```
//*****************************************************************************
**********************
// University of Illinois at Chicago, Dept. of Electrical and Computer Engineering
// ECE 367 -Microprocessor-Based Design
// Semester: Spring 2013

// Experiment Title: Real Time Clock
// Experiment Description: This experiment is for a real time clock and the extra
credit stop-watch.  The
//                            clock is capable of 12 or 24 hour mode and can be
reset at any time.  The
// Date: 4/20/2013
// Updated: 4/20/2013
// Version: 1
// Programmer: Mitchell Hedditch
// Lab Session: Tuesday 8AM-10:50AM
//*****************************************************************************
**********************
/* Some include (header) files needed by Codewarrior with machine info for the
NanoCore12                */

#include <hidef.h>                                          /* common defines and
macros                  */
#include "derivative.h"                                     /* derivative-specific
definitions           */

/* Here we give function prototypes before we start on the main code */
extern void near tch0ISR(void);                            // tch0ISR() prototype
extern void near tch1ISR(void);                             // tch1ISR() prototype

#pragma CODE_SEG __NEAR_SEG NON_BANKED                       // required pragma

#pragma CODE_SEG DEFAULT                                    // required pragma


typedef void (*near tIsrFunc) (void);                        // required typedef

const tIsrFunc _vect[] @0xFFEC = {                           // vector array setup

     tch1ISR,                                               // 0xFFEC timer Ch1

     tch0ISR,                                                // 0xFFEE timer Ch0

};


/* We need to define some constants. Similar to EQU's in assembly
                    */
#define     IOREGS_BASE  0x0000

#define     _IO8(off)    *(unsigned char  volatile *)(IOREGS_BASE + off) //define
form prototype 8-bit
#define     _IO16(off)   *(unsigned short volatile *)(IOREGS_BASE + off) //define
form prototype 16-bit


//#define   PORTT     _IO8(0x240)
/* portT data register is unsigned 8-bit at address $0240
                    */
/* because of the form prototype defines above this is the same as
                    */

/* #define PORTT *(unsigned char  volatile *) (0x240);    Means PORTT points to
```

```
address $0240            */
/* the statement PORTT = 0x34; means to store $34 at location $0240
                  */
/* if the contents of PORTT is 0xd3 then the assignment x = PORTT; means x is now
equal to 0xd3          */
/*************************************************************************
**********************/
/* The commented out defines already exist in one of the above header files. The
compiler                */
/* does not like the redundancy. So, they are commented out with the // symbols
                  */
//#define     TSCR1   _IO8(0x46)            //timer system control register
//#define     PTT     _IO8(0x240)           //portt data register
//#define     DDRT    _IO8(0x242)           //portt direction register
//#define     CRGFLG  _IO8(0x37)            //pll flags register
//#define     SYNR    _IO8(0x34)            //synthesizer / multiplier register
//#define     REFDV   _IO8(0x35)            //reference divider register
//#define     CLKSEL  _IO8(0x39)            //clock select register
//#define     PLLCTL  _IO8(0x3a)            //pll control register
#define   PORTT    _IO8(0x240)             // PortT data register
#define   PORTTi   _IO8(0x241)             // portT data register
#define   PORTM    _IO8(0x250)             // portM data register
#define   MCCTL    _IO8(0x66)              //modulus down conunter control
#define   MCFLG    _IO8(0x67)              //down counter flags

//TIMER INTERRUPTS INCLUDED IN HEADER FILE ALREADY

#define   SPCR1    _IO8(0xD8)              //SPI SPCR1 REGISTER LOCATION
#define   SPCR2    _IO8(0xD9)              //SPI SPCR2 REGISTER LOCATION
#define   SPIB     _IO8(0xDA)              //SPI SPIB REGISTER LOCATION
#define   SPSR     _IO8(0xDB)              //SPI SPSR REGISTER LOCATION
#define   SPDR     _IO8(0xDD)              //SPI SPDR REGISTER LOCATION
#define   MCCNT    _IO16(0x76)             //modulus down counter register
#define   keypad   PORTT

// Let's define some bit locations for some flags and config bits
#define   PLLSEL  0x80
#define   LOCK    0x08
#define   TFFCA   0x10
#define   MCZF    0x80
#define   BIT0    0x01
#define   BIT1    0x02
#define   BIT2    0x04
#define   BIT3    0x08
#define   BIT4    0x10
#define   BIT5    0x20
#define   BIT6    0x40
#define   BIT7    0x80
#define   ENABLE  0x02                                  // LCD ENABLE AT PM1
#define   RCK     0x08                                  // RCK CONNECTED TO PM3
#define   RS      0x01                                  // REGISTER SELECT (RS)
AT PM0 (0=COMMAND,1=DATA)



// Let's define our general variables
unsigned char CLOCK_FLAG;                               // FLAG FOR CLOCK 1
INTERRUPT
unsigned char STOP_WATCH_START;                         // TOGGLE FLAG FOR STOP
WATCH
unsigned char CLOCK_SET;                                // SET MODE FOR CLOCK
unsigned char FORMAT;                                   // 12 OR 24 HOUR MODE
FOR CLOCK
```

```
unsigned char SPEED;                                    // CLOCK SPEED (NORMAL
OR FAST)
unsigned char BLINK;                                    // BLINK FLAG FOR TIME
ENTRY MODE
unsigned int CLOCK_TIMER_COUNT;                         // INTERRUPT TIMER COUNT
FOR CLOCK
unsigned long CLOCK_TIME;                               // CLOCK TIME COUNT IN
SECONDS
unsigned long STOP_WATCH_TIME;                          // STOP WATCH TIME IN
MICROSECONDS


char getkey(void);
void SetClk8(void);
void delayby1ms(int k);
void keyrelease(void);
void Command(char a);
void Print(char a);
void Clear(void);
void delay(void);
void delay3(void);
void systemInitialize(void);
void SPIInitialize(void);
void InitLCD(void);
void updateDisplay(void);
void updateStopWatch(void);
void displayTime24Hr(void);
void displayTime12Hr(void);
void displayStopWatch(void);
void initTimerInterrupt(void);
void startTimerInterrupt(void);
void printString(char *string);
void printParseDecimal(char val);
void setClock(void);
void blinking(void);

/*****************************************************************/
/*  This is the main code where everything should go,            */
/*  All program function is directed by this code                */
/*****************************************************************/
void main(void)
{
     char key1;

        systemInitialize();                             //Run system
initialization

     updateDisplay();                                   //Print out the display

     setClock();                                        //Let the user set the
time

     while(1)                                           // this is and infinite
while loop
     {
          /* OK. 1ms gone by. Let's see if user pressed the A key or the B key. */
               key1  = getkey();                        // go check for keypress
               if(key1 < 0x1f)                          // did we get a key? If
so, do the next three statements
               {
                    keyrelease();                       //check for keyrelease
                    delay();                            //short delay
                       while((PORTTi & 0x08));                   // if column 3
```

```
is HIGH wait here until LOW
                    if(key1 >= 10)                          //Did user press
anything above 9?
                    {
                            if(key1==10)                    // If user pressed A do
the following
                            {
                                CLOCK_SET = 0x01;           // PUT CLOCK IN ENTRY
MODE USE SET FLAG
                                updateDisplay();            // UPDATE THE DISPLAY
                    }
                    else if(key1==11)                       // IF USER PRESSED B...
                            {
                                 FORMAT = ~FORMAT;          // TOGGLE THE FORMAT
FLAG
                                 updateDisplay();           //Print out the display
                    }
                    else if(key1 == 12)
                    {
                         SPEED = ~SPEED;                    // IF USER PRESSED C,
CHANGE SPEED
                             }
                             else if(key1 == 14)            // IF USER PRESSED E...
                    {
                         STOP_WATCH_START = 0x00;           // STOP STOP-WATCH
                         STOP_WATCH_TIME = 0;               // RESET STOP_WATCH TIME
TO 0
                             }
                             else if(key1 == 15)            // IF USER PRESSED F....
                    {
                         STOP_WATCH_START = ~STOP_WATCH_START;   // TOGGLE
START/STOP OF STOPWATCH

                         if(STOP_WATCH_START == 0x00) {     // ARE WE STOPPING THE
STOP WATCH?
                              updateDisplay();              // UPDATE THE DISPLAY
                         }
                             }
                     }
                 }

              if(CLOCK_SET == 0x01)                         // ARE WE IN SET MODE?
              {
                setClock();                                 // LET USER SET CLOCK

              }
              else if (CLOCK_FLAG == 0x01)                  // IS THE CLOCK FLAG
SET?
              {
                CLOCK_TIME++;                               // INCREASE TIME BY 1
                if(CLOCK_TIME >= 86400) CLOCK_TIME = 0;     // IF AT END OF TIME,
RESET TIME
                   updateDisplay();                            // UPDATE DISPLAY
                CLOCK_FLAG = 0x00;                          // RESET CLOCK INTERRUPT
FLAG
              }


     }
}
```

```
/*********************************************************************/
/*  This is the initialization controller, it initializes program   */
/*  vars & calls other systems to initialize in the hardware        */
/*********************************************************************/
void systemInitialize(void)
{
     SetClk8();                                                  // go setup the PLL

     // setup the data direction registers
     DDRM = 0xfc;                                                // set data direction
register for PortM
     DDRT = 0xf0;                                                // set data direction
register for PortT
     PORTM = 0;                                                  // clear PortM

     SPIInitialize();                                            // INITIALIZE THE SPI
SYSTEM

     InitLCD();                                                  // INITIALIZE THE LCD

     initTimerInterrupt();                                       // INITIALIZE THE
INTERNAL TIMER INTERRUPTS
     startTimerInterrupt();

     CLOCK_SET = 0x01;                                           // INITIALIZE IN
CLOCK_SET MODE
     CLOCK_FLAG = 0x00;                                          // CLOCK INTERRUPT FLAG
NOT SET
     FORMAT = 0x00;                                              // INITIALIZE IN 24HR
MODE
     SPEED = 0x00;                                               // INITIALIZE IN NORMAL
SPEED
     BLINK = 0x01;                                               // INITIALIZE BLINK ON
     CLOCK_TIMER_COUNT = 0;                                      // INTERRUPT COUNTER AT
0
     STOP_WATCH_START = 0x00;                                    // STOPWATCH NOT RUNNING
     CLOCK_TIME = 43200;                                         // SET CLOCK TO NOON
     STOP_WATCH_TIME = 0;                                        // STOP WATCH TO 0
}



/*********************************************************************/
/* This function enables PLL and use an 8-MHz crystal oscillator to */
/* generate 24-MHz E clock. Same as done in assembler.               */
/*********************************************************************/
void SetClk8(void)
{
     asm(sei);                                                   // turn of interrupts
     CLKSEL  &= PLLSEL;                                          // disengage PLL from
system
     SYNR    = 0x02;                                             // set SYSCLK to 24 MHz
from a 4-MHz oscillator
     REFDV   = 0;                                                //
     PLLCTL  = 0x40;                                             // turn on PLL, set
automatic
     while(!(CRGFLG & LOCK));                                    // wait for HIGN on LOCK
bit at address CRGFLG
     asm(nop);                                                   // very short delays
     asm(nop);
     CLKSEL  |= PLLSEL;                                          // clock derived from
PLL
     asm(cli);                                                   // turn on interrups
```

```
}


/*****************************************************************/
/*  This subroutine initializes the SPI system on the HC12S      */
/*                                                               */
/*****************************************************************/
void SPIInitialize(void)
{
     SPIB = 0x22;                                          //SPI CLOCKS A 1/24 OF
E-CLOCK
     DDRM = 0x3B;                                          //SETUP PORTM DATA
DIRECTION
     SPCR1 = 0x50;                                         //ENABLE SPI AND SET
MODE AS MASTER
     SPCR2 = 0x00;                                         //RESETS SPCR2 TO $00
(ALSO DOES AT RESET)
     PORTM = PORTM | RCK;                                  //SET RCK TO IDLE HIGH
     PORTM = PORTM & ~ENABLE;                              //ENABLE TO IDLE LOW
}


/*****************************************************************/
/*  This subroutine initializes the LCD screen                   */
/*                                                               */
/*****************************************************************/
void InitLCD(void)                                         //Cheap and dirty method
to initialize LCD
{
        Command(0x30);                                        //Call command method
with 0x30
        delay3();                                             //Allow the command
to take place
        Command(0x30);                                        //Call command method
with 0x30
        delay3();                                             //Allow the command
to take place
        Command(0x30);                                        //Call command method
with 0x30
        delay3();                                             //Allow the command
to take place
        Command(0x38);                                        //Call command method
with 0x38
        delay3();                                             //Allow the command
to take place
        Command(0x0C);                                        //Call command method
with 0x0C
        delay3();                                             //Allow the command
to take place
        Clear();                                              //Clear the
homescreen
}


/*****************************************************************/
/*  This subroutine initializes the timer interrupt              */
/*                                                               */
/*****************************************************************/
void initTimerInterrupt(void)
{
```

```
     TSCR2 = 0x04;                                          //CONFIGURE PRESCALE
FACTOR 16 (2/3 usec) 1500 = 1msec
     TIOS = 0x03;                                           //ENABLE OC0,OC1 FOR
OUTPUT COMPARE
     TSCR1 = 0x90;                                          //ENABLE TCNT & FAST
FLAGS CLEAR
     TIE = 0x03;                                            //ENABLE INTERRUPTS TC1
& TC2
}


/*********************************************************************/
/*  This subroutine starts the timer interrupts up                   */
/*                                                                   */
/*********************************************************************/
void startTimerInterrupt(void)
{
     TC0 = TCNT + 3750;                                     //INCREMENT COUNT BY
3750 (1SECOND)
     TC1 = TCNT + 3750;                                     //INCREMENT TC1 BY 3750
     TFLG1 = 0x03;                                          //SET BOTH FLAGS FOR
TIMER INTERRUPTS
}


/*********************************************************************/
/*  This subroutine blinks the entire display for clock set mode     */
/*                                                                   */
/*********************************************************************/
void blinking(void)
{
     if(CLOCK_FLAG == 0x01)                                 //IF OUR CLOCK FLAG IS
SET
     {
          if(BLINK == 0x01)                                 //AND OUR BLINK FLAG IS
SET
          {
               Command(0x08);                               //TURN THE LCD Off
               BLINK = 0x00;                                //CHANGE BLINK FLAG
          } else {
               Command(0x0C);                               //TURN THE LCD ON
               BLINK = 0x01;                                //CHANGE THE BLINK FLAG
          }

          CLOCK_FLAG = 0x00;                                //RESET THE CLOCK FLAG
     }
}



/*********************************************************************/
/*                   */
/*                                                                   */
/*********************************************************************/
void setClock()
{
     unsigned long hour;                                    //DEFINE VARS
     unsigned char minute, second, key, key_flag;           //DEFINE VARS
     key_flag = 0;                                          //INITIALIZE KEY FLAG
```

```
     while(key != 0x0a)                                    //LOOP UNTIL USER
PRESSES A AGAIN
     {
          /* OK. 1ms gone by. Let's see if user pressed the A key or the B key. */
               key  = getkey();                            //GO CHECK FOR KEYPRESS
               if(key < 0x1f)                              //DID WE GET A KEY? THE
PERFORM ACTION
               {
                    keyrelease();                          //WAIT FOR USER TO
RELEASE KEY
                    delay();                               //DELAY
                       while((PORTTi & 0x08));             //IF COLUMN 3 IS HIGH
WAIT HERE UNTIL LOW
                       if(key == 11)                       //DID USER PRESS B?
                       {
                      key_flag++;                          //MOVE TO NEXT TIME
SEGMENT (HOURS/MINS/SECS)

                      if(key_flag > 2) key_flag = 0;       //IF WE'RE AT SECONDS
RESET TO HOURS
                }
                     else if (key == 12)                   //DID USER PRESS C?
                     {
                          hour = ((CLOCK_TIME/3600));      //CALCULATE CURRENT HOUR


                    minute = (((CLOCK_TIME % 3600)/60));   //CALCULATE CURRENT
MINUTE

                    second = (((CLOCK_TIME % 3600) % 60));  //CALCULATE CURRENT
SECONDS

                    if(key_flag == 0)                      //IF WE'RE AT HOURS...
                    {
                         hour++;                           //INCREASE HOURS BY 1
                         if(hour >= 24) hour = 0;          //IF IT'S AT 24, RESET
TO 0
                    }
                    else if (key_flag == 1)                //IF WE'RE AT
MINUTES....
                    {
                         minute++;                         //INCREASE MINUTES BY 1
                         if(minute >= 60) minute = 0;      //IF IT'S AT 60 MINS,
RESET TO 0
                    }
                    else if (key_flag == 2)                //IF WE'RE AT
SECONDS....
                    {
                         second++;                         //INCREASE SECONDS BY 1
                         if(second >= 60) second = 0;      //IF IT'S AT 60 SECONDS,
RESET TO 0
                    }

                    CLOCK_TIME = (hour*3600) + (minute*60) + (second);
//RECONSTRUCT CLOCK TIME IN SECONDS
                        }

               updateDisplay();                            //UPDATE OUR DISPLAY
AGAIN
             }

             blinking();                                   //CHECK TO SEE IF WE
NEED TO BLINK
```

```
     }

     Command(0x0C);                                          //MAKE SURE THE LCD IS
ON IF WE'RE DONE
     CLOCK_SET = 0x00;                                       //CHANGE CLOCK SET MODE
     startTimerInterrupt();                                  //START UP OUR TIMER
INTERRUPTS
}




/*************************************************************************/
/*                                                                       */
/* The getkey functions gets the key value from a 4X4 matrix keypad connected    */
/* PortT. Rows (0,1,2,3) = P4,P5,P6,P7                                   */
/*     Columns (0,1,2,3) = P0,P1,P2,P3                                   */
/* The strategy used here is nessted if -else statements and is similar to what  */
/* we did in assembly language. There are more efficient and elegant strategies. */
/*                                                                       */
/*************************************************************************/
char getkey(void)
{                                                            // We test the keys in
sequence - row 0 columns 0,1,2,3
                                                             // row 2 columns 0,1,2,3
etc. until we have checked
   char keyX;                                                // all of the keys. If a
key is pressed then we save the
                                                             // value in keyX and
jump down to return without
                                                             // checking any more
keys.  Note that there many
                                                             // more ways to do this.

   PORTT = 0x00;                                             // clear portT
   asm(NOP);                                                 // short wait times with
assembler NOP
   PORTT |= 0x10;                                            // PORTT = PORTT | 0x10;
OR PORTT with $10. ie. set row 0 (PT4) High
   asm(nop);
   asm(nop);
   asm(nop);

   if(PORTT & BIT0)                                          // AND PORT with 0x01
and check if ans is 1 (TRUE). ie. Check column 0 for HIGH. If High
           keyX = 1;                                         // then set keyX to 1
and jump to return.
   else if(PORTT & BIT1)                                     // Check column 1
           keyX = 2;
   else if(PORTT & BIT2)                                     // Check column 2
           keyX = 3;
   else if(PORTT & BIT3)                                     // Check column 3
           keyX = 10;
   else {
           PORTT = 0x00;                                     // Clear PortT and start
on row 1
           PORTT |= 0x20;                                    //  Set row 1 High
           asm(nop);
           asm(nop);
           asm(nop);

           if(PORTT & BIT0)                                  // Check column 0 etc.,
```

```
etc.
           keyX = 4;
          else if(PORTT & BIT1)
           keyX = 5;
          else if(PORTT & BIT2)
           keyX = 6;
          else if(PORTT & BIT3)
           keyX = 11;
          else {
              PORTT = 0x00;
              PORTT |= 0x40;                                  // row 2 High
              asm(nop);
              asm(nop);
              asm(nop);

              if(PORTT & BIT0)
                keyX = 7;
              else if(PORTT & BIT1)
                keyX = 8;
              else if(PORTT & BIT2)
                keyX = 9;
              else if(PORTT & BIT3)
                keyX = 12;
              else {
                PORTT = 0x00;
                PORTT |= 0x80;                                // row 3 High
                asm(nop);
                asm(nop);
                asm(nop);

                if(PORTT & BIT0)
                  keyX = 0 ;
                else if(PORTT & BIT1)
                  keyX = 15;
                else if(PORTT & BIT2)
                  keyX = 14;
                else if(PORTT & BIT3)
                  keyX = 13;
                else                                          // if we get to here ==>
no key pressed
                  keyX = 0x1f;                                // nokey signal
              }
          }
   }
   return (keyX);                                             // return the key value
}

/*************************************************************/
/*                                                           */
/* Key release routine. Check each coulmn bit. If HIGH wait */
/* until it goes LOW to break out of the while statement.    */
/* Note that we are reading the input register of PortT      */
/* which is at address $0241 and is called (here) PORTTi     */
/*                                                           */
/*************************************************************/
void keyrelease(void)
{
     //PORTT = 0xf0                                           // Set all rows high
(not needed here. Why?)
     while((PORTTi & 0x01));                                  // if column 0 is HIGH
wait here until LOW
     while((PORTTi & 0x02));                                  // if column 1 is HIGH
wait here until LOW
```

```
     while((PORTTi & 0x04));                                    // if column 2 is HIGH
wait here until LOW
     while((PORTTi & 0x08));                                    // if column 3 is HIGH
wait here until LOW
}



/*********************************************************************/
/*  This subroutine creates a small delay which counts clock cycles */
/*                                                                   */
/*********************************************************************/
void delay(void)                                                    //This will be the delay
for LCD commands - count clock cycles
{
     int y = 2000;                                                  //Initialize Y as 8000
        int i = 0;                                                  //Initialize i as 0
        for(i; i<=y;i++);                                           //Do the delay 8000
times
}



/*********************************************************************/
/*  This is a slightly larger delay than delay(), it uses a nested  */
/*  loop to increase the time spent here                             */
/*********************************************************************/
void delay3(void)                                                   //This delay has nested
while loops - count clock cycles
{
        int y = 0x0F;                                               //Iniialize Y as $0F
        while (y!=0){                                               //Loop while Y!=0
                int x = 0xFFFF;                                     //Initialize X as $FFFF
                while(x!=0){                                        //Loop while X!=0
                        x--;                                        //Decrement X
                }
                y--;                                                //Decrement Y
        }
}



/*********************************************************************/
/* The following function creates a time delay which is equal to the */
/* multiple of 1ms. The value passed in k specifies the number of    */
/* milliseconds to be delayed.                                       */
/*********************************************************************/
void delayby1ms(int k)                                              // k*1ms delay with
embedded key press check
{
     /* Standard Timer Setup */
        int ix;

        TSCR1 = 0x90;                                               /* enable TCNT and fast
timer flag clear */
        TSCR2 = 0x06;                                               /* disable timer
interrupt, set prescaler to 64 */
        TIOS  |= BIT0;                                              /* enable OC0 */
        TFLG1 &= BIT0;                                              /* clear timer flag
OC0F*/
        TC0 = TCNT + 375;                                           /* add 375 to the
tcount*/
```

```
        for(ix = 0; ix < k; ix++)                       // Do this loop k times.
Where k*1ms is the ~time wait we need. Not necessarily 1 second.
        {
             while(!(TFLG1 & BIT0));                    // ASM==> Here BRCLR
TFLAG1, $01, Here
                TC0 += 375;                             // If we get here
TFLAG1's BIT0 became HIGH
     }
        TIOS  &= (~BIT0);                               /* disable OC0  and
exit.  note no return statement required*/
}



/*****************************************************************/
/*  This function clears the LCD screen                          */
/*                                                               */
/*****************************************************************/
void Clear(void)                                        //Clears the LCD screen
{
        Command(0x01);                                  //Sends the clear
command to LCD
        delay();                                        //Allows the command to
go through
        delay();                                        //Allows the command to
go through
}



/*****************************************************************/
/*  This subroutine sends a command to the LCD, for example to move */
/*  the cursor to the beginning of the screen                    */
/*****************************************************************/
void Command(char a)                                    //Method to send
commands to LCD via SPI to SIPO system
{
        while(!(SPISR & 0x20));                         //Wait for register
empty flag (SPIEF)
        SPIDR = a;                                      //Output command via SPI
to SIPO
        while(!(SPISR & 0x80));                         //Wait for SPI Flag
        a = SPIDR;                                      //Equate a with SPIDR
        asm(nop);                                       //Wait for 1 cycle
        PORTM &= ~RCK;                                  //Pulse RCK
        asm(nop);                                       //Wait for 1 cycle
        asm(nop);                                       //Wait for 1 cycle
        PORTM |= RCK;                                   //Command now available
for LCD
        PORTM &= ~RS;                                   //RS = 0 for commands
        asm(nop);                                       //Wait for 1 cycle
        asm(nop);                                       //Wait for 1 cycle
        asm(nop);                                       //Wait for 1 cycle
        PORTM |= ENABLE;                                //Fire ENABLE
        asm(nop);                                       //Wait for 1 cycle
        asm(nop);                                       //Wait for 1 cycle
        PORTM &= ~ENABLE;                               //ENABLE off
        delay();                                        //Delay
        delay();                                        //Delay
}
```

```
/*********************************************************************/
/*  This subroutine prints an ASCII character to the screen          */
/*                                                                   */
/*********************************************************************/
void Print(char a)                                          // Method to send data
to LCD via SPI to SIPO system
{
        while(!(SPISR & 0x20));                             //Wait for register
empty flag (SPIEF)
        SPIDR = a;                                          //Output command via SPI
to SIPO
        while(!(SPISR & 0x80));                             //Wait for SPI Flag
        a = SPIDR;                                          //Equate a with SPIDR
        asm(nop);                                           //Wait for 1 cycle
        PORTM &= ~RCK;                                      //Pulse RCK
        asm(nop);                                           //Wait for 1 cycle
        asm(nop);                                           //Wait for 1 cycle
        PORTM |= RCK;                                       //Command now available
for LCD
        PORTM |= RS;                                        //RS = 1 for data
        asm(nop);                                           //Wait for 1 cycle
        asm(nop);                                           //Wait for 1 cycle
        asm(nop);                                           //Wait for 1 cycle
        PORTM |= ENABLE;                                    //Fire ENABLE
        asm(nop);                                           //Wait for 1 cycle
        asm(nop);                                           //Wait for 1 cycle
        PORTM &= ~ENABLE;                                   //ENABLE off
        delay();                                            //Delay
}


/*********************************************************************************
*****/
/* printStringPrint determines the number of characters in the string so that we can
   */
/* send the correct number of characters to the LCD print command.
   */
/* Then, the characters are printed. There are built in functions in the string.h
   */
/* library but we are not using that library. So, we will use this home made
function.  */
/* Not pretty but it works.
   */
/*********************************************************************************
*****/
void printString(char *string)
{
    int i, n;
    const char *tmp = string;

    // NOTE: does NOT check for string == NULL

    while(*tmp != '\0') {                                   // C strings end with \0
        tmp++;
    }
    n = tmp - string;                                       // OK. Now we know how
many characters to print

    for(i=0; i<n; i++) {
        Print(string[i] );                                  // Call LCD print
command
    }
```

```
}


/*****************************************************************/
/*  This command takes a value (max 99) and parses it to decimal    */
/*                                                                 */
/*****************************************************************/
void printParseDecimal(char val)
{
    Print((val/10)+0x30);                                  //PRINT TENS DIGIT

    Print((val % 10)+0x30);                                //PRINT ONES DIGIT
}



/*****************************************************************/
/*  This function updates the display by printing characters and    */
/*  Commands to it                                                 */
/*****************************************************************/
void updateDisplay(void)
{
     char i;                                                   //DECLARE I AS CHAR

     Clear();                                                  //CLEAR DISPLAY

     Command(0x02);                                            //MOVE CURSOR TO HOME
POSITION

        if(FORMAT == 0x00) {                                   //IF WE'RE IN 24HR MODE
             displayTime24Hr();                                //DISPLAY 24HR TIME
        } else {
             displayTime12Hr();                                //DISPLAY 12HR TIME
        }

        Command(0xC0);                                         //MOVE CURSOR TO NEW
LINE

     for(i=0; i<11;i++)                                        //LOOP 8 TIMES
     {
          Print(0x20);                                         //PRINT A SPACE " "
     }

     displayStopWatch();                                       //DISPLAY STOP_WATCH
TIME
}



/*****************************************************************/
/*  This function updates the display by printing characters and    */
/*  Commands to it                                                 */
/*****************************************************************/
void updateStopWatch(void)
{
     Command(0x10);                                            //PRINT A SPACE " "
     Command(0x10);                                            //PRINT A SPACE " "
     Command(0x10);                                            //PRINT A SPACE " "
     Command(0x10);                                            //PRINT A SPACE " "
     Command(0x10);                                            //PRINT A SPACE " "

     Command(0x06);                                            //MAKE SURE WE'RE
```

```
PRINTING IN THE RIGHT DIRECTION

     displayStopWatch();                                    //DISPLAY STOPWATCH
}



/*********************************************************************/
/*  This sub displays the number that the count is currently at.     */
/*                                                                   */
/*********************************************************************/
void displayTime24Hr(void)
{
     char hour;                                             //DECLARE hour VAR
     char i;                                                //DECLARE i VAR
     for(i=0; i<11;i++)                                     //LOOP 11 TIMES
     {
          Print(0x20);                                      //PRINT A SPACE " "
     }

     hour = ((CLOCK_TIME/3600));                            //CALCULATE CURRENT HOUR

     if(hour < 10)                                          //IF HOUR IS LESS THAN
10
     {
          Print(0x20);                                      //PRINT A SPACE " "
          Print(hour+0x30);                                 //PRINT CURRENT HOUR
     }
     else
     {
          printParseDecimal(hour);                          //PRINT HOUR (GREATER
THAN 9)
     }

     if(BLINK == 0x01) {                                    //CHECK TO SEE IF COLON
SHOULD BLINK?
          Print(0x3A);                                      //IF IT'S OFF, TURN IT
ON
          BLINK = 0x00;                                     //SET BLINK TO ZERO
     } else {
          Print(0x20);                                      //IF IT'S ON, TURN IT
OFF
          BLINK = 0x01;                                     //SET BLINK TO ONE
     }

     printParseDecimal((CLOCK_TIME % 3600)/60);             //PRINT MINUTES

     //Print(0x3A);                                         //PRINT COLON

     //printParseDecimal(((CLOCK_TIME % 3600) % 60));       //PRINT ONES DIGIT
}



/*********************************************************************/
/*  This sub displays the number that the count is currently at.     */
/*                                                                   */
/*********************************************************************/
void displayTime12Hr(void)
{
     char hour, meridian;                                   //DECLARE hour, meridian
VAR
```

```
     int i;                                                    //DECLARE i VAR
     for(i=0; i<9;i++)                                         //LOOP 9 TIMES
     {
          Print(0x20);                                         //PRINT A SPACE " "
     }

     hour = ((CLOCK_TIME/3600));                               //CALCULATE CURRENT HOUR

     if(hour < 12) {                                           //IF IT'S LESS THAN 12
          meridian = 0x00;                                     //THEN IT'S AM
     } else {
          meridian = 0x01;                                     //ELSE IT'S PM
     }

     if(hour == 0) {                                           //IF HOUR IS ZERO
          hour = 12;                                           //THEN HOUR IS TWELVE
     } else if (hour > 12) {                                   //IF HOUR IS GREATER
THAN TWELVE
          hour = hour - 12;                                    //THEN SUBTRACT TWELVE
FROM IT
     }

     if(hour < 10)                                             //IF HOUR IS LESS THAN
10
     {
          Print(0x20);                                         //PRINT A SPACE
          Print(hour+0x30);                                    //PRINT HOUR
     }
     else
     {
          printParseDecimal(hour);                             //ELSE JUST PRINT THE
HOUR
     }

     if(BLINK == 0x01) {                                       //CHECK TO SEE IF COLON
SHOULD BLINK?
          Print(0x3A);                                         //IF IT'S OFF, TURN IT
ON
          BLINK = 0x00;                                        //SET BLINK TO ZERO
     } else {
          Print(0x20);                                         //IF IT'S ON, TURN IT
OFF
          BLINK = 0x01;                                        //SET BLINK TO ONE
     }

     printParseDecimal((CLOCK_TIME % 3600)/60);                //PRINT MINUTES


     //Print(0x3A);                                            //PRINT COLON

     //printParseDecimal(((CLOCK_TIME % 3600) % 60));          //PRINT ONES DIGIT


     if(meridian == 0x00) {                                    //IS IT AM?
          Print(0x41);                                         //PRINT A
          Print(0x4d);                                         //PRINT M
     } else {
          Print(0x50);                                         //PRINT P
          Print(0x4d);                                         //PRINT M
     }

}
```

```
/*****************************************************************/
/*  This sub displays the number that the count is currently at.    */
/*                                                                 */
/*****************************************************************/
void displayStopWatch(void)
{
     char sec, msec, i;                                    //DECLARE VARS
     long t;

        t = STOP_WATCH_TIME;                               //GET STOP_WATCH_TIME

     sec = ((t / 100));                                    //CALCULATE SECONDS


     if(sec < 10)                                          //IF MIN < 10
     {
          Print(0x20);                                     //PRINT SPACE
          Print(sec+0x30);                                 //PRINT MIN
     }
     else
     {
          printParseDecimal(sec);                          //PRINT MIN (GREATER
THAN 10)
     }

     Print(0x3A);                                          //PRINT A COLON

     msec = (t % 100);                                     //CALCULATE MILLISECONDS

     printParseDecimal(msec);                              //PRINT MILLISECONDS
}



/*****************************************************************/
/*  Timer Interrupt Service Routine for Time 0, which is to be      */
/*  used for the clock counter                                     */
/*****************************************************************/
interrupt void tch0ISR(void)
{

     TC0 = TC0 + 15000;                                    //INCREMENT COUNT BY
15000 (.01SECOND)

     if(SPEED == 0x00)                                     //ARE WE AT NORMAL
SPEED?
     {
          CLOCK_TIMER_COUNT++;                             //INCREMENT CLOCK TIMER
COUNT
          if(CLOCK_TIMER_COUNT == 100)                     //IF IT'S AT 100
          {
               CLOCK_FLAG = 0x01;                          //SET THE FLAG

               CLOCK_TIMER_COUNT = 0;                      //RESET THE COUNT
          }

     }
     else
     {
          CLOCK_FLAG = 0x01;                               //ELSE JUST SET THE FLAG
(1000X FASTER)
```

```
     }
}



/*****************************************************************/
/*  Timer Interrupt Service Routine for Time 1, which is to be      */
/*  used for the stop-watch counter                                 */
/*****************************************************************/
interrupt void tch1ISR(void)
{

     TC1 = TC1 + 15000;                                          //INCREMENT COUNT BY
9000
     if(STOP_WATCH_START != 0x00)                               //IF STOPWATCH IS ON
     {
          STOP_WATCH_TIME++;                                    //INCREASE STOPWATCH
TIME
          if(STOP_WATCH_TIME >= 366100) STOP_WATCH_TIME = 0; //IF IT'S AT MAX, RESET
          updateStopWatch();                                    //UPDATE STOPWATCH
DISPLAY
     }
}
```