

应用资料 3

JF24C编程指南

应用资料1 详细介绍了JF24C模块的性能与单片机的接口电路及应用指南。

应用资料2 详细描述了JF24C模块芯片MCU的工作程序及工作流程示意图，SPI协议时序图及各种数据。

应用资料3 详细介绍了JF24C模块与单片机应用编程指南供参考。

目前2.4G产品应用比较广泛，有些芯片性能也很不错，但价位都比较偏高，很难进入量产的产品。为降低成本JF24C模块采用裸片绑定，虽然性能指标略低于目前具有代表性的 nRF2401 CC2500 A7105但它的价格要比它们低很多，完全可以满足一般需要双向数据传输及双向遥控的短距离产品应用。

单发单收的产品使用比较简单，加电加信号就发射，收到信号就有输出，纯硬件产品单向传输，不需要软件程序的支持就可以完成收发功能。2.4G产品就比较复杂化了，芯片内有CPU需要软件程序的支持，必须要有单片机的指令才可以完成双向收发功能。单发单收的产品成本低廉应用广泛，但存在着严重的无法避免的同频干扰，2.4G产品具有跳频功能一般都有几十至100多个通道可以避开干扰。但2.4G产品复杂的软件程序也使一些不懂单片机的工程师望而怯步，同时2.4G产品的功耗及成本还有对墙体的穿透性能下降也影响到在低端产品的普及应用。

确定 JF24C 和单片机的硬件连接后，开始对单片机编程

编程顺序

一、定义单片机的引脚功能。

二、定义单片机通用寄存器。

三、写模块初始化寄存器的值。

四、写主程序：

1. 设置单片机端口

2. 写复位 JF24C

3. 写初始化 JF24C 微处理器寄存器 reg48---reg57

4. 写初始化 JF24C RF 寄存器 reg0---reg28

5. 写进入空闲模式

五、循环主程序

1. 写进入接收模式，接收数据后由第一个字节判断是否丢失数据，丢失则返回主循环。接收正确的数据后给发射机返回数据，收发不正确返回主程序。
2. 写检测是否按下发射键，如果有数据要发送则进入发射模式，发送错误返回主循环。
3. 写检测到 10ms 后进入空闲模式，
4. 写检测到 100ms 后进入发射模式。

编程流程图

2. 4G 测试板程序 (EM78P156)

2009-8-3 更改

2. 4G 模块测试板 (双向数据返回) 测试程序:

接通电源后 **电源指示灯亮** → 接收处于周期性的休眠与唤醒状态 → 按下主机发送按键 → **主机发送指示灯闪亮** → 从机收到数据后 **接收指示灯闪亮**, 同时从机自动返回主机确认数据, 从机 **发送指示灯闪亮** → 主机 **接收指示灯闪亮** → 发送接收成功。程序如下:

```

;*****
;GENERaL REGISTER DEFINE
;*****
INDF          EQU    0X00
TCC           EQU    0X01
PCL           EQU    0X02
STATUS       EQU    0X03
C             EQU    0      ;CARRY FLAG, 1=CARRY
DC           EQU    1      ;AUXILIARY CARRY FLAG, 1=AUXILIARY CARRY
Z           EQU    2      ;ZERO FLAG, 1=LOGIC OPERATION IS      ZERO
P           EQU    3      ;POWER DOWN BIT
T           EQU    4      ;TIME-OUT BIT
PS0          EQU    5      ;

PS1          EQU    6      ;PS1-PS0 PAGE SELECT BIT
GP           EQU    7      ;GENERAL READ/WRITE BIT
FSR          EQU    0X04

ISR          EQU    0X3F      ;INTERRUPT STaTUS REG
TCIF         ==    0      ;TCC OVERFOLW INTERRUPT FLaG
;ICIF        ==          ;PORT6 INPUT STaTUS CHaNGE INTERRUPT FLaG
EXIF         ==    1      ;EXTERNaL INTERRUPT FLaG, SET BY FaLLING
EDGE /INT PIN

;*****
;MaCRO DIFINE
;*****

```

```

BaNK0  MACRO
        BC FSR, 7
        BC FSR, 6
        ENDM
BaNK1  MACRO
        BC FSR, 7
        BS FSR, 6
        ENDM
BaNK2  MACRO
        BS FSR, 7
        BC FSR, 6
        ENDM
BaNK3  MACRO
        BS FSR, 7
        BS FSR, 6
        ENDM

```

```

;*****

```

```

;EM78P156 PORT6 aSSIGNMENT

```

```

;*****

```

```

PORT6      EQU      0X06
PRESET_N   ==      6
PPKT_FLG   ==      5
PSPI_MOSI  ==      6
PSPI_MISO  ==      6
PSPI_CLK   ==      6
PSPI_SS    ==      6
;PFIFO_FLG ==      6

```

```

SPI_MISO   ==      4
RESET_N    ==      3
SPI_CLK    ==      5
SPI_MOSI   ==      6
SPI_SS     ==      7
;FIFO_FLG  ==      6
PKT_FLG    ==      1

```

```

;*****

```

```

;EM78P156 PORT5 aSSIGNMENT

```

```

;*****

```

```

PORT5      EQU      0X05

```

```

PUP_LED ==      5
PDOWN_LED          ==      5
PRIGHT_LED         ==      5
PLEFT_LED          ==      5
PMI_LED            ==      5
PHI_LED            ==      5
UP_LED ==        2
RIGHT_LED          ==      3
MI_LED             ==      2
HI_LED             ==      3
;PORT9            EQU 0X09
;*****
;*****
IOCA               EQU    0X0A
IOCB               EQU    0X0B
IOCC               EQU    0X0C
IOCD               EQU    0X0D
IOCE               EQU    0X0E
IOCF               EQU    0X0F
;*****
;BANK0
;*****
A_BUFFER          EQU    0X10
FSR_BUFFER        EQU    0X11
STATUS_BUFFER     EQU    0X12
FLAG              EQU    0X13
;;;0              IS 0 TX 发送 DaTa
;;
;;               IS 1 RX 发送 DaTa
;;1              IS    1 10MS 到了需要进入 IDE 模式
;;2              IS 1   100MS 到了需要进入 RX 模式
SER_DIR           ==     0
SCAN_10_FINISH   ==     1
HAS_MAX_SIGNAL    ==     2
SPIRB             EQU    0X14          ;SPI REaD FIFO REGISTER
SPIWB             EQU    0X15          ;SPI WRITE FIFO REGISTER
SUM_UP            EQU    0X16
SUM_DOWN          EQU    0X17
SUM_LEFT          EQU    0X18
SUM_RIGHT         EQU    0X19
TIMEINC1          EQU    0X1a
TIMEINC2          EQU    0X1B
CODE_LEFT         EQU    0X1C
CODE_RIGHT        EQU    0X1D

```

```

ADDR          EQU    0X1E          ;6210 REGISTER aDRESS VaLUE
VALUE_H       EQU    0X1F
VALUE_L       EQU    0X20
CH_NO_INDEX   EQU    0X21
CH_NO         EQU    0X22          ;COM CHaNNEL BUF
TEMPO         EQU    0X23
TIMECNT1      EQU    0X26
TEMP1         EQU    0X2E
TEMP2         EQU    0X2F
CNT1          EQU    0X24
CNT2          EQU    0X25
TABLE_INDEX   EQU    0X32          ;6210 REGISTER aDRESS INDEX
TIMECNT       EQU    0X33
COMTEMP       EQU    0X34
TEMP3         EQU    0X35
BYTECNT       EQU    0X36
RX_BUF0       EQU    0X2A
RX_BUF1       EQU    0X2B
RX_BUF2       EQU    0X2C
RX_BUF3       EQU    0X2D

;*****
;CONSTANT DEFINE
;*****
READ_MASK     EQU    0X80          ;SPI REaD IS WRITTED "1"
RX_MASK       EQU    0X80          ;6210 RX CHANNEL MASK
FIFO_REG      EQU    0X50          ;6210 TX_FIFO REGISTER80
FIFO_PTR      EQU    0X52          ;6210 FIFO_RD_PTR REGISTER82
CLR_W_PTR     EQU    0X80          ;6210 FIFO_RD_PTR REGISTER82 BIT[15]=1 CLaR
TX_FIFO POINT TO 0
CLR_R_PTR     EQU    0X80          ;6210 FIFO_RD_PTR REGISTER82 BIT[7]=1 CLaR
RX_FIFO POINT TO 0

;*****
;CODE START HERE
;*****
        ORG    0X00
RESET:
        JMP    MAIN_LOOP
        ORG    0X08

INT_INT:
        RETI

```

```

;*****
;LOOKUP TABLE
;FaMER REGISTER INITIaL VaLUE (REG48~57)
;*****
FRAME_TABLE:                                     ;6210 寄存器(REG48-57)参数表
    ADD    PCL, A
    RETL   @48                                     ;REGISTER 48
    RETL   @0X98                                   ;DaTa HIGH BYTE
    RETL   @0X00                                   ;DaTa LOW BYTE

    RETL   @49
    RETL   @0XFF                                   ;VALUE_H
    RETL   @0X8F                                   ;VALUE_L

    RETL   @50
    RETL   @0X50          ;0X96
    RETL   @0X14          ;0X28;18

    RETL   @51
    RETL   @0X50          ;0X80
    RETL   @0X52          ;0X56

    RETL   @52          ;R52~R55 BE DEFINE BY THUNDER USERS
    RETL   @0X80          ;IT IS SYNC WORD, THE VaLUE MUST BE SaME aS

DONGLE
    RETL   @0X01

;RETL   @53
;RETL   @0XAA
;RETL   @0X55

    RETL   @54
    RETL   @0XB7
    RETL   @0X5C

    RETL   @55
    RETL   @0XD6
    RETL   @0X18

    RETL   @56
    RETL   @0X44          ;PKT_FLG/FIFO_FLG HIGNT aCTIVE
    RETL   @0X04          ;0X04          ;0XC4

```

```

RETL    @57                      ;REG57[13]=1
RETL    @0XE0                    ;0XC0          ;0XE0
RETL    @0X00                    ;0X80          ;0X00

```

```

RETL    @0XFF
RETL    @0XFF
RETL    @0XFF

```

```

RETL    @0XFF

```

```

;*****

```

```

;LOOKUP TABLE

```

```

;RF TRAnSCeIVER REGISTER INITIaL VaLUE (REG0~28)

```

```

;*****

```

```

RF_TABLE:

```

```

ADD     PCL, A
RETL    @9                      ;REGISTER9
RETL    @0X21
RETL    @0X03                    ;0X82

RETL    @0                      ;REGISTER 0
RETL    @0X35                    ;DaTa HIGH BYTE
RETL    @0X4D                    ;DaTa LOW BYTE

RETL    @2
RETL    @0X1E
RETL    @0X01

RETL    @4
RETL    @0XBC
RETL    @0XF0

RETL    @5
RETL    @0X00
RETL    @0XA1

RETL    @8
RETL    @0X80
RETL    @0X00

RETL    @10
RETL    @0X00
RETL    @0X04

```


RETL @11
RETL @0X40
RETL @0X41

RETL @12
RETL @0X80
RETL @0X00

RETL @13
RETL @0X00
RETL @0X00

RETL @14
RETL @0X16
RETL @0X9B;D

RETL @15
RETL @0X0D;90
RETL @0XED;AD

RETL @16
RETL @0XB0
RETL @0X00

RETL @18
RETL @0XE0
RETL @0X00

RETL @19
RETL @0XA1
RETL @0X14

RETL @20
RETL @0X81
RETL @0X95

RETL @21
RETL @0X69 ;0X69
RETL @0X62

RETL @22
RETL @0X00
RETL @0X02

```
RETL    @23
RETL    @0X00
RETL    @0X02
```

```
RETL    @24
RETL    @0XB1
RETL    @0X40
```

```
RETL    @25
RETL    @0XA8
RETL    @0X0F
```

```
RETL    @26
RETL    @0X3E
RETL    @0X07;4
```

```
RETL    @28
RETL    @0X58
RETL    @0X00
```

```
RETL    @0XFF
RETL    @0XFF
RETL    @0XFF
```

```
;*****
```

```
;LOOKUP TABLE
```

```
;RF TRaNSCEIVER CHaNNEL 40
```

```
;*****
```

```
;MaIN_LOOP
```

```
;*****
```

```
MAIN_LOOP:
```

```
    MOV    A, @0B11000111          ;PRESCaLER 1:256
```

```
    CONTW
```

```
    MOV    A, @0X02          ;@0XFC          ;MISO=1, SEaRCH KEY
```

```
    IOW   IOCD          ;ENaBLE MISO, P90 INTERNaL PULL-HIGH
```

```
FUNCTION
```

```
    MOV    A, @0X10;0;80          ;DISaBLE WDT FUNCTION
```

```
    IOW   IOCE
```

```
    MOV    A, @0XFF
```

```
    IOW   PORT5
```

```
    MOV    a, @0XFF
```

```
    MOV    PORT5, A
```

```

MOV    A, @0B00010001
IOW    PORT6
MOV    a, @0X00
MOV    PORT6, A
BS     PORT6, 2
BS     PORT6, 1
MOV    A, @100
CALL   DELAY_X1MS
MOV    A, @100
CALL   DELAY_X1MS
CLR    FLaG
BS     FLaG. 1
CLR    TIMEINC1
CLR    TIMEINC2
;
RF_RST:
BC     PRESET_N, RESET_N
CALL   DELAY_3US
BS     PRESET_N, RESET_N      ;RESET RF MODULE
REGISTER_INIT:
MOV    A, @1
CALL   DELAY_X1MS
CALL   FRAME_REG_INIT      ;初始化 RF 寄存器 R48-R57      ;88888;;
CALL   FRAME_REG_TEST      ;THE TEST PRO IS FOR PRODUCT
CALL   DELAY_3US
CALL   RF_REG_INIT         ;初始化 RF 寄存器 R0-R28
CALL   RF_REG_TEST         ;THE TEST PRO IS FOR PRODUCT
IDLE:
CALL   ENTER_IDLE_STATE
CALL   DELAY_3US
CALL   IDLE0
;*****
;进入主循环
;*****
MaIN_WaIT_KEY_RECIVER:
        CaLL   ENTER_RX_STaTE
;        CALL   DELAY_3US
;        CALL   DELAY_3US
        BC     FLAG. 1
        CLR    TIMEINC1
        CLR    TIMEINC2
RD_KEY_AND_RECIVER:
        BS     PORT6, 1

```

```

CALL    DELAY_3US
CALL    DELAY_3US
JBS     PPKT_FLG, PKT_FLG
JMP     $+7
nop
CALL    RECIVER_DATA
;nop
CALL    BACK_SEND_DATA
MOV     A, @4
CALL    DELAY_X1MS
BS      PORT6, 1
JBS     PORT5, 3
JMP     ENTER_TX_STATE
CALL    TIME_HAVE_10MS
JBC     FLAG, 1
JMP     INTO_IDLE
JMP     RD_KEY_AND_RECIVER

INTO_IDLE:
BC      FLAG. 2
CLR     TIMEINC1
CLR     TIMEINC2
CALL    ENTER_IDLE_STATE
CALL    DELAY_3US
CALL    IDLE0

INTO_IDLE_HAVE_KEY:
CALL    TIME_HAVE_100MS
JBC     FLAG. 2
JMP     MAIN_WAIT_KEY_RECIVER
NOP
JBC     PORT5, 3
JMP     INTO_IDLE_HAVE_KEY
BC      FLAG. 2
CLR     TIMEINC1
CLR     TIMEINC2
BC      FLAG. 1
JMP     MaIN_WaIT_KEY_RECIVER

TIME_HAVE_10MS:
INC     TIMEINC1
MOV     A, TIMEINC1
SUB     A, @0X8F
JBC     STATUS, Z

```

```
BS          FLAG, 1
RET
```

TIME_HAVE_100MS:

```
INC          TIMEINC1
MOV         A, TIMEINC1
SUB        A, @0xFF
JBC        STATUS, Z
INC          TIMEINC2

MOV         A, TIMEINC2
SUB        A, @0x3          ; ;A, #0x09
JBC        STATUS, Z
BS          FLAG, 2
RET
```

RECIVER_DATA:

```
BC          PORT6, 1
BC          PSPI_SS, SPI_SS
MOV         A, @80
OR          A, @READ_MASK
MOV         SPIWB, A
CALL        SIM_SPI_WRITE_AN
CALL        SIM_SPI_READ_AN
MOV         A, SPIRB
MOV         BYTECNT, A
XOR         A, @5          ; 8
JBS        STATUS, Z
JMP        ERR1
CALL        SIM_SPI_READ_AN
MOV         A, SPIRB
MOV         RX_BUF0, A
CALL        SIM_SPI_READ_AN
MOV         A, SPIRB
MOV         RX_BUF1, A
CALL        SIM_SPI_READ_AN
MOV         A, SPIRB
MOV         RX_BUF2, A
CALL        SIM_SPI_READ_AN
MOV         A, SPIRB
```

```

MOV RX_BUF3, A
BS PSPI_SS, SPI_SS
RET

```

ENTER_TX_STATE:

```

Bc PORT6, 2
JMP TX_STATE

```

BACK_SEND_DATA:

```

Bc PORT6, 2
BS FLAG. 0

```

TX_STATE:

```

CALL ENTER_IDLE_STATE
CALL DELAY_3US
CALL DELAY_3US
BC FLAG. 1
CLR TIMEINC1
CLR TIMEINC2
MOV A, @82 ;CLEAR FIFO WRITE POINT
MOV ADDR, A
MOV A, @0X80
MOV VALUE_H, A
MOV A, @0X00
MOV VALUE_L, A
CALL WRITE_SPI_REG

```

```

MOV A, @7 ;TX ON AND SELECT CHANNEL
MOV ADDR, A
MOV A, @0X01
MOV VALUE_H, A
MOV A, @0X05 ;0X04
MOV VALUE_L, A
CALL WRITE_SPI_REG

```

```

BC PSPI_SS, SPI_SS
MOV A, @80 ;所发送数据写入到 FIFO
MOV SPIWB, A

```

```

CALL SIM_SPI_WRITE_AN
MOV A, @5
MOV SPIWB, A

```

```
CALL SIM_SPI_WRITE_AN
```

```
MOV    A, @0X0E
MOV SPIWB, A
CALL SIM_SPI_WRITE_AN
```

```
MOV    A, @0X0F
MOV SPIWB, A
CALL SIM_SPI_WRITE_AN
MOV    A, @0X0F
MOV SPIWB, A
CALL SIM_SPI_WRITE_AN
MOV    A, @0X0F
MOV SPIWB, A
CALL SIM_SPI_WRITE_AN
```

```
BS     PSPI_SS, SPI_SS
```

```
MM:
```

```
CALL    DELAY_3US
JBS PPKT_FLG, PKT_FLG
JMP MM
NOP
```

```
; JBC FLAG. 0
; JMP RETURNMAIN
CALL RECEIVE_SPI
JMP WAIT_PKT_HI
```

```
RETURNMAIN:
```

```
BC     FLAG. 0
Bs PORT6, 2
RET
```

```
WAIT_PKT_HI:
```

```
MOV A, @200
MOV TEMPO, A
```

```
RD:
```

```
Bs PORT6, 2
CALL    DELAY_3US
CALL    DELAY_3US
JBS PPKT_FLG, PKT_FLG
JMP TIMEOUT
```

```
CALL RECIVER_DATA
JMP MAIN_WAIT_KEY_RECIVER
```

TIMOUT:

```
DJZ TEMPO
JMP RD
JMP MAIN_WAIT_KEY_RECIVER
```

RECEIVE_SPI:

```
MOV A, @82 ;CLEAR FIFO WRITE POINT
MOV ADDR, A
MOV A, @0X00
MOV VALUE_H, A
MOV A, @0X80
MOV VALUE_L, A
CALL WRITE_SPI_REG

MOV A, @7 ;RX ON AND SELECT CHANNEL
MOV ADDR, A
MOV A, @0X00
MOV VALUE_H, A
MOV A, @0X85
MOV VALUE_L, A
CALL WRITE_SPI_REG

MOV A, @64
MOV ADDR, A
CALL READ_SPI_REG

MOV A, TEMP1
XOR A, @0XD0 ;0B00011000
JBC STATUS, Z ;CHECK IF IS DISABLEPA3 STATUS
RET
JMP RF_RST
RET
```

ERR1:

```
BS PSPI_SS, SPI_SS
JMP MAIN_WAIT_KEY_RECIVER
```

ENTER_RX_STATE:


```

MOV     A, @82                ;CLEAR FIFO WRITE POINT
MOV     ADDR, A
MOV     A, @0X00
MOV     VALUE_H, A
MOV     A, @0X80
MOV     VALUE_L, A
CALL    WRITE_SPI_REG

MOV     A, @7                  ;RX ON AND SELECT CHANNEL
MOV     ADDR, A
MOV     A, @0X00
MOV     VALUE_H, A
MOV     A, @0X85
MOV     VALUE_L, A
CALL    WRITE_SPI_REG
NOP
RET

ENTER_IDLE_STATE:
MOV     A, @7
MOV     ADDR, A
MOV     A, @0X00
MOV     VALUE_H, A
MOV     VALUE_L, A           ;ENTER IDLE STATUS
CALL    WRITE_SPI_REG
RET

INTO_SLEEP:
CALL    ENTER_IDLE_STATE
CALL    DELAY_3US
;      CALL    IDLE0
BC      FLAG. 1
CLR     TIMEINC1
CLR     TIMEINC2

LOOP_KEY:
NOP
JBS     PORT5, 2
JMP     MAIN_WAIT_KEY_RECIVER
JBS     PORT5, 3
JMP     MAIN_WAIT_KEY_RECIVER
JMP     LOOP_KEY

```

;读 REG64, 检查是否进入 IDLE 状态

IDLE0:

MOV A, @64

MOV ADDR, A

CALL READ_SPI_REG

RRC TEMP1

RRC TEMP1

RRC TEMP1

RRC TEMP1

MOV A, @0X0F

AND A, TEMP1

XOR A, @0X0C ;0X0C

JBS STATUS, Z

JMP ERR1

NOP

RET

;SUBROUTINE PROGRAM

/*

INIT_GENERAL_REG:

MOV A, @0X10

MOV FSR, A

CLR_GENERAL_REG:

;WDTC ;CLEAR THE ALL USER'S RAM

CLR INDF

INC FSR

MOV A, FSR

JBC STATUS, Z ;等于零时结束

JMP INIT_GENERAL_REG_END

;AND A, @0X2F

XOR A, @0X3F

JBS STATUS, Z

JMP CLR_GENERAL_REG

;MOV A, @0X10

;ADD FSR, A

;JMP CLR_GENERAL_REG

INIT_GENERAL_REG_END:

RET

*/

;*****

FRAME_REG_INIT:

CLR TABLE_INDEX

CONFIG_FRAME:

;WDTC

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV ADDR, A

INC TABLE_INDEX

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV VALUE_H, A

INC TABLE_INDEX

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV VALUE_L, A

CALL WRITE_SPI_REG

INC TABLE_INDEX

MOV A, VALUE_L

XOR A, @0xFF

JBS STATUS, Z

;CHECK IF WRITE FINISH

JMP CONFIG_FRAME

;*****

;AFTER INITIAL FRAMER REGISTER RF STATUS WILL BE AUTOMATICALLY

;ENTER IDLE STATUS FROM SLEEP MODE WHEN WAITING CERTAIN TIMING

;*****

RF_ON:

;WDTC

;检查模组是否进入了 IDLE 状态

MOV A, @5

CALL DELAY_X1MS

MOV A, @64

MOV ADDR, A

CALL READ_SPI_REG

```

MOV    A, TEMP1
XOR    A, @0XC0                ;0B00011000
JBC    STATUS, Z                ;CHECK IF IS DISABLEPA3 STATUS
RET    ;JMP    RF_ON

```

```

MOV    A, @57
MOV    ADDR, A
MOV    A, @0XC0                ;CRC, SCRAMBLE IS ON
MOV    VALUE_H, A
MOV    A, @0X00
MOV    VALUE_L, A
CALL   WRITE_SPI_REG

```

```

MOV    A, @5
CALL   DELAY_X1MS

```

```

MOV    A, @64
MOV    ADDR, A
CALL   READ_SPI_REG

```

```

MOV    A, TEMP1
XOR    A, @0XC0
JBC    STATUS, Z                ;CHECK IF IS IDLE STATUS
RET

```

SPI_ERROR:

```

;WDT
; BC PORT5, 1
; BC PORT5, 3
; BC PORT5, 0
; BS PORT5, 2
;BS    PRIGHT_LED, RIGHT_LED
BC PORT6, 1                ;IF FRAMER REGISTER HAS ERROR, THEN LED FLASH
32HZ

```

MOV A, @200;128 ; 如果模组寄存器初始化失败，将给出错误信号指示灯

```

CALL   DELAY_X1MS
MOV    A, @200;128
CALL   DELAY_X1MS
; BC PORT5, 2
BS PORT6, 1
MOV    A, @50;16
CALL   DELAY_X1MS

```

JMP SPI_ERROR

FRAME_REG_TEST:

CLR TABLE_INDEX

FRAME_REG_TEST0:

;WDTC

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV ADDR, A

INC TABLE_INDEX

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV VALUE_H, A

INC TABLE_INDEX

MOV A, TABLE_INDEX

CALL FRAME_TABLE

MOV VALUE_L, A

CALL READ_SPI_REG

INC TABLE_INDEX

MOV A, VALUE_L

XOR A, @0xFF

JBC STATUS, Z

RET

MOV A, VALUE_H

SUB A, TEMP1

JBS STATUS, Z

JMP ERROR_FRAME

MOV A, VALUE_L

SUB A, TEMP2

JBC STATUS, Z

JMP FRAME_REG_TEST0

ERROR_FRAME:

; BS PUP_LED, UP_LED

; BC PORT6, 7

```

; BC PORT5, 1
; BC PORT5, 3
; BC PORT5, 0
; WDTC
; BS PLO_LED, LO_LED ; IF FRAMER REGISTER HAS ERROR,
THEN LED FLASH 32HZ
MOV A, @32
CALL DELAY_X1MS
; BC PUP_LED, UP_LED
MOV A, @32
CALL DELAY_X1MS
JMP ERROR_FRAME
RET

;*****
;RF MODULE REGISTER0~28 INITIAL
;*****
RF_REG_INIT:
CLR TABLE_INDEX
CONFIG_RF:
; WDTC
MOV A, TABLE_INDEX
CALL RF_TABLE
MOV ADDR, A

INC TABLE_INDEX
MOV A, TABLE_INDEX
CALL RF_TABLE
MOV VALUE_H, A

INC TABLE_INDEX
MOV A, TABLE_INDEX
CALL RF_TABLE
MOV VALUE_L, A

CALL WRITE_SPI_REG
INC TABLE_INDEX

MOV A, VALUE_L
XOR A, @0xFF
JBS STATUS, Z
JMP CONFIG_RF
RET

;*****

```

```
;TEST PROGRAM
;*****
RF_REG_TEST:
    CLR     TABLE_INDEX
RF_REG_TEST0:
    ;WDTC
    MOV     A, TABLE_INDEX
    CALL    RF_TABLE
    MOV     ADDR, A

    INC     TABLE_INDEX
    MOV     A, TABLE_INDEX
    CALL    RF_TABLE
    MOV     VALUE_H, A

    INC     TABLE_INDEX
    MOV     A, TABLE_INDEX
    CALL    RF_TABLE
    MOV     VALUE_L, A
    CALL    READ_SPI_REG
    INC     TABLE_INDEX

    MOV     A, VALUE_L
    XOR     A, @0xFF
    JBC     STATUS, Z
    RET

    MOV     A, VALUE_H
    SUB     A, TEMP1
    JBS     STATUS, Z
    JMP     ERROR_FRAME

    MOV     A, VALUE_L
    SUB     A, TEMP2
    JBS     STATUS, Z
    JMP     ERROR_FRAME
    JMP     RF_REG_TEST0

;*****
;R/W REGISTER SPI SUBROUTINE
;*****
WRITE_SPI_REG:
```

```
BC    PSPI_SS, SPI_SS
MOV   A, ADDR
MOV   SPIWB, A
CALL  SIM_SPI_WRITE_AN
```

```
MOV   A, ADDR
SUB   A, @0X1F
JBC   STATUS, C
CALL  DELAY_3US
;WDTC
MOV   A, VALUE_H
MOV   SPIWB, A
CALL  SIM_SPI_WRITE_AN
;WDTC
MOV   A, VALUE_L
MOV   SPIWB, A
CALL  SIM_SPI_WRITE_AN
```

```
BS    PSPI_SS, SPI_SS
RET
```

READ_SPI_REG:

```
BC    PSPI_SS, SPI_SS
MOV   A, ADDR
OR    A, @READ_MASK
MOV   SPIWB, A
CALL  SIM_SPI_WRITE_AN
```

```
MOV   A, ADDR
SUB   A, @0X1F
JBC   STATUS, C
CALL  DELAY_3US
;WDTC
CALL  SIM_SPI_READ_AN
MOV   A, SPIRB
MOV   TEMP1, A
;WDTC
CALL  SIM_SPI_READ_AN
MOV   A, SPIRB
MOV   TEMP2, A
```

```
BS    PSPI_SS, SPI_SS
RET
```



```

;*****
;THE FOLLOWING IS SPI COMMUNICATION USE I/O, NON 451 SPI
;INTERFACE, SPI IS READY DATA ON FALLING EDGE AND READ DATA
;ON RISING EDGE, 250KBPS BAUD RETE
;*****

```

```
SIM_SPI_WRITE_AN:
```

```

BC    PSPI_CLK, SPI_CLK
JMP   $+1
;BC   PSPI_CLK, SPI_CLK
JBS   SPIWB, 7
BC    PSPI_MOSI, SPI_MOSI
JBC   SPIWB, 7
BS    PSPI_MOSI, SPI_MOSI

BC    PSPI_CLK, SPI_CLK
JMP   $+1
BC    PSPI_CLK, SPI_CLK
JBS   SPIWB, 6
BC    PSPI_MOSI, SPI_MOSI
JBC   SPIWB, 6
BS    PSPI_MOSI, SPI_MOSI

BC    PSPI_CLK, SPI_CLK
JMP   $+1
BC    PSPI_CLK, SPI_CLK
JBS   SPIWB, 5
BC    PSPI_MOSI, SPI_MOSI
JBC   SPIWB, 5
BS    PSPI_MOSI, SPI_MOSI

BC    PSPI_CLK, SPI_CLK
JMP   $+1
BC    PSPI_CLK, SPI_CLK
JBS   SPIWB, 4
BC    PSPI_MOSI, SPI_MOSI
JBC   SPIWB, 4
BS    PSPI_MOSI, SPI_MOSI

BC    PSPI_CLK, SPI_CLK
JMP   $+1
BC    PSPI_CLK, SPI_CLK
JBS   SPIWB, 3

```

BC PSPI_MOSI, SPI_MOSI
 JBC SPIWB, 3
 BS PSPI_MOSI, SPI_MOSI

BS PSPI_CLK, SPI_CLK
 JMP \$+1
 BC PSPI_CLK, SPI_CLK
 JBS SPIWB, 2
 BC PSPI_MOSI, SPI_MOSI
 JBC SPIWB, 2
 BS PSPI_MOSI, SPI_MOSI

BS PSPI_CLK, SPI_CLK
 JMP \$+1
 BC PSPI_CLK, SPI_CLK
 JBS SPIWB, 1
 BC PSPI_MOSI, SPI_MOSI
 JBC SPIWB, 1
 BS PSPI_MOSI, SPI_MOSI

BS PSPI_CLK, SPI_CLK
 JMP \$+1
 BC PSPI_CLK, SPI_CLK
 JBS SPIWB, 0
 BC PSPI_MOSI, SPI_MOSI
 JBC SPIWB, 0
 BS PSPI_MOSI, SPI_MOSI

BS PSPI_CLK, SPI_CLK
 BC PSPI_MOSI, SPI_MOSI
 BC PSPI_CLK, SPI_CLK
 RET

SIM_SPI_READ_AN:

MOV A, @0xFF
 MOV SPIRB, A

BC PSPI_CLK, SPI_CLK
 JMP \$+1
 BS PSPI_CLK, SPI_CLK
 JBS PSPI_MISO, SPI_MISO
 BC SPIRB, 7

BC PSPI_CLK, SPI_CLK

JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 6
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 5
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 4
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 3
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 2
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 1
BC	PSPI_CLK, SPI_CLK
JMP	\$+1
BS	PSPI_CLK, SPI_CLK
JBS	PSPI_MISO, SPI_MISO
BC	SPIRB, 0
BC	PSPI_CLK, SPI_CLK
RET	

```

;*****
;TIMERBASE PROGRAM
;*****
DELAY_3US:
    JMP    $+1            ;0.5US
    JMP    $+1            ;1US
    JMP    $+1            ;2US
    JMP    $+1
    NOP
    RET

;*****
DELAY_X1MS:
    MOV    CNT1, A
DELAY_1MS:
    MOV    A, @250        ;250*4US=1000US
DELAY_X4US:
    MOV    CNT2, A
DELAY_1MS0:
    NOP
    JMP    $+1
    JMP    $+1
    JMP    $+1
    JMP    $+1
    JMP    $+1
    JMP    $+1
    DJZ    CNT2
    JMP    DELAY_1MS0
    DJZ    CNT1
    JMP    DELAY_1MS
    RET

;*****
DELAY_X100MS:
    MOV    TEMP3, A
DELAY_X1S0:
    MOV    A, @100        ;100*1MS=100MS
    MOV    CNT1, A
DELAY_X1S1:
    MOV    A, @250        ;250*4US=1000US
    MOV    CNT2, A
DELAY_X1S2:
    NOP

```

```
JMP    $+1
JMP    $+1
JMP    $+1
JMP    $+1
JMP    $+1
JMP    $+1
DJZ    CNT2
JMP    DELAY_X1S2
DJZ    CNT1
JMP    DELAY_X1S1
DJZ    TEMP3
JMP    DELAY_X1S0
RET
```

```
*****
;
;   RESET VECTOR
;*****
;
;   ORG 0X3FF
;   JMP RESET
;
;*****
;
;   END OF PROGRAM
;*****

EOP
END
```

安阳市新世纪电子研究所有限公司

地 址：中国.河南省安阳市西环城路南 1 号

电 话：86(0372)5968708 5968993

传 真：86(0372)5968993-803

网 址：www.ayxsj.com

市 场 部：

电 话：0372--5968708-801 0372-5968993-801

E-mail：ay5968708@163.com

技 术 部：

电 话：0372--5968708-802 0372-5968993-802

E-mail：ay5968993@163.com

深圳经销商

华强电子世界

深圳圣世佳禾电子公司

地址：深圳市华强电子世界三号楼一楼 A413 柜

电话：0755-83759042 手机：13724309618

联系人：王经理

北京经销商

北京知春电子城

地址：北京市知春电子城二楼 C009 柜台

电话：010-62637297

联系人：杨 飞

郑州经销商

郑州中州商场

郑州新星电子科技

地址：郑州市中州商场三楼东区 17 号

电话：0371-66961968 手机：13503815523

联系人：李经理