```
/*************************************************************************/
/*                                                                       */
/*                                 LCDOUT                                */
/*                            LCD Output Functions                       */
/*                         Digital Oscilloscope Project                  */
/*                                 EE/CS 52                              */
/*                                                                       */
/*************************************************************************/

/*
   This file contains the functions for doing output to the LCD screen for the
   Digital Oscilloscope project.  The functions included are:
      clear_region - clear a region of the display
      plot_char    - output a character
      plot_hline   - draw a horizontal line
      plot_string  - output a string
      plot_vline   - draw a vertical line
      plot_cursor  - plot the cursor

   The local functions included are:
      none

   The locally global variable definitions included are:
      none


   Revision History
      3/8/94   Glen George       Initial revision.
      3/13/94  Glen George       Updated comments.
      3/13/94  Glen George       Simplified code in plot_string function.
      3/17/97  Glen George       Updated comments.
      3/17/97  Glen George       Change plot_char() and plot_string() to use
                                 enum char_style instead of an int value.
      5/27/98  Glen George       Change plot_char() to explicitly declare the
                                 size of the external array to avoid linker
                                 errors.
      6/3/14   Santiago Navonne  Changed UI display colors, added support for
                                 highlighted characters.
*/



/* library include files */
  /* none */

/* local include files */
#include  "interfac.h"
#include  "scopedef.h"
#include  "lcdout.h"


extern int pixel_color(int, int);



/*
   clear_region

   Description:      This function clears the passed region of the display.
                     The region is described by its upper left corner pixel
                     coordinate and the size (in pixels) in each dimension.

   Arguments:        x_ul (int)   - x coordinate of upper left corner of the
                    region to be cleared.
              y_ul (int)   - y coordinate of upper left corner of the
                     region to be cleared.
              x_size (int) - horizontal size of the region.
              y_size (int) - vertical size of the region.
   Return Value:     None.

   Input:            None.
   Output:           A portion of the screen is cleared (set to PIXEL_CLEAR).

   Error Handling:   No error checking is done on the coordinates.

```

```
   Algorithms:       None.
   Data Structures:  None.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    June 03, 2014

*/

void  clear_region(int x_ul, int y_ul, int x_size, int y_size)
{
    /* variables */
    int  x;     /* x coordinate to clear */
    int  y;     /* y coordinate to clear */



    /* loop, clearing the display */
    for (x = x_ul; x < (x_ul + x_size); x++)  {
        for (y = y_ul; y < (y_ul + y_size); y++)  {

        /* clear this pixel */
        plot_pixel(x, y, PIXEL_CLEAR);
        }
    }


    /* done clearing the display region - return */
    return;

}




/*
   plot_hline

   Description:      This function draws a horizontal line from the passed
                     position for the passed length.  The line is always drawn
                     with the color PIXEL_LINE.  The position (0,0) is the
             upper left corner of the screen.

   Arguments:        start_x (int) - starting x coordinate of the line.
             start_y (int) - starting y coordinate of the line.
             length (int)  - length of the line (positive for a line
                         to the "right" and negative for a line to
                     the "left").
   Return Value:     None.

   Input:            None.
   Output:           A horizontal line is drawn at the specified position.

   Error Handling:   No error checking is done on the coordinates.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    June 03, 2014

*/

void  plot_hline(int start_x, int start_y, int length)
{
    /* variables */
    int  x;     /* x position while plotting */

    int  init_x;    /* starting x position to plot */
    int  end_x;     /* ending x position to plot */


```

```
151
152     /* check if a line to the "right" or "left" */
153     if (length > 0)  {
154
155         /* line to the "right" - start at start_x, end at start_x + length */
156     init_x = start_x;
157     end_x = start_x + length;
158     }
159     else  {
160
161         /* line to the "left" - start at start_x + length, end at start_x */
162     init_x = start_x + length;
163     end_x = start_x;
164     }
165
166
167     /* loop, outputting points for the line (always draw to the "right") */
168     for (x = init_x; x < end_x; x++)
169         /* plot a point of the line */
170     plot_pixel(x, start_y, PIXEL_LINE);
171
172
173     /* done plotting the line - return */
174     return;
175
176 }
177
178
179
180
181 /*
182    plot_vline
183
184    Description:      This function draws a vertical line from the passed
185                      position for the passed length.  The line is always drawn
186                      with the color PIXEL_LINE.  The position (0,0) is the
187              upper left corner of the screen.
188
189    Arguments:        start_x (int) - starting x coordinate of the line.
190              start_y (int) - starting y coordinate of the line.
191              length (int)  - length of the line (positive for a line
192                          going "down" and negative for a line
193                      going "up").
194    Return Value:     None.
195
196    Input:            None.
197    Output:           A vertical line is drawn at the specified position.
198
199    Error Handling:   No error checking is done on the coordinates.
200
201    Algorithms:       None.
202    Data Structures:  None.
203
204    Global Variables: None.
205
206    Author:           Glen George
207    Last Modified:    June 03, 2014
208
209 */
210
211 void  plot_vline(int start_x, int start_y, int length)
212 {
213     /* variables */
214     int  y;     /* y position while plotting */
215
216     int  init_y;    /* starting y position to plot */
217     int  end_y;     /* ending y position to plot */
218
219
220
221     /* check if an "up" or "down" line */
222     if (length > 0)  {
223
224         /* line going "down" - start at start_y, end at start_y + length */
225     init_y = start_y;
```

```
    end_y = start_y + length;
    }
    else  {

        /* line going "up" - start at start_y + length, end at start_y */
    init_y = start_y + length;
    end_y = start_y;
    }


    /* loop, outputting points for the line (always draw "down") */
    for (y = init_y; y < end_y; y++)
        /* plot a point of the line */
    plot_pixel(start_x, y, PIXEL_LINE);


    /* done plotting the line - return */
    return;

}




/*
   plot_char

   Description:      This function outputs the passed character to the LCD
                     screen at passed location.  The passed location is given
                     as a character position with (0,0) being the upper left
              corner of the screen.  The character can be drawn in
              "normal video" (gray on black), "reverse video" (black
              on gray), or highlighted (white on black).

   Arguments:        pos_x (int)             - x coordinate (in character
                                 cells) of the character.
              pos_y (int)             - y coordinate (in character
                                 cells) of the character.
              c (char)                - the character to plot.
              style (enum char_style) - style with which to plot the
                                    character (NORMAL or REVERSE).
   Return Value:     None.

   Input:            None.
   Output:           A character is output to the LCD screen.

   Error Handling:   No error checking is done on the coordinates or the
              character (to ensure there is a bit pattern for it).

   Algorithms:       None.
   Data Structures:  The character bit patterns are stored in an external
              array.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    June 03, 2014

*/

void  plot_char(int pos_x, int pos_y, char c, enum char_style style)
{
    /* variables */

    /* pointer to array of character bit patterns */
    extern const unsigned char   char_patterns[(VERT_SIZE - 1) * 128];

    int  bits;          /* a character bit pattern */

    int  col;       /* column loop index */
    int  row;           /* character row loop index */

    int  x;     /* x pixel position for the character */
    int  y;     /* y pixel position for the character */

```

```
    int color = PIXEL_TEXT_N; /* pixel drawing color */



    /* setup the pixel positions for the character */
    x = pos_x * HORIZ_SIZE;
    y = pos_y * VERT_SIZE;


    /* loop outputting the bits to the screen */
    for (row = 0; row < VERT_SIZE; row++)  {

        /* get the character bits for this row from the character table */
    if (row == (VERT_SIZE - 1))
        /* last row - blank it */
        bits = 0;
    else
        /* in middle of character, get the row from the bit patterns */
            bits = char_patterns[(c * (VERT_SIZE - 1)) + row];

    /* take care of "normal/reverse video" */
    if (style == REVERSE)
        /* invert the bits for "reverse video" */
        bits = ~bits;
  if (style == HIGHLIGHTED)
      color = PIXEL_TEXT_H;

        /* get the bits "in position" (high bit is output first */
    bits <<= (8 - HORIZ_SIZE);


    /* now output the row of the character, pixel by pixel */
    for (col = 0; col < HORIZ_SIZE; col++)  {

            /* output this pixel in the appropriate color */
        if ((bits & 0x80) == 0)
            /* blank pixel - output in PIXEL_CLEAR */
        plot_pixel(x + col, y, PIXEL_CLEAR);
        else
            /* black pixel - output in PIXEL_TEXT */
        plot_pixel(x + col, y, color);

        /* shift the next bit into position */
        bits <<= 1;
        }


    /* next row - update the y position */
    y++;
    }


    /* all done, return */
    return;

}



/*
   plot_string

   Description:      This function outputs the passed string to the LCD screen
                     at passed location.  The passed location is given as a
                     character position with (0,0) being the upper left corner
             of the screen.  There is no line wrapping, so the entire
             string must fit on the passed line (pos_y).  The string
             can be drawn in "normal video" (black on white) or
             "reverse video" (white on black).

   Arguments:        pos_x (int)             - x coordinate (in character
                            cells) of the start of the
                          string.
             pos_y (int)             - y coordinate (in character
```

```
                                cells) of the start of the
                              string.
              s (const char *)        - the string to output.
              style (enum char style) - style with which to plot
                                        characters of the string.
   Return Value:     None.

   Input:            None.
   Output:           A string is output to the LCD screen.

   Error Handling:   No checking is done to insure the string is fully on the
             screen (the x and y coordinates and length of the string
             are not checked).

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    Mar. 17, 1997

*/

void  plot_string(int pos_x, int pos_y, const char *s, enum char_style style)
{
    /* variables */
      /* none */



    /* loop, outputting characters from string s */
    while (*s != '\0')

        /* output this character and move to the next character and screen position */
    plot_char(pos_x++, pos_y, *s++, style);


    /* all done, return */
    return;

}
```