单片机实验报告

--RTC时钟修改显示

小组成员

17343031 辜宇然

17343075 刘皓铧

17343050 黄昱珲

17343092 潘鹏程

1. 实验原理：

在STC开发板上有时钟芯片RTC，通过keyboard输入通过I2C总线读入数据，通过中断设置芯片时间

1. 例程代码重要部分解释：

读取RTC函数

void ReadRTC(void)

{

u8 tmp[3];

ReadNbyte(2, tmp, 3);

second = ((tmp[0] >> 4) & 0x07) \* 10 + (tmp[0] & 0x0f);

minute = ((tmp[1] >> 4) & 0x07) \* 10 + (tmp[1] & 0x0f);

hour = ((tmp[2] >> 4) & 0x03) \* 10 + (tmp[2] & 0x0f);

}

写入RTC函数

void WriteRTC(void)

{

u8 tmp[3];

tmp[0] = ((second / 10) << 4) + (second % 10);

tmp[1] = ((minute / 10) << 4) + (minute % 10);

tmp[2] = ((hour / 10) << 4) + (hour % 10);

WriteNbyte(2, tmp, 3);

}

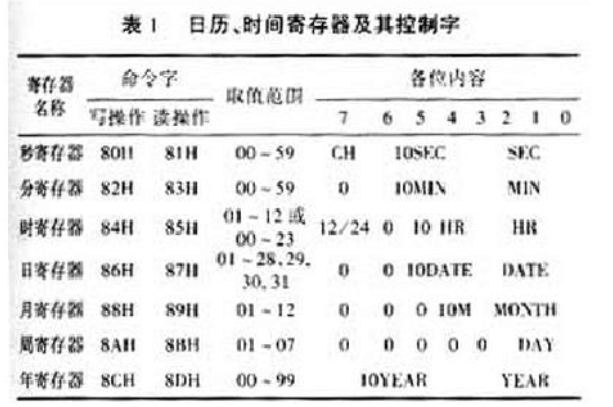
解释：

单片机通过这两个函数控制RTC芯片中时间变量

tmp[3]表示时分秒三个变量位

Hour minute second 分别表示三个RTC变量

这里需要说明，tmp是一个专门服务于RTC芯片时间显示的寄存器组，一共有7个：



Tmp[0]控制秒，由于second变量前四位是秒钟十位，后四位是秒钟个位，由于tmp遵循高低四位设置方式，故写为tmp[0] = ((second / 10) << 4) + (second % 10) ，下同

tmp[1]控制分，tmp[2]控制时，tmp[3]控制日，tmp[4]控制月，tmp[6]控制年，这些都与上述一致，但tmp[5]是周寄存器，专门显示星期，我们此次不用她，故置为0

再解释一下，由于second最大值为60，所以高四位最多使用位数为三位，故second = ((tmp[0] >> 4) & 0x07) \* 10 + (tmp[0] & 0x0f);

所以为了达到预置年月日的效果：

首先在本地变量区定义RTC变量

u8 year,month,day

修改上述函数为：

void ReadRTC(void)

{

u8 tmp[7];

ReadNbyte(2, tmp, 7);

second = ((tmp[0] >> 4) & 0x07) \* 10 + (tmp[0] & 0x0f);

minute = ((tmp[1] >> 4) & 0x07) \* 10 + (tmp[1] & 0x0f);

hour = ((tmp[2] >> 4) & 0x03) \* 10 + (tmp[2] & 0x0f);

day = ((tmp[3] >> 4) & 0x07） \* 10 + (tmp[3] & 0x0f);

month = ((tmp[4] >> 4) & 0x01) \* 10 + (tmp[4] & 0x0f);

year = ((tmp[6] >> 4) & 0x0f) \* 10 + (tmp[6] & 0x0f);

}

写入RTC函数

void WriteRTC(void)

{

u8 tmp[7];

tmp[0] = ((second / 10) << 4) + (second % 10);

tmp[1] = ((minute / 10) << 4) + (minute % 10);

tmp[2] = ((hour / 10) << 4) + (hour % 10);

tmp[3] = ((day / 10) << 4) + (day % 10)

tmp[4] = 0x80 || ((month / 10) << 4) + (month % 10);

tmp[5] = 0;

tmp[6] = ((year / 10) << 4) + (year % 10);

WriteNbyte(2, tmp, 7);

}

此处将tmp[4]最高位置1后可自行设置rtc芯片世纪位

所以为了达到两种显示的效果：

注释原来的Display函数并将其用Display1(显示时分秒)和Display2(显示年月日)部分代替

修改按键操作其显示

1. 代码

由前述，代码为：

/\*\*\*\*\*\*\*\*\*\*\*\*\* 本程序功能说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*

读写RTC, IC为PCF8563.

用STC的MCU的IO方式控制74HC595驱动8位数码管。

用户可以修改宏来选择时钟频率.

使用Timer0的16位自动重装来产生1ms节拍,程序运行于这个节拍下, 用户修改MCU主时钟频率时,自动定时于1ms.

用户可以在显示函数里修改成共阴或共阳.推荐尽量使用共阴数码管.

8位数码管显示时间(小时-分钟-秒)/(年月日).

行列扫描按键键码为17~32.

按键只支持单键按下, 不支持多键同时按下, 那样将会有不可预知的结果.

键按下超过1秒后,将以10键/秒的速度提供重键输出. 用户只需要检测KeyCode是否非0来判断键是否按下.

调整时间键:

键码 13：切换时钟显示模式.

键码 14：年份+.

键码 15：月份+.

键码 16：日期+.

键码17: 小时+.

键码18: 小时-.

键码19: 分钟+.

键码20: 分钟-.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAIN\_Fosc 22118400L //定义主时钟

#include "STC15Fxxxx.H"

#include "config.h"

#include "USART.h"

#include "delay.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define DIS\_DOT 0x20

#define DIS\_BLACK 0x10

#define DIS\_ 0x11

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 用户定义宏 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define Timer0\_Reload (65536UL -(MAIN\_Fosc / 1000)) //Timer 0 中断频率, 1000次/秒

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\* 本地常量声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 code t\_display[]={ //标准字库

// 0 1 2 3 4 5 6 7 8 9 A B C D E F

0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71,

//black - H J K L N o P U t G Q r M y

0x00,0x40,0x76,0x1E,0x70,0x38,0x37,0x5C,0x73,0x3E,0x78,0x3d,0x67,0x50,0x37,0x6e,

0xBF,0x86,0xDB,0xCF,0xE6,0xED,0xFD,0x87,0xFF,0xEF,0x46}; //0. 1. 2. 3. 4. 5. 6. 7. 8. 9. -1

u8 code T\_COM[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80}; //位码

/\*\*\*\*\*\*\*\*\*\*\*\*\* IO口定义 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sbit P\_HC595\_SER = P4^0; //pin 14 SER data input

sbit P\_HC595\_RCLK = P5^4; //pin 12 RCLk store (latch) clock

sbit P\_HC595\_SRCLK = P4^3; //pin 11 SRCLK Shift data clock

/\*\*\*\*\*\*\*\*\*\*\*\*\* 本地变量声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 LED8[8]; //显示缓冲

u8 display\_index; //显示位索引

bit B\_1ms; //1ms标志

u8 IO\_KeyState, IO\_KeyState1, IO\_KeyHoldCnt; //行列键盘变量

u8 KeyHoldCnt; //键按下计时

u8 KeyCode; //给用户使用的键码, 1~16有效

u8 cnt50ms;

bit flag; //时钟显示标志

u8 hour,minute,second; //RTC变量

u16 msecond;

/\*\*\*\*\*\*\*\*\*\*\*\*\* 本地函数声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CalculateAdcKey(u16 adc);

void IO\_KeyScan(void); //50ms call

void WriteNbyte(u8 addr, u8 \*p, u8 number);

void ReadNbyte( u8 addr, u8 \*p, u8 number);

void DisplayRTC(void);

void ReadRTC(void);

void WriteRTC(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 外部函数声明和外部变量声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UART\_config(void)

{

COMx\_InitDefine COMx\_InitStructure; //结构定义

COMx\_InitStructure.UART\_Mode = UART\_8bit\_BRTx; //模式, UART\_ShiftRight,UART\_8bit\_BRTx,UART\_9bit,UART\_9bit\_BRTx

COMx\_InitStructure.UART\_BRT\_Use = BRT\_Timer1; //使用波特率, BRT\_Timer1, BRT\_Timer2 (注意: 串口2固定使用BRT\_Timer2)

COMx\_InitStructure.UART\_BaudRate = 115200ul; //波特率, 一般 110 ~ 115200

COMx\_InitStructure.UART\_RxEnable = ENABLE; //接收允许, ENABLE或DISABLE

COMx\_InitStructure.BaudRateDouble = DISABLE; //波特率加倍, ENABLE或DISABLE

COMx\_InitStructure.UART\_Interrupt = ENABLE; //中断允许, ENABLE或DISABLE

COMx\_InitStructure.UART\_Polity = PolityLow; //中断优先级, PolityLow,PolityHigh

COMx\_InitStructure.UART\_P\_SW = UART1\_SW\_P30\_P31; //切换端口, UART1\_SW\_P30\_P31,UART1\_SW\_P36\_P37,UART1\_SW\_P16\_P17(必须使用内部时钟)

COMx\_InitStructure.UART\_RXD\_TXD\_Short = DISABLE; //内部短路RXD与TXD, 做中继, ENABLE,DISABLE

USART\_Configuration(USART1, &COMx\_InitStructure); //初始化串口1 USART1,USART2

COMx\_InitStructure.UART\_Mode = UART\_8bit\_BRTx; //模式, UART\_ShiftRight,UART\_8bit\_BRTx,UART\_9bit,UART\_9bit\_BRTx

COMx\_InitStructure.UART\_BaudRate = 57600ul; //波特率, 110 ~ 115200

COMx\_InitStructure.UART\_RxEnable = ENABLE; //接收允许, ENABLE或DISABLE

COMx\_InitStructure.UART\_Interrupt = ENABLE; //中断允许, ENABLE或DISABLE

COMx\_InitStructure.UART\_Polity = PolityLow; //中断优先级, PolityLow,PolityHigh

COMx\_InitStructure.UART\_P\_SW = UART2\_SW\_P10\_P11; //切换端口, UART2\_SW\_P10\_P11,UART2\_SW\_P46\_P47

USART\_Configuration(USART2, &COMx\_InitStructure); //初始化串口2 USART1,USART2

PrintString1("STC15F2K60S2 UART1 Test Prgramme!\r\n"); //SUART1发送一个字符串

PrintString2("STC15F2K60S2 UART2 Test Prgramme!\r\n"); //SUART2发送一个字符串

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

u8 j;

u8 i;

UART\_config();

EA = 1;

flag = 0;

P0M1 = 0; P0M0 = 0; //设置为准双向口

P1M1 = 0; P1M0 = 0; //设置为准双向口

P2M1 = 0; P2M0 = 0; //设置为准双向口

P3M1 = 0; P3M0 = 0; //设置为准双向口

P4M1 = 0; P4M0 = 0; //设置为准双向口

P5M1 = 0; P5M0 = 0; //设置为准双向口

P6M1 = 0; P6M0 = 0; //设置为准双向口

P7M1 = 0; P7M0 = 0; //设置为准双向口

display\_index = 0;

AUXR = 0x80; //Timer0 set as 1T, 16 bits timer auto-reload,

TH0 = (u8)(Timer0\_Reload / 256);

TL0 = (u8)(Timer0\_Reload % 256);

ET0 = 1; //Timer0 interrupt enable

TR0 = 1; //Tiner0 run

EA = 1; //打开总中断

for(i=0; i<8; i++) LED8[i] = 0x10; //上电消隐

ReadRTC();

F0 = 0;

if(year > 99) F0 = 1;

if(month >= 13) F0 = 1;

if(day >= 32 && (month == 1 || mobth == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12)) F0 = 1;

if (day >= 31 && (month == 4 || mobth == 6 || month == 9 || month == 11)) F0 = 1;

if (day >= 29 && month == 2 && !((year % 400) == 0 || (year % 4 == 0 && (year % 100) != 0))) F0 = 1;

if (day >= 30 && month == 2 && ((year % 400) == 0 || (year % 4 == 0 && (year % 100) != 0))) F0 = 1;

if(second >= 60) F0 = 1; //错误

if(minute >= 60) F0 = 1; //错误

if(hour >= 24) F0 = 1; //错误

if(F0) //有错误, 默认12:00:00

{

second = 0;

minute = 0;

hour = 12;

WriteRTC();

}

DisplayRTC();

LED8[2] = DIS\_;

LED8[5] = DIS\_;

KeyHoldCnt = 0; //键按下计时

KeyCode = 0; //给用户使用的键码, 1~16有效

IO\_KeyState = 0;

IO\_KeyState1 = 0;

IO\_KeyHoldCnt = 0;

cnt50ms = 0;

while(1)

{

if(B\_1ms) //1ms到

{

B\_1ms = 0;

if(++msecond >= 1000) //1秒到

{

msecond = 0;

ReadRTC();

DisplayRTC();

}

if(++cnt50ms >= 50) //50ms扫描一次行列键盘

{

cnt50ms = 0;

IO\_KeyScan();

}

if(KeyCode != 0) //有键按下

{

if (KeyCode == 13) //切换显示模式

{

flag = ~flag;

WriteRTC();

DisplayRTC();

}

if (KeyCode == 14)

{

if (++year >= 99) year = 1;

{

WriteRTC();

DisplayRTC();

}

}

if (KeyCode == 15)

{

if (++month >= 13) month = 1;

{

WriteRTC();

DisplayRTC();

}

}

if (KeyCode == 16)

{

if (++day >= 32 && (month == 1 || mobth == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12)) day = 1;

else if (++day >= 31 && (month == 4 || mobth == 6 || month == 9 || month == 11)) day = 1;

else if (++day >= 29 && month == 2 && !((year % 400) == 0 || (year % 4 == 0 && (year % 100) != 0))) day = 1;

else if (++day >= 30 && month == 2 && ((year % 400) == 0 || (year % 4 == 0 && (year % 100) != 0))) day = 1;

{

WriteRTC();

DisplayRTC();

}

}

if(KeyCode == 17) //hour +1

{

if(++hour >= 24) hour = 0;

WriteRTC();

DisplayRTC();

}

if(KeyCode == 18) //hour -1

{

if(--hour >= 24) hour = 23;

WriteRTC();

DisplayRTC();

}

if(KeyCode == 19) //minute +1

{

second = 0;

if(++minute >= 60) minute = 0;

WriteRTC();

DisplayRTC();

}

if(KeyCode == 20) //minute -1

{

second = 0;

if(--minute >= 60) minute = 59;

WriteRTC();

DisplayRTC();

}

KeyCode = 0;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 显示时钟函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayRTC(void)

{

if (flag) {

if (hour >= 10) LED8[0] = hour / 10;

else LED8[0] = DIS\_BLACK;

LED8[1] = hour % 10;

LED8[2] = DIS\_;

LED8[3] = minute / 10;

LED8[4] = minute % 10;

LED8[6] = second / 10;

LED8[7] = second % 10;

}

else {

//世纪位自行预设

LED8[0] = 2;

LED8[1] = 0;

LED8[2] = year / 10;

LED8[3] = year % 10；

LED8[4] = month / 10;

LED8[5] = month % 10;

LED8[6] = day / 10;

LED8[7] = day % 10;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 读RTC函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadRTC(void)

{

u8 tmp[3];

ReadNbyte(2, tmp, 3);

second = ((tmp[0] >> 4) & 0x07) \* 10 + (tmp[0] & 0x0f);

minute = ((tmp[1] >> 4) & 0x07) \* 10 + (tmp[1] & 0x0f);

hour = ((tmp[2] >> 4) & 0x03) \* 10 + (tmp[2] & 0x0f);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 写RTC函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteRTC(void)

{

u8 tmp[3];

tmp[0] = ((second / 10) << 4) + (second % 10);

tmp[1] = ((minute / 10) << 4) + (minute % 10);

tmp[2] = ((hour / 10) << 4) + (hour % 10);

WriteNbyte(2, tmp, 3);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

行列键扫描程序

使用XY查找4x4键的方法，只能单键，速度快

Y P04 P05 P06 P07

| | | |

X | | | |

P00 ---- K00 ---- K01 ---- K02 ---- K03 ----

| | | |

P01 ---- K04 ---- K05 ---- K06 ---- K07 ----

| | | |

P02 ---- K08 ---- K09 ---- K10 ---- K11 ----

| | | |

P03 ---- K12 ---- K13 ---- K14 ---- K15 ----

| | | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 code T\_KeyTable[16] = {0,1,2,0,3,0,0,0,4,0,0,0,0,0,0,0};

void IO\_KeyDelay(void)

{

u8 i;

i = 60;

while(--i) ;

}

void IO\_KeyScan(void) //50ms call

{

u8 j;

j = IO\_KeyState1; //保存上一次状态

P0 = 0xf0; //X低，读Y

IO\_KeyDelay();

IO\_KeyState1 = P0 & 0xf0;

P0 = 0x0f; //Y低，读X

IO\_KeyDelay();

IO\_KeyState1 |= (P0 & 0x0f);

IO\_KeyState1 ^= 0xff; //取反

if(j == IO\_KeyState1) //连续两次读相等

{

j = IO\_KeyState;

IO\_KeyState = IO\_KeyState1;

if(IO\_KeyState != 0) //有键按下

{

F0 = 0;

if(j == 0) F0 = 1; //第一次按下

else if(j == IO\_KeyState)

{

if(++IO\_KeyHoldCnt >= 20) //1秒后重键

{

IO\_KeyHoldCnt = 18;

F0 = 1;

}

}

if(F0)

{

j = T\_KeyTable[IO\_KeyState >> 4];

if((j != 0) && (T\_KeyTable[IO\_KeyState& 0x0f] != 0))

KeyCode = (j - 1) \* 4 + T\_KeyTable[IO\_KeyState & 0x0f] + 16; //计算键码，17~32

}

}

else IO\_KeyHoldCnt = 0;

}

P0 = 0xff;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 向HC595发送一个字节函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Send\_595(u8 dat)

{

u8 i;

for(i=0; i<8; i++)

{

dat <<= 1;

P\_HC595\_SER = CY;

P\_HC595\_SRCLK = 1;

P\_HC595\_SRCLK = 0;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 显示扫描函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DisplayScan(void)

{

Send\_595(~T\_COM[display\_index]); //输出位码

Send\_595(t\_display[LED8[display\_index]]); //输出段码

P\_HC595\_RCLK = 1;

P\_HC595\_RCLK = 0; //锁存输出数据

if(++display\_index >= 8) display\_index = 0; //8位结束回0

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Timer0 1ms中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void timer0 (void) interrupt TIMER0\_VECTOR

{

DisplayScan(); //1ms扫描显示一位

B\_1ms = 1; //1ms标志

}

#define SLAW 0xA2

#define SLAR 0xA3

sbit SDA = P1^1; //定义SDA PIN5

sbit SCL = P1^0; //定义SCL PIN6

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2C\_Delay(void) //for normal MCS51, delay (2 \* dly + 4) T, for STC12Cxxxx delay (4 \* dly + 10) T

{

u8 dly;

dly = MAIN\_Fosc / 2000000UL; //按2us计算

while(--dly) ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2C\_Start(void) //start the I2C, SDA High-to-low when SCL is high

{

SDA = 1;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

SDA = 0;

I2C\_Delay();

SCL = 0;

I2C\_Delay();

}

void I2C\_Stop(void) //STOP the I2C, SDA Low-to-high when SCL is high

{

SDA = 0;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

SDA = 1;

I2C\_Delay();

}

void S\_ACK(void) //Send ACK (LOW)

{

SDA = 0;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

SCL = 0;

I2C\_Delay();

}

void S\_NoACK(void) //Send No ACK (High)

{

SDA = 1;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

SCL = 0;

I2C\_Delay();

}

void I2C\_Check\_ACK(void) //Check ACK, If F0=0, then right, if F0=1, then error

{

SDA = 1;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

F0 = SDA;

SCL = 0;

I2C\_Delay();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2C\_WriteAbyte(u8 dat) //write a byte to I2C

{

u8 i;

i = 8;

do

{

if(dat & 0x80) SDA = 1;

else SDA = 0;

dat <<= 1;

I2C\_Delay();

SCL = 1;

I2C\_Delay();

SCL = 0;

I2C\_Delay();

}

while(--i);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 I2C\_ReadAbyte(void) //read A byte from I2C

{

u8 i,dat;

i = 8;

SDA = 1;

do

{

SCL = 1;

I2C\_Delay();

dat <<= 1;

if(SDA) dat++;

SCL = 0;

I2C\_Delay();

}

while(--i);

return(dat);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteNbyte(u8 addr, u8 \*p, u8 number) /\* WordAddress,First Data Address,Byte lenth \*/

//F0=0,right, F0=1,error

{

I2C\_Start();

I2C\_WriteAbyte(SLAW);

I2C\_Check\_ACK();

if(!F0)

{

I2C\_WriteAbyte(addr);

I2C\_Check\_ACK();

if(!F0)

{

do

{

I2C\_WriteAbyte(\*p); p++;

I2C\_Check\_ACK();

if(F0) break;

}

while(--number);

}

}

I2C\_Stop();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadNbyte(u8 addr, u8 \*p, u8 number) /\* WordAddress,First Data Address,Byte lenth \*/

{

I2C\_Start();

I2C\_WriteAbyte(SLAW);

I2C\_Check\_ACK();

if(!F0)

{

I2C\_WriteAbyte(addr);

I2C\_Check\_ACK();

if(!F0)

{

I2C\_Start();

I2C\_WriteAbyte(SLAR);

I2C\_Check\_ACK();

if(!F0)

{

do

{

\*p = I2C\_ReadAbyte(); p++;

if(number != 1) S\_ACK(); //send ACK

}

while(--number);

S\_NoACK(); //send no ACK

}

}

}

I2C\_Stop();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

原理图：

RTC构造：

