```
// filename  ***************  LCD.C ******************************
// LCD Display (HD44780) on Port H for the 9S12DP512
// Jonathan W. Valvano 9/18/09

//  This example accompanies the books
//   "Embedded Microcomputer Systems: Real Time Interfacing",
//        Thompson, copyright (c) 2006,
//   "Introduction to Embedded Systems: Interfacing to the Freescale 9S12",
//        Cengage Publishing 2009, ISBN-10: 049541137X | ISBN-13: 9780495411376

// Copyright 2009 by Jonathan W. Valvano, valvano@mail.utexas.edu
//    You may use, edit, run or distribute this file
//    as long as the above copyright notice remains

/*
  size is 1*16
  if do not need to read busy, then you can tie R/W=ground
  ground = pin 1    Vss
  power  = pin 2    Vdd   +5V
  ground = pin 3    Vlc   grounded for highest contrast
  PH4    = pin 4    RS    (1 for data, 0 for control/status)
  PH5    = pin 5    R/W   (1 for read, 0 for write)
  PH6    = pin 6    E     (enable)
  PH3    = pin 14   DB7   (4-bit data)
  PH2    = pin 13   DB6
  PH1    = pin 12   DB5
  PH0    = pin 11   DB4
16 characters are configured as 2 rows of 8
addr  00 01 02 03 04 05 06 07 40 41 42 43 44 45 46 47
*/

#include <mc9s12dp512.h>     /* derivative information */
#include "LCD.H"
#define BUSY 1
#define NOTBUSY 0
static unsigned short OpenFlag=0;
static unsigned short ClearFlag=1;
static unsigned short CharFlag=1;
static unsigned short StringFlag=1;
static unsigned short GotoFlag=1;

//---------------------wait---------------------
// Time delay
// Input: Time in 0.667 usec
// Output: None
// Returns: None
void static wait(unsigned short delay){
unsigned short startTime;
  startTime = TCNT;
  while((TCNT-startTime) <= ((delay/24)+1)){}  // Divide by 24 to scale for 8MHz
}
//---------------------1mswait---------------------
// time delay
// Input: time in msecs
// Output: none
// Returns: none
void static wait1ms(unsigned short msec){
  for(;msec;msec--){
    wait(1500);    // 1ms wait
  }
}

//---------------------readStatus----------------------
```

```
// Checks the LCD's internal busy flag
// Input: None
// Output: None
// Returns: State of busy flag: BUSY = 1 and NOTBUSY = 0
unsigned char readStatus(void) {
  unsigned char busy, trash;
                                                //4-bit protocol read Busy
  DDRH &= ~0x0F;    //1) data direction input on 4 data bits
  PTH_PTH5 = 1;     //2) R/W=1, RS=0
  PTH_PTH4 = 0;
  PTH_PTH6 = 1;     //3) E=1
  asm nop           //4) Wait a little time (2 nops)
  asm nop                       //   [it does not work without delay]
  busy = ((PTH & 0x08) >> 3); //5) Read 4-bit MS nibble data (bit 3 is busy)
  PTH_PTH6 = 0;     //6) E=0
  PTH_PTH6 = 1;     //7) E=1
  asm nop           //8) Wait a little time (2 nops)
  asm nop
  trash = PTH;      //9) Read 4-bit LS nibble data (nothing interesting)
  PTH_PTH6 = 0;     //10) E=0
  PTH_PTH5 = 0;     //11) R/W=0 (default settings)
  DDRH |= 0x0F;     //12) direction on four data lines go back to outputs
                                        //    (default settings)
  return busy;  //BUSY = 1 and NOTBUSY = 0
}

//---------------------checkStatus----------------------
// Checks the LCD's internal busy flag and timeouts out after a given amount of
cycles
// Input: Cycles to wait before timing-out
// Output: None
// Returns: BUSY if the flag is set or it times out
unsigned char checkStatus(unsigned short cycles) {
  unsigned short startTime, tempTime;
  startTime = TCNT;
  tempTime = 0;
  while(tempTime <= cycles && readStatus() == BUSY) {
    tempTime = TCNT-startTime;
  }
  if(tempTime > cycles){ // Wait time exceeded
    return BUSY;
  }
  return NOTBUSY;
}




//---------------------outCsrNibble---------------------
// Sends one command code to the LCD control/status
// Input: Command is 4-bit function to execute
// Output: 4-bit command to LCD peripheral
// Returns: None
static void outCsrNibble(unsigned char command){
  PTH = (PTH&0x80)+(command&0x0F);    // nibble, E=0, RS=0
  PTH |= 0x40;             // E goes 0,1
  asm nop
  asm nop                  // 5 cycles wide = 208ns
  PTH &= ~0x40;            // E goes 1,0
}

//---------------------outCsr---------------------
// Sends one command code to the LCD control/status
// Input: Command is 8-bit function to execute
// Output: None
// Returns: None
static void outCsr(unsigned char command){
  outCsrNibble(command>>4);   // ms nibble, E=0, RS=0
  outCsrNibble(command);      // ls nibble, E=0, RS=0
  wait(135);
}

//---------------------LCD_Clear---------------------
// Clear the LCD display, send cursor to home
// Input: None
// Output: Sets internal flag if LCD is not open or LCD is busy.
// Returns: None
void LCD_Clear(void){
  if(OpenFlag==0){
    ClearFlag = 0;     // Not open, set error flag
    return;
  }
  outCsr(0x01);        // Clear Display
  if(checkStatus(350) == BUSY) {
    ClearFlag = 0;     // Set error flag, LCD is busy
    return;
  }
  outCsr(0x02);        // Cursor to home
  if(checkStatus(350) == BUSY) {
    ClearFlag = 0;     // Set error flag, LCD is busy
    return;
  }
  ClearFlag = 1;           // Success
  return;
}
#define LCDINC 2
#define LCDDEC 0
#define LCDSHIFT 1
#define LCDNOSHIFT 0
#define LCDCURSOR 2
#define LCDNOCURSOR 0
#define LCDBLINK 1
#define LCDNOBLINK 0
#define LCDSCROLL 8
#define LCDNOSCROLL 0
#define LCDLEFT 0
#define LCDRIGHT 4
#define LCD2LINE 8
#define LCD1LINE 0
#define LCD10DOT 4
#define LCD7DOT 0

//---------------------LCD_Open---------------------
// Initialize the LCD display, called once at beginning
// Input: None
// Output: Sets internal flag if Open succeeds
// Returns: None
void LCD_Open(void){
  if(OpenFlag){
    return;      // error if already open
  }
  DDRH |= 0x7F;    // PH6-0 output to LCD
  PTH &= ~0x20;    // PH5=R/W=0 means write
  //TSCR1 = 0x80;    // Enable TCNT, 24MHz boot mode, 4MHz in run mode
  //TSCR2 = 0x04;    // divide by 16 TCNT prescale, TCNT at 667nsec
  PACTL = 0;       // timer prescale used for TCNT
```

```
/* Bottom three bits of TSCR2 (PR2,PR1,PR0) determine TCNT period
    divide  FastMode(24MHz)    Slow Mode (8MHz)
000   1     42ns  TOF  2.73ms  125ns TOF 8.192ms
001   2     84ns  TOF  5.46ms  250ns TOF 16.384ms
010   4    167ns  TOF  10.9ms  500ns TOF 32.768ms
011   8    333ns  TOF  21.8ms    1us TOF 65.536ms
100  16    667ns  TOF  43.7ms    2us TOF 131.072ms
101  32   1.33us  TOF  87.4ms            4us TOF 262.144ns
110  64   2.67us  TOF 174.8ms    8us TOF 524.288ms
111 128   5.33us  TOF 349.5ms   16us TOF 1.048576s */
// Be careful, TSCR1 and TSCR2 maybe set in other rituals
  wait1ms(20);        // to allow LCD powerup
  outCsrNibble(0x03); // (DL=1 8-bit mode)
  wait1ms(5);         //  blind cycle 5ms wait
  outCsrNibble(0x03); // (DL=1 8-bit mode)
  wait(150);          // blind cycle 100us wait
  outCsrNibble(0x03); // (DL=1 8-bit mode)
  wait(150);          //  blind cycle 100us wait (not called for, but do it anyway)
  outCsrNibble(0x02); // (DL=0 4-bit mode)
  wait(150);          // blind cycle 100 us wait
/* Entry Mode Set 0,0,0,0,0,1,I/D,S
     I/D=1 for increment cursor move direction
        =0 for decrement cursor move direction
     S  =1 for display shift
        =0 for no display shift  */
  outCsr(0x04+LCDINC+LCDNOSHIFT);        // I/D=1 Increment, S=0 no displayshift
/* Display On/Off Control 0,0,0,0,1,D,C,B
     D  =1 for display on
        =0 for display off
     C  =1 for cursor on
        =0 for cursor off
     B  =1 for blink of cursor position character
        =0 for no blink  */
  outCsr(0x0C+LCDNOCURSOR+LCDNOBLINK);   // D=1 displayon, C=0 cursoroff, B=0 blink
off
/* Cursor/Display Shift  0,0,0,1,S/C,R/L,*,*
     S/C=1 for display shift
        =0 for cursor movement
     R/L=1 for shift to left
        =0 for shift to right    */
  outCsr(0x10+LCDNOSCROLL+LCDRIGHT);   // S/C=0 cursormove, R/L=1 shiftright
/* Function Set   0,0,1,DL,N,F,*,*
     DL=1 for 8 bit
       =0 for 4 bit
     N =1 for 2 lines
       =0 for 1 line
     F =1 for 5 by 10 dots
       =0 for 5 by 7 dots */
  outCsr(0x20+LCD2LINE+LCD7DOT);   // DL=0 4bit, N=1 2 line, F=0 5by7 dots
  OpenFlag = 1;         // device open
  return;   // clear display
}

//---------------------LCD_OutChar---------------------
// Sends one ASCII to the LCD display
// Input: Letter is ASCII code
// Output: Sets internal error flag if failure occurs
// Returns: None
void LCD_OutChar(unsigned char letter){
  if(OpenFlag==0){
    CharFlag = 0;  // not open
    return;
  }
  PTH = (PTH&0x80)+(0x10+(0x0F&(letter>>4)));   // ms nibble, E=0, RS=1
  PTH |= 0x40;       // E goes 0,1
  asm nop
  asm nop            // 5 cycles wide = 208ns
  PTH &= ~0x40;      // E goes 1,0
  PTH = (PTH&0x80)+(0x10+(letter&0x0F));        // ls nibble, E=0, RS=1
  PTH |= 0x40;       // E goes 0,1
  asm nop
  asm nop            // 5 cycles wide = 208ns
  PTH &= ~0x40;      // E goes 1,0
  if(checkStatus(6) == BUSY) { // 6 cycle timeout
    CharFlag = 0;
    return;
  }
  CharFlag = 1;          // success
  return;
}

//---------------------LCD_OutString--------------
// Display String
// Input: Pointer to NULL-terminationed ASCII string
// Output: Set internal error code if failure occurs
// Returns: None
void LCD_OutString(char *pt){
  if(OpenFlag==0){
    StringFlag=0;
    return;
  }
  while(*pt){
    LCD_OutChar((unsigned char)*pt);
    pt++;
  }
  StringFlag=1;   // success
  return;
}

//---------------------LCD_GoTo--------------
// Move the cursor to a particular row and column
// Input: Parameters (row, column)   First row and and column is 0
// Output: Sets internal error code if failure occurs
// Returns: None
void LCD_GoTo(unsigned char row, unsigned char col){
  //int i;
  if(OpenFlag==0 || col > 7 || row > 1){
    GotoFlag = 0;  // not open
    return;
  }
  if(row) {
    outCsr(0xC0 + col); // Jump to second 8 characters then to correct column
  }
  else {
    outCsr(0x80 + col); // Jump to correct column
  }
  GotoFlag = 1;         // success
  return;
}

//---------------------LCD_ErrorCheck--------------
// LCD_ErrorCheck Check to see if the LCD driver has had any errors
// Returns an error code if LCD has had any errors since initialization or since the
last call to ErrorCheck
// Input Parameter(none)
// Output Parameter(error code)
// Typical calling sequence
// Err = ErrorCheck();
// if(Err) Handle(Err);
short LCD_ErrorCheck(void) {
  short error = (StringFlag<<4)+(CharFlag<<3)+(GotoFlag<<2)+(ClearFlag<<1)+OpenFlag;
  return error&0x1F;
}
```