```
// --- LCD Display Functions for CL2-R.OLIVA 2009-2017
// Filename: LCD_CL2_3.c
// Updated 04-11-2017
// Adapted from indoor.c & Other..
// 3-5-2010
// Hardware supposed (CL2bm1):
// * PC.4-PC.7 LCDDATA
// * PD.7 LCD_E
// * PC.3 LCD_RW
// * PC.2 LCD_RS
// we need to change wr_half() and wr_disp() to reflect this..
// vCL2 - 9 for CVAVR 2.045 21.1.2010
// Externals eliminated, LCD_Shadow not used..
// VCL2Bm1 -7.5.2010 Use 4x20 display - Project 64-4 onwards
// Rename with underscores LCD_CL2_3.h for portability
// Version 01-2017 - Add Extended LDC1_INCL_H guards (previous LDC1_INCL only half the file(??))
// Rev 4.11.2017 after KIERAS Rule compliance analysis in:
// C:\cvavr328\Work3\CL2\CL2_Drivers\LCD\LCD4x20(2010)\DOC
//      \LCD_4x20(2010)_TESTING FOR HEADER RULES COMPLIANCE(1stIteration)_v04-11-2017.docx
// Rule #5 & #6 violations: Local functions and variables should be kept visible only to .c file,
// not exposed in .h, and defined as static.


// PORT Definitions here:
#include "../inc/lcd_CL2_3.h"


// 21.1.010 Comment out externals..

//extern unsigned char LCD_ADDR, LCD_LINE;
//extern const unsigned char flash addLUT[];
//unsigned char LCD_shadow;
//extern unsigned char *pvdata;
//extern unsigned char vdata;

// Variables for LCD bit access - idea is not to
// disturb I2C bits PC0,1 - Defined in lcd-cl2-1.h
// unsigned char *pvdata;
// unsigned char vdata = 0;
// LCD Port defines
// #define LCD_DB0 PORTC.4
// #define LCD_DB1 PORTC.5
// #define LCD_DB2 PORTC.6
// #define LCD_DB3 PORTC.7
// pvdata = &vdata;
// #define LCD_VAR (*(bits *) pvdata)

// Tested in main() for LCD variable on upper PORTC nibble only.. 6.2.09
//      vdata++;
//      LCD_DB0 = LCD_VAR.bit_0;
//      LCD_DB1 = LCD_VAR.bit_1;
//      LCD_DB2 = LCD_VAR.bit_2;
//      LCD_DB3 = LCD_VAR.bit_3;
//      if(vdata == 15) vdata=0; // 4 bits only.. 6.2.09
//      delay_ms(1000);
//      printf("-LCDVAR=%d ", *pvdata);  // LCD_VAR here gives "6" constant..

// ************************************************************
// LOCAL VARIABLES
// These are local variables, defined as static
// within LCD_CL2_3.c  to comply rules #5, #6 04-11-2017
// Not exposed in LCD_CL2_3.h
// ************************************************************
static unsigned char *pvdata;
static unsigned char vdata = 0;

// Give values of 0 28-01-2017
```

```
static unsigned char LCD_ADDR = 0;
static unsigned char LCD_LINE = 0;

// this Look-Up-Table translates LCD line/cursor positions
// into the displays' interal memory addresses 22.3.2006
unsigned char flash addLUT[4] = {0x80,0xC0,0x94,0xD4};

// Changed place 04-11-2017, local to .c function
#define LCD_VAR (*(bits *) pvdata)

// ************************************************************

// ************************************************************
// LOCAL FUNCTIONS
// These are local functions, defined as static
// within LCD_CL2_3.c  to comply rules #5, #6 04-11-2017
// Not exposed in LCD_CL2_3.h
// ************************************************************
static void wr_half(unsigned char data);
static void wr_disp(unsigned char data);
static void line(char which_line);

// ************************************************************
// Start Display Functions *****
// ************************************************************
void wr_half(unsigned char data)
{
    pvdata = &vdata; // Set pointer to pvdata == LCD_VAR
    LCD_RW = 0;      // set WR active

    LCD_E = 1;       // raise E
    // LCD_PORT = (data & 0x0F); // put data on port - OK for low nibble..
    // LCD_PORT = (data & 0xF0);     // this for HI nibble case 23.3.06 (don't work)
    // LCD_shadow = (LCD_shadow & 0x0F);
    // LCD_shadow |= (data & 0xF0);
    // LCD_PORT = LCD_shadow;
    vdata = (data & 0x0F);    // Lower half of data to global variable..6.2.09
    // contents of pvdata, bit accessible vía LCD_VAR, point to vdata..
    // -------- one bit at a time 6.2.09 -----------
    LCD_DB0 = LCD_VAR.bit_0;
    LCD_DB1 = LCD_VAR.bit_1;
    LCD_DB2 = LCD_VAR.bit_2;
    LCD_DB3 = LCD_VAR.bit_3;
    // ------------------------------------------
    LCD_E = 0;       // drop E

    LCD_RW = 1;      // disable WR

    delay_ms(3);     // allow display time to think
}

void wr_disp(unsigned char data)
{
    pvdata = &vdata; // Set pointer to pvdata == LCD_VAR
    LCD_RW = 0;      // enable write..

    LCD_E = 1;

    // LCD_PORT= (data >> 4);    23.3 Invert
    order..
    // LCD_PORT = (data & 0xF0);     // this for HI nibble case 23.3.06 (don't work)
    vdata = (data >> 4);    // Upper half of data to global variable..6.2.09
    // contents of pvdata, bit accessible vía LCD_VAR, point to vdata..
    // -------- one bit at a time 6.2.09 -----------
    LCD_DB0 = LCD_VAR.bit_0;
    LCD_DB1 = LCD_VAR.bit_1;
    LCD_DB2 = LCD_VAR.bit_2;
```

```
    LCD_DB3 = LCD_VAR.bit_3;
    // ---------------------------------------------
    LCD_E = 0;      // strobe upper half of data

    LCD_E = 1;      // out to the display
    // LCD_PORT = (data & 0x0F);  23.3 Invert
    order..
    // LCD_PORT = ((data << 4) & 0xF0); // Send lower half "up"..23.3.06 don't work..
    vdata = (data & 0x0F);    // Lower half of data to global variable..6.2.09
    // contents of pvdata, bit accessible vía LCD_VAR, point to vdata..
    // -------- one bit at a time 6.2.09 -----------
    LCD_DB0 = LCD_VAR.bit_0;
    LCD_DB1 = LCD_VAR.bit_1;
    LCD_DB2 = LCD_VAR.bit_2;
    LCD_DB3 = LCD_VAR.bit_3;
    // ---------------------------------------------
    LCD_E = 0;      // now, strobe the lower half

    LCD_RW = 1;     // disable write

    delay_ms(3);    // allow time for LCD to react
}

void init_display(void)
{
    char i;

    LCD_RW = 1;
    LCD_E = 0;        // preset interface signals..
    LCD_RS = 0;       // command mode..
    delay_ms(50);

    wr_half(0x33);    // This sequence enables the display
    wr_half(0x33);    // for 4-bit interface mode
    wr_half(0x33);    // these commands can be found
    wr_half(0x22);    // in the manufacturer's data sheets

    wr_disp(0x28);    // Enable the internal 5x7 font
    wr_disp(0x01);
    wr_disp(0x10);    // set cursor to move (instead of display shift)
    wr_disp(0x06);    // set the cursor to move right, and not shift the display
    wr_disp(0x0c);    // turns display on, cursor off, and cursor blinking off

    for(i=0x40; i<0x5F; i++) // init character font ram to "00"
    {
        delay_ms(10);
        LCD_RS = 0;
        wr_disp(i);
        delay_ms(10);
        disp_char(0);
    }
    LCD_RS = 1;         // data mode
}

void line(char which_line)
{
    LCD_RS = 0;

    LCD_ADDR = addLUT[which_line-1];
    wr_disp(LCD_ADDR); // move to which_line line

    LCD_RS = 1;
    LCD_ADDR = 0;
    LCD_LINE = which_line;
}

void clear_display(void)
```

```
{
    LCD_RS = 0;
    wr_disp(0x01);            // clear display and home cursor
    delay_ms(10);
    LCD_RS = 1;
    line(1);
}

void set_LCD_cur(char LCDrow, char LCDcol)
{
    LCD_RS = 0;

    LCD_ADDR = addLUT[LCDrow] + LCDcol;

    wr_disp(LCD_ADDR); // move to row and column

    LCD_RS = 1;
    LCD_LINE = LCDrow;
}

void disp_char(unsigned char c)
{
    LCD_RS = 1;
    wr_disp(c);
    LCD_ADDR++;             // automatically wrap text as required
    if(LCD_ADDR == 20)
        line(2);
    if(LCD_ADDR == 40)
        line(3);
    if(LCD_ADDR == 60)
        line(4);
    if (LCD_ADDR == 80)
        line(1);
}

void disp_str(unsigned char *sa)    // display string from RAM
{
    while(*sa != 0)
      disp_char(*sa++);
}

void disp_cstr(unsigned char flash *sa) /* display string from ROM */
{
    while(*sa != 0)
      disp_char(*sa++);
}
```