

```

1 // Graphics LCD Library
2 // Jason Losh
3
4 //-----
5 // Hardware Target
6 //-----
7
8 // Target Platform: EK-TM4C123GXL with LCD/Keyboard Interface
9 // Target uC:      TM4C123GH6PM
10 // System Clock:   40 MHz
11
12 // Hardware configuration:
13 // ST7565R Graphics LCD Display Interface:
14 //   MOSI on PD3 (SSI1Tx)
15 //   SCLK on PD0 (SSI1Clk)
16 //   ~CS on PD1 (SSI1Fss)
17 //   A0 connected to PD2
18
19 //-----
20 // Device includes, defines, and assembler directives
21 //-----
22
23 #include <stdint.h>
24 #include <stdbool.h>
25 #include <string.h>
26 #include "tm4c123gh6pm.h"
27 #include "graphics_lcd.h"
28 #include "gpio.h"
29 #include "spi1.h"
30
31 // Pins
32 #define A0 PORTD,2
33
34 //-----
35 // Global variables
36 //-----
37
38 uint8_t pixelMap[1024];
39 uint16_t txtIndex = 0;
40
41 // 96 character 5x7 bitmaps based on ISO-646 (BCT IRV extensions)
42 const uint8_t charGen[100][5] = {
43     // Codes 32-127
44     // Space ! " % $ % & ' ( ) * + , - . /
45     {0x00, 0x00, 0x00, 0x00, 0x00},
46     {0x00, 0x00, 0x4F, 0x00, 0x00},
47     {0x00, 0x07, 0x00, 0x07, 0x00},
48     {0x14, 0x7F, 0x14, 0x7F, 0x14},
49     {0x24, 0x2A, 0x7F, 0x2A, 0x12},
50     {0x23, 0x13, 0x08, 0x64, 0x62},
51     {0x36, 0x49, 0x55, 0x22, 0x40},
52     {0x00, 0x05, 0x03, 0x00, 0x00},
53     {0x00, 0x1C, 0x22, 0x41, 0x00},
54     {0x00, 0x41, 0x22, 0x1C, 0x00},
55     {0x14, 0x08, 0x3E, 0x08, 0x14},
56     {0x08, 0x08, 0x3E, 0x08, 0x08},
57     {0x00, 0x50, 0x30, 0x00, 0x00},
58     {0x08, 0x08, 0x08, 0x08, 0x08},
59     {0x00, 0x60, 0x60, 0x00, 0x00},
60     {0x20, 0x10, 0x08, 0x04, 0x02},
61     // 0-9
62     {0x3E, 0x51, 0x49, 0x45, 0x3E},
63     {0x00, 0x42, 0x7F, 0x40, 0x00},
64     {0x42, 0x61, 0x51, 0x49, 0x46},
65     {0x21, 0x41, 0x45, 0x4B, 0x31},
66     {0x18, 0x14, 0x12, 0x7F, 0x10},
67     {0x27, 0x45, 0x45, 0x45, 0x39},
68     {0x3C, 0x4A, 0x49, 0x49, 0x30},
69     {0x01, 0x71, 0x09, 0x05, 0x03},

```

```
70 {0x36, 0x49, 0x49, 0x49, 0x36},
71 {0x06, 0x49, 0x49, 0x29, 0x1E},
72 // : ; < = > ? @
73 {0x00, 0x36, 0x36, 0x00, 0x00},
74 {0x00, 0x56, 0x36, 0x00, 0x00},
75 {0x08, 0x14, 0x22, 0x41, 0x00},
76 {0x14, 0x14, 0x14, 0x14, 0x14},
77 {0x00, 0x41, 0x22, 0x14, 0x08},
78 {0x02, 0x01, 0x51, 0x09, 0x3E},
79 {0x32, 0x49, 0x79, 0x41, 0x3E},
80 // A-Z
81 {0x7E, 0x11, 0x11, 0x11, 0x7E},
82 {0x7F, 0x49, 0x49, 0x49, 0x36},
83 {0x3E, 0x41, 0x41, 0x41, 0x22},
84 {0x7F, 0x41, 0x41, 0x22, 0x1C},
85 {0x7F, 0x49, 0x49, 0x49, 0x41},
86 {0x7F, 0x09, 0x09, 0x09, 0x01},
87 {0x3E, 0x41, 0x49, 0x49, 0x3A},
88 {0x7F, 0x08, 0x08, 0x08, 0x7F},
89 {0x00, 0x41, 0x7F, 0x41, 0x00},
90 {0x20, 0x40, 0x41, 0x3F, 0x01},
91 {0x7F, 0x08, 0x14, 0x22, 0x41},
92 {0x7F, 0x40, 0x40, 0x40, 0x40},
93 {0x7F, 0x02, 0x0C, 0x02, 0x7F},
94 {0x7F, 0x04, 0x08, 0x10, 0x7F},
95 {0x3E, 0x41, 0x41, 0x41, 0x3E},
96 {0x7F, 0x09, 0x09, 0x09, 0x06},
97 {0x3E, 0x41, 0x51, 0x21, 0x5E},
98 {0x7F, 0x09, 0x19, 0x29, 0x46},
99 {0x46, 0x49, 0x49, 0x49, 0x31},
100 {0x01, 0x01, 0x7F, 0x01, 0x01},
101 {0x3F, 0x40, 0x40, 0x40, 0x3F},
102 {0x1F, 0x20, 0x40, 0x20, 0x1F},
103 {0x3F, 0x40, 0x70, 0x40, 0x3F},
104 {0x63, 0x14, 0x08, 0x14, 0x63},
105 {0x07, 0x08, 0x70, 0x08, 0x07},
106 {0x61, 0x51, 0x49, 0x45, 0x43},
107 // [ \ ] ^ _ `
108 {0x00, 0x7F, 0x41, 0x41, 0x00},
109 {0x02, 0x04, 0x08, 0x10, 0x20},
110 {0x00, 0x41, 0x41, 0x7F, 0x00},
111 {0x04, 0x02, 0x01, 0x02, 0x04},
112 {0x40, 0x40, 0x40, 0x40, 0x40},
113 {0x00, 0x01, 0x02, 0x04, 0x00},
114 // a-z
115 {0x20, 0x54, 0x54, 0x54, 0x78},
116 {0x7F, 0x44, 0x44, 0x44, 0x38},
117 {0x38, 0x44, 0x44, 0x44, 0x20},
118 {0x38, 0x44, 0x44, 0x48, 0x7F},
119 {0x38, 0x54, 0x54, 0x54, 0x18},
120 {0x08, 0x7E, 0x09, 0x01, 0x02},
121 {0x0C, 0x52, 0x52, 0x52, 0x3E},
122 {0x7F, 0x08, 0x04, 0x04, 0x78},
123 {0x00, 0x44, 0x7D, 0x40, 0x00},
124 {0x20, 0x40, 0x44, 0x3D, 0x00},
125 {0x7F, 0x10, 0x28, 0x44, 0x00},
126 {0x00, 0x41, 0x7F, 0x40, 0x00},
127 {0x7C, 0x04, 0x18, 0x04, 0x78},
128 {0x7C, 0x08, 0x04, 0x04, 0x78},
129 {0x38, 0x44, 0x44, 0x44, 0x38},
130 {0x7C, 0x14, 0x14, 0x14, 0x08},
131 {0x08, 0x14, 0x14, 0x18, 0x7C},
132 {0x7C, 0x08, 0x04, 0x04, 0x08},
133 {0x48, 0x54, 0x54, 0x54, 0x20},
134 {0x04, 0x3F, 0x44, 0x40, 0x20},
135 {0x3C, 0x40, 0x40, 0x20, 0x7C},
136 {0x1C, 0x20, 0x40, 0x20, 0x1C},
137 {0x3C, 0x40, 0x20, 0x40, 0x3C},
138 {0x44, 0x28, 0x10, 0x28, 0x44},
```

```

139     {0x0C, 0x50, 0x50, 0x50, 0x3C},
140     {0x44, 0x64, 0x54, 0x4C, 0x44},
141     // { | } ~ cc
142     {0x00, 0x08, 0x36, 0x41, 0x00},
143     {0x00, 0x00, 0x7F, 0x00, 0x00},
144     {0x00, 0x41, 0x36, 0x08, 0x00},
145     {0x0C, 0x04, 0x1C, 0x10, 0x18},
146     {0x00, 0x00, 0x00, 0x00, 0x00},
147     // Custom assignments beyond ISO646
148     // Codes 128+: right arrow, left arrow, degree sign
149     {0x08, 0x08, 0x2A, 0x1C, 0x08},
150     {0x08, 0x1C, 0x2A, 0x08, 0x08},
151     {0x07, 0x05, 0x07, 0x00, 0x00},
152 };
153
154 //-----
155 // Subroutines
156 //-----
157
158 // Blocking function that writes data to the SPI bus and waits for the data to complete
159 // transmission
160 void sendGraphicsLcdCommand(uint8_t command)
161 {
162     setPinValue(A0, 0);           // clear A0 for commands
163     writeSpi1Data(command);
164 }
165
166 // Blocking function that writes data to the SPI bus and waits for the data to complete
167 // transmission
168 void sendGraphicsLcdData(uint8_t data)
169 {
170     setPinValue(A0, 1);           // set A0 for data
171     writeSpi1Data(data);
172 }
173
174 void setGraphicsLcdPage(uint8_t page)
175 {
176     sendGraphicsLcdCommand(0xB0 | page);
177 }
178
179 void setGraphicsLcdColumn(uint8_t x)
180 {
181     sendGraphicsLcdCommand(0x10 | (x >> 4));
182     sendGraphicsLcdCommand(0x00 | (x & 0x0F));
183 }
184
185 void refreshGraphicsLcd()
186 {
187     uint8_t x, page;
188     uint16_t i = 0;
189     for (page = 0; page < 8; page++)
190     {
191         setGraphicsLcdPage(page);
192         setGraphicsLcdColumn(0);
193         for (x = 0; x < 128; x++)
194             sendGraphicsLcdData(pixelMap[i++]);
195     }
196 }
197
198 void clearGraphicsLcd()
199 {
200     uint16_t i;
201     // clear data memory pixel map
202     for (i = 0; i < 1024; i++)
203         pixelMap[i] = 0;
204     // copy to display
205     refreshGraphicsLcd();
206 }

```

```

206 void drawGraphicsLcdPixel(uint8_t x, uint8_t y, enum operation op)
207 {
208     uint8_t data, mask, page;
209     uint16_t index;
210
211     // determine pixel map entry
212     page = y >> 3;
213
214     // determine pixel map index
215     index = page << 7 | x;
216
217     // generate mask
218     mask = 1 << (y & 7);
219
220     // read pixel map
221     data = pixelMap[index];
222
223     // apply operator
224     switch(op)
225     {
226         case CLEAR: data &= ~mask; break;
227         case SET: data |= mask; break;
228         case INVERT: data ^= mask; break;
229     }
230
231     // write to pixel map
232     pixelMap[index] = data;
233
234     // write to display
235     setGraphicsLcdPage(page);
236     setGraphicsLcdColumn(x);
237     sendGraphicsLcdData(data);
238 }
239
240 void drawGraphicsLcdRectangle(uint8_t xul, uint8_t yul, uint8_t dx, uint8_t dy, enum
operation op)
241 {
242     uint8_t page, page_start, page_stop;
243     uint8_t bit_index, bit_start, bit_stop;
244     uint8_t mask, data;
245     uint16_t index;
246     uint8_t x;
247
248     // determine pages for rectangle
249     page_start = yul >> 3;
250     page_stop = (yul + dy - 1) >> 3;
251
252     // draw in pages from top to bottom within extent
253     for (page = page_start; page <= page_stop; page++)
254     {
255         // calculate mask for this page
256         if (page > page_start)
257             bit_start = 0;
258         else
259             bit_start = yul & 7;
260         if (page < page_stop)
261             bit_stop = 7;
262         else
263             bit_stop = (yul + dy - 1) & 7;
264         mask = 0;
265         for (bit_index = bit_start; bit_index <= bit_stop; bit_index++)
266             mask |= 1 << bit_index;
267
268         // write page
269         setGraphicsLcdPage(page);
270         setGraphicsLcdColumn(xul);
271         index = (page << 7) | xul;
272         for (x = 0; x < dx; x++)
273         {

```

```

274         // read pixel map
275         data = pixelMap[index];
276         // apply operator (0 = clear, 1 = set, 2 = xor)
277         switch(op)
278         {
279             case CLEAR: data &= ~mask; break;
280             case SET:   data |= mask; break;
281             case INVERT: data ^= mask; break;
282         }
283         // write to pixel map
284         pixelMap[index++] = data;
285         // write to display
286         sendGraphicsLcdData(data);
287     }
288 }
289 }
290
291 void setGraphicsLcdTextPosition(uint8_t x, uint8_t page)
292 {
293     txtIndex = (page << 7) + x;
294     setGraphicsLcdPage(page);
295     setGraphicsLcdColumn(x);
296 }
297
298 void putcGraphicsLcd(char c)
299 {
300     uint8_t i, val;
301     uint8_t uc;
302     // convert to unsigned to access characters > 127
303     uc = (uint8_t) c;
304     for (i = 0; i < 5; i++)
305     {
306         val = charGen[uc-' '][i];
307         pixelMap[txtIndex++] = val;
308         sendGraphicsLcdData(val);
309     }
310     pixelMap[txtIndex++] = 0;
311     sendGraphicsLcdData(0);
312 }
313
314 void putsGraphicsLcd(char str[])
315 {
316     uint8_t i = 0;
317     while (str[i] != 0)
318         putcGraphicsLcd(str[i++]);
319 }
320
321 void initGraphicsLcd()
322 {
323     // Initialize SPI1 interface
324     initSpil(USE_SSI_FSS);
325     setSpilBaudRate(1e6, 40e6);
326     setSpilMode(1, 1);
327
328     // Enable clocks
329     enablePort(PORTD);
330
331     // Configure A0 for graphics LCD
332     selectPinPushPullOutput(A0);
333
334     // Initialize display
335     sendGraphicsLcdCommand(0x40); // set start line to 0
336     sendGraphicsLcdCommand(0xA1); // reverse horizontal order
337     sendGraphicsLcdCommand(0xC0); // normal vertical order
338     sendGraphicsLcdCommand(0xA6); // normal pixel polarity
339     sendGraphicsLcdCommand(0xA2); // set led bias to 1/9 (should be A2)
340     sendGraphicsLcdCommand(0x2F); // turn on voltage booster and regulator
341     sendGraphicsLcdCommand(0xF8); // set internal volt booster to 4x Vdd
342     sendGraphicsLcdCommand(0x00);

```

```
343     sendGraphicsLcdCommand(0x27); // set contrast
344     sendGraphicsLcdCommand(0x81); // set LCD drive voltage
345     sendGraphicsLcdCommand(0x16);
346     sendGraphicsLcdCommand(0xAC); // no flashing indicator
347     sendGraphicsLcdCommand(0x00);
348     clearGraphicsLcd();           // clear display
349     sendGraphicsLcdCommand(0xAF); // display on
350 }
351
```