```
/**********************************************************************
*                                                                     *
*   Filename:      chessclock.c                                       *
*   Date:          2011.12.-2012.01                                   *
*   File Version:  3.0                                                *
*                                                                     *
*   Author:        Peter Borkuti                                      *
*   Company:                                                          *
*                                                                     *
***********************************************************************
*                                                                     *
*   Architecture:  Mid-range PIC                                      *
*   Processor:     16F684                                             *
*   Compiler:      HI-TECH C                                          *
*                                                                     *
***********************************************************************
*                                                                     *
*   Files required: delay.h                                           *
*                                                                     *
***********************************************************************
*                                                                     *
*   Description:    Chessclock                                        *
*                                                                     *
*   PIC ---- 74HC595 serial to parallel ----- 2x7 segment LED display   *
                                                                    *
***********************************************************************
*                                                                     *
*   Pin assignments:                                                  *
*       RC0 - digital out - data  line to 74hc595                     *
*       RC1 - digital out - clock line to 74hc595                     *
*       RC2 - digital out - latch line to 74hc595                     *
*       RC3 - digital in  - start/stop button                         *
*       RC4 - digital in  - player switch button                      *
*       RC5 - digital in  - set button                                *
*                                                                     *
**********************************************************************/

//in CLOCK_DEBUG mode, time will be faster than as usual
//#define CLOCK_DEBUG

#include <htc.h>
#include "delay.h"

// Config: ext reset, no code protect, no watchdog, 4MHz int clock
__CONFIG(MCLRE_ON & CP_OFF & WDTE_OFF & FOSC_INTOSCIO);

//for timing the interrupt
#define TMR0_SETTING 0

//Buttons

// Start/Stop button. Its switches the state of the program between
// TIMER and IDLE
#define BTN_START_STOP RC3
// in TIMER mode:
//    Players push this button after moving
// in IDLE mode:
//    Switch the digits to set
#define BTN_PLAYER_SWITCH RC4

// Sets the digit in IDLE mode
#define BTN_SET RC5

//the chessclock has two states: IDLE and TIMER
#define STATUS_IDLE 0
#define STATUS_TIMER 1

//show timer helps to blinking the leds and digits
#define SHOW_TIMER 2

//when pic clocks at 4MHz and presclaer is 1:256 (111),
//there are 15 interrupts in every seconds
#ifdef CLOCK_DEBUG
#define CLICKS_IN_SECS 3
#endif
#ifndef CLOCK_DEBUG
```

```
#define CLICKS_IN_SECS 15
#endif

//If I wanted to test the clock it will be useful to speeding the "time"
//so if you want to test the chessclock, set CLICKS_IN_SECONDS to 2 and
//CLICKS_IN_SECS to 10
#ifdef CLOCK_DEBUG
#define SECONDS_IN_MINUTE 10
#endif
#ifndef CLOCK_DEBUG
#define SECONDS_IN_MINUTE 60
#endif

//The elapsed time for the players
volatile char number[2];
//Second mode for the players: if timer is in second mode,
// show seconds, not minutes
volatile char second_mode[2];
//current status of the chessclock
volatile bit status = STATUS_IDLE;
//current player (1/0)
volatile bit player = 0;
//Whether the led must be on or the digit must be on
// 1: ON
// 0: OFF
volatile bit show = 0;

//The actual digit when setting the time (0..3)
volatile char digit = 0;

//The segments to display for the numbers 0-9
//L means Led
const char pat7seg[10] = {

//        76543210
//        Labcdefg
        0b01111110,   // 0
        0b00110000,   // 1
        0b01101101,   // 2
        0b01111001,   // 3
        0b00110011,   // 4
        0b01011011,   // 5
        0b01011111,   // 6
        0b01110000,   // 7
        0b01111111,   // 8
        0b01110011    // 9
};

/*  Transmit and Receive port bits */
// !! Ports must be set up as OUTPUT before running

/*
Serial Data Input. The data on this pin is shifted into the
8-bit serial shift register.
*/
#define TxData      RC0             /* Map TxData to pin */

/*
Shift Register Clock Input. A low– to–high transition on this
input causes the data at the Serial Input pin to be shifted into
the 8–bit shift register.
*/
#define TxClock     RC1             /* Map Clock  to pin */

/*
Storage Latch Clock Input. A low–to–high transition on this
input latches the shift register data.
*/
#define TxLatch     RC2             /* Map Latch to pin  */


//sends 8 bits in serial
//If you want to debug, put LEDs and resistors to RC0,RC1,RC2
//and sets the delays to 10 in DelayMs(1) in order to see
//the communication
void putch(char c)
```

```
{
    TxClock = 0;
    for(char i =0; i<8; i++) {
        DelayMs(1);
        TxClock = 0;
        if(c & 1)
            TxData = 1;
        else
            TxData = 0;
        DelayMs(1);
        TxClock = 1;
        c = (c >> 1) | 0x80;
    };
NOP();
}


//Displays a two-digit number (16 bits)
//
//in TIMER mode, the LEDs are blinking, see the show*(player==1)
//in IDLE mode the led is ON but the appropriate digit is blinking
void displayNum(char x){
    TxLatch = 0;
    if (status==STATUS_TIMER) {
        putch(pat7seg[x/10]+128*show*(player==1));
        putch(pat7seg[x%10]+128*show*(player==0));
    } else {
        if (digit%2) {
            putch(pat7seg[x/10]*show+128*(digit>1));
            putch(pat7seg[x%10]+128*(digit<2));
        } else {
            putch(pat7seg[x/10]+128*(digit>1));
            putch(pat7seg[x%10]*show+128*(digit<2));
        }
    }
    TxLatch = 1;
    DelayMs(1);
    TxLatch = 0;
}

void setup(){
    //Processor speed setup, 4MHz, Internal Clock
    IRCF2=1;
    IRCF1=1;
    IRCF0=0;
    SCS=1;


    // input/output
    ANSEL  = 0b00000000;        // only digital I/O
    CMCON0 = 0b00000111;        // Comparators off. CxIN pins are configured as digital I/O
    TRISC  = 0b00111000;        // output: RC0,RC1,RC2; input:  RC3,RC4,RC5

    RC0=0;
    RC1=0;
    RC2=0;

    //timer

    asm("clrwdt");
    OPTION_REG&=0B11000111;    // turn off bottom 6 bits to configure tmr0
    TMR0   = TMR0_SETTING;     // reset timer (and prescaler!)

    //interrupt
    T0IE=1;
    GIE=1;

}

//players and idle mode have its own counter for seconds
volatile int count[]={0,0,0};

void interrupt isr(void)
{
  if (T0IF) {
    count[SHOW_TIMER]++;
    if (count[SHOW_TIMER]>=CLICKS_IN_SECS) {
        show^=1;
        count[SHOW_TIMER]=0;
    }
    if (status==STATUS_TIMER) {
        count[player]++;
        if (count[player]>=CLICKS_IN_SECS*SECONDS_IN_MINUTE ||
            (count[player]>=CLICKS_IN_SECS && second_mode[player])) {
            //1 minute is elapsed. Decrement the number
            count[player]=0;
            if (number[player]>0) {
                number[player]--;
            }
            if (number[player]==0 && !second_mode[player]) {
                number[player]=SECONDS_IN_MINUTE;
                second_mode[player]=1;
            }
        }
    }
   TMR0=TMR0_SETTING;
   T0IF=0;
  }
}

//Default time for players are 15-15
//It stores the digits in reverse order

#ifdef CLOCK_DEBUG
volatile char times[] = {5,0,5,0};
#endif
#ifndef CLOCK_DEBUG
volatile char times[] = {5,1,5,1};
#endif


void init_vars(){
    number[0]=times[1]*10+times[0];
    number[1]=times[3]*10+times[2];

    second_mode[0]=0;
    second_mode[1]=0;
    count[0]=0;
    count[1]=0;
    count[2]=0;
    player=0;
}

void main()
{
    //stores the previously displayed number
    //if the number is not changed, wont be displayed again
    char prevNum = -1;

    //when blinking a digit, it must be refreshed wheter it is changed or not
    char prevShow = 0;

    //This is the number to display
    char num2dsp = 0;

    //1, if timer was set by the user in IDLE mode
    char timerWasSet = 1;

    //fill with default values
    init_vars();
    setup();
    //RC0=1;
    for (;;){

        //check start/stop button with debouncing
        if (BTN_START_STOP==1) {
         DelayMs(20);
         while (BTN_START_STOP==1);
         if (status==STATUS_TIMER) {
            timerWasSet=0;
         } else {
            if (timerWasSet) {
                //If time was set by user, use the newly set timer values
                init_vars();
            }
         }
         status^= 1;
         DelayMs(20);
        }

        //check player button with debouncing
        if (BTN_PLAYER_SWITCH==1) {
         DelayMs(20);
         while (BTN_PLAYER_SWITCH==1);
         // in IDLE mode it steps the digit to set
         // in TIMER mode switches the player
         if (status==STATUS_IDLE) {
            digit++;
            digit%=4;
         } else
            player^=1;
         DelayMs(20);
        }

        //check set button with debouncing
        if (BTN_SET==1) {
         DelayMs(20);
         while (BTN_SET==1);
         //in IDLE mode, increases the actual digit
         //in TIMER mode, restart the timer and switches to IDLE mode
         if (status==STATUS_IDLE) {
            times[digit]++;
            times[digit]%=10;
            timerWasSet=1;
         } else {
            status=STATUS_IDLE;
            init_vars();
         }
         DelayMs(20);
        }

        //in TIMER mode, display the counter, else display the
        //starting-time
        if (status==STATUS_TIMER)
            num2dsp=number[player];
        else {
            char d = digit/2;
            d*=2;
            num2dsp=times[d+1]*10+times[d];
        }

        if (num2dsp!=prevNum||show!=prevShow) {
            displayNum(num2dsp);
            prevNum=num2dsp;
            prevShow=show;
            DelayMs(50);
        }
    }
}
```