键,执行程序后,要机器回到初始化状态,必须按复位键;
(2) 按数字键显示对应数字,并有小数点作为光标,提示下一步工作位置,
按命令键(先按 SHIFT)执行相应命令;
(3) 对应功能入口地址(地址数字后零可省)
0070H-01FFH 读、写内部 SRAM(监控规定 SRAM 读写范围)
0000H-01FFH 读片内 E2PROM 数据
0200H-歌曲-祝你生日快乐,万水千山总是情
0300H-LED 上 8 字循环显示
0320H-LED 上 0-F 字符循环显示
程序清单见光盘文件 SLAVR73B.ASM

7.3.1 在 SL-AVR 开发实验器 LED 上实现字符 8 的循环移位显示程序

```
; 源程序: SLAVR731.ASM;本程序在 SL-AVR 开发实验器上通过
;请你 1.如何修改字形; 2.改变字符个数,二位或三位或一隔一显示;
;3. 改变字形移动方向; 4.改变字符移位速度;
;
.include"8515def.inc"          ;器件配置文件
.def temp=r16                  ;数据暂存器
.def scndp=r22                 ;LED 显示位置暂存器
```

h	g	f	e	d	c	b	a	十六进制 码	字形
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0		
0	0	1	1	1	1	1	1	3F	0
0	0	0	0	0	1	1	0	06	1
0	1	0	1	1	0	1	1	5B	2
0	1	0	0	1	1	1	1	4F	3
0	1	1	0	0	1	1	0	66	4
0	1	1	0	1	1	0	1	6D	5
0	1	1	1	1	1	0	1	7D	6
0	0	0	0	0	1	1	1	07	7
0	1	1	1	1	1	1	1	7F	8
0	1	1	0	1	1	1	1	6F	9
0	1	1	1	0	1	1	1	77	A
0	1	1	1	1	1	0	0	7C	B
0	0	1	1	1	0	0	1	39	C
0	1	0	1	1	1	1	0	5E	D
0	1	1	1	1	0	0	1	79	E
0	1	1	1	0	0	0	1	71	F
1	1	1	1	0	0	1	1	F3	P.
0	1	1	1	0	1	1	0	76	H

硬件设定高电平
LED笔划点亮
低电平LED管选中

```
.org $0000
    rjmp reset
.org $030
reset:  ldi temp,low(ramend);设置堆栈指针。
    out spl,temp
    ldi temp,high(ramend)
    out sph,temp
    ldi temp,$ff     ;设置 B、D 口输出。
    out ddrb,temp
    out ddrd,temp
    out portd,temp
    ldi temp,$7f     ;字形 8 的代码为$7F(可修改)。
    out portb,temp
again:  sec          ;置进位位为 1(低电平 LED 亮,高电平 LED 灭)
    ldi scndp,0b11011111 ;扫描显示 SCANDP(可修改)
route:  out portd,scndp      ;从 LED 最左一位(D5)右移(可修改)
    ldi temp,$40     ;设置延时常数(可修改)。
    rcall delay      ;调用延时
    ror scndp        ;右循环(可修改)
```

```
    brcc again      ;显示下一位
    rjmp route      ;循环显示
delay:              ;通用延时子程序略。
```

7.3.2 电脑放音机

源程序：SLAVR732.ASM

AVR 单片机在儿童智能玩具中的应用--音乐玩具

利用单片机开发儿童智力玩具大有作为,尤其单片机扩展存贮器方便，而大容量存贮器价格也很低,64KB 的 EPROM 可存放 300 多首歌曲,8M 位 EPROM 可存放 5000 多首歌曲,几个芯片就可组成一个音乐库,这是用其它方法难办的。

利用 AVR 单片机产生乐曲音符,再把乐谱翻译成计算机音乐语言，由单片机进行信息处理,再经过信号放大,由耳机或喇叭放出乐曲声．由于音符和节拍是由计算机产生的,所以发音音符和节拍准确,可见音乐从娃娃开始抓起，音乐玩具是儿童第一个好老师。利用单片机的中断,I/O 口控制功能，可以做到电脑放音机有自动连续放音功能,乐曲全部放完自动从头开始连续放音,循环不断。

如何产生音乐频率:

1.要产生音频脉冲,只要算出某一音频的周期(1/频率),然后将此周期除以 2，即为半周期的时间,然后利用计时器计时此半周期时间,每当计时到后就将输出脉冲对 I/O 口反相,然后重复计时此半周期时间再对 I/O 口反相,如此就可在 I/O 口引脚上得到此频率的脉冲(程序驱动 I/O 口反相,即正、负各半周期为一个周期,才能使喇叭"吸、放"发声);

2．利用 AVR 单片机的内部计时器让其工作在计数模式 MODE1(16 位定时计数器)下,改变计数值 TCNT1H 及 TCNT1L 以产生不同的频率;

3．例如以 6MHZ 晶振为例:要产生频率为 523HZ,其周期 T= 1/523= 1912us,其半周期为 1912/2=956us,因此只要令计数器计时 956us/1us=956(为半周期).所以在每计数 956 次时将 I/O 反相,就可得到中音 DO(523HZ).

计数脉冲值与频率的关系公式如下:

N=Fi(6MHz 晶振,CPU 产生的频率)÷2(半周期)÷Fr

N:计数值

Fi:以 6MHZ 晶振为例,内部计时(数)一次需 2us，频率单位为 1 周期/秒,即 HZ

1 周期/2us=1 周期/2X10^{-6}秒=500000 次/秒=500000HZ

故其频率为 500000HZ

Fr:要产生的频率

4．其计数值的求法如下:

T(16 位计数器计多少后溢出)=65536(16 位二进制计数器,计满数溢出时的计数值为 2 的 16 次方)-N=65536-Fi/2/Fr

例如:求低音 DO(262HZ),中音 DO(523HZ),高音 DO(1046HZ) 的计数值?

设 K=65535　F=500000=Fi=0.5MHZ

T=65536-N=65536-Fi/2/Fr=65536-500000/2/Fr=65536-250000/Fr

低音 DO 的 T=65535-1908=63627(十进制数)

中音 DO 的 T=65535-0956=64579(十进制数)

高音 DO 的 T=65535-0478=65057(十进制数)

5．C 调各音符频率与计数值 T 的对照表：

音符	频率 HZ	半周期	TCNT 值	音符	频率 HZ	半周期	TCNT 值
低 1DO	262	1908μS	63627	#4FA#	740	0676μS	64859
#1DO#	277	1805	63730	中 5SO	784	0638	64897
低 2RE	294	1700	63835	#5SO#	831	0602	64933
#2RE#	311	1608	63927	中 6LA	880	0568	64967
低 3M	330	1516	64020	#6LA#	932	0536	64999
低 4FA	349	1433	64012	中 7SI	988	0506	65029
#4FA#	370	1350	64185	高 1DO	1046	0478	65057
低 5SO	392	1276	64259	#DO#	1109	0451	65084
#5SO#	415	1205	64330	高 2RE	1175	0426	65109
低 6LA	440	1136	64399	#2RE#	1245	0402	65133
#6LA#	466	1072	64463	高 3M	1318	0372	65156
低 7SI	494	1012	64523	高 4FA	1397	0358	65177
中 1DO	523	0956	64579	#4FA#	1480	0338	65197
#1DO#	554	0903	64632	高 5SO	1568	0319	65216
中 2RE	578	0842	64683	#5SO#	1661	0292	65243
#2RE#	622	0804	64731	高 6LA	1760	0284	65251
中 3M	659	0759	64776	#6LA#	1865	0268	65267
中 4FA	698	0716	64819	高 7SI	1976	0253	65282

"#"表示半音,用于上升或下降半个音

如何产生节拍：

每个音符使用 1 个字节，每个节拍使用 1 个字节,AVR 程序存储器可以设为 16 位,即 1 个字,或称双字节,所以一个字的高 8 位存放音符码,低 8 位存放节拍码。如果 1 拍节为 0.4 秒则 1/4 拍是 0.1 秒,只要设定延迟时间就可求得节拍的时间,我们假设 1/4 拍为 1 DELY 单位,则 1 拍应为 4 个 DELY,以此类推,只要求得 1/4 拍的 DELY 单位时间,其余的节拍就是它的倍数.

1/4 拍的延迟时间=187 毫秒

节拍与节拍码对照表

节拍码	节拍数(拍)	节拍码	节拍数(拍)
1	1/4	1	1/8
2	2/4	2	1/4
3	3/4	3	3/8
4	1	4	1/2
5	1 又 1/4	5	5/8
6	1 又 1/2	6	3/4
8	2	8	1
10	2 又 1/2	10	1 又 1/4
12	3	12	1 又 1/2
16	1 又 3/4		

建立音乐的步骤：

找出乐曲，然后对照音符表,翻译出乐曲码,用程序伪指令 DB 输入曲码和节拍码;也可直接

在调试窗口的程序存储器窗口$0100 地址输入曲码和节拍码(只适用于在线实时仿真器)。

例:音符表练习,

1.把简谱翻译成曲码代码;

以下音符均设为一拍，代码为 4

1 2 3 4 5 6 7(低八度音)1 2 3 4 5 6 7 (中音) 1(高音) 1(高音) 7 6 5 4 3 2 1(中音) 7 6 5 4 3 2 1(低八度音)

曲码	1	3	5	6	8	10	12	13	15	17	18	20	22	24	25
简码	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1
	低八度音							中　音							高音

曲码	36	34	32	30	29	27	25	24	22	20	18	17	15	13	12
简码	7	6	5	4	3	2	1	7	6	5	4	3	2	1	7
	高八度音							中　音							低音

最后翻译成乐曲加节拍代码为:

01,04,03,04,05,04,06,04,08,04.10,04,12,04,13,04,13,04,15,04,17,04,18,04,20,04,
22,04,24,04,25,04,
25,04,36,04,34,04,32,04,30,04,30,04,29,04,27,04,25,04,24,04,22,04,20,04,18,04,
17,04,15,04,13,04,12,04

以上乐曲数据用伪指令 DB 方式输入"乐曲.ASM"的$0100 地址,再汇编一次就可下载试听,

注意: 音符节拍间用逗号隔开,不要不小心键入小数点,因为逗号键右边是小数点键,

键入小数点,程序汇编时将造成计算机死机!

00 00(4 个零为所有曲结束标志)

1. 把乐曲代码输入计算机

把 SL-AVR 实验器与 PC 机连机,U4 插上 AT90S8515 芯片,插上音响器短路块,开机通电。进入 AVR 下载窗口,进行下载操作,下载结束应能听到乐曲声。

```
;************* 乐曲程序 SLAVR732.ASM ************************
;* 标题:AT90S8515 C 口输出乐曲声–电脑放音机
;* 版本:        1.0
;* 最后更新日期: 2000.08.08
;* 支援  E-mail:     gzsl@sl.com.cn
;* 描述
;*    用 SL-AVR 万用下载开发实验器做样机,在 AT90S8515 的 C 口接喇叭发出乐曲声,
;* 请你把最喜爱的乐曲送入单片机! 起始地址为$0100,也可把曲码、节拍码在调试窗口中的
;* 程序存储器窗口(Program     Memory)内,从$0100 地址,
;*    用键盘直接输入乐曲(仅适合 ICE-200 实时仿真器)
;* 作者: SL.
;*程序适用于所有单片机
;*****************************************************************************

.include"8515def.inc"          ;文件头 AT90S8515 器件配置文件,不同的器件有不同的器件配置文件
      rjmp RESET               ;AVR 重新定位
.def  TEMPDH      =r2          ;寄存器定义
.def  TEMPDL =r3
.def  CNT =r10
```

```
.def  SCNN    =r11
.def  KEYN    =r12
.def  SCNK    =r13
.def  SCNDP   =r14
.def  KSNI    =r15
.def  TEMP    =r16          ;数据暂存器
.def  TEMP1   =r17
.def  TEMP2   =r18
.def  TEMP3   =r19
.def  SCNTT   =r26;
.def  MUSN    =r22          ;输出乐曲声暂存器
.def  TONL    =r21          ;节拍码低位
.def  TONH    =r20          ;节拍码高位
.def  PLYTON =r25           ;存乐曲码
.def  TONSET =r24
.def  TONLNG      =r23      ;存节拍码
.cseg
.org 0x06                   ;TIM1__OVF 定时器 1 溢出中断处理入口地址
intt1:      RJMP    OUTPM   ;转定时器 1 溢出中断处理,发音周期到,则跳转到发音输出态
.cseg
.org 0x010                  ; 定时器 1 溢出中断处理程序,发音起始地址
                            ;发音周期到重新装入计时值并将输出到 PORTC 口
OUTPM: OUT      TCNT1H,TONH ;重新将 TONH 新计时值载入 TCNT1H 内
       OUT      TCNT1L,TONL ; 重新将 TONL 新计时值载入 TCNT1L 内
       SBIS     PORTC,00    ;先检测 PORTC 口是否为 1 而跳转
       RJMP     SETOP1      ;若是 PORTC 口为 0 则跳到 SETOP1 令 PORTC 口转为 1
SETOP0: CBI  PORTC,00       ;若 PORTC 为 1 则令 PORTC 转为 0
       LDI  MUSN,$00        ;同时令 MUSN 为 00 值
       RETI                 ;回中断前主程序并令可再次中断返回
SETOP1: SBI  PORTC,00       ;若 PORTC 为 0 则令 PORTC 转为 1
       LDI  MUSN,$01        ;同时令 MUSN 为 01 值
       RETI                 ; 回中断前主程序并令可再次中断返回
.cseg
.org 0x020                  ;主程序起始地址,必须跳过中断区
RESET:
       ldi    temp,low(RAMEND)     ;RAMEND 为 8515def.inc 内建值为 025FH
       out    SPL,temp             ;启始堆栈指针低位将 TEMP=02H 放入 SP=3DH
;若硬件堆栈或者片 AVR 片内含 SRAM 小于 256B 时,下列二行程序可省略,
       ldi    temp,high(RAMEND)    ;以 TEMP=R16<5F 为数据装入缓冲暂存器
       out    SPL+1,temp           ;堆栈指针高位将 R16=TEMP=5FH 放入 SPL+1=3EH
       wdr                         ;在使用看门狗计时器前需重设看门狗计时器,为
                                   ;避免在接下来程序前就因 WDT 已快计时溢出而重设
       ldi    temp,$0F             ;WDTCR 地址$21 设定以 TEMP 缓冲令 WDE=D3=1
       out    WDTCR,temp           ;并会预除为 2048mS 设定 WDE=1=D3 输出到 WDTCR 内
```

```
        LDI  MUSN,$00                ;令 MUSN 为 00 值
        ldi    temp,$00              ;令 TEMP 暂存器放入 00
        OUT      TCCR1A,TEMP         ;TEMP=00 内含输出到 TCCR1A 内禁止比较器及 PWM 动作
        OUT      TCCR1B,TEMP         ;将 TEMP=00 内含输出到 TCCR1B 内停止 TC1 计时及捕抓
        LDI      TEMP,$02            ;将 02 值预存入 SRAM 的 0100H 内作 TC1 的
        STS      $0100,TEMP          ;TCCR1B 控制内含令 TC1 为计时预除 8
LEDA:   CLI                          ;中断总开关 sreg=d7=i=1
        Ldi      r16,0b10000000          ;令 toiel=1 触发中断
        out      timsk,r16               ;将 R16 的 D7=1 令 TOIE1=1 触发中断
        LDI      TEMP,$FF            ;设 AVR 单片机 I/O 口方向宏寄存器为输出
        OUT      DDRA,TEMP           ;A 口为输出
        OUT      DDRB,TEMP           ;B 口为输出
        OUT      DDRC,TEMP           ;C 口为输出
;       OUT      DDRD,TEMP           ;D 口为输出
        LDI      TEMP,0B11111111     ;关灭 I/O 口的 LED 发光二极管
        OUT      PORTC,TEMP          ;C 口输出乐曲声
        OUT      PORTA,TEMP          ;关 A 口 LED 灯,硬件设定高电平 LED 暗
        OUT      PORTB,TEMP          ;关 B 口 LED 灯,硬件设定高电平 LED 暗
        OUT      PORTD,TEMP          ;关 D 口 LED 灯,硬件设定高电平 LED 暗
        CLR      TEMP2               ;暂存器清零
        CLR      TEMP1               ;暂存器清零
        CLR      KSNI                ;暂存器清零
        LDI      SCNTT,$02           ;
        CLR      TONLNG              ;暂存器清零
STARTP: WDR                          ;关看门狗
        LDI      ZH,HIGH(PLYTAB*2)   ;启动演奏则令数据装入 Z 地址
        LDI      ZL,LOW(PLYTAB*2)    ;音乐演奏乐谱存放在 PLYTAB*2 起始地址
NEXMUT:     LPM                  ;将 Z 所指程序存储器乐曲,依次取音符码及节拍码置于 R0
        MOV      PLYTON,R0       ;将取出的第一个音符码装入 PLYTON 作周期控制
        LD       R0,Z+           ;以 LD R0,Z+指令使得 Z 间接寻址加 1
        LPM                      ;将 Z 所指程序存储器乐曲,依次取节指码置于 R0
        MOV      TONLNG,R0       ;将取出的第一个节拍码装入 PLYTON 作节拍控制
        OR       R0,PLYTON       ;将此 R0 节拍码与音符码 PLYTON 作 OR 运算
        LD       R0,Z+           ;以 LD R0,Z+指令使得 Z 间接寻址加 1
        BRNE     PLAYM           ;若音符码及节拍码非为全 00 值,则跳到 PLAYM 演奏
        LDI      TEMP,$00        ;若音符码及节拍码全为零(0000),则为乐曲结束标记
        OUT      TCCR1B,TEMP     ;将 TEMP=00 内含输出到 TCCR1B 内停止 TC1 计时及捕抓
        CLI                      ;令中断总开关 SREG 的 I 标志清零,而禁止中断
        SBI      PORTD,00        ;令 PINC=1 将喇叭输出 OFF
        RJMP     STARTP          ;循环演奏
PLAYM:  PUSH     ZH              ;进栈保存数据
        PUSH     ZL              ;进栈保存数据
        TST      PLYTON          ;检测 PLYTON 是否为 0
        BREQ     MUSTD           ;若为 0 则跳至 MUSTD 作节拍等待
```

```
    LDI  ZH,HIGH(MUSTAB*2)  ;乐曲码装入 Z 地址
    LDI  ZL,LOW(MUSTAB*2)   ;计时器值存放于 MUSTAB*2 起始地址
    MOV     TEMP,PLYTON     ;将乐曲码 PLYTON 装入 TEMP 寄存器内
    DEC TEMP                ;寄存器 TEMP 减 1
    LSL     TEMP            ;寄存器 TEMP 左移即 X2
    ADD     ZL,TEMP         ;将正确的计时器控制乐曲码的存表位移且使 TEMP 加入 ZL
    LDI     TEMP,$00        ;令 TEMP=00 以便让 ZH 与进行标志位 C 相加
    ADC     ZH,TEMP         ;将 ZH 与 TEMP=00 以及进行标志位 C 相加得到真正的 Z 地址值
    LPM                     ;将 Z 所指 PROM 的预存乐曲码计时长度低位值装入 R0
    MOV     TONL,R0         ;将乐曲码计时长度低位值 R0 装入 TONL 内
    OUT     TCNT1L,R0       ;将乐曲码计时长度低位值 R0 也装入 TCNT1L 内
    LD      R0,Z+           ;以 LD R0，Z＋指令使 Z 间接寻址加 1
    LPM                     ;将 Z 所指 PROM 之预存乐曲码之计时节拍码置于 R0
    MOV     TONH,R0         ;将节拍高位值 R0 装入 TONH 内
    OUT     TCNT1H,R0       ;  将节拍高位值 R0 装入 TCNT1H 内
    POP     ZL              ;出栈将 ZL,ZH 由堆栈指针依次取回
    POP     ZH              ;出栈
    LDS     TEMP,$0100      ;将 SRAM 地址 0100H 之内容装入 TEMP 内
    OUT     TCCR1B,TEMP     ;将原 0100H 的 SRAM 内容输出到 TCCR1B 控制 TC1
    SEI                     ;内容 02 令 TC1 为计时预除 8,令中断总开关 I=1 触发
MUSTD: RCALL    PLYDEL      ;调用延时子程序 0.2S
    DEC TONLNG              ;将节拍码 TONLNG 减 1
    BRNE    MUSTD           ;若节拍码 TONLNG 不为 0 则转回,再发音,为 0 则顺执
    RJMP    NEXMUT          ;继续到 NEXMUT 取乐曲码和节拍码
PLYDEL:LDI      TEMP,185    ;延时子程序,185x1mS=185mS,
                            ;即 PLYDEL=185mS 为十进制时间常数
DT3:    LDI     TEMP1,04    ;  送时间常数 4X250μS=1mS,DT3 约为 1mS
DT2:    LDI     TEMP2,250   ;250 为十进制时间常数,250X8X125nS=250mS,dt2=250μS
DT1:    WDR                 ;1T
        WDR                 ;2T
        WDR                 ;3T
        WDR                 ;4T
        WDR                 ;5T
        DEC TEMP2           ;6T,TEMP-1
        BRNE    DT1         ;8T,TEMP2 不为 0(则共执行 250X8XT=250μS)转,
                            ;为 0 按顺序执行
    DEC TEMP1               ;TEMP1-1
    BRNE    DT2             ;TEMP1  不为 0(则共执行 250X4US=1mS)转,为 0 按顺序执行
    DEC TEMP                ;TEMP-1
    BRNE    DT3             ;TEMP  不为 0(则共执行 185X1mS=185mS)转,为 0 按顺序执行
    RET                     ;子程序返回

    ;约定:因为计算机不能表示简谱乐曲,低音用数字后一点表示,
    ;高音用数字前一点表示,
```

;半音用#号,'为隔开音符,
;乐曲节拍应对照简谱查看,音长为节拍,
;一拍为 04,3/4 拍为 03,1/2(2/4)拍为 02,1/4 拍为 01,
;00 为表示休止符,
;00 00 连续两个字节为零,表示乐曲结束

;　　乐曲低八度音

曲码号	1	2	3	4	5	6	7	8	9	10	11	12
音符号	1	#1	2	#2	3	4	#4	5	#5	6	#6	7

;**********

;　　乐曲中音

曲码号	13	14	15	16	17	18	19	20	21	22	23	24
音符代	1	#1	2	#2	3	4	#4	5	#5	6	#6	7

;

;**********

;　　乐曲高八度音

曲码号	25	26	27	28	29	30	31	32	33	34	35	36
音符号	1	#1	2	#2	3	4	#4	5	#5	6	#6	7

;

;**********

```
.EQU    PLYTAB=0X0100           ;乐曲存放首地址
.EQU    MUSTAB=0X00A0           ;乐曲音符表存放首地址
.cseg                           ;实例
.org PLYTAB     ;"祝你生日快乐" 乐曲 1=C 3/4  乐曲存放起始地址,请查看对照简谱乐曲
;
```

例：生日快乐歌　1＝C 3/4

曲码

简码 | 5. 5　6　5 | i 7- | 5. 5　6　5 | 2　i- |

曲码

简码 | 5. 5　5　3 | i 7　6 | 4 · 4　3　i | 2　i-: ||

```
;
; 注意：曲码中不能用小数点,只能用逗号隔开,否则汇编时造成死机
    ; |   5             5,      6       5 |
    .DB 20,02,00,01,20,01,22,04,20,04;
    ; |  .1       7       - |
```

```
    .DB 25,04,24,04,00,04
    ;|    5                5,        6        5|
    .DB 20,02,00,01,20,01,22,04,20,04
    ;|   .2       .1        -|
    .DB 27,04,25,04,00,04
    ;|    5                5,     .5       .3|
    .DB 20,02,00,01,20,01,32,04,29,04
    ;|   .1        7        6|
    .DB 25,04,24,04,22,04
    ;|   .4                .4       .3       .1|
    .DB 30,02,00,01,30,01,29,04,25,04
    ;|   .2       .1        -  |
    .DB 27,04,25,04,00,04
    ;REAGAIN
    ;|    5                5,        6        5|
    .DB 20,02,00,01,20,01,22,04,20,04
    ;|.1          7        -  |
    .DB 25,04,24,04,00,04
    ;|    5                5,        6        5|
    .DB 20,02,00,01,20,01,22,04,20,04
    ;|   .2       .1        -  |
    .DB 27,04,25,04,00,04
    ;|    5                5,     .5       .3|
    .DB 20,02,00,01,20,01,32,04,29,04
    ;|   .1        7        6|
    .DB 25,04,24,04,22,04
    ;|   .4                .4       .3       .1|
    .DB 30,02,00,01,30,01,29,04,25,04
    ;|   .2       .1        -  |
    .DB 27,04,25,04,00,04
;               万水千山总是情
.db 17,04,18,04,20,06,20,02,22,04,20,04,17,12,15,04  ;
.db 13,06,17,02,15,04,13,04,10,12,10,04,8,8,13,04    ;注意:08 应写成 8 才能编译通过
.db 15,04,17,04,20,04,22,04,17,04,15,15,15,04,00,04  ;
.db 17,04,18,04,20,06,20,02,22,04,20,04,17,12,15,04  ;
.db 13,06,17,02,15,04,13,04,10,12,10,04,8,8,13,06    ;
.db 17,02,15,06,13,02,13,04,10,04,13,15,13,8,17,04   ;
.db 20,04,22,12,25,10,22,04,18,04,20,06,22,02,20,12  ;
.db 17,04,20,8,17,04,20,04,22,12,25,04,25,04,22,04   ;
.db 20,04,17,04,15,15,15, 8,17,04,18,04,20,06,20,02  ;
.db 22,04,20,04,17,12,15,04,13,06,17,02,15,04,13,04  ;
.db 10,12,10,04,8,8,13,04,17,04,15,06,13,02,10,04
.db 12,04,13,15,13,15    ;
    .DB 00,00        ;END
```

```
.cseg
.org MUSTAB              ;音符表地址标号
    ;约定:低音为数字后一点表示,高音为数字前一点表示,
          ;  半音为#号,'为隔开音符

    ;1      2     3      4     5      6     7      8     9
    ;1.   '#1.   '2.   '#2.   '3.    '4.   '#4.    '5.   '#5.

    ;10   11   12   13    14   15   16   17   18
    ;6.   '#6.  '7.  '1    '#1  '2   '#2   '3   '4
    .DW 63627,63730,63835,63927
    .DW 64020,64102,64185,64259
    .DW 64330,64399,64463,64523
    .DW 64579,64632,64683,64731
    .DW 64776,64819

    ;19   20   21   22   23   24   25    26    27
    ;#4   '5   '#5   '6   '#6   '7    ' 1    '#.1   '.2

    ;   28    29   30    31   32    33   34    35   36
    ;   '#.2  '.3  '.4   '#.4  '.5   '#.5  '.6   '#.6  '.7
    .DW 64859,64897
    .DW 64933,64967,64999,65029
    .DW 65057,65084,65109,65133
    .DW 65156,65177,65197,65216
    .DW 65243,65251,65267,65282
```

7.3.3 键盘扫描程序说明

源程序见 SLAVR73A(73B).ASM(**其中部分程序,键盘扫描程序,可供调用**)

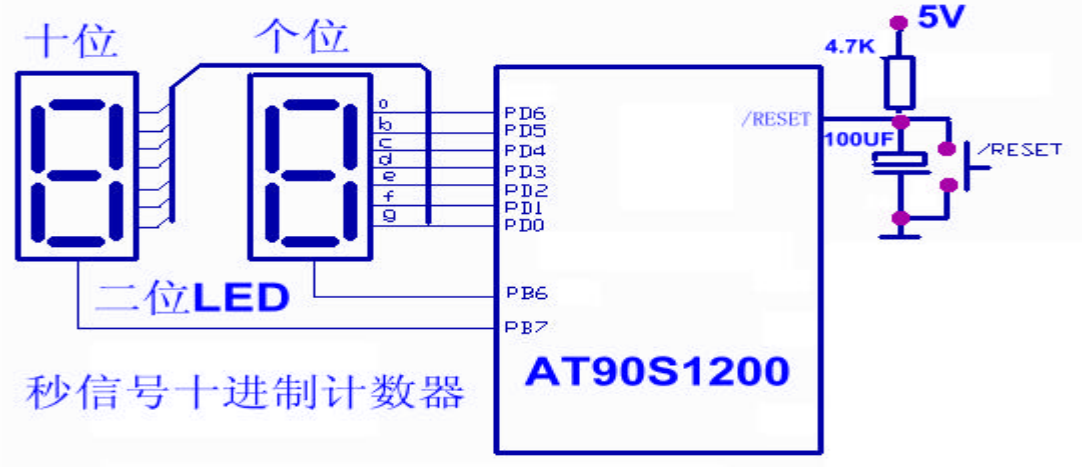
```
SCAN1:  PUSH XH          ;键扫显示子程序。
        PUSH XL          ;将 xl 压入堆栈
        PUSH TEMP3
        PUSH TEMP2
        PUSH TEMP1
        PUSH TEMP
        LDI XL,$60
            SET                     ;T 标志为 1 表示未按键
            LDI  SCNN,$00           ;按键起始扫描码 SCNN 为 00
            LDI  SCNDP,0B11011111   ;令 6 位七段 LED 扫描显示码初始为 11011111
            LDI  CNT,$06            ;七段 LED 共 6 位故 CNT=6 为位数计数
            LDI  KSNI,0B11110111    ;4*4 键盘扫描码 KSNI 初始为 11110111
      COL1: LDI  TEMP,$FF           ;PORTB 设定为输出
            OUT  DDRb,TEMP
```

```
    OUT   DDRC,TEMP       ;PORTC 设定为输出
OUT   PORTC,TEMP
    OUT   DDRd,TEMP       ;PORTD 设定为输出
OUT   PORTd,SCNDP         ;6 位七段 LED 扫描显示码输出到 PORTD
    LD    R1,X+      ;要显示于七段 LED 的间接寄存器 X 中的内容送入 R1 并令 X 加 1
    OUT   PORTb,R1            ;显示内容输出到 PORTB 以驱动 LED 显示
    RCALL   DELAY             ;调用延时以显示此位数一段时间
    MOV   TEMP,CNT            ;LED 位数为 6 而按键码行数为 4 故需作 CNT 值检测
    SUBI   TEMP,$03           ;CNT=TEMP 与 3 相减比较
    BRCS   NOSK               ;位数扫描 CNT 超过 3 则 C 为 1 跳到 NOSK 不作按键处理
    LDI   TEMP1,$04           ;一共要检查 4 个按键
    LDI   TEMP,0B00001111     ;设定 PC0-PC3 为输出 PC4-PC7 为输入
    OUT   DDRc,TEMP
    OUT   PORTc,KSNI          ;KSNI 输出到 PORTC 并令 PC7-PC4 为上拉电阻输入态
    RCALL   DELYT             ;调用延时以稳定读取键盘 I/O 输入端
    IN    TEMP,PINc           ;读取 C 口检测 PC7-PC4 看是否有按键低电位输入
    ANDI   TEMP,0B11110000    ;取 TEMP 的高 4 位
    SWAP   TEMP           ;键码顺序为 PC4-PC7 故将 TEMP 的高低 4 位互换成 D0-D3
  KROW: SEC           ;令 C 标志为 1 以便将键盘码 D0-D3 移到 C 标志位检测
ROR   TEMP        ;TEMP 的内容右移 1 位将第一个键码 D0=PC4 移到 C 标志位检测
    BRCS   NOKEY        ;若有键按下则测到 PC4=D0=0，若 C=1 无按键则转到 NOKEY
    CLT                 ;若 PC4=D0=CF=0 表示有按键令 T=0 表示有按键
    MOV    KEYN,SCNN    ;把按键扫描码 SCNN 送键码 KEYN 中保存
SBIS PINd,$07
ADIW KEYN,$10           ;判定 SHIFT 键是否按下，按下则键值加 10
 NOKEY: INC   SCNN      ;按键扫描码 SCNN 加 1
    DEC   TEMP1         ;扫描读取键数 TEMP1 减 1
    BRNE   KROW         ;每行有 4 个按键如 TEMP1 不为 0 则跳到 KROW 再检测 PC5-PC7
    SEC       ;此行 4 个键码检测完后令 C 为 1 以方便键盘扫描码 KSNI 内容的移位
    ROR   KSNI ;键盘扫描码 KSNI=CF=1>11110111 移位以进行下一行按键扫描
  NOSK: SEC             ;令进位标志 CF=1
    ROR   SCNDP         ;将扫描显示码 SCNDP 左移作下一位扫描
    DEC   CNT           ;共需作 6 位数扫描显示故 CNT 减 1
    BRNE   COL1         ;CNT 减 1 不为 0 则跳回 COL1 再作扫描显示及读取键盘输入
    LDI    TEMP,$FF     ;若已完成全部扫描显示和读取按键则令 TEMP=Off
    OUT    DDRC,TEMP    ;TEMP 输出到 DDRC 设定 PORTC 为输出驱动 LED
OUT    PORTC,TEMP
POP TEMP                ;出栈
POP TEMP1
POP TEMP2
POP TEMP3
POP XL
POP XH
RET                     ;子程序返回
```

7.3.4 十进制计数显示

源程序:SLAVR734.ASM
应用例子*.ASM,必须编译生成*.OBJ 文件才可调试,如要修改*.ASM,必须修改文件属性,去掉*.ASM 只读文件属性。
★ 必须按下图接线才能正常工作!

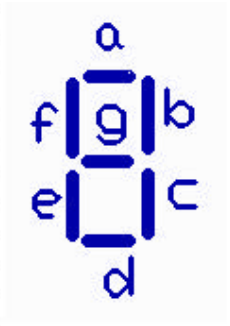
```
;*********************十进制计数程序 **********************************
;* 标题:用二位 LED 显示十进制计数
;* 版本:    1.0
;*最后更新日期: 2000.08.08
;* 支援 E-mail: gzsl@sl.com.cn
;* 描述
;* 用 AVR AT90S1200 的 D 口接二只 LED 数目管,PB7,PB6 作片选,硬件设定 D 口高电平 LED 灯
;* 亮,B 口低电平选中 LED,即选用共阴极数目管,最大显示十进制 99。硬件接线原理图如下:
;* 作者: SL.Z
;* 程序适用于所有单片机
;*************************************************************************
```




```
.include"1200def.inc"    ;AT90S1200 配置文件
.org $0000
    rjmp reset
.org $0010
  reset:ldi r20,$ff      ;设 B 口、D 口为输出
        out ddrb,r20     ;B 口方向寄存器
        out ddrd,r20     ;D 口方向寄存器
        sbi $18,7        ;硬件设定 B 口低电平选中 LED,PB.7 关高位 LED,
        sbi $18,6        ;PB.6 关低位 LED
        ldi r20,$fe      ;建字形表
    mov r0,r20           ;0
        ldi r20,$b0
    mov r1,r20           ;1
```

```
    ldi r20,$ed
mov r2,r20          ;2
    ldi r20,$f9
```

h	a	b	c	d	e	f	g	十六进制码	字形	R
1	1	1	1	1	1	1	0	FEH	0	R0
1	0	1	1	0	0	0	0	B0H	1	R1
1	1	1	0	1	1	0	1	EDH	2	R2
1	1	1	1	1	0	0	1	F9H	3	R3
1	0	1	1	1	0	1	1	B3H	4	R4
1	1	0	1	1	0	1	1	DBH	5	R5
1	1	0	1	1	1	1	1	DFH	6	R6
1	1	1	1	0	0	0	0	F0H	7	R7
1	1	1	1	1	1	1	1	FFH	8	R8
1	1	1	1	0	0	1	1	F3H	9	R9
1	1	1	1	0	1	1	1	F7H	A	R10
1	0	0	1	1	1	1	1	9FH	B	R11
1	1	0	0	1	1	1	0	CEH	C	R12
1	0	1	1	1	1	0	1	BDH	D	R13
1	1	0	0	1	1	1	1	CFH	E	R14
1	1	0	0	0	1	1	1	C7H	F	R15



硬件设定高电平

LED数目管点亮

```
mov r3,r20          ;3
    ldi r20,$b3
mov r4,r20          ;4
    ldi r20,$db
mov r5,r20          ;5
    ldi r20,$df
mov r6,r20          ;6
    ldi r20,$f0
mov r7,r20          ;7
    ldi r20,$ff
mov r8,r20          ;8
    ldi r20,$f3
mov r9,r20          ;9
    ldi r20,$f7
mov r10,r20         ;A
    ldi r20,$9f
mov r11,r20         ;B
    ldi r20,$ce
mov r12,r20         ;C
    ldi r20,$bd
mov r13,r20         ;D
    ldi r20,$cf
mov r14,r20         ;E
    ldi r20,$c7
mov r15,r20         ;F
bclr 7              ;清 I 标志,关中断
```

```
    clr r28             ;(28)=$00
main:   ldi r20,$28     ;扫描次数
start:  mov r30,r28     ; 十位显示字符值,第一次显示 0
display:andi r30,$f0    ; 取十位
    swap r30        ;半字节交换,获取 Z 地址
ledh:   ld r25,z    ;LED 高位,复位后(R31)=$00
        out portd,r25   ;送 D 口显示
        sbi $18,6       ;关个位,硬件设定高电平不亮
        cbi $18,7       ;选通十位,硬件设定低电平亮
        ldi r27,$10     ;延时常数
delay1: dec r26         ;-1,延时
    brne delay1         ;不为 0 转
    dec r27             ; -1,延时
    brne delay1         ; 不为 0 转
        sbi $18,7       ; 关十位
        nop
    mov r30,r28         ;个位显示字符值, 第一次显示 0
    andi r30,$0f    ;取个位
ledl:   ld r25,z    ;荻显示字符
        out portd,r25   ;送 D 口显示
        sbi $18,7       ; 关十位,硬件设定高电平不亮
        cbi $18,6       ; 选通个位,硬件设定低电平亮
        ldi r27,$10     ;延时常数
delay2: dec r26         ; -1,延时
        brne delay2     ; 不为 0 转
    dec r27             ; -1,延时
        brne delay2     ; 不为 0 转
        sbi $18,6       ; 关个位
    nop
        dec r20         ;显示数-1
        brne start      ;(R20)不为 0 转,
        inc r28         ;+1
        rjmp main       ;返回主程序
```

7.3.5 廉价的 A/D 转换器

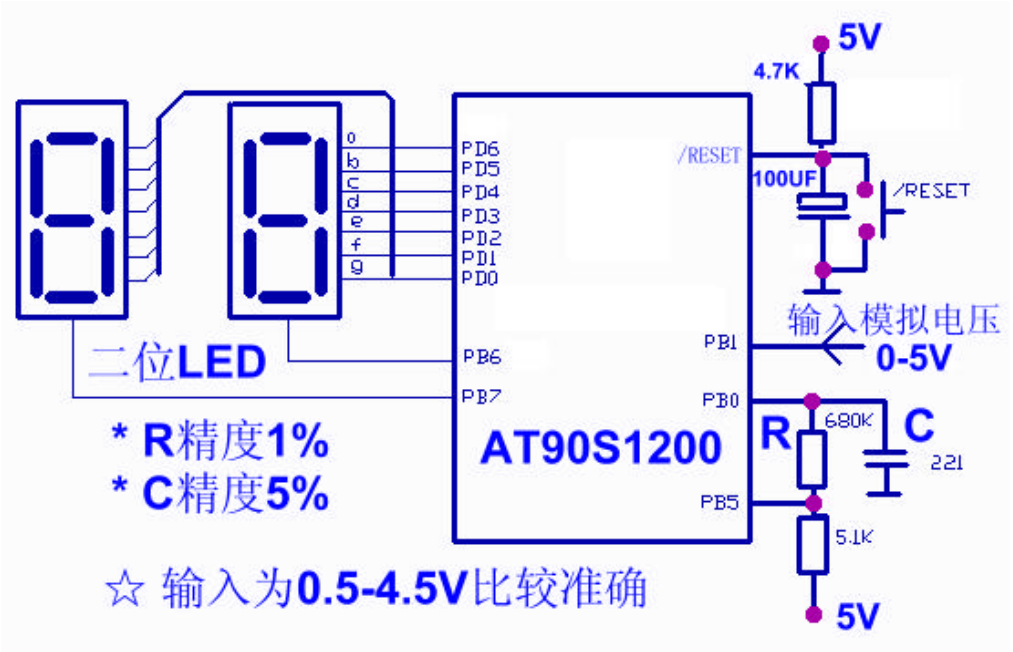
源程序:SLAVR735.ASM

★ 必须按下图接线才能正常工作!

AVR 单片机的 AT90 系列片内置有模拟比较器。这一节介绍用 AT90S1200 单片机实现廉价 A/D 转换器

一、硬件设计

使用 AVR 单片机及一个外部电阻和一个外部电容器设计成一个 A/D 转换器,并使用片内的定时器/计数器中断和模拟比较器中断。采用 RC 模拟转换原理。这程转方法在精确度和转换时间的花费上是较低的,适用一般要求不高的场合。硬件连接如图 6.1



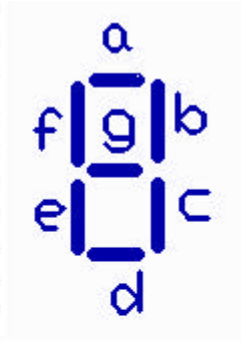
二、软件编程（源程序为:模拟比较 AD.ASM)

```
;********************AVR 单片机实用实验程序 ************************
;* 标题:     廉价的 A/D 转换器
;* 版本:     1.0
;* 最后更新日期:2000.08.08
;*
;* 支援 E-mail: gzsl@sl.com.cn
;* 描述
;*  用 AVR Studio 调试软件窗口观察指令执行变化情况
;* 作者: SL.Z
;* 程序适用于所有 AT90S 系列单片机
;*********************************************************************

;*********************************************************************
;* 硬件电路及说明阅<<廉价的 A/D 转换器>>一文
;* 本程序实测调试通过
;*********************************************************************
.include"1200def.inc"    ;应用器件配置文件,*.ASM 所在文件夹中不能缺,不然汇编出错提示
.org $0000
    rjmp reset       ;复位处理
.org $0002           ;EXT_INT0 外部中断入口地址
    rjmp inter           ;转 T0 溢出中断服务程序
.org $0003           ;TIMI_CAPT 定时器外部中断入口地址
    rjmp inter           ;转模拟比较器处理中断服务程序
.org $0010           ;主程序
reset:  ldi r20,$ff      ;设置 D 口为输出
    out ddrd,r20     ;送 D 口方向寄存器
```

```
sbi $18,7        ;置 I/O 寄存器 PORTB 的 7 位,LED 数目管十位片选
sbi $18,6            ;置 I/O 寄存器 PORTB 的 6 位,数目管个位片选
```

h	a	b	c	d	e	f	g	十六进制码	字形	R
1	1	1	1	1	1	1	0	FEH	0	R0
1	0	1	1	0	0	0	0	B0H	1	R1
1	1	1	0	1	1	0	1	EDH	2	R2
1	1	1	1	1	0	0	1	F9H	3	R3
1	0	1	1	1	0	1	1	B3H	4	R4
1	1	0	1	1	0	1	1	DBH	5	R5
1	1	0	1	1	1	1	1	DFH	6	R6
1	1	1	1	0	0	0	0	F0H	7	R7
1	1	1	1	1	1	1	1	FFH	8	R8
1	1	1	1	0	0	1	1	F3H	9	R9
1	1	1	1	0	1	1	1	F7H	A	R10
1	0	0	1	1	1	1	1	9FH	B	R11
1	1	0	0	1	1	1	0	CEH	C	R12
1	0	1	1	1	1	0	1	BDH	D	R13
1	1	0	0	1	1	1	1	CFH	E	R14
1	1	0	0	0	1	1	1	C7H	F	R15



硬件设定高电平

LED数目管点亮

```
ldi r20,$fe      ;R0-R15 存放显示字符.见表 6.1
mov r0,r20       ;0
ldi r20,$b0
mov r1,r20       ;1
ldi r20,$ed
mov r2,r20       ;2
ldi r20,$f9
mov r3,r20       ;3
ldi r20,$b3
mov r4,r20       ;4
ldi r20,$db
mov r5,r20       ;5
ldi r20,$df
mov r6,r20       ;67
ldi r20,$f0
mov r7,r20       ;7
ldi r20,$ff
mov r8,r20       ;8
ldi r20,$f3
mov r9,r20       ;9
ldi r20,$f7
mov r10,r20      ;A
ldi r20,$9f
mov r11,r20      ;B
ldi r20,$ce
mov r12,r20      ;C
ldi r20,$bd
```

```
    mov r13,r20      ;D
    ldi r20,$cf
    mov r14,r20      ;E
    ldi r20,$c7
    mov r15,r20      ;F

main:   rcall conini ;初始化 A/D 转换器
    sei              ;开中断
    ldi r16,$fc      ;置 B 口为输出,PB.0,PB.1 为输入,0B1111 1100
    out ddrb,r16
ddelay: clr r16          ;延时,清暂存计数器 1
        ldi r17,$f1  ;复位暂存计数器 2
 loop:  inc r16      ;清暂存计数器 1,加 1 计数
    brne loop            ;检查暂存计数器 1 不为 0 转,为 0 顺执
    inc r17          ;清暂存计数器 2,加 1 计数
    brne loop            ;检查暂存计数器 2 不为 0 转,为 0 顺执
    rcall adconv     ;调用启动转换器
wait:   brtc wait        ;等待中断,T 标志被清零转移,为 1 顺执
    clt                  ;清 T 标志
        rcall fetch      ;调用取显示
        ldi r20,$38      ;反复显示次数
start:  mov r30,r28          ;R28 送 Z 寄存器低位
display:andi r30,$f0     ;显示高位,与,即屏蔽高位,
    swap r30             ;交换半字节, 取得高位数据地址
ledh:   ld r25,z         ;取 LED 高位数据
    out portd,r25        ;高位显示送 D 口
    sbi $18,6            ;置位,关低位 LED 显示,
    cbi $18,7            ;清零,硬件设定低电平选中高位 LED
    ldi r27,$10      ;延时常数$10
delay1: dec r26          ;延时
    brne delay1
    dec r27
    brne delay1
    sbi $18,7            ;置位,关高位 LED
    nop
    mov r30,r28      ;显示低位
    andi r30,$0f
ledl:   ld r25,z
    out portd,r25
    sbi $18,7            ; 置位,关高位 LED 显示,
    cbi $18,6            ; 清零,硬件设定低电平 LED 亮,
    ldi r27,$10      ;延时常数$10
delay2: dec r26        ;延时
    brne delay2
```

```
    dec r27
    brne delay2
    sbi $18,6           ;关低位 LED
    nop
        dec r20     ;-1,
    brne start      ;不为 0 转,原(R20)=$38 即扫描显示 38 次
    rjmp main       ;返回主程序
inter:  in r28,tcnt0    ;中断服务程序
    clr r16         ;关闭 T0
    cli
    out tccr0,r16
    cbi portb,5
    set

    reti
conini: ldi r16,$0b         ;
    out acsr,r16
        ldi r16,$02
    out timsk,r16
        cbi portb,5
    ret

adconv: ldi r16,$40         ;
    out tcnt0,r16
    clt
        ldi r16,$02
    out tccr0,r16
    sbi portb,5
    ret

fetch:  mov r18,r28     ;取显示
        swap r18        ;半字节交换
        andi r18,$0f    ;与,屏蔽低位
        dec r18         ;-1
        dec r18
        dec r18
        dec r18
        brne l50        ;R18 不为 0 转
        mov r18,r28     ;
        andi r18,$0f    ;与,屏蔽低位
        rjmp fee
   l50: dec r18         ;
        brne l60
        mov r18,r28
```

```
        andi r18,$0f
        ori r18,$10
        rjmp fee
   l60: dec r18          ;
        brne l70
        mov r18,r28
        andi r18,$0f
        ori r18,$20
        rjmp fee
   l70: dec r18      ;-1
        brne l80     ;不为 0 转,为 0 顺执
        mov r18,r28
        andi r18,$0f;与
        ori r18,$30 ;或
        rjmp fee
   l80: ldi r28,$ff      ;(R28)=$FF
        ret

   fee: cbi eecr,0       ;清 I/O 寄存器 EEPROM 控制寄存器 EECR 的 0 位,设为读操作
        out eear,r18;R18 送 EEPROM 地址口,输出地址
        sbi eecr,0       ;置位 I/O 寄存器 EEPROM 控制寄存器的 0 位,读选通
        in r28,eedr ;获得数据,EEPROM 数据寄存器内容送 R28
        ret

.eseg
.org $0000           ; EEPROM 数据地址首址
.db 0x00,0x00,0x03,0x14,0x21,0x25,0x2f,0x33
.db 0x38,0x3f,0x47,0x52,0x59,0x6a,0x78,0x7d
.db 0x81,0x86,0x8b,0x90,0x9a,0x9f,0xa4,0xa7
.db 0xac,0xaf,0xbe,0xc5,0xc9,0xca,0xce,0xd0
.db 0xd3,0xd5,0xd7,0xd9,0xda,0xdf,0xe0,0xe1
.db 0xe2,0xe3,0xe4,0xe5,0xe6,0xe8,0xe9,0xec
.db 0xed,0xee,0xef,0xf0,0xf3,0xf4,0xf5,0xf6
.db 0xf8,0xf9,0xfa,0xfc,0xff,0xff,0xff,0xff
```

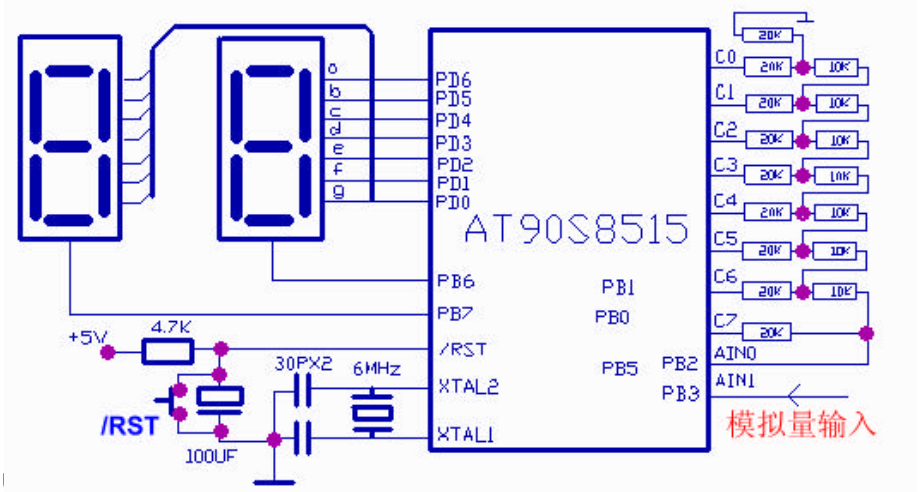
7.3.6 高精度廉价的 A/D 转换器

--- 用网络电阻组成的高精度 A/D 转换器

源程序:SLAVR736.ASM

★ 必须按下图接线才能正常工作!

```
;*****************************AVR 单片机实用程序 ************************
;* 标题:     高精度廉位的 A/D 转换器
;* 版本:     1.0
;* 最后更新日期:2000.08.08
;* 支援 E-mail: gzsl@sl.com.cn
;* 描述
;*  用网络电阻实现高精度廉位的 A/D 转换,本程序实测调试通过
;* 作者: SL.Z
;* 程序适用于所有单片机
```




```
.incl
.org $0000
    rjmp reset
.def temp=r16
.def temp1=r17
.equ label=$0100     ;字型表首址
.org $0010
reset:  ldi r20,$02      ;设堆栈指针
    out sph,r20
    out spl,r20
    ldi r20,$ff      ;设置 D 口、C 口为输出
    out ddrd,r20
    out ddrc,r20
    ldi r20,$f0          ;设 PB7~PB4 为输出,PB3~PB0 为输入
    out ddrb,r20
    out portb,r20
    clr r20              ;清 C 口
    out portc,r20
```

```
    sbi $18,7       ;关显示,硬件设定片选 LED 数目管低电平亮,高电平关
    sbi $18,6
    cli                 ;关中断
    ldi zh,high(label*2);
main:   ldi temp,$00
    nop
loop1:  out portc,temp
    nop
    nop
    nop
    in temp1,acsr       ;读模拟比较器控制和状态寄存器
    sbrs temp1,5
    rjmp naco       ;模拟比较器输出为 0 转
    rjmp haco       ;模拟比较器输出为 1 转
naco:   inc temp;+1
    brne loop1      ;不为 0 转
    ldi temp,$ff
haco:   mov r28,temp;暂存
    ldi r20,$38
display:mov temp,r28     ;显示十位字型
    andi temp,$f0
    swap temp
    clr zl
    add zl,temp
ledh:   lpm             ;取十位字型
    out portd,r0
    sbi $18,6           ;关 PB.6 硬件设定片选 LED 数目管高电平关，低电平开(灯亮)
    cbi $18,7           ;开 PB.7
    rcall delay
    mov temp,r28     ;显示个位
    andi temp,$0f
    clr zl
ledl:   add zl,temp
    lpm             ;取个位字型
    out portd,r0
    sbi $18,7       ;关 PB.7 硬件设定片选 LED 数目管高电平关，低电平开(灯亮)
    cbi $18,6       ;开 PB.6
    rcall delay     ;调用延时
        dec r20
    brne display
    rjmp main
delay:  ldi r27,$10 ;延时子程序
delay1: dec r26
    brne delay1
```

```
    dec r27
    brne delay1
    sbi $18,7          ;关 PB.7
    ret                ;子程序返回
```

h	a	b	c	d	e	f	g	十六进制码	字形
1	1	1	1	1	1	1	0	FEH	0
1	0	1	1	0	0	0	0	B0H	1
1	1	1	0	1	1	0	1	EDH	2
1	1	1	1	1	0	0	1	F9H	3
1	0	1	1	1	0	1	1	B3H	4
1	1	0	1	1	0	1	1	DBH	5
1	1	0	1	1	1	1	1	DFH	6
1	1	1	1	0	0	0	0	F0H	7
1	1	1	1	1	1	1	1	FFH	8
1	1	1	1	0	0	1	1	F3H	9
1	1	1	1	0	1	1	1	F7H	A
1	0	0	1	1	1	1	1	9FH	B
1	1	0	0	1	1	1	0	CEH	C
1	0	1	1	1	1	0	1	BDH	D
1	1	0	0	1	1	1	1	CFH	E
1	1	0	0	0	1	1	1	C7H	F

硬件设定高电平

LED数目管点亮

```
.cseg
.org $0100          ;字型表首址
.dw 0xb0fe,0xf9ed,0xdbb3,0xf0df
.dw 0xf3ff,0x9ff7,0xbdce,0xc7cf
```

7.3.7　星星灯

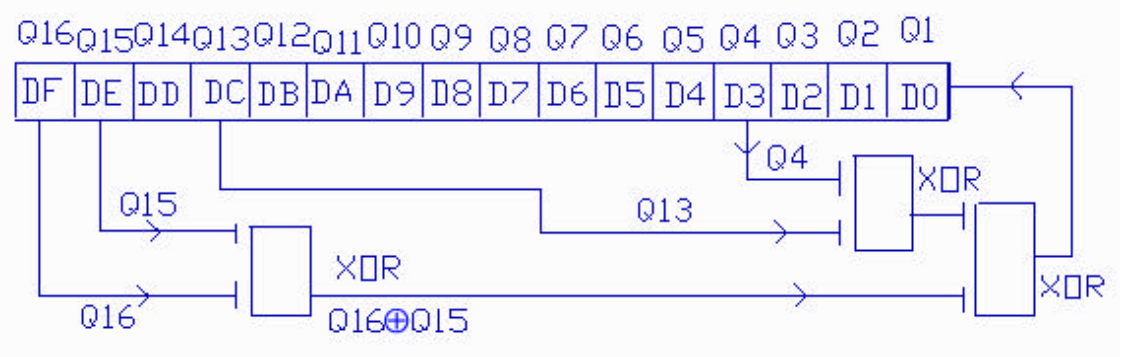
源程序:SLAVR737.ASM

用 AVR 单片机 8 位数据产生随机数,由 PORTA 口及 PORTC 口输出随机数,在 8X8 LED 上显示,硬件接线电路见“7.3.8 按钮猜数”。随机数的种子由程序设定(也可外接开关设定),启动种子后,由移位寄存器以互斥的异或逻辑组合返回循环产生。

```
.include"8515def.inc"
    rjmpRESET
.def    temp    =r16    ;暂存器
.deftemp1   =r17    ;暂存器 1
.defudata   =r21    ;存随机数送 A 口
.defddata   =r22    ;存随机数送 C 口
.cseg
.org    0x10
RESET:  ldi temp,high(RAMEND);设堆栈指针
    out SPH,temp
    ldi temp,low(RAMEND)
    out SPL,temp
```

```
        ldi     temp,0xff       ;设 A 口、C 口为输出
        out     ddra,temp       ;送方向寄存器 A
        out     ddrc,temp       ;送方向寄存器 C
start:  wdr                     ;关看门狗
        ldi     udata,0x6a      ; 设置随机数初值
        ldi     ddata,0x3c      ;
startp: out     porta,udata     ;输出到 A 口
        out     portc,ddata     ;输出到 C 口
        ldi     temp,0x80       ;设延时常数
        rcall   delay           ;调用延时子程序
        rcall   randm           ;调用十六位随机数子程序
        rjmp    startp
delay:                  ; 通用延时子程序从略
....
```

16 位移位产生随机数原理图



8--16 位移位寄存器产生随机数循环组合

位 数	循环输入组合 S=2^n-1 Qn XOR Qm
8	Q2 ⊕ Q3 ⊕ Q4 ⊕ Q8 (现程序按钮猜数采用 8 位数)
9	Q5 ⊕ Q9
10	Q7 ⊕ Q10
11	Q9 ⊕ Q11
12	Q2 ⊕ Q10 ⊕ Q11 ⊕ Q12
13	Q1 ⊕ Q11 ⊕ Q12 ⊕ Q13
14	Q2 ⊕ Q12 ⊕ Q13 ⊕ Q14
15	Q14 ⊕ Q15
16	Q4 ⊕ Q13 ⊕ Q15 ⊕ Q16

```
randm:                          ;产生十六位随机数子程序
          mov   temp,udata      ;产生 A 口随机数
          mov   temp1,udata     ;
          rol   temp            ;通过进位位左循环移位
          eor   temp1,temp      ;异或
```

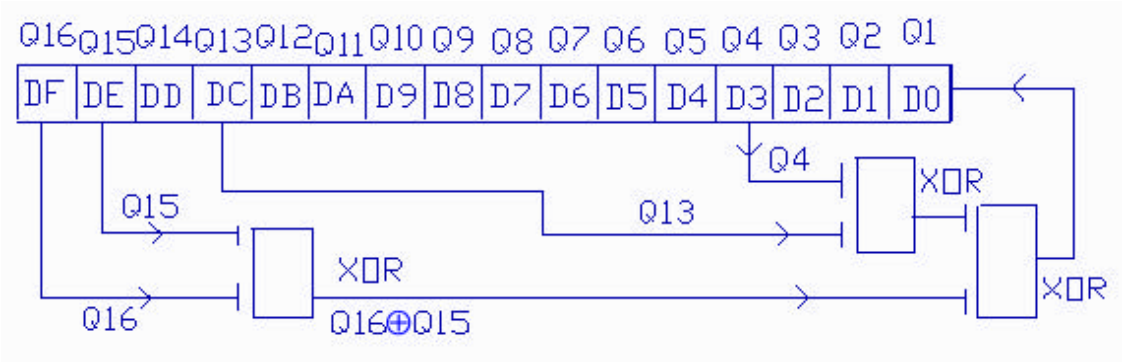
```
rol   temp        ; 通过进位位左循环移位
rol   temp        ; 通过进位位左循环移位
eor   temp1,temp  ;异或
mov   temp,ddata  ; 产生 C 口随机数
swap  temp        ; 通过进位位左循环移位
eor   temp,temp1  ;异或通过进位位左循环移位
rol   temp        ; 通过进位位左循环移位
rol   ddata       ; 通过进位位左循环移位
rol   udata       ; 通过进位位左循环移位
ret               ;子程序返回
```

7.3.8　按钮猜数程序

源程序:SLAVR738.ASM

许多场合如按钮猜数(电脑摇奖,电脑选出幸运号),游戏开始按钮等待一个不规则且不定序的数据产生,即须要随机数发生器。随机数的种子由程序设定(也可外接开关设定),启动种子后,由移位寄存器以互斥的异或逻辑组合返回循环产生。产生随机数的原理图如下:

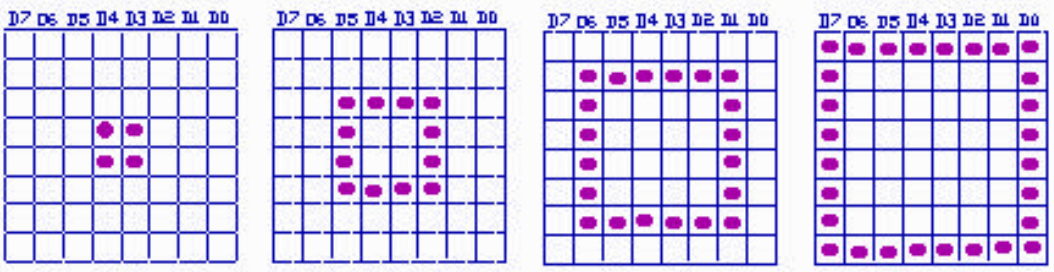
16 位移位产生随机数原理图

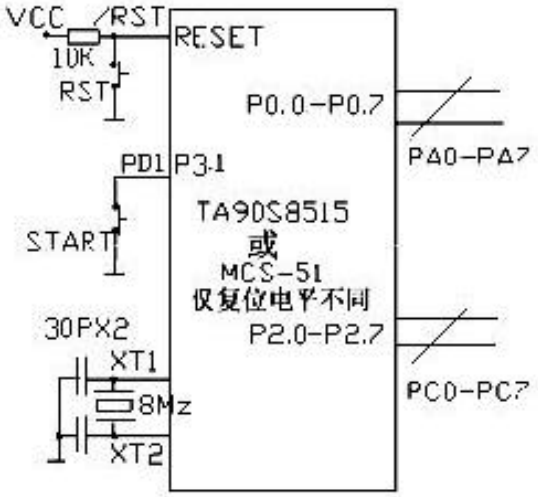


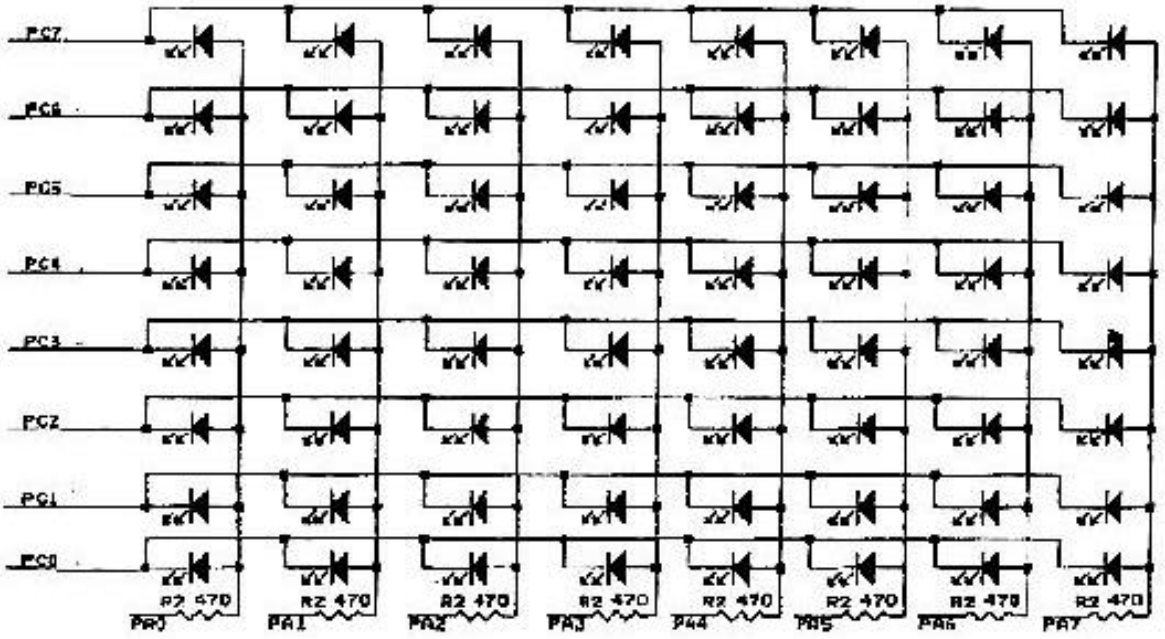
8—16 位移位寄存器产生随机数循环组合

位 数	循环输入组合 S=2^n-1 Qn XOR Qm
8	Q2 ⊕ Q3 ⊕ Q4 ⊕ Q8 (现程序按钮猜数采用 8 位数)
9	Q5 ⊕ Q9
10	Q7 ⊕ Q10
11	Q9 ⊕ Q11
12	Q2 ⊕ Q10 ⊕ Q11 ⊕ Q12
13	Q1 ⊕ Q11 ⊕ Q12 ⊕ Q13
14	Q2 ⊕ Q12 ⊕ Q13 ⊕ Q14
15	Q14 ⊕ Q15
16	Q4 ⊕ Q13 ⊕ Q15 ⊕ Q16

以 8X8 LED 阵列,开机时为了避免被使用者预测出压按时间对应随机数的变化值,故 LED 字幕以广告动画画面显示,并令随机数随着变化使无法预测随机数起始值,广告动画面共有四张,每张有 8 位数据。见”org　dpfstb”;







AVR 直接驱动 8X8 LED 按钮猜数电路及接线配置图

由按钮(PD1)按下,AVR 用 8 位数据产生随机数,由 PORTA 口及 PORTC 口输出随机数,在 8X8 LED 上显示好玩的真实的按钮猜数。

```
.include "8515def.inc"
.def    peed    =r16
.def    dspn    =r17    ;存显示初始动画次数
.def    temp2   =r18
```

```
.def    temp1   =r19
.def    temp    =r20
.def    scndp   =r21
.def    cnt     =r22
.def    rdata   =r23     ;存随机种子数
.def    rdata9  =r24
.equ    dpfstb  =0x01e0      ;大小矩形图表首址
.equ    randtb  =0x0210      ;随机数种子表首址
.equ    numbertb=0x0240      ;0-9 数字表首址
.org  $0000
    rjmpRESET        ;Reset Handle
.cseg
.org $0010
RESET:  ldi     peed,high(RAMEND)    ;设置堆栈$25F,见器件配置文件"8515def.inc"
        out     SPH,peed
        ldi     peed,low(RAMEND)
        out     SPL,peed
        ldi     peed,0xff     ;对口初始化,
        out     ddra,peed     ;设 A 口为输出
        out     ddrc,peed     ;设 C 口为输出
        ldi     peed,0xfd     ;PD1 作输入,且接内部上拉电阻
        out     ddrd,peed     ;PD1 为输入,其余为输出
        ldi     peed,0xff     ;关 D 口
        out     portd,peed
        ldi     peed,0x13     ;显示画面次数
start:  ldi     dspn,0x06     ;显示初始动画
        ldi     zh,high(dpfstb*2)
        ldi     zl,low(dpfstb*2)
dspfm:  rcall   ldtb8         ;调用程序区数送到内存 RAM
        ldi     temp2,0xa0    ;显示动画面次数
dspfm1: rcall   scan1         ;调用从内存取数显示一次
        sbis    pind,01       ;I/O 口的位被置位跳行,检测到 PD1 按下否
        rjmp    getseed       ; 检测到 PD1 按下转
        dec     temp2         ;-1
        brne    dspfm1        ;不为 0 转
        dec     dspn     ; 初始画面次数-1
        brne    dspfm         ; 不为 0 转
        rjmp    start         ;转到显示初始动画
getseed:inc     temp          ;+`1,根据 PD1 按下的时间,选择随机数种子
        sbis    pind,01       ; I/O 口的位被置位跳行,检测到 PD1 按下否
        rjmp    getseed       ; 检测到 PD1 按下,继续计数
        andi    temp,0x1f     ;按钮松开,取随机数种子与 0X0F 加
        ldi     zh,high(randtb*2)
        ldi     zl,low(randtb*2)
```

```
        add     zl,temp
        lpm
        mov     rdata,r0     ;得到随机数种子
next:   ldi     dspn,0x08    ;显示 8 个不同的随机数;
repeet: rcall   randm        ;调用产生随机数子程序
        rcall   dspnumber    ;调用显示 8 个不同的随机数
        dec     dspn     ;-1
        brne    repeet       ;dspn 不为 0 转
        rcall   randm        ; 调用产生随机数子程序
guess1: rcall   dspnumber    ;调用显示同一随机数,直到有键按下
        sbic    pind,01      ;松开后再往下执行(I/O 口清零跳行)
        rjmp    guess1       ;转显示同一随机数,直到有键按下
wait:   rcall   dspnumber    ;
        sbis    pind,01
        rjmp    wait         ;等待按钮按下
        ldi     rdata9,0x03      ;显示动画三次
start0: ldi     dspn,0x06        ;每次显示六幅画面
        ldi     zh,high(dpfstb*2)
        ldi     zl,low(dpfstb*2)
dspfm0: rcall   ldtb8            ;调用从 Z 指向的程序区取数据送到内存 0080-0087 中
        ldi     temp2,0xa0       ;显示次数
dspfm1a:rcall   scan1            ;调用从内存 0080-0087 中取数据显示一次
        dec     temp2            ;-1
        brne    dspfm1a          ;不为 0 转
        dec     dspn         ;显示初始动画次数-1
        brne    dspfm0           ;不为 0 转
        dec     rdata9           ;显示动画三次-1
        brne    start0           ;不为 0 转
        rjmp    next              ;转显示 8 个不同的随机数
dspnumber:                        ;显示一个 0-9 数字的子程序
        ldi     zh,high(numbertb*2)
        ldi     zl,low(numbertb*2)
        add     zl,rdata9
        rcall   ldtb8            ;取数
        ldi     temp2,0xa0    ;该数字重复显示 A0H 次
dspn1:  rcall   scan1
        dec     temp2
        brne    dspn1
        ret
scan1:  push    xl                ;从内存 0080-0087 中取数据显示一次
        ldi     temp,0b01111111
        mov     scndp,temp
        ldi     cnt,0x08
col1:   out     portc,scndp      ;显示屏幕的一列
```

```
        ld      r1,x+
        out     porta,r1
        rcall   delay
        sec
        ror     scndp
        dec     cnt
        brne    col1
        pop     xl
        ret
ldtb8:  ldi     xl,0x80      ;从 Z 指向的程序区取数据送到内存 0080-0087 中
        ldi     xh,0x00
        ldi     temp1,0x08
        push    xl
nexld1: lpm
        st      x+,r0
        ld      r0,z+
        dec     temp1
        brne    nexld1
        pop     xl
        ret
delay:           ;通用延时子程序从略
....
randm:  mov     temp,rdata     ;产生 8N(0≤N≤9)随机数子程序
        mov     temp1,rdata
        swap    temp1
        eor     temp,temp1
        rol     temp1
        eor     temp,temp1
        rol     temp1
        eor     temp,temp1
        rol     temp
        rol     rdata
        mov     rdata9,rdata
        andi    rdata9,0x0f
        cpi     rdata9,0x0a
        brsh    randm          ;产生了一个 0≤RDATA9≤9 的随机数
        lsl     rdata9
        lsl     rdata9
        lsl     rdata9
        ret

.cseg
.org      dpfstb;            ;大小方框字形表
;small  o
```

```
.db     0b00000000,0b00000000,0b00000000,0b00011000
.db     0b00011000,0b00000000,0b00000000,0b00000000
.db     0b00000000,0b00000000,0b00111100,0b00100100
.db     0b00100100,0b00111100,0b00000000,0b00000000
.db     0b00000000,0b01111110,0b01000010,0b01000010
.db     0b01000010,0b01000010,0b01111110,0b00000000
;big   o
.db     0b11111111,0b10000001,0b10000001,0b10000001
.db     0b10000001,0b10000001,0b10000001,0b11111111
.db     0b00000000,0b01111110,0b01000010,0b01000010
.db     0b01000010,0b01000010,0b01111110,0b00000000
.db     0b00000000,0b00000000,0b00111100,0b00100100
.db     0b00100100,0b00111100,0b00000000,0b00000000
.cseg
.org    randtb                  ;随机数种子表
.db     0x5a,0x7b,0x5b,0x4f,0x66,0x6d,0x7d,0x07
.db     0x3b,0x8c,0x67,0x9a,0x99,0x7e,0x2d,0x3e
.db     0x5c,0x6d,0x5b,0x7e,0xf6,0xe7,0x4c,0xc8
.db     0x69,0x9c,0xe2,0x75,0x6c,0xd3,0xe8,0x9a
.cseg
.org    numbertb                ;0-9 数字字形表
;0
.db     0b00111000,0b01000100,0b01000100,0b01000100
.db     0b01000100,0b01000100,0b01000100,0b00111000
;1
.db     0b00010000,0b00011000,0b00010000,0b00010000
.db     0b00010000,0b00010000,0b00010000,0b00111000
;2
.db     0b00011100,0b00100010,0b00100000,0b00010000
.db     0b00001000,0b00000100,0b00000010,0b00111110
;3
.db     0b00111100,0b00010000,0b00001000,0b00010000
.db     0b00100000,0b00100000,0b00100010,0b00011100
;4
.db     0b00100000,0b00110000,0b00101000,0b00100100
.db     0b00100010,0b11111110,0b00100000,0b00100000
;5
.db     0b01111110,0b00000010,0b00111110,0b01000000
.db     0b01000000,0b01000000,0b01000010,0b00111100
;6
.db     0b00110000,0b00001000,0b00000100,0b00111100
.db     0b01000100,0b01000100,0b01000100,0b00111000
;7
.db     0b01111100,0b01000000,0b00100000,0b00010000
```

```
.db     0b00001000,0b00001000,0b00001000,0b00001000
;8
.db     0b00111000,0b01000100,0b01000100,0b00111000
.db     0b01000100,0b01000100,0b01000100,0b00111000
;9
.db     0b00111000,0b01000100,0b01000100,0b01111000
.db     0b01000000,0b01000000,0b01000100,0b00111000
```

7.3.9 汉字的输入

源程序:SLAVR739.ASM

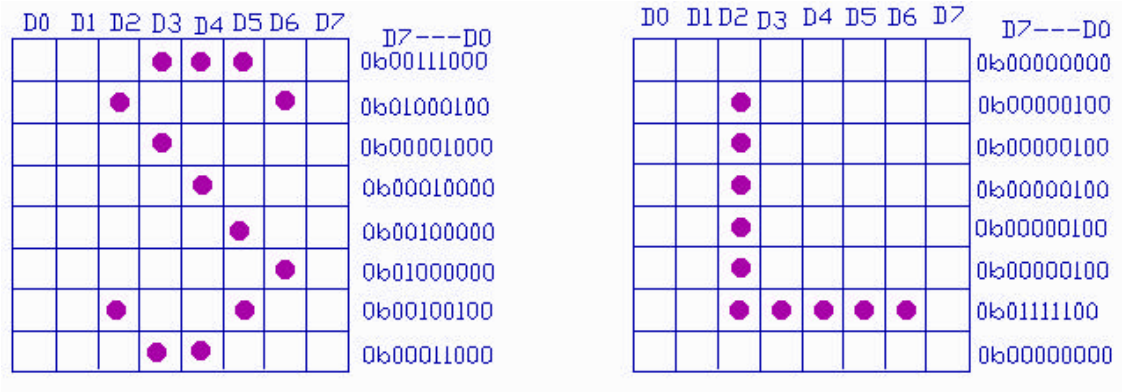
硬件电路见“7.3.8 按钮猜数程序”

```
;**********************AVR 单片机实验测试程序 **********************
;* 标题:    汉字的输入
;* 版本:          1.0
;* 最后更新日期:    2000.08.08
;* 支援  E-mail: gzsl@sl.com.cn
;* 描述
;*   用 AVR Studio 调试软件窗口观察指令执行变化情况
;* 作者: SL.Z
;* 程序适用于所有单片机
;*****************************************************************

.include "8515def.inc"
.def      dspn      =r23
.def      temp2     =r24
.def      temp1     =r17
.def      temp      =r18
.def      scndp     =r19
.def      cnt       =r20
.equ      dpfstb    =0x01e0
.org    $0000
   rjmp RESET          ;Reset Handle
.org $0010
RESET:
   ldi    r16,high(RAMEND)      ;设堆栈为$025F
   out    SPH,r16
   ldi    r16,low(RAMEND)
   out    SPL,r16
          ldi       r16,0xff     ;设 A 口、C 口为输出
          out       ddra,r16     ;A 口方向寄存器
          out       ddrc,r16     ;C 口方向寄存器
```

```
dspfst:  ldi     dspn,0x07              ;显示次数
         ldi     zh,high(dpfstb*2)      ;高位取数
         ldi     zl,low(dpfstb*2)       ;低位取数
dspfm:   rcall   ldtb8                  ;调用取字形子程序
         ldi     temp2,0xa0;循环次数
dspfm1:  rcall   scan1
         dec     temp2
         brne    dspfm1
         dec     dspn
         brne    dspfm
         rjmp    dspfst
scan1:   push    xl                     ;XL 进栈
         ldi     temp,0b01111111;  第一次选中 PC.7,硬件设定低电平 LED 亮
         mov     scndp,temp
         ldi     cnt,0x08               ;取一个字符需 8 次取数
col1:    out     portc,scndp  ;选通数据送 C 口,第一次选中 PC.7
         ld      r1,x+        ;取数后地址指针加 1
         out     porta,r1     ;数据送 A 口
         ldi     r16,0x10     ;送延时常数
         rcall   delay                  ;调用延时
         sec                            ;置位进位位
         ror     scndp        ;通过进位位右循环
         dec     cnt          ;-1
         brne    col1                   ;不为 0 转
         pop     xl           ;为 0,XL 出栈
         ret                            ;子程序返回
ldtb8:   ldi     xl,0x80      ;取数(字形)子程序
         ldi     xh,0x00      ;
         ldi     temp1,0x08   ;取数(字符)次数
         push    xl                     ;XL 进栈
nexld1:  lpm                  ;从程序存储器取数,将 Z 寄存器指向的一个字节装入 R0
         st      x+,r0                  ;X 寄存器内容(字形)送 R0,后 X 指针加 1
         ld      r0,z+                  ; Z 寄存器内容(字形)送 R0,后 Z 指针加 1
         dec     temp1        ;-1
         brne    nexld1       ;未完继续取数
         pop     xl           ;XL 出栈
         ret                            ;子程序返回
delay:                        ;通用延时子程序从略
.org     dpfstb               ;
```

;在 8X8 的方格中填字,硬件设定高电平为 1 点亮 LED,注意:编码的高、低位与习惯编码书写方法正好相反,其目的为汉字、字符书写符合人们习惯(正写)!




```
;字符 S
.db     0b00111000,0b01000100,0b00001000,0b00010000
.db     0b00100000,0b01000000,0b00100100,0b00011000
;字符 L
.db     0b00000000,0b00000100,0b00000100,0b00000100
.db     0b00000100,0b00000100,0b01111100,0b00000000
;字符“双龙电子”字形表略
```

7.4.1　10 位 AD/转换

源程序:SLAVR741.ASM

AVR 单片机中 AT90S8535 是功能较强的单片机.有如下特点:

1. AVR RISC 结构
2. AVR—高性能、低功耗 RISC 结构
   —118 条指令——大多数为单指令周期
   —32 个 8 位通用（工作）寄存器
   —工作在 8MHz 时具有 8MIPS 的性能
3. 数据和非易失性程序内存
   —4K/8K 字节的在线可编程 FLASH（擦除次数：1000 次）
   —256/512 字节 SRAM
   —256/512 字节在线可编程 EEPROM（寿命：100000 次）
   —程序加密位
4. 外围（Peripheral）特点
   —两个具有比较模式的可预分频（Prescale）8 位定时器/计数器
   —一个可预分频、具有比较、捕捉和两个 8/9/10 位 PWM 功能的 16 位定时器/计数器
   —片内模拟比较器
   —可编程的看门狗定时器（由片内振荡器生成）
   —8 通道 10 位 ADC
   —全双工 UART
5. 特别的 MCU 特点
   —上电复位电路
   —具有记数功能、有独立振荡器的实时时钟（RTC）
   —低功耗空闲、省电和掉电模式
   —内外部中断源