```
hold macro
        nop
        nop
        nop
        nop
        endm

disp_str macro string                   ;Macro for sending string to LCD
        irpc char, <string>
                if nul 'char'
                        exitm
                endif
                mov a,#'char'
                lcall data_in
        endm
        endm

build_char macro P1,P2,P3,P4,P5,P6,P7,P8  ;Macro for building a custom character
        irp arg, <P1,P2,P3,P4,P5,P6,P7,P8>
                mov a,#arg
                acall data_in
        endm
        endm

        sec equ 30h                     ;second register
        min equ 31h                     ;minutes register
        hour equ 32h                    ;hour register
```

```
days equu 33h                   ;days register
date equ 34h                    ;date register
month equ 35h                   ;month register
year equ 36h                    ;year register
alarm equ 37h                   ;alarm register
sig equ 38h                     ;signature for data checking in RTC RAM
ahour equ 39h                   ;alarm hour register
amin equ 3Ah                    ;alarm minute register
sda equ P1.1                    ;Serial data pin
scl equ P1.0                    ;Serial clock pin
rw equ p3.6                     ;read write pin of LCD
en equ p3.5                     ;enable pin of LCD
rs equ p3.7                     ;register select pin of LCD
alarm_key equ p3.3              ;Alarm set key
time_key equ p3.2               ;Time ser key
increment equ p1.3              ;Increment value
decrement equ p1.2              ;Decrement Value
alarm_port equ p1.4             ;Alarm Buzzer port
alarm_flag equ 20h              ;flag for alarm
time_flag equ 21h               ;flag for time
ampm_flag equ 22h               ;am/pm flag
alarm_on equ 23h                ;Alarm on flag
alarm_off equ 24h               ;alarm off flag
alarm_ring equ 25h              ;alarm ring flag
alarm_hrs equ 58h               ;Alarm hour register
alarm_min equ 59h               ;Alarm Minutes register
alarm_chk_hour equ 5Ah                  ;Alarm hour check register
```

```
alarm_chk_min equ 5Bh          ;Alarm min check register
ampm equ 61h                   ;AM/PM register
BELL equ 7h                    ;Bell icon
SPEAKER_OFF equ 1h             ;speaker off icon
SPEAKER_ON equ 2h              ;speaker on icon
CLOCK equ 3h                   ;clock icon
OK equ 4h                      ;ok icon
HEART equ 5h                   ;heart icon
MUSIC equ 6h                   ;music icon

org 0000h                      ;Starting of main program
ljmp start

org 03h                        ;interrupt for set time
setb time_flag
reti

org 13h                        ;Interrupt for set alarm
setb alarm_flag                ;and switch off the alarm if on
jnb alarm_off,hmmm
setb alarm_port
mov a,#1h
lcall command
mov a,#81h
lcall command
mov a,#OK
acall data_in
```

```
        mov r7,#35
        disp_str < Alarm Off!>
        setb tr0
loooop:     jnb tf0,$
        clr tf0
        djnz r7,loooop
        clr tr0
        mov a,#1h
        acall command
        acall disp_const
        setb alarm_off
        setb alarm_on
        clr alarm_flag
        clr alarm_flag
        clr alarm_on
        setb alarm_port
        clr alarm_off
        clr alarm_flag
        clr alarm_ring
hmmm:       reti

start:
        ;clear RAM
        mov R0,#7FH
        clr A
clr_ram:
        mov @R0,A
```

```
        djnz R0, clr_ram

        ;Init I2C Port
        lcall i2cinit

        mov tmod,#1h                    ;start routine
        mov sp,#10h
        acall initial
        acall build
        mov a,#80h
        acall command
        mov a,#CLOCK
        acall data_in
        disp_str < Digital Clock >      ;Name display
        mov r7,#50
        setb tr0
wloop:
        jnb tf0,$
        clr tf0
        djnz r7,wloop
        clr time_flag
        clr alarm_flag
        clr alarm_on
        clr ampm_flag
        setb alarm_port

        lcall startc                    ;Reading invalid hour entry
```

```
        mov a,#0d0h                     ;initialize the RTC if necessary
        lcall send
        lcall ack
        mov a,#02h
        lcall send
        lcall ack
        lcall rstart
        mov a,#0d1h
        acall send
        acall ack
        acall read
        acall nak
        acall stop
        anl a,#1fh
        cjne a,#12h,step
step:
        jnc done
        acall startc                ;Reading signature byte if same then no need
        mov a,#0d0h                     ;of initialization else initialize the RTC
        acall send
        acall ack
        mov a,#08h
        acall send
        acall ack
        acall rstart
        mov a,#0d1h
        acall send
```

```
        acall ack
        acall read
        acall nak
        acall stop
        cjne a,#'~',done
        sjmp run
done:
        ;acall initial
        ;acall disp_const
        mov sec,#00h                    ;For more information about the RTC DS1307
        mov min,#0h                     ;Please refer to the datasheet
        mov hour,#52h                           ;21h for 24 hour format
        mov days,#01h
        mov date,#1h
        mov month,#1h
        mov year,#06h
        mov alarm,#0
        mov sig,#'~'
        mov ahour,#0
        mov amin,#0
        mov r6,#0bh
        mov r0,#sec
        mov r1,#00h
        acall rtc_ini                   ;initializing RTC if required
run:
        mov ie,#85h
        acall initial
```

```
        acall disp_const
run1:
        acall startc                    ;Actual Starting
        mov a,#0d0h
        acall send
        acall ack
        mov a,#00h
        acall send
        acall ack
        mov r6,#0fh
        acall rstart                    ;Reading the RCT content
        mov a,#0d1h
        acall send
        acall ack
        mov r0,#40h
here:
        lcall read
        mov @r0,a
        inc r0
        cjne r6,#1,sendack
        acall nak
        sjmp naksent
sendack:
        acall ack
naksent:
        djnz r6,here
        acall stop
```

```
        acall display                   ;Display RTC
        jnb time_flag,noset             ;check for time flag if set the set the clock
        clr time_flag
        clr ea
        acall time_set
        clr time_flag
        mov r6,#09h
        mov r0,#30h
        mov r1,#00h
        acall rtc_ini
        clr time_flag
        mov a,#1h
        acall command
        mov a,#81h
        acall command
        mov a,#OK
        acall data_in
        clr time_flag
        disp_str < Time Set!>
        mov r7,#35
        setb tr0
looooop:jnb tf0,$
        clr tf0
        djnz r7,looooop
        clr tr0
        mov a,#1h
```

```
        acall command
        acall disp_const
        setb ea
noset:
        jnb alarm_flag,noset1           ;check for alarm flag if set then set alarm
        clr alarm_flag
        clr ea
        lcall alarm_set
        setb ea
        setb alarm_on
        clr alarm_flag
        setb alarm_on
noset1:
        jnb alarm_ring,noset2           ;check if alarm ring flag is set
        lcall alarm_alarm               ;if set then ring the alarm
noset2: ljmp run1                       ;Do everything again

;*************************************
;Initializing I2C Bus Communication
;*************************************
i2cinit:
      setb sda
      setb scl
      ret

;*************************************
;ReStart Condition for I2C Communication
```

```
;*************************************
rstart:
        clr scl
        setb sda
        setb scl
        clr sda
        ret

;*************************************
;Start Condition for I2C Communication
;*************************************
startc:
        setb scl
        clr sda
        clr scl
        ret

;*************************************
;Stop Condition For I2C Bus
;*************************************
stop:
        clr scl
        clr sda
        setb scl
        setb sda
        ret
```

```
;****************************************
;Sending Data to slave on I2C bus
;****************************************
send:
        mov r7,#08
back:
        clr scl
        rlc a
        mov sda,c
        setb scl
        djnz r7,back
        clr scl
        setb sda
        ret

;****************************************
;ACK and NAK for I2C Bus
;****************************************
ack:
        clr sda
        setb scl
        clr scl
        setb sda
        ret

nak:
        setb sda
```

```
        setb scl
        clr scl
        setb scl
        ret

;****************************************
;Receiving Data from slave on I2C bus
;****************************************
read:
        mov r7,#08
back2:
        clr scl
        setb scl
        mov c,sda
        rlc a
        djnz r7,back2
        clr scl
        setb sda
        ret

recv:                                   ;Recieving data from I2C bus
            lcall read
            lcall ack
            mov @r0,a
            inc r0
            ret
```

```
rtc_ini:acall startc                    ;RTC initialization subroutine
        mov a,#0d0h
        acall send
        acall ack
        mov a,r1
        acall send
        acall ack
rtc_ini_loop:
        mov a,@r0
        acall send
        acall ack
        inc r0
        inc r1
        ;acall stop
        djnz r6,rtc_ini_loop
        acall stop
        ret

initial:mov a,#38h                      ;LCD initialization subroutine
        acall command
        mov a,#0ch
        acall command
        mov a,#01h
        acall command
        mov a,#06h
        acall command
        ret
```

```
build:  mov a,#40h                      ;Building custom character routine
        acall command
        build_char 0h,0h,0h,0h,0h,0h,0h,0h    ;BELL
        build_char 1h,3h,0fh,0fh,0fh,3h,1h,0h       ;SPEAKER OFF
        build_char 8h,10h,0h,18h,0h,10h,8h,0h       ;SPEAKER ON
        build_char 0h,0eh,15h,17h,11h,0eh,0h,0h     ;CLOCK
        build_char 0h,1h,3h,16h,1ch,8h,0h,0h        ;OK
        build_char 0ah,1fh,1fh,1fh,0eh,4h,0h,0h     ;HEART
        build_char 2h,3h,2h,0eh,1eh,0ch,0h,0h       ;MUSIC
        build_char 4h,0eh,0eh,0eh,1fh,0h,4h,0h      ;BELL
        ret

display:mov r1,#40h                     ;Displaying RTC content
        mov a,#0Cah
        acall command
        mov a,@r1
        mov 50h,@r1
        acall disp_val
PASS:   inc r1
        mov a,#0C7h
        acall command
        mov a,@r1
        mov 51h,@r1
        mov alarm_chk_min,@r1
        acall disp_val
PASS1:      inc r1
```

```
        mov a,#0c4h
        acall command
        mov a,@r1
        mov 52h,@r1
        mov alarm_chk_hour,@r1
        mov r4,a
        anl a,#1fh
        acall disp_val
        mov a,r4
        anl a,#20h
        cjne a,#00h,pm
        mov a,#0cdh
        acall command
        disp_str <am>
        sjmp pass2
pm:     mov a,#0cdh
        acall command
        disp_str <pm>
PASS2:      inc r1
        mov a,#80h
        acall command
        mov a,@r1
        mov 53h,@r1
        acall day
PASS3:      inc r1
        mov a,#88h
        acall command
```

```
        mov a,@r1
        mov 54h,@r1
        acall disp_val
PASS4:  inc r1
        mov a,#8bh
        acall command
        mov a,@r1
        mov 55h,@r1
        acall disp_val
PASS5:  inc r1
        mov a,#8eh
        acall command
        mov a,@r1
        mov 56h,@r1
        acall disp_val
        jnb alarm_on,PASS6
        inc r1
        inc r1
        inc r1
        mov a,@r1
        cjne a,52h,PASS6
        inc r1
        mov a,@r1
        cjne a,51h,PASS6
        setb alarm_ring
        setb alarm_ring
PASS6:  ret
```

```
disp_val:                    ;Display 8-bit Decimal Value
        push acc
        mov r5,a
        anl a,#0f0h
        swap a
        mov dptr,#ascii
        movc a,@a+dptr
        acall data_in
        mov a,r5
        anl a,#0fh
        movc a,@a+dptr
        acall data_in
        pop acc
        ret

disp_const:                  ;Display constant characters
        mov a,#8ah
        acall command
        mov a,#'/'
        acall data_in
        mov a,#8dh
        acall command
        mov a,#'/'
        acall data_in
        mov a,#0c6h
        acall command
```

```
        mov a,#':'
        acall data_in
        mov a,#0c9h
        acall command
        mov a,#':'
        acall data_in
        mov a,#0c0h
        acall command
        mov a,#BELL
        acall data_in
        mov a,#SPEAKER_OFF
        acall data_in
        ret

command:acall busy                  ;Sending command to LCD
        MOV P2,A
        CLR rs
        CLR rw
        SETB en
        CLR en
        RET

data_in:acall busy                  ;Sending data to LCD
        SETB rs
        MOV P2,A
        CLR rw
        SETB en
```

```
        CLR en
        RET

busy:   ;SETB P2.7                  ;Checking Busy flag
        mov P2, #0FFH
        CLR rs
        SETB rw
wait:   CLR en
        SETB en
        JB P2.7,wait
        RET

day:    push acc                    ;Displaying day subroutine for RTC
        cjne a,#01,skip
        disp_str < *Sun*>
        ljmp exit
skip:   cjne a,#02,skip2
        disp_str < *Mon*>
        ljmp exit
skip2:  cjne a,#03,skip3
        disp_str < *Tue*>
        ljmp exit
skip3:  cjne a,#04,skip4
        disp_str < *Wed*>
        sjmp exit
skip4:  cjne a,#05,skip5
        disp_str< *Thu*>
```

```
        sjmp exit
skip5:  cjne a,#06,skip6
        disp_str < *Fri*>
        sjmp exit
skip6:  disp_str < *Sat*>
exit:   pop acc
        ret

time_set:                                   ;Setting time subroutine
        jnb time_key,$
        mov a,#1
        acall command
        mov a,#80h
        acall command
        mov a,#CLOCK
        acall data_in
        disp_str < set minutes:>
        mov a,#0c5h
        acall command
        mov a,51h
        acall disp_val
        lcall hexdec
key:    push acc
        mov a,#0c5h
        acall command
        pop acc
        jb increment,chk1
```

```
        inc a
        cjne a,#60,ok1
        mov a,#0
ok1:    jnb increment,$
        lcall dechex
        acall disp_val
        lcall hexdec
        sjmp key
chk1:   jb decrement,chk2
        dec a
        cjne a,#0ffh,ok2
        mov a,#59
ok2:    jnb decrement,$
        lcall dechex
        acall disp_val
        lcall hexdec
        sjmp key
chk2:   jb time_key,key

        lcall dechex
        mov min,a
        jnb time_key,$

        mov a,#1
        acall command
        mov a,#80h
        acall command
```

```
        mov a,#CLOCK
        acall data_in
        disp_str < set hours:>
        mov a,#0c5h
        acall command
        mov a,52h
        anl a,#1fh
        acall disp_val
        lcall hexdec
key1:   push acc
        mov a,#0c5h
        acall command
        pop acc
        jb increment,chk3
        inc a
        cjne a,#13,ok3
        mov a,#1
ok3:    jnb increment,$
        lcall dechex
        acall disp_val
        lcall hexdec
        sjmp key1
chk3:   jb decrement,chk4
        dec a
        cjne a,#0,ok4
        mov a,#12
ok4:    jnb decrement,$
```

```
        lcall dechex
        acall disp_val
        lcall hexdec
        sjmp key1
chk4:   jb time_key,key1

        lcall dechex
        mov hour,a
        jnb time_key,$

        mov a,#1
        acall command
        mov a,#80h
        acall command
        mov a,#CLOCK
        acall data_in
        disp_str < set am/pm:>
        mov a,52h
        anl a,#20h
        cjne a,#00h,pm1
        mov a,#0c5h
        acall command
        disp_str <am>
        setb ampm_flag
        sjmp key2
pm1:    mov a,#0c5h
        acall command
```

```
        disp_str <pm>
        clr ampm_flag

key2:   mov a,#0c5h
        acall command
        jb increment,chk5
        jb ampm_flag,ok5
        setb ampm_flag
        disp_str <am>
        hold
        jnb increment,$
        sjmp key2
ok5:    clr ampm_flag
        disp_str <pm>
        hold
        jnb increment,$
        sjmp key2
chk5:   jb decrement,chk6
        jb ampm_flag,ok6
        setb ampm_flag
        disp_str <am>
        hold
        jnb decrement,$
        sjmp key2
ok6:    clr ampm_flag
        disp_str <pm>
        hold
```

```
        jnb decrement,$
        sjmp key2
chk6:   jb time_key,key2

        jnb ampm_flag,noam
        mov a,hour
        add a,#40h
        sjmp nopm
noam:   mov a,hour
        add a,#60h
nopm:   mov hour,a
        jnb time_key,$

        mov a,#1
        acall command
        mov a,#80h
        acall command
        mov a,#CLOCK
        acall data_in
        disp_str < set days:>
        mov a,#0c5h
        lcall command
        mov a,53h
        push acc
        lcall day
        pop acc
key3:   push acc
```

```
        mov a,#0c5h
        lcall command
        pop acc
        jb increment,chk7
        inc a
        cjne a,#8,ok7
        mov a,#1
ok7:    jnb increment,$
        push acc
        lcall day
        pop acc
        sjmp key3
chk7:   jb decrement,chk8
        dec a
        cjne a,#0,ok8
        mov a,#7
ok8:    jnb decrement,$
        push acc
        lcall day
        pop acc
        sjmp key3
chk8:   jb time_key,key3

        mov days,a
        jnb time_key,$

        mov a,#1
```

```
        lcall command
        mov a,#80h
        lcall command
        mov a,#CLOCK
        lcall data_in
        disp_str < set date:>
        mov a,#0c5h
        lcall command
        mov a,54h
        lcall disp_val
        lcall hexdec
key4:   push acc
        mov a,#0c5h
        lcall command
        pop acc
        jb increment,chk9
        inc a
        cjne a,#32,ok9
        mov a,#1
ok9:    jnb increment,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key4
chk9:   jb decrement,chk10
        dec a
        cjne a,#0,ok10
```

```
        mov a,#31
ok10:   jnb decrement,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key4
chk10: jb time_key,key4

        lcall dechex
        mov date,a
        jnb time_key,$

        mov a,#1
        lcall command
        mov a,#80h
        lcall command
        mov a,#CLOCK
        lcall data_in
        disp_str < set month:>
        mov a,#0c5h
        lcall command
        mov a,55h
        lcall disp_val
        lcall hexdec
key5:   push acc
        mov a,#0c5h
        lcall command
```

```
        pop acc
        jb increment,chk11
        inc a
        cjne a,#13,ok11
        mov a,#1
ok11:   jnb increment,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key5
chk11: jb decrement,chk12
        dec a
        cjne a,#0,ok12
        mov a,#12
ok12:   jnb decrement,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key5
chk12: jb time_key,key5

        lcall dechex
        mov month,a
        jnb time_key,$

        mov a,#1
        lcall command
```

```
        mov a,#80h
        lcall command
        mov a,#CLOCK
        lcall data_in
        disp_str < set year:>
        mov a,#0c5h
        lcall command
        mov a,56h
        lcall disp_val
        lcall hexdec
key6:   push acc
        mov a,#0c5h
        lcall command
        pop acc
        jb increment,chk13
        inc a
        cjne a,#100,ok13
        mov a,#0
ok13:   jnb increment,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key6
chk13: jb decrement,chk14
        dec a
        cjne a,#0ffh,ok14
        mov a,#99
```

```
ok14:   jnb decrement,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp key6
chk14: jb time_key,key6

        jnb time_key,$
        lcall dechex
        mov year,a
        mov alarm,#00
        mov sig,#'~'
        clr time_flag
        ret

alarm_set:                              ;Setting Alarm Subroutine
        jnb alarm_key,$
        mov a,#1
        lcall command
        mov a,#80h
        lcall command
        mov a,#BELL
        lcall data_in
        disp_str < set minutes:>
        mov a,#0c5h
        lcall command
        mov a,#30h
```

```
        lcall disp_val
        lcall hexdec
akey:   push acc
        mov a,#0c5h
        lcall command
        pop acc
        jb increment,achk1
        inc a
        cjne a,#60,aok1
        mov a,#0
aok1:   jnb increment,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp akey
achk1: jb decrement,achk2
        dec a
        cjne a,#0ffh,aok2
        mov a,#59
aok2:   jnb decrement,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp akey
achk2: jb alarm_key,akey

        lcall dechex
```

```
        mov alarm_min,a
        jnb alarm_key,$

        mov a,#1
        lcall command
        mov a,#80h
        lcall command
        mov a,#BELL
        lcall data_in
        disp_str < set hours:>
        mov a,#0c5h
        lcall command
        mov a,#44h
        anl a,#1fh
        lcall disp_val
        lcall hexdec
akey1:  push acc
        mov a,#0c5h
        lcall command
        pop acc
        jb increment,achk3
        inc a
        cjne a,#13,aok3
        mov a,#1
aok3:   jnb increment,$
        lcall dechex
        lcall disp_val
```

```
        lcall hexdec
        sjmp akey1
achk3:  jb decrement,achk4
        dec a
        cjne a,#0,aok4
        mov a,#12
aok4:   jnb decrement,$
        lcall dechex
        lcall disp_val
        lcall hexdec
        sjmp akey1
achk4:  jb alarm_key,akey1

        lcall dechex
        mov alarm_hrs,a
        jnb alarm_key,$

        mov a,#1
        lcall command
        mov a,#80h
        lcall command
        mov a,#BELL
        lcall data_in
        disp_str < set am/pm:>
        mov a,#0c5h
        lcall command
        mov a,alarm_hrs
```

```
        anl a,#20h
        cjne a,#20h,its_pm
        disp_str <am>
        setb ampm_flag
        sjmp akey2
its_pm:     disp_str <pm>
        clr ampm_flag
akey2:  mov a,#0c5h
        lcall command
        jb increment,achk5
        jb ampm_flag,aok5
        setb ampm_flag
        disp_str <am>
        hold
        jnb increment,$
        sjmp akey2
aok5:   clr ampm_flag
        disp_str <pm>
        hold
        jnb increment,$
        sjmp akey2
achk5:  jb decrement,achk6
        jb ampm_flag,aok6
        setb ampm_flag
        disp_str <am>
        hold
        jnb decrement,$
```

```
        sjmp akey2
aok6:   clr ampm_flag
        disp_str <pm>
        hold
        jnb decrement,$
        sjmp akey2
achk6:  jb alarm_key,akey2
        jnb alarm_key,$
        mov a,alarm_hrs
        jb ampm_flag,anoam
        add a,#60h
        sjmp anopm
anoam:  mov a,alarm_hrs
        add a,#40h
anopm:  mov alarm_hrs,a
        mov ahour,alarm_hrs
        mov amin,alarm_min
        mov r6,#02h
        mov r0,#39h
        mov r1,#09h
        lcall rtc_ini
        mov a,#1h
        lcall command
        mov a,#81h
        lcall command
        mov a,#OK
        lcall data_in
```

```
        disp_str < Alarm On!>
        mov a,#0c4h
        lcall command
        mov a,alarm_hrs
        anl a,#1fh
        lcall disp_val
        mov a,#':'
        lcall data_in
        mov a,alarm_min
        lcall disp_val
        mov a,alarm_hrs
        anl a,#20h
        cjne a,#00h,ppm
        disp_str < am>
        sjmp ppass2
ppm:    disp_str < pm>
ppass2:         mov r7,#35
        setb tr0
looop:  jnb tf0,$
        clr tf0
        djnz r7,looop
        clr tr0
        mov a,#1h
        lcall command
        lcall disp_const
        mov a,#SPEAKER_ON
        lcall data_in
```

```
        setb alarm_off
        setb alarm_on
        clr alarm_flag
        clr alarm_flag
        ret

alarm_alarm:                    ;Alarm Ring subroutine
        clr alarm_port
        mov r3,#0ffh
a_loop:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop
        setb alarm_port
        mov r3,#0ffh
a_loop1:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop1
        clr alarm_port
        mov r3,#0ffh
a_loop2:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop2
        setb alarm_port
        mov r3,#0ffh
a_loop3:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop3
```

```
        clr alarm_port
        mov r3,#0ffh
a_loop4:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop4
        setb alarm_port
        mov r3,#0ffh
a_loop5:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop5
        clr alarm_port
        mov r3,#0ffh
a_loop6:mov r4,#0ffh
        djnz r4,$
        djnz r3,a_loop6
        setb alarm_port
        ret

dechex:     mov b,#0ah              ;Decimal to Hexadecimal conversion
        div ab
        mov 60h,b
        mov b,#10h
        mul ab
        add a,60h
        ret

hexdec:     mov b,#10h              ;Hexadecimal to deciaml conversion
```

```
        div ab
        mov 60h,b
        mov b,#0ah
        mul ab
        add a,60h
        ret

ascii:                                  ;ASCII lookup table
db      30h,31h,32h,33h,34h,35h,36h,37h,38h,39h

        end
```