

```

1  /*****
2  /*
3  /*                      LCDOUT                      */
4  /*                      LCD Output Functions          */
5  /*                      Digital Oscilloscope Project  */
6  /*                      EE/CS 52                      */
7  /*
8  /*****
9
10 /*
11 This file contains the functions for doing output to the LCD screen for the
12 Digital Oscilloscope project. The functions included are:
13     clear_region - clear a region of the display
14     plot_char   - output a character
15     plot_hline  - draw a horizontal line
16     plot_string - output a string
17     plot_vline  - draw a vertical line
18     plot_cursor - plot the cursor
19
20 The local functions included are:
21     none
22
23 The locally global variable definitions included are:
24     none
25
26
27 Revision History
28     3/8/94   Glen George      Initial revision.
29     3/13/94  Glen George      Updated comments.
30     3/13/94  Glen George      Simplified code in plot_string function.
31     3/17/97  Glen George      Updated comments.
32     3/17/97  Glen George      Change plot_char() and plot_string() to use
33                               enum char_style instead of an int value.
34     5/27/98  Glen George      Change plot_char() to explicitly declare the
35                               size of the external array to avoid linker
36                               errors.
37     6/3/14   Santiago Navonne Changed UI display colors, added support for
38                               highlighted characters.
39 */
40
41
42
43 /* library include files */
44 /* none */
45
46 /* local include files */
47 #include "interfac.h"
48 #include "scopedef.h"
49 #include "lcdout.h"
50
51
52 extern int pixel_color(int, int);
53
54
55
56 /*
57     clear_region
58
59 Description:      This function clears the passed region of the display.
60                   The region is described by its upper left corner pixel
61                   coordinate and the size (in pixels) in each dimension.
62
63 Arguments:      x_ul (int)   - x coordinate of upper left corner of the
64                   region to be cleared.
65                   y_ul (int) - y coordinate of upper left corner of the
66                   region to be cleared.
67                   x_size (int) - horizontal size of the region.
68                   y_size (int) - vertical size of the region.
69 Return Value:    None.
70
71 Input:           None.
72 Output:          A portion of the screen is cleared (set to PIXEL_CLEAR).
73
74 Error Handling:  No error checking is done on the coordinates.
75

```

```

76 Algorithms:      None.
77 Data Structures: None.
78
79 Global Variables: None.
80
81 Author:          Glen George
82 Last Modified:   June 03, 2014
83
84 */
85
86 void clear_region(int x_ul, int y_ul, int x_size, int y_size)
87 {
88     /* variables */
89     int x;        /* x coordinate to clear */
90     int y;        /* y coordinate to clear */
91
92
93
94     /* loop, clearing the display */
95     for (x = x_ul; x < (x_ul + x_size); x++) {
96         for (y = y_ul; y < (y_ul + y_size); y++) {
97
98             /* clear this pixel */
99             plot_pixel(x, y, PIXEL_CLEAR);
100         }
101     }
102
103
104     /* done clearing the display region - return */
105     return;
106 }
107
108
109
110
111
112 /*
113 plot_hline
114
115 Description:      This function draws a horizontal line from the passed
116                   position for the passed length. The line is always drawn
117                   with the color PIXEL_LINE. The position (0,0) is the
118                   upper left corner of the screen.
119
120 Arguments:        start_x (int) - starting x coordinate of the line.
121                   start_y (int) - starting y coordinate of the line.
122                   length (int) - length of the line (positive for a line
123                               to the "right" and negative for a line to
124                               the "left").
125 Return Value:     None.
126
127 Input:            None.
128 Output:           A horizontal line is drawn at the specified position.
129
130 Error Handling:   No error checking is done on the coordinates.
131
132 Algorithms:      None.
133 Data Structures: None.
134
135 Global Variables: None.
136
137 Author:          Glen George
138 Last Modified:   June 03, 2014
139
140 */
141
142 void plot_hline(int start_x, int start_y, int length)
143 {
144     /* variables */
145     int x;        /* x position while plotting */
146
147     int init_x;    /* starting x position to plot */
148     int end_x;     /* ending x position to plot */
149
150

```

```

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;

```

```

226     end_y = start_y + length;
227 }
228 else {
229
230     /* line going "up" - start at start_y + length, end at start_y */
231     init_y = start_y + length;
232     end_y = start_y;
233 }
234
235
236 /* loop, outputting points for the line (always draw "down") */
237 for (y = init_y; y < end_y; y++)
238     /* plot a point of the line */
239     plot_pixel(start_x, y, PIXEL_LINE);
240
241
242 /* done plotting the line - return */
243 return;
244
245 }
246
247
248
249
250 /*
251  plot_char
252
253  Description:      This function outputs the passed character to the LCD
254                    screen at passed location. The passed location is given
255                    as a character position with (0,0) being the upper left
256                    corner of the screen. The character can be drawn in
257                    "normal video" (gray on black), "reverse video" (black
258                    on gray), or highlighted (white on black).
259
260  Arguments:      pos_x (int)          - x coordinate (in character
261                    cells) of the character.
262                    pos_y (int)        - y coordinate (in character
263                    cells) of the character.
264                    c (char)           - the character to plot.
265                    style (enum char_style) - style with which to plot the
266                    character (NORMAL or REVERSE).
267
268  Return Value:   None.
269
270  Input:          None.
271  Output:         A character is output to the LCD screen.
272
273  Error Handling:  No error checking is done on the coordinates or the
274                    character (to ensure there is a bit pattern for it).
275
276  Algorithms:     None.
277  Data Structures: The character bit patterns are stored in an external
278                    array.
279
280  Global Variables: None.
281
282  Author:         Glen George
283  Last Modified:  June 03, 2014
284 */
285
286 void plot_char(int pos_x, int pos_y, char c, enum char_style style)
287 {
288     /* variables */
289
290     /* pointer to array of character bit patterns */
291     extern const unsigned char char_patterns[(VERT_SIZE - 1) * 128];
292
293     int bits;          /* a character bit pattern */
294
295     int col;           /* column loop index */
296     int row;           /* character row loop index */
297
298     int x;             /* x pixel position for the character */
299     int y;             /* y pixel position for the character */
300

```

```

301 int color = PIXEL_TEXT_N; /* pixel drawing color */
302
303
304
305 /* setup the pixel positions for the character */
306 x = pos_x * HORIZ_SIZE;
307 y = pos_y * VERT_SIZE;
308
309
310 /* loop outputting the bits to the screen */
311 for (row = 0; row < VERT_SIZE; row++) {
312
313     /* get the character bits for this row from the character table */
314     if (row == (VERT_SIZE - 1))
315         /* last row - blank it */
316         bits = 0;
317     else
318         /* in middle of character, get the row from the bit patterns */
319         bits = char_patterns[(c * (VERT_SIZE - 1)) + row];
320
321     /* take care of "normal/reverse video" */
322     if (style == REVERSE)
323         /* invert the bits for "reverse video" */
324         bits = ~bits;
325     if (style == HIGHLIGHTED)
326         color = PIXEL_TEXT_H;
327
328     /* get the bits "in position" (high bit is output first */
329     bits <= (8 - HORIZ_SIZE);
330
331
332     /* now output the row of the character, pixel by pixel */
333     for (col = 0; col < HORIZ_SIZE; col++) {
334
335         /* output this pixel in the appropriate color */
336         if ((bits & 0x80) == 0)
337             /* blank pixel - output in PIXEL_CLEAR */
338             plot_pixel(x + col, y, PIXEL_CLEAR);
339         else
340             /* black pixel - output in PIXEL_TEXT */
341             plot_pixel(x + col, y, color);
342
343         /* shift the next bit into position */
344         bits <= 1;
345     }
346
347
348     /* next row - update the y position */
349     y++;
350 }
351
352
353 /* all done, return */
354 return;
355
356 }
357
358
359
360
361 /*
362 plot_string
363
364 Description:      This function outputs the passed string to the LCD screen
365                   at passed location. The passed location is given as a
366                   character position with (0,0) being the upper left corner
367                   of the screen. There is no line wrapping, so the entire
368                   string must fit on the passed line (pos_y). The string
369                   can be drawn in "normal video" (black on white) or
370                   "reverse video" (white on black).
371
372 Arguments:      pos_x (int)          - x coordinate (in character
373                   cells) of the start of the
374                   string.
375                   pos_y (int)        - y coordinate (in character

```

```

376         cells) of the start of the
377         string.
378         s (const char *)           - the string to output.
379         style (enum char style) - style with which to plot
380         characters of the string.
381 Return Value:      None.
382
383 Input:             None.
384 Output:           A string is output to the LCD screen.
385
386 Error Handling:    No checking is done to insure the string is fully on the
387                   screen (the x and y coordinates and length of the string
388                   are not checked).
389
390 Algorithms:        None.
391 Data Structures:   None.
392
393 Global Variables:  None.
394
395 Author:            Glen George
396 Last Modified:     Mar. 17, 1997
397
398 */
399
400 void plot_string(int pos_x, int pos_y, const char *s, enum char_style style)
401 {
402     /* variables */
403     /* none */
404
405
406
407     /* loop, outputting characters from string s */
408     while (*s != '\0')
409
410         /* output this character and move to the next character and screen position */
411         plot_char(pos_x++, pos_y, *s++, style);
412
413
414     /* all done, return */
415     return;
416
417 }
418

```