```
// filename ******** Main.C **************
// LCD Display (HD44780) on Port H for the 9S12DP512
// Jonathan W. Valvano 9/18/09
// TCNT runs at 667ns,

//  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
// Purpose: test program for 4-bit LCD.C driver

/*
  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 <hidef.h>      /* common defines and macros */
#include <mc9s12dp512.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dp512"

#include <stdio.h>
#include "LCD.H"
#include "PLL.H"
#include "fixed.h"

//--------------------TimerInit---------------
// initialize timer module to 0.667us(Boot Mode) TCNT clock
// inputs: none
// outputs: none
void TimerInit(void){
  TSCR1 = 0x80;   // Enable TCNT, 24MHz in both boo tand run modes
  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
}
```

```
//---------------------mwait---------------------
// wait specified number of msec
// Input: number of msec to wait
// Output: none
// assumes TCNT timer is running at 667ns
void mwait(unsigned short msec){
unsigned short startTime;
  for(; msec>0; msec--){
    startTime = TCNT;
    while((TCNT-startTime) <= 1500){}
  }
}

//---------------------check---------------------
// if LCD broken toggle LED1 at 2Hz
// Input: last LCD status, 0 means bad
// Output: none
// Error: if status is zero, this function will not return
void check(short status){        // 0 if LCD is broken
  if(status ==0){
    for(;;) {
      PTP ^= 0x80;   // fast toggle LED
      mwait(250);    // 0.25 sec wait
    }
  }
}
//---------------------swait---------------------
// wait specified 2 sec, then clear LCD
// Input: none
// Output: none
// uses mswait and TCNT timer
void swait(void){
  PTP ^= 0x80;    // toggle LED0
  mwait(2000);    // 2 sec wait
  check(LCD_Clear());
}

//---------------------blank---------------------
// move cursor to second half of LCD display
// 32 spaces from address 08 to 40
// Input: none
// Output: none
void blank(void){
  check(LCD_OutString("                                "));
}

void main(void) {
  PLL_Init();       // set E clock to 24 MHz
  TimerInit();      // enable timer
  DDRP |= 0x80;     // PortP bit 7 is output to LED, used for debugging
  check(LCD_Open());
  check(LCD_Clear());
  for(;;) {         // Tests the three functions of Fixed.c
    LCD_OutString("Fixed_uD"); blank();
    LCD_OutString("ecOut2");
    swait();
    LCD_OutString("0    = "); blank();
    Fixed_uDecOut2(0);
    swait();
    LCD_OutString("1    = "); blank();
    Fixed_uDecOut2(1);
    swait();
    LCD_OutString("99   = "); blank();
```

```
Fixed_uDecOut2(99);
swait();
LCD_OutString("100   = "); blank();
Fixed_uDecOut2(100);
swait();
LCD_OutString("999   = "); blank();
Fixed_uDecOut2(999);
swait();
LCD_OutString("1000  = "); blank();
Fixed_uDecOut2(1000);
swait();
LCD_OutString("9999  = "); blank();
Fixed_uDecOut2(9999);
swait();
LCD_OutString("10000 = "); blank();
Fixed_uDecOut2(10000);
swait();
LCD_OutString("65534 = "); blank();
Fixed_uDecOut2(65534);
swait();
LCD_OutString("65535 = "); blank();
Fixed_uDecOut2(65535);
swait();

LCD_OutString("Fixed_sD"); blank();
LCD_OutString("ecOut3");
swait();
LCD_OutString("-32768 ="); blank();
Fixed_sDecOut3(-32768);
swait();
LCD_OutString("-10000 ="); blank();
Fixed_sDecOut3(-10000);
swait();
LCD_OutString("-9999  ="); blank();
Fixed_sDecOut3(-9999);
swait();
LCD_OutString("-999   ="); blank();
Fixed_sDecOut3(-999);
swait();
LCD_OutString("-1     ="); blank();
Fixed_sDecOut3(-1);
swait();
LCD_OutString("0      ="); blank();
Fixed_sDecOut3(0);
swait();
LCD_OutString("123    ="); blank();
Fixed_sDecOut3(123);
swait();
LCD_OutString("1234   ="); blank();
Fixed_sDecOut3(1234);
swait();
LCD_OutString("9999   ="); blank();
Fixed_sDecOut3(9999);
swait();
LCD_OutString("32767 = "); blank();
Fixed_sDecOut3(32767);
swait();

LCD_OutString("Fixed_uB"); blank();
LCD_OutString("inOut8");
swait();
LCD_OutString("0     = "); blank();
Fixed_uBinOut8(0);
```

```
    swait();
    LCD_OutString("2     = "); blank();
    Fixed_uBinOut8(2);
    swait();
    LCD_OutString("64    = "); blank();
    Fixed_uBinOut8(64);
    swait();
    LCD_OutString("100   = "); blank();
    Fixed_uBinOut8(100);
    swait();
    LCD_OutString("500   = "); blank();
    Fixed_uBinOut8(500);
    swait();
    LCD_OutString("512   = "); blank();
    Fixed_uBinOut8(512);
    swait();
    LCD_OutString("5000  = "); blank();
    Fixed_uBinOut8(5000);
    swait();
    LCD_OutString("30000 = "); blank();
    Fixed_uBinOut8(30000);
    swait();
    LCD_OutString("65534 = "); blank();
    Fixed_uBinOut8(65534);
    swait();
    LCD_OutString("65535 = "); blank();
    Fixed_uBinOut8(65535);
    swait();

  }

}
```

```
//---------------------Fixed_uDecOut2---------------------
// Takes an unsigned 16-bit integer part of the
// fixed-point number and outputs the fixedpoint
// value on the LCD
// Input: 16-bit unsigned integer
// Output: true if successful
unsigned short Fixed_uDecOut2(unsigned short integer);

//---------------------Fixed_sDecOut3---------------------
// Takes an signed 16-bit integer part of the
// fixed-point number and outputs the fixed point
// value on the LCD
// Input: 16-bit signed integer
// Output: true if successful
unsigned short Fixed_sDecOut3(signed short integer);

//---------------------Fixed_uBinOut8---------------------
// Takes an unsigned 16-bit integer part of the
// binary fixed-point number and outputs the fixed-point value on
// the LCD
// Input: 16-bit unsigned integer
// Output: true if successful
unsigned short Fixed_uBinOut8(unsigned short integer);
```

```
#include <stdio.h>

//---------------------Fixed_uDecOut2---------------------
// Takes an unsigned 16-bit integer part of the
// fixed-point number and outputs the fixedpoint
// value on the LCD
// Input: 16-bit unsigned integer
// Output: true if successful
unsigned short Fixed_uDecOut2(unsigned short integer) {
  if(integer > 65534) { // Out of range is an error
    printf("***.**");
    return 0;
  }
  else { // Splits into integer part and decimal part
    printf("%3d.%02d", integer/100, integer%100);
    return 1;
  }
}


unsigned short Fixed_sDecOut3(signed short integer) {
  if(integer < -9999 || integer > 9999) { // Out of range is an error
    printf(" *.***");
    return 0;
  }
  else { // Splits into integer part and decimal part
    if(integer < 0) { // Prints negative sign
      integer *= -1;
      printf("-%1d.%03d", integer/1000, integer%1000);
    }
    else { // Does not print negative sign
      printf(" %1d.%03d", integer/1000, integer%1000);
    }
    return 1;
  }
}


unsigned short Fixed_uBinOut8(unsigned short integer) {
  if(integer > 65534) { // Out of range is an error
    printf("***.**");
    return 0;
  }
  else {
    unsigned short newInt = (((unsigned long) integer)*100) >> 8; // Bit shifts to
proper value
    printf("%3d.%02d", newInt/100, newInt%100); // Splits into integer part and
decimal part
    return 1;
  }
}
```

```
// filename***************  LCD.H *************************
// 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
*/

//---------------------LCD_Open---------------------
// initialize the LCD display, called once at beginning
// Input: none
// Output: true if successful
short LCD_Open(void);

//---------------------LCD_Clear---------------------
// clear the LCD display, send cursor to home
// Input: none
// Output: true if successful
short LCD_Clear(void);


//---------------------LCD_OutChar---------------------
// sends one ASCII to the LCD display
// Input: letter is ASCII code
// Output: true if successful
short LCD_OutChar(unsigned char letter);


//---------------------LCD_OutString--------------
// Display String
// Input: pointer to NULL-terminationed ASCII string
// Output: true if successful
short LCD_OutString(char *pt);


//---------------------TERMIO_PutChar---------------------
// sends one ASCII to the LCD display
// Input: letter is ASCII code
// handles at least two special characters, like CR LF or TAB
// Output: true if successful
```

```
#define CR 13 // \r
#define TAB 9 // \n
#define LF 10 // \r
short TERMIO_PutChar(unsigned char letter);
```

```
// 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"

static unsigned short OpenFlag=0;

//---------------------wait---------------------
// time delay
// Input: time in 0.667usec
// Output: none
void static wait(unsigned short delay){
unsigned short startTime;
  startTime = TCNT;
  while((TCNT-startTime) <= delay){}
}
//---------------------1mswait---------------------
// time delay
// Input: time in msec
// Output: none
void static wait1ms(unsigned short msec){
  for(;msec;msec--){
    wait(1500);    // 1ms wait
  }
}

//---------------------outCsrNibble---------------------
// sends one command code to the LCD control/status
// Input: command is 4-bit function to execute
// Output: 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
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);                  // blind cycle 90 us wait
}

//---------------------LCD_Clear---------------------
// clear the LCD display, send cursor to home
// Input: none
// Output: true if successful
short LCD_Clear(void){
  if(OpenFlag==0){
    return 0;  // not open
  }
  outCsr(0x01);        // Clear Display
  wait(2460);          // 1.64ms wait
  outCsr(0x02);        // Cursor to home
  wait(2460);          // 1.64ms wait
  return 1;                      // success
}
#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: true if successful
short LCD_Open(void){
  if(OpenFlag){
    return 0;      // 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 1;   // clear display
}

//---------------------LCD_OutChar---------------------
// sends one ASCII to the LCD display
// Input: letter is ASCII code
// Output: true if successful
short LCD_OutChar(unsigned char letter){
  if(OpenFlag==0){
    return 0;  // not open
  }
  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
  wait(135);         // 90 us wait
  return 1;              // success
}

//---------------------LCD_OutString--------------
// Display String
// Input: pointer to NULL-terminationed ASCII string
// Output: true if successful
short LCD_OutString(char *pt){
  if(OpenFlag==0){
    return 0;  // not open
  }
  while(*pt){
    if(LCD_OutChar((unsigned char)*pt)==0){
      return 0;
    }
    pt++;
  }
  return 1;       // success
}


//---------------------TERMIO_PutChar---------------------
// sends one ASCII to the LCD display
// Input: letter is ASCII code
// handles at least two special characters, like CR LF or TAB
// Output: true if successful
#define CR 13 // \r
#define TAB 9 // \n
#define LF 10 // \r
short TERMIO_PutChar(unsigned char letter) {
  if(letter == CR) {
    if(OpenFlag==0){
      return 0;  // not open
    }
    outCsr(0x02);        // Cursor to home
    wait(2460);          // 1.64ms wait
  }
  if(letter == LF) {
    return LCD_Clear();  // Clearscreen
  }
  return LCD_OutChar(letter); // Outputs character to LCD
}
```