STM8与汇编语言（ 17）——蜂鸣器

蜂鸣器是现在单片机应用系统中很常见的，常用于实现报警功能。为此 STM8 特别集成了蜂鸣器模块，应用起来非常方便。

在应用蜂鸣器模块时，首先要打开片内的低速 RC振荡器（应该也能使用外部的高速时钟，不过本人没实验过），其频率为 128KHZ。然后通过设置蜂鸣器控制寄存器 BEEP_CSR中的 BEEPDIV[4:0] 来获取 8KHZ的时钟，再通过 BEEPSEL 最终产生 1KHZ或 2KHZ或 4KHZ的蜂鸣器时钟，最后使能该寄存器中的 BEEPEN 位，产生蜂鸣器的输出。

下面的实验程序首先初始化低速振荡器，然后启动蜂鸣器，再延时 2.5 秒，然后关闭蜂鸣器。

同样还是利用 ST的开发工具，生成一个汇编程序的框架，然后修改其中的 main.asm，修改后的代码如下。

stm8/

```
    #include "mapping.inc"
    #include "STM8S207C_S.INC"

; 定义堆栈空间的起始位置和结束位置
stack_start.w  EQU    $stack_segment_start
stack_end.w    EQU    $stack_segment_end



        segment 'rom'          ;                下面开始定义一个段，该段位于 ROM中
main.l                         ;                定义复位后的第一条指令的标号 （即入口地址）
;
; 首先要初始化堆栈指针
    LDW     X,#stack_end
    LDW     SP,X


    LD      A,CLK_ICKR
    OR      A,#$08
    LD      CLK_ICKR,A         ;                打开芯片内部的低速振荡器 LSI
WAIT_LSI_READY.L
```

```
        LD      A,CLK_ICKR
        AND     A,#$10
        JREQ    WAIT_LSI_READY    ;           等待振荡器稳定

        LD      A,#$2e            ; BEEPDIV[1:0] = 00
                                  ; BEEPDIV[4:0] = 0e
                                  ; BEEPEN       = 1
;   输出频率  = Fls / ( 8 * (BEEPDIV + 2) )= 128K / (8 * 16) = 1K
        LD      BEEP_CSR,A        ;           打开蜂鸣器

        LD      A,#10             ;           延时 250MS*10
DELAY_1.L
        PUSH    A
        LD      A,#250            ;           延时 250MS
        CALL    DELAY_MS
        POP     A
        DEC     A
        JRNE    DELAY_1

        LD      A,#$1E            ;           关闭蜂鸣器
        LD      BEEP_CSR,A

MAIN_LOOP.L
        JRA     MAIN_LOOP

;   函数功能：延时
;   输入参数：寄存器   A  －－  要延时的毫秒数，这里假设    CPU的主频为  2MHZ
;   输出参数：无
;   返  回  值：无
;   备        注：无
DELAY_MS.L
        PUSH    A               ;           将入口参数保存到堆栈中
            LD      A,#250          ;           寄存器  A<-250, 作为下面的循环数
DELAY_MS_1.L
        NOP                     ;           用空操作指令进行延时    4T
            NOP
            NOP
            NOP
```

```
        NOP
        DEC    A              ;              寄存器 A<-A-1，本条指令执行之间为  1
T
        JRNE   DELAY_MS_1     ;     若不等于 0，则循环,
                              ;              本条指令执行时间为  2T（跳时）或 1T
（不跳时）
  POP    A              ;              从堆栈中恢复入口参数
        DEC    A              ;              将要延时的 MS数－1
        JRNE   DELAY_MS       ;       若不等于 0，则循环
        RET                   ;              函数返回

  interrupt NonHandledInterrupt
NonHandledInterrupt.l
        iret

;   下面定义中断向量表
       segment 'vectit'
       dc.l {$82000000+main}                          ; reset
            dc.l {$82000000+NonHandledInterrupt}          ; trap
       dc.l {$82000000+NonHandledInterrupt}          ; irq0
       dc.l {$82000000+NonHandledInterrupt}          ; irq1
       dc.l {$82000000+NonHandledInterrupt}          ; irq2
       dc.l {$82000000+NonHandledInterrupt}          ; irq3
       dc.l {$82000000+NonHandledInterrupt}          ; irq4
          dc.l {$82000000+NonHandledInterrupt}          ; irq5
       dc.l {$82000000+NonHandledInterrupt}          ; irq6
       dc.l {$82000000+NonHandledInterrupt}          ; irq7
       dc.l {$82000000+NonHandledInterrupt}          ; irq8
       dc.l {$82000000+NonHandledInterrupt}          ; irq9
       dc.l {$82000000+NonHandledInterrupt}          ; irq10
       dc.l {$82000000+NonHandledInterrupt}          ; irq11
       dc.l {$82000000+NonHandledInterrupt}          ; irq12
       dc.l {$82000000+NonHandledInterrupt}          ; irq13
       dc.l {$82000000+NonHandledInterrupt}          ; irq14
       dc.l {$82000000+NonHandledInterrupt}          ; irq15
       dc.l {$82000000+NonHandledInterrupt}          ; irq16
       dc.l {$82000000+NonHandledInterrupt}          ; irq17
       dc.l {$82000000+NonHandledInterrupt}          ; irq18
```

```
    dc.l {$82000000+NonHandledInterrupt}        ; irq19
    dc.l {$82000000+NonHandledInterrupt}        ; irq20
    dc.l {$82000000+NonHandledInterrupt}        ; irq21
    dc.l {$82000000+NonHandledInterrupt}        ; irq22
    dc.l {$82000000+NonHandledInterrupt}        ; irq23
    dc.l {$82000000+NonHandledInterrupt}        ; irq24
    dc.l {$82000000+NonHandledInterrupt}        ; irq25
    dc.l {$82000000+NonHandledInterrupt}        ; irq26
    dc.l {$82000000+NonHandledInterrupt}        ; irq27
    dc.l {$82000000+NonHandledInterrupt}        ; irq28
    dc.l {$82000000+NonHandledInterrupt}            ; irq29

end
```