

密码锁规格

文件编码：HA0096S

简介

通过扫描按键来获得数据，比对后相应发出使能或禁能开门信号。

使用说明

按键

- 一个密码设置键：红色开关
- 三个功能键：“#”、“*”、“0”
- 十个数字键：“1”、“2”、“3”、“4”、“5”、“6”、“7”、“8”、“9”

芯片规格

使用微控制器母体：HT48RA0-2、HT24LC02

额定电源：DC 6V(4 节 1.5V 电池)

工作电压 VDD：DC 6 Volts(系统要求)

DC 3.3 Volts(芯片规格)

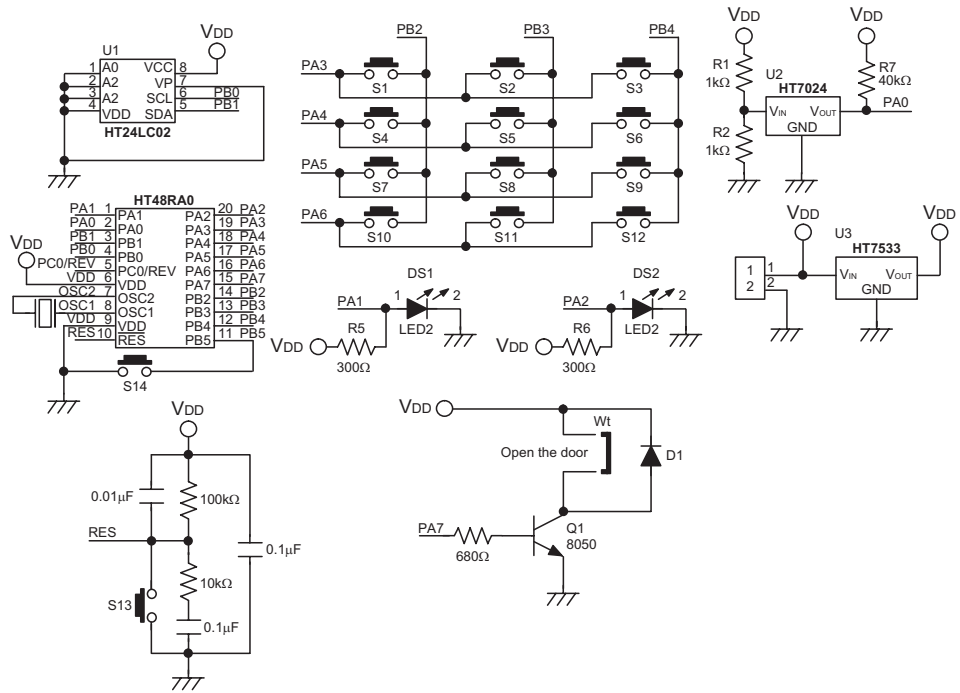
工作频率 OSC：Crystal 1MHz

功能描述

1. 出厂的统一密码为个人码“12345678”，管理码“88888888”。
2. 上电时密码锁红绿灯由点点亮到熄灭，表示可以开始输入密码。
3. 操作键板上“#”键为激活键，按“#”键可开启线路板进入工作状态。
4. 操作键板上“*”键在数据输入状态时为清除键，用于输入错误数字的清除，每按一次“*”清除一位数字，如长按 2 秒，红绿灯前后分别亮一下，表示清除所有数字。
5. 每按一次数字键绿灯闪一下，表示数字已输入。
6. 任何键按下 25 秒左右，无后续操作，计算机板自动进入睡眠状态。
7. 一旦进入睡眠状态则所有的操作终止，未完成动作视为无效操作。须按下“#”唤醒重新进入功能操作。
8. 当红灯点亮时，表示电压不足状态，需更换电池。

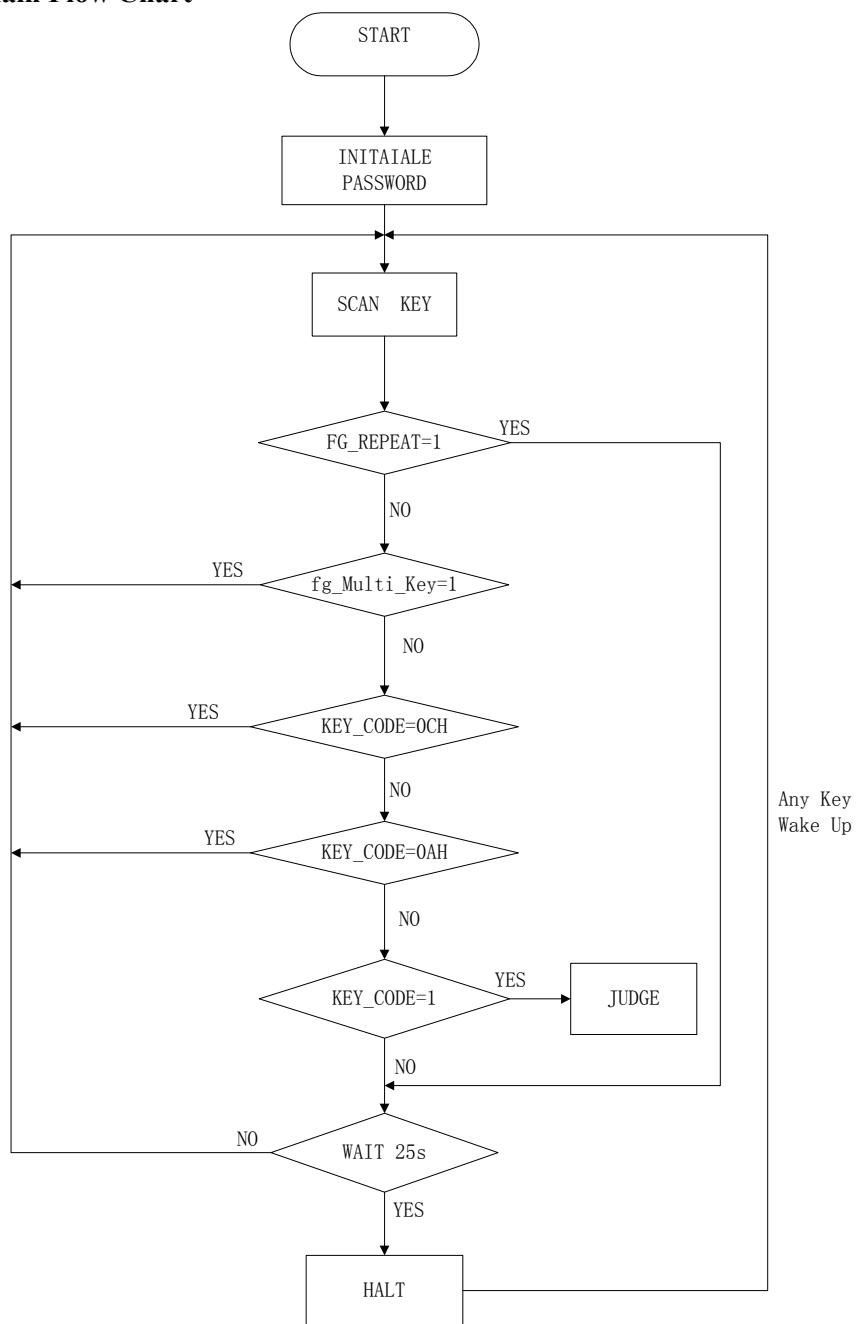
- 个人密码的设置与更改步骤：
 1. 先按下设置按钮（红色开关），然后输入管理码，红绿灯交替闪亮，表示进入个人密码设置或更改状态。
 2. 任意输入 8 位数字键后，绿灯点亮 1 秒，表示个人密码已设置或更改。如果已设置了密码，则再进行密码设置将只是对原有密码的更改。
 3. 红绿灯同时闪一下，表示个人密码修改成功。
- 管理码的设置与更改步骤：
 1. 先按下设置按钮（红色开关），按“0”键一次，再输入原管理码，红绿灯交替闪亮，进入管理码设置或更改状态。
 2. 任意输入 8 位数字键后，绿灯点亮 1 秒，表示个管理码已设置或更改。如果已设置了管理码，则再进行管理码设置将只是对原有管理码的更改。
 3. 红绿灯同时闪一下，表示管理码修改成功。
- 开门
 1. 按“#”键唤醒，输入 8 位密码。
 2. 输入个人码后，绿色灯点亮，表示密码正确，电磁铁吸合 10 秒，可进行开门动作。如 10 秒内未完成开门动作，则需重新输入密码。
 3. 如果密码输入不正确，则红灯点亮 1 秒，并重新切换到等待输入状态。
- 其它
 1. 计算机板使用四节 1.5V 碱性电池，当电池电压低于 5V 时红灯点亮，表示电池电压不足，提醒您更换电池。
 2. 如电池电压不足，可使用外接电源操作。

应用电路

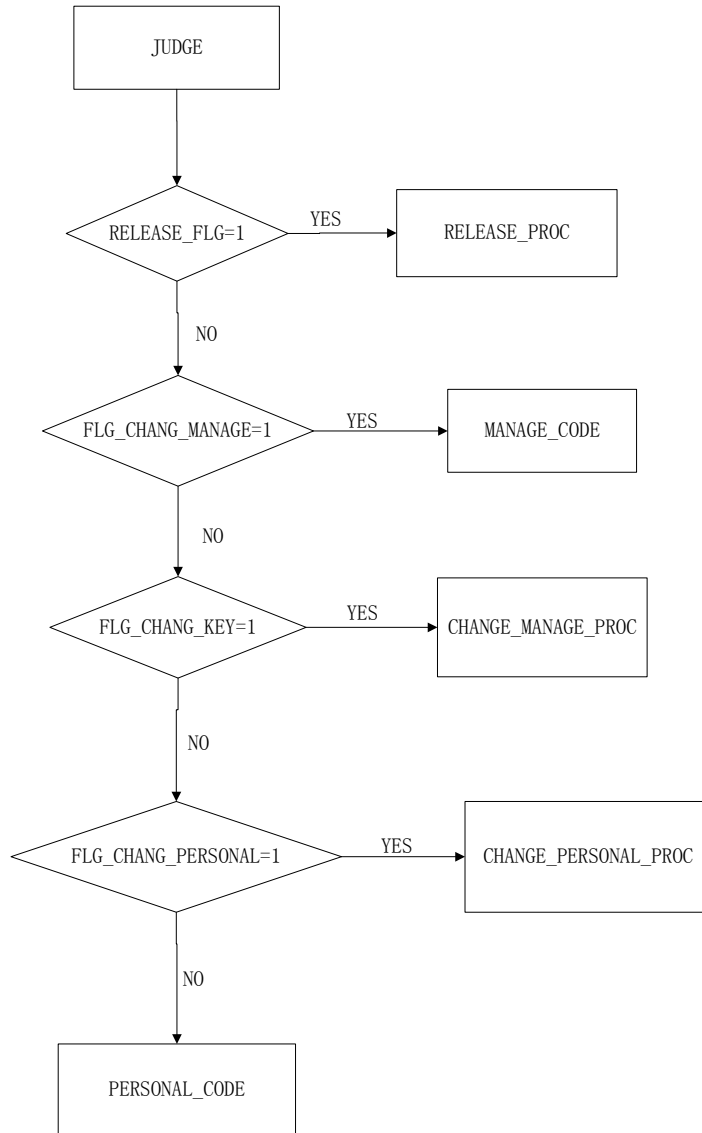


流程图

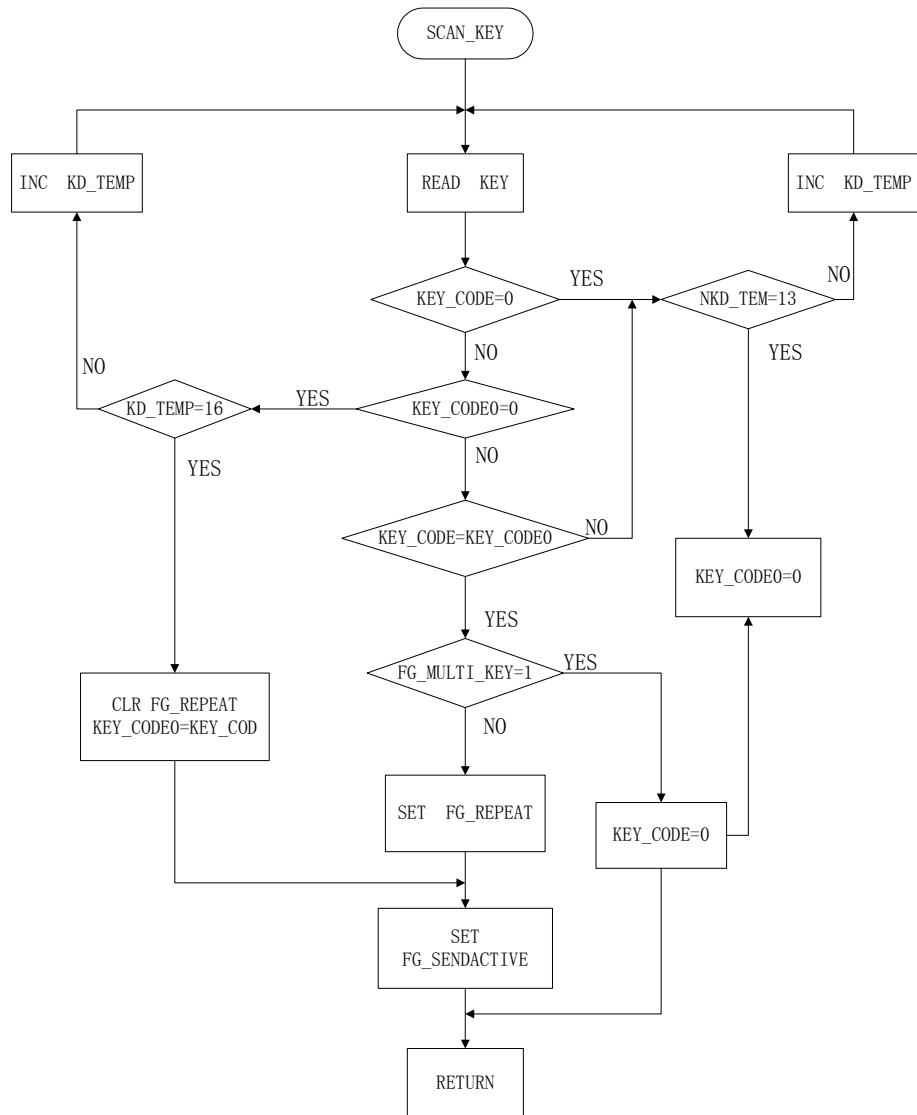
Main Flow Chart



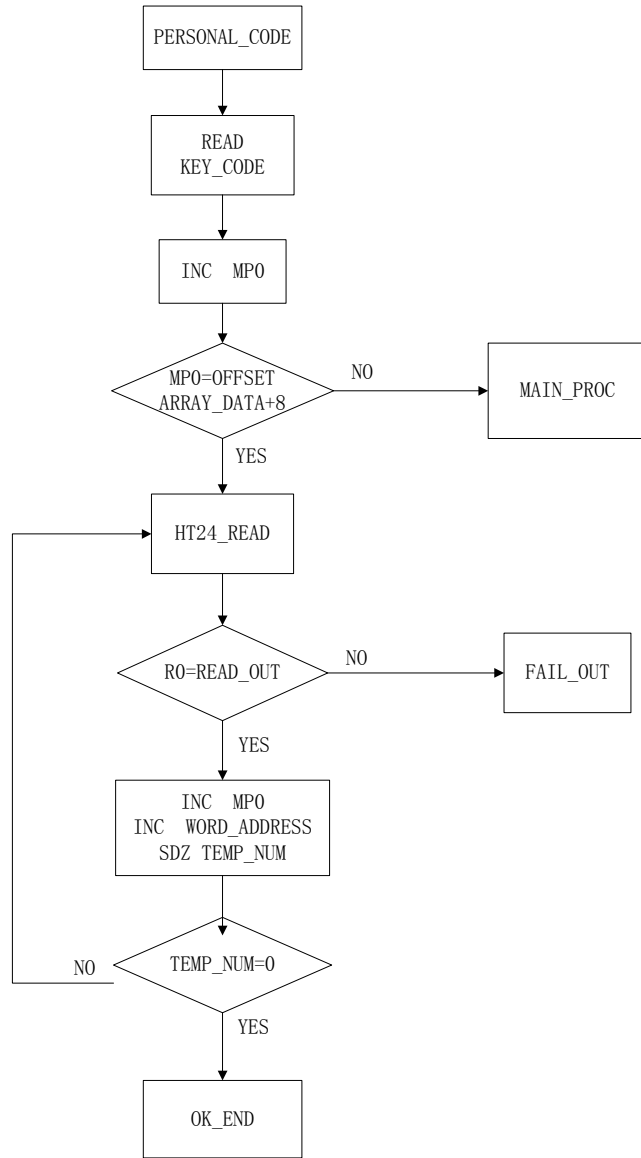
Judge Flow Chart



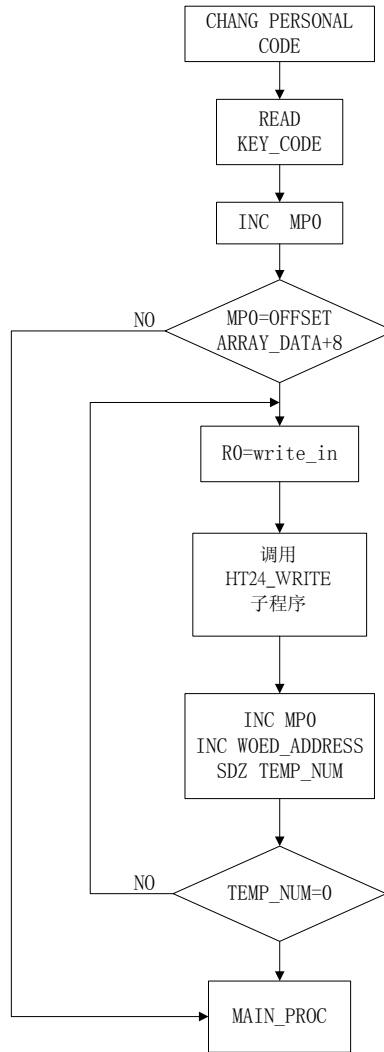
SCAN_KEY Flow Chart



Personal Code Flow Chart



Change Personal Code Flow Chart



主程序

```

;*****
;@*****      Filename : CLOCK.ASM      *****
;@*****      OSC:1MHz      *****
;@*****      HT48RA0-2      *****
;*****
;*****
; I/O Structure:
;PB1 -- SCL
;PB0 -- SDA
;;4 scan line for output PA3~PA6
;;3 data line for input PB2~PB4
;;key matrix=3x4 12keys
;;PB5 is for "#" input
;;SYSTEM VOLT = 3V
;;WDT DISABLE
;;PB0~PB4 NO WAKEUP ,PB5 WAKEUP
;*****

# include      ht48ra0-2.inc

;=====
; Define area
debounce_num   equ      16
scl             equ      pb.1           ; serial clock data input
sda             equ      pb.0           ; Serial data I/O
read_out       equ      [3ch]         ; write register
write_in       equ      [3dh]         ; Read register
word_address   equ      [3eh]
data_8         equ      [3fh]
;-----

;-----
; Define Macro
; Delay Macro, delay 100us
d_1 macro
    local      label
    mov        a,64h
    mov        delay,a
label:
    sdz        delay
    jmp        label
endm
    
```

```

*****
SAMPLE SECTION 'DATA'

;; ---Temp RAM---
    a_sr10      db      ?           ;;WORK Temp Register
    a_sr11      db      ?           ;;
    temp_num    db      ?

;;-----CODE RAM-----
    array_data  db 8 dup (?)        ;;8 bit password

;;-----Program RAM-----
    column      db      ?           ;;Scan Key Column Step Number
    key_code0   db      ?           ;;Save Previous Key Number
    key_code    db      ?           ;;Scan Key Number
    col_value   db      ?           ;;Read on column value
                                           ;;to a_COL_VALUE
    kd_temp     db      ?           ;;Key Debounce Check Times
    nkd_temp    db      ?
    ;;Key Release Confirm Times
    Key_data    db      ?
    ;;PA temp register
    Key_data1   db      ?

    Delay       db      ?
    delay0      db      ?
    delay1      db      ?
    t_count     db      ?
    t_count1    db      ?
    release_num db      ?           ;;count release key

;;---User FLAG define---
    fg_SendActive dbit           ;;VALIDITY KEY Flag
    fg_PressRec   dbit           ;;Check Column Multi_Key Flag
    fg_repeat     dbit           ;;REPEAT KEY Flag
    fg_Multi_Key  dbit           ;;INVALID KEY Flag
    release_flg   dbit
    flg_chang_key dbit
    lg_chang_personal dbit
    flg_chang_manage dbit
    flg_chang     dbit
    
```

```

;*****
code .section      at 0 'code'

        org          00h
START:
        set          pa
        mov          a, 20H
        mov          mp0, a
        mov          a, 50H
CLR_RAM:
        clr          r0
        inc          mp0
        sdz          acc
        jmp          clr_ram

        mov          a,070h
        mov          t_count,a
        mov          a,050h
        mov          t_count1,a
        mov          a,07fh
        mov          release_num,a

ini_clock:
        mov          a, offset personal_clock
        mov          tblp,a
        mov          a,00h                ; write 00H to eeprom address,it is personal
password
        mov          word_address,a
ini_personal_clock:
        tabrdl       write_in
        call         ht24_write
        inc          TBLP
        inc          word_address
        mov          a,offset personal_clock+8
        xor          a,TBLP
        sz           acc
        jmp          ini_personal_clock

        mov          a, offset manage_clock
        mov          tblp,a
        mov          a,010h                ; write 10H to eeprom address,it is personal
password
        mov          word_address,a
    
```

```

ini_manage_clock:
    tabrdl    write_in
    call     ht24_write
    inc      TBLP
    inc      word_address
    mov      a,offset manage_clock+8
    xor      a,TBLP
    sz       acc
    jmp      ini_manage_clock

    clr      pa.1
    clr      pa.2
    mov      a, offset array_data
    mov      mp0, a

;@*****Polling System Event*****
MAIN_PROC:
    call     scan_key_proc          ;SCAN KEY PROCEDURE
    mov      a,0bh                ;* key
    xor      a,key_code
    sz       acc
    jmp      $+2
    jmp      release_proc

    sz       fg_repeat
    jmp      sleep

    sz       fg_Multi_Key          ;Check Invalid Key
    jmp      main_proc ;

    mov      a,0ch                  ;red key
    xor      a,key_code
    sz       acc
    jmp      $+4
    set      flg_chang_key
    set      flg_chang_personal
    jmp      main_proc

    mov      a,0ah                  ;"0" key
    xor      a,key_code
    sz       acc
    jmp      $+4
    clr      flg_chang_personal
    set      flg_chang_manage
    jmp      main_proc

    sz       key_code ;Check NO Key Press
    jmp      judge
    jmp      sleep
    
```

```

judge:
    set        pa.1
    call       delay30ms
    clr        pa.1

    mov        a,070h
    mov        t_count,a
    mov        a,050h
    mov        t_count1,a

    sz         release_flg
    jmp        release_proc          ;;deal with "*" key
    sz         flg_chang_key
    jmp        manage_code           ;;input mange password
    sz         flg_chang_manage
    jmp        chang_manage_proc     ;;change mange password
    sz         flg_chang_personal
    jmp        chang_personal_proc   ;;change personal password
    jmp        personal_code         ;;input personal password

SLEEP:
    sz         pa.0
    jmp        $+2
    jmp        low_voltage           ;;deal with low_voltage

    sdz        t_count1             ;\
    jmp        main_proc             ;\
    sdz        t_count               ;/if no key code wait 25s
    jmp        $-3                   ;/

    sz         PB.5
    halt                               ;System Halt
    clr        release_flg
    clr        flg_chang_key
    clr        flg_chang_manage
    clr        flg_chang_personal
    mov        a,070h
    mov        t_count,a
    mov        a,050h
    mov        t_count1,a
    jmp        main_proc ;Any Key WakeUp
;@*****SUBROUTINE[xx]: Scan 12 Key Matrix (3x4)*****
    
```

```

SCAN_KEY_PROC:
;---READ KEY PROCEDURE---
    ;;Initiate Process
    clr        key_code  ;
    clr        fg_PressRec      ;
    clr        column      ;
    clr        fg_Multi_Key     ;

    mov        a,0f9h
    mov        pa,a
    mov        a,0fdh
    mov        key_data,a
    mov        a,0ch
    mov        key_data1,a
NEXT_COLUMN:
    mov        a,key_data1
    xor        a,key_data
    rl        key_data1
    mov        key_data,a
    mov        pa,a                ;Scan Column PA PORT OUTPUT
    nop
    nop                            ;
;---READ PB PORT---
    mov        a,pb                ;;Read on column value
                                    ;;to a_COL_VALUE

    mov        col_value,a;;
    mov        a,1
    mov        a_sr11,a
    mov        a,1111011b        ;;
    mov        a_sr10,a          ;; This column value to ACC
    snz        col_value.4
    jmpP       key_row_4
    snz        col_value.3
    jmp        key_row_3
    snz        col_value.2
    jmp        key_row_2
    jmp        $+1
    nop
    jmp        col_without_key
key_row_4:
    rl        a_sr10                ;; a_SR10 rotate to "1110111b"
    inc        a_sr11
key_row_3:
    rl        a_sr10                ;; a_SR10 rotate to "1111011b"
    inc        a_sr11
    
```

```

key_row_2:
    mov     a,01ch
    andm   a,a_sr10
    and    a,col_value
    xor    A,a_sr10 ;;Check ROW Multi-Key
    sz     z
    jmp    col_with_key
col_error_key:
    set    fg_Multi_Key

col_without_key:
    jmp    $+1
    jmp    $+1
    jmp    $+1
    jmp    ready_2_next_column
col_with_key:
    mov     a,2                ;;\
    mov    temp_num,a         ;;\
    mov    a,column           ;;\
    add    a,column           ;;/COLUMN×3→ACC
    ;mov   column,a          ;;/
    sdz   temp_num           ;;/
    jmp   $-2                ;;/
    add   a,a_sr11          ;;
    mov   key_code,a        ;;
    sz    fg_PressRec       ;;Check Column Multi-Key
    set   fg_Multi_Key      ;;
    set   fg_PressRec       ;;

ready_2_next_column:
    inc   column
    mov   A,column
    xor   a,4
    snz  z
    jmp  next_column
    nop

READ_KEY_COMPLETE:

;@***** SUBROUTINE[xx]: Check Key Active *****
;@--- Scan Key Complete, Check key valid ? ---
CHECK_KEY_ACTIVE:
    sz    key_code ;;If a_KEY_NUM=FFH,invalid key
    jmp   valid_key_chk
    jmp   invalid_key_chk

;@-----
VALID_KEY_CHK:
    sz    key_code0 ;;Check New Key
    jmp   valid_key_pro
    jmp   key_deb
    
```

```

VALID_KEYC_PRO:
    mov     A, key_code
    xor     A, key_code0        ;;Check Repeat key
    snz     z
    jmp     valid_key
;;-----
VALID_RPSEND_KEY:
    snz     fg_Multi_key        ;Check Multi-Key Clear KEY_CODE
    jmp     repeat_key
    clr     key_code
    ret

REPEAT_KEY:
    set     fg_REPEAT          ;;SET Repeat Flag
    jmp     $+1
    jmp     set_validfg

valid_key:
    mov     a,key_code0        ;;Check Change Key
    xor     a,0ffh            ;;Check Debounce END
    snz     z
    jmp     chang_key

;;--- NEW KEY Initiate
    clr     fg_REPEAT
    mov     a,key_code
    mov     key_code0,a

;;--- SET VALIDITY KEY
SET_VALIDFG:
    set     fg_SendActive
    ret

CHANG_KEY:
    clr     ke_temp
    jmp     rkey_pro

;@-----
INVALID_KEY_CHK:
    clr     kd_temp

RKEY_PRO:                                ;Release Key Confirm
    inc     nkd_temp
    mov     a,nkd_temp
    xor     a,13
    snz     z
    jmp     scan_key_proc
    dec     nkd_temp
    
```

```

NO_KEY_PROC:
    clr        fg_SendActive
    jmp        $+1
    jmp        $+1
    nop

EXIT_INVALID_KEY:
    mov        a, key_code           ;Clear KEY_CODE0
    mov        key_code0,a
    jmp        $+1
    ret

KEY_DEB:
                                           ;Check Key Debounce

    inc        kd_temp
    mov        A, kd_temp
    xor        a,debounce_num
    snz        z
    jmp        scan_key_proc

    mov        a,0ffh               ;SAVE KEY Debounce Check END
    mov        key_code0,A         ;0FFH to KEY_CODE0
    jmp        scan_key_proc

;*****read personal clock*****
personal_code:
    mov        a,key_code
    mov        r0, a
    inc        mp0
    rl         release_num
    mov        a,mp0
    and        a,7fh
    xor        a,offset array_data+8
    sz         acc                  ;;input 8 bit personal password
    jmp        main_proc

read_code1:
    mov        a,07fh
    mov        release_num,a
    mov        a,8
    mov        temp_num, a
    mov        a,0
    mov        word_address, a
    mov        a,offset array_data
    mov        mp0,a

read_n1:
    call       ht24_read
    mov        a,r0
    xor        a,read_out           ;;compare 8 bit personal
                                           ;;password
    
```

```

        sz          acc
        jmp         fail_out          ;;error password
        inc        mp0
        inc        word_address
        sdz        temp_num
        jmp         read_n1
        jmp         ok_end            ;;right password
;-----
chang_personal_proc:
    mov          a,key_code
    mov          r0,          a
    inc        mp0
    rl          release_num
    mov          a,mp0
    and         a,7fh
    xor         a,offset array_data+8
    sz          acc                  ;;input 8 bit manage password
    jmp         main_proc

    mov          a,07fh
    mov          release_num,a
    mov          a,8
    mov          temp_num, a
    mov          a,0
    mov          word_address, a
    mov          a,offset array_data
    mov          mp0,a
personal_1:
    mov          a,r0
    mov          write_in,a
    call         ht24_write          ;;input 8 bit to ht24
    inc        mp0
    inc        word_address
    sdz        temp_num
    jmp         personal_1
    clr        flg_chang_key
    clr        flg_chang_personal
    call        chang_end
    mov          a,offset array_data
    mov          mp0,a
    jmp         main_proc
    
```

```

;-----read manage clock-----
manage_code:
    mov     a,key_code
    mov     r0,a
    inc     mp0
    rl      release_num
    mov     a,mp0
    and     a,7fh
    xor     a,offset array_data+8
    sz      acc
    jmp     main_proc
read_code2:
    mov     a,07fh
    mov     release_num,a
    mov     a,8
    mov     temp_num,a
    mov     a,010h
    mov     word_address,a
    mov     a,offset array_data
    mov     mp0,a
read_n2:
    call    ht24_read
    mov     a,r0
    xor     a,read_out
    sz      acc
    jmp     fail_out
    inc     mp0
    inc     word_address
    sdz     temp_num
    jmp     read_n2
    clr     flg_chang_key
    call    chang_proc
    mov     a,offset array_data
    mov     mp0,a
    jmp     main_proc

;-----
chang_manage_proc:
    ;call    chang_proc
    mov     a,key_code
    mov     r0,a
    inc     mp0
    rl      release_num
    mov     a,mp0
    and     a,7fh
    xor     a,offset array_data+8
    sz      acc
    jmp     main_proc
    
```

```

        mov     a,07fh
        mov     release_num,a
        mov     a,8
        mov     temp_num,a
        mov     a,010h
        mov     word_address,a
        mov     a,offset array_data
        mov     mp0,a
manage_0:
        mov     a,r0
        mov     write_in,a
        call    ht24_write
        inc     mp0
        inc     word_address
        sdz     temp_num
        jmp     manage_0
        clr     flg_chang_key
        clr     flg_chang
        clr     flg_chang_manage
        call    chang_end
        mov     a,offset array_data
        mov     mp0,a
        jmp     main_proc
;-----
release_proc:
        snz     fg_repeat
        jmp     release_1
        call    delay30ms
        inc     delay0
        mov     a,13
        xor     a,delay0
        sz     acc
        jmp     main_proc
        jmp     release_end

release_1:
        snz     release_num.0           ;;which key will be clear
        mov     a,offset array_data
        snz     release_num.1
        mov     a,offset array_data+1
        snz     release_num.2
        mov     a,offset array_data+2
        snz     release_num.3
        mov     a,offset array_data+3
        snz     release_num.4
        mov     a,offset array_data+4
        snz     release_num.5
        mov     a,offset array_data+5
        snz     release_num.6
    
```

```

        mov     a,offset array_data+6
        snz    release_num.7
        mov     a,offset array_data+7

        mov     mp0,a
        clr     r0                ;;if pressed 2s,clear all
        clr     release_flg
        rr     release_num
        jmp     main_proc
;***** Write*****
ht24_write:
        set     sda
        d_1
        set     scl
        d_1
        clr     sda                ;start signal

        clr     scl
        set     sda                ;1
        d_1
        set     scl
        d_1

        clr     scl
        clr     sda                ;0
        set     scl
        d_1

        clr     scl
        set     sda                ;1
        d_1
        set     scl
        d_1

        clr     scl
        clr     sda                ;0
        set     scl
        d_1

        clr     scl
        clr     sda                ;a2,a1,a0=0
        set     scl
        d_1

        clr     scl
        set     scl
        d_1
    
```

```

        clr      scl
        set      scl
        d_1

        clr      scl
        clr      sda      ;0 write mode
        set      scl
        d_1

        clr      scl
        set      sda      ; 1 for ack,set input, accept the answering
signal
        d_1
        set      scl      ;read_modify_write
        d_1

skch:
        sz      sda      ;respond signal
        jmp     skch
        clr     scl
        mov     a,08h
        mov     data_8,a      ;8 bit
write_address_in:
        clr     sda
        sz     word_address.7
        set     sda
        d_1
        set     scl
        d_1

        clr     scl
        rl     word_address
        sdz    data_8
        jmp    write_address_in
        set     sda
        d_1
        set     scl
        d_1

wdow:
        sz     sda
        jmp    wdow
        clr    scl
        mov    a,08h
        mov    data_8,a
    
```

```

write_data_in:
    clr        sda
    sz        write_in.7
    set        sda
    d_1
    set        scl
    d_1

    clr        scl
    rl        write_in
    sdz        data_8
    jmp       write_data_in

    clr        sda
    set        scl
    d_1
    clr        scl
    set        scl
    d_1
    set        sda                ;stop signal
    d_1
    clr        scl
    ret

;***** Read*****
ht24_read:
    set        sda
    d_1
    set        scl
    d_1
    clr        sda                ;start signal

    clr        scl
    set        sda                ;1
    d_1
    set        scl
    d_1

    clr        scl
    clr        sda                ;0
    set        scl
    d_1

    clr        scl
    set        sda                ;1
    d_1

    set        scl
    d_1

```

```

        clr      scl
        clr      sda          ;0
        set      scl
        d_1
        clr      scl
        clr      sda          ;a2,a1,a0=0,0,0
        set      scl
        d_1

        clr      scl
        set      scl
        d_1

        clr      scl
        set      scl
        d_1

        clr      scl
        clr      sda          ;0 write mode
        set      scl
        d_1

        clr      scl
        set      sda          ;for ack
        set      scl
        d_1
    fl1:
        sz      sda
        jmp     ht24_read
        clr      scl
        mov     a,08h
        mov     data_8,a
    read_address_in:
        clr      sda
        sz      word_address.7
        set      sda
        d_1
        set      scl
        d_1
        clr      scl
        rl      word_address
        sdz     data_8
        jmp     read_address_in

        set      sda          ;for ack
        d_1
        set      scl
        d_1
    
```

```

skco:
    sz      sda
    jmp     skco
    clr     scl
restart:
    set     sda
    d_1
    set     scl
    d_1
    clr     sda                ;start signal

    clr     scl
    set     sda                ;1
    d_1
    set     scl
    d_1

    clr     scl
    clr     sda                ;0
    set     scl
    d_1

    clr     scl
    set     sda                ;1
    d_1
    set     scl
    d_1

    clr     scl
    clr     sda                ;0
    set     scl
    d_1

    clr     scl
    clr     sda                ;a2,a1,a0=0
    set     scl
    d_1

    clr     scl
    set     scl
    d_1

    clr     scl
    set     scl
    d_1

    clr     scl
    set     sda                ;1 read mode
    d_1

```

```

        set        scl
        d_1

        clr        scl
        set        sda                ;for ack
        d_1
        set        scl
        d_1
ewfp:
        sz        sda
        jmp        ewfp
        mov        a,08h
        mov        data_8,a
flow_out:
        clr        scl
        set        sda                ;input I/O
        d_1
        clr        read_out.7
        sz        sda
        set        read_out.7
        d_1
        set        scl
        d_1
        rl        read_out
        sdz        data_8
        jmp        flow_out

        clr        scl
        clr        sda
        set        scl
        d_1
        set        sda                ;stop signal
        d_1
        ret
;***** Deal with LED*****

fail_out:                ;deal with fail
        set        pa.2
        call        delay_200ms
        call        delay_200ms
        call        delay_200ms
        call        delay_200ms
        call        delay_200ms
clr_array_data:        ;;clrea 8 bit password
        mov        a,8
        mov        temp_num,a
        mov        a,offset array_data
        mov        mp0,a
    
```

```

out:
    clr        r0
    inc        mp0
    sdz        temp_num
    jmp        out
    mov        a,04h
    xorm       a,pa
    clr        delay0
    mov        a,offset array_data
    mov        mp0,a
    jmp        main_proc
;-----
release_end:
bright
    mov        a,02h
    xorm       a,pa
    call       delay_200ms
    call       delay_200ms
    mov        a,06h
    xorm       a,pa
    call       delay_200ms
    call       delay_200ms
    clr        delay0
    jmp        clr_array_data
;-----
chang_proc:
    mov        a,02h
    xorm       a,pa

    clr        delay
    mov        a,030h
    mov        delay1,a
;
$3:
    sdz        delay
    jmp        $3

    sdz        delay1
    jmp        $3
    clr        delay1
    clr        delay

    inc        delay0
    mov        a,04h
    xorm       a,pa

    clr        delay
    mov        a,030h
    mov        delay1,a
;
    
```

;; After the green LED bright,the red LED

```

$4:
    sdz        delay
    jmp        $4

    sdz        delay1
    jmp        $4
    clr        delay1
    clr        delay

    mov        a,3
    xor        a,delay0
    sz         acc
    jmp        chang_proc
    mov        a,0
    xorm       a,pa
    clr        delay0
    ret

chang_end:
    mov        a,06h
    xorm       a,pa
    clr        delay
    mov        a,196h
    mov        delay1,a
;

$5:
    sdz        delay
    jmp        $5

    sdz        delay1
    jmp        $5
    mov        a,06h
    xorm       a,pa
    ret
;=====
ok_end:
    clr        delay0
    mov        a,082h
    xorm       a,pa
    call       delay_200ms
    mov        a,25
    inc        delay0
    xor        a,delay0
    sz         acc
    jmp        $-5
    mov        a,080h
    xorm       a,pa
; Password is right,open the door
; PA.7 is low, open the magnet

```

```

        clr          delay0
        mov          a,offset array_data
        mov          mp0,a
        jmp          main_proc
;=====
low_voltage:
        set          pa.2
        jmp          $                ;;If low_voltage,No aciton

;*****delay program*****
delay_200ms:
        clr          delay
        mov          a,196h
        mov          delay1, a
$0:
        sdz          delay
        jmp          $0

        sdz          delay1
        jmp          $0
        ret
;-----
delay30ms:
        clr          delay
        mov          a,030h
        mov          delay1,a
$1:
        sdz          delay
        jmp          $1

        sdz          delay1
        jmp          $1
        clr          delay1
        clr          delay
        ret
;*****

```

,

ORG	03e0h
manage_clock:	
DC	8
DC	8
DC	8
DC	8
DC	8
DC	8
DC	8
DC	8

ORG	03f0h
personal_clock:	
DC	1
DC	2
DC	3
DC	4
DC	5
DC	6
DC	7
DC	8