uc/os 实验设计 - 万年历

**一、 实验要求:**

 1、基于uc/os II 操作系统在三星S3C2440开发板进行设计

 2、实时显示:不断显示当前的年月日、时间信息;

 3、时间设置:可设置万年历的时钟和年月日信息,完成设置后,可更新

显示。

**二、 实验设计:**

 1. 模块划分 首先要进行模块划分。在硬件上,使用 LED 数码管进行时间的显示,使

用四个按钮进行时间和闹钟设置,时间基准使用开发板上已集成的 RTC 模块。 由此,可以把编程分为几个模块:

* 1)  RTC 设置与读取:用于读取和设置时间;
* 2)  数码管显示:用于显示当前时间;
* 3)  按钮设置:用于捕获按钮活动;
* 4)  顶层模块:连接所有其他模块。

**三、 实验内容**

1.显示时间

查S3C2440开发板manual和自带例程RTC.mcp可知年月日时分秒都放在如下寄存器中，每次时间变化遍历更新这些寄存器。

while (1) { while (1) {

year = 0 x2000 + rBCDYEAR ; month = rBCDMON ; weekday = rBCDDAY ; date = rBCDDATE ;

hour = rBCDHOUR ; min = rBCDMIN ; sec = rBCDSEC ; If ( sec != tmp )

{ tmp = sec ;

break ; }

} // time updated

// print out current time }

 上面采用的是阻塞查询方式来获知时间更新的时刻,更好的是使用 S3C2440 的 RTC 的功能:TickTimeInterrupt。这里把 TICNT 设为 0x 即可实现每秒一次中断请求:

pISR\_TICK = ( unsigned ) Rtc\_Tick ;

rTICNT = (1 < <7) + 127; rINTMSK = ~( BIT\_TICK ) ; void \_\_irq Rtc\_Tick ( void ) {

rSRCPND = BIT\_TICK ; rINTPND = BIT\_TICK ; rINTPND ; Uart\_Printf ( " %03 d sec " , sec\_tick ++) ;

}

2.设置时间

rRTCCON = rRTCCON & ~(0 xf ) | 0 x1 ; // RTC控制使能

 rBCDYEAR = (( syear /10) < <4) +( syear %10) ;

 rBCDMON = (( smonth /10) < <4) +( smonth %10) ;

rBCDDAY = sday ; // SUN :1 MON :2 TUE :3 WED :4 THU :5 FRI :6 SAT :7

rBCDDATE = sdate ; rBCDHOUR = shour ; rBCDMIN = smin ;

rBCDSEC = ssec ; rRTCCON = 0 x0 ;

这样就可以完成 RTC 时间的读取和设置了。

3. LED数码管显示

  使用 8 个七段数码管来显示当前的时间。 初始化 IIC 接口就可以更直接地控制数码管了。

如下:

void xzSetSeg ( U8 pos , U8 seg [8]) {

U8 i ; IIC\_Init () ; for ( i =0; i <8; i ++) {

if (( pos & (1 < <(7 - i ) ) ) != 0) { \_WrIIC (0 x70 ,0 x17 -i , DTab [ seg [ i ]]) ;

} }

IIC\_Restore () ; }

上面的函数可以进行数码管显示设置,pos 的每一位指示刷新哪些个位 置,seg 里则是刷新为哪个数字。这样就可以操作数码管进行时间显示了。

 4. 按钮设置

,我们用上图四个按钮实现对时间和闹钟的设置。

 可以看到上图四个按钮都可以产生外部中断EINT0，EINT2,EINT11，EINT19,只要配置好相应地控制寄存器值把按钮设置为下降沿中断,在中断处理程序

中进行进一步地处理。举例设置SW1的中断如下:

rGPFCON = (( rGPFCON & 0 xff00 ) | (1 < <1) ); // 中断0使能

rEXTINT0 |= (0 x2 < <0) ; // 设置为下降沿触发

pISR\_EINT0 =( U32 ) Eint0Int ; // ISR entry

rINTMSK &=~(1 < <0) ;

 static void \_\_irq Eint0Int ( void ) {//中断0的服务程序

OSIntEnter () ;

ClearPending ( BIT\_EINT0 ) ;

 OSIntExit () ;

}

5. 顶层模块

顶层模块设计了三个=个任务： Task \_Display:更新数码管显示; Task \_RTC:时间更新 Task \_Proc Key:处理按键; 并且创建了两个信号量: Sem \_for \_update:时间更新。 Sem \_for \_Alarm:闹钟响。 用于等待把新时间显示到数码管的任务如下:

void Task\_Display ( void \* pdata ) {

U8 Reply =0; while (1) {

OSSemPend ( Sem\_for\_update , 0 , & Reply ) ; // 得到(占有)sem1

}

xzSetSeg ( Pos , SegData ) ;//更新LED数码管显示

OSTimeDly (3) ; }

}

当时间更新时,会释放 Sem \_for\_ update,当这个任务得到更新信号时就 更新显示。

等待按键如下:

void Task\_Proc\_Key ( void \* pdata ) { U8 key0 , key1 , key2 , key3 ; U8 func =0; U8 p1 , p2 , dat ;

U8 flag =0; while (1) {

if (( rGPFDAT & 0 x01 ) !=0) { key0 =0;} else { key0 =1;} if (( rGPFDAT & 0 x04 ) !=0) { key1 =0;} else { key1 =1;} If (( rGPGDAT & 0 x08 ) !=0) { key2 =0;} else { key2 =1;} if (( rGPGDAT & (0 x01 < <11) ) !=0) { key3 =0;} else { key3 =1;} if ( key0 || key1 || key2 || key3 ) {

// (key0 ==1)说明有键按下

 {

xz\_Rtc\_TimeSet ( func ,1 , p1 , p2 ) ; } //设置时间

} else if ( key1 ==1) {

if ( flag ==1) { if ( func ==0) Alarm\_hour - -; if ( func ==1) Alarm\_min - -;

}

else { // set time xz\_Rtc\_TimeSet ( func ,2 , p1 , p2 ) ;

} }

else if ( key2 ==1) { func ++;

func = func %3;

Uart\_Printf ( " Set :% d \ n " , func ) ; }

else if ( key3 ==1) { if ( flag ==0) {

flag =1;

}

OSTimeDly (15) ; }

else { // no key down

OSTimeDly (15) ; }

} }

时间更新任务:

void Task\_RTC ( void \* pdata ) {

while (1) { Display\_Rtc () ; }

} //一直执行

其中的函数,Display \_Rtc 里,每次时间更新都会刷新显示, 如下:

void Display\_Rtc ( void ) { while (1) {

while (1) { get current time

if ( time updated ) { break ;

}

} //时间更新

 OSSemPost ( Sem\_for\_update ) ; //释放信号量给数码管显示任务

} }

//////////////// end ///////////////