```
//18111C Master
1
2
    /*----*/
3
    // DriverLCD.c
    /*----*/
4
5
    // Description : Utilitaire qui gère l'initialization de
6
    //
           l'écran ainsi que d'autres fonctions
    //
7
    // Auteur : Paulo Gomes
// Version : V1.0
8
9
10
11
12
    /*----*/
13
    // Section: Included Files
    /*----*/
14
15
    #include "DriverLcd.h"
16
    #include "Mc32Delays.h"
17
    #include "DefineLCD.h"
18
    #include "glcdfont.c"
19
    #include "app.h"
20
21
    #include <stdlib.h>
    #include <stdio.h>
23
    #include <string.h>
24
25
    #define D C low PLIB PORTS PinWrite(PORTS ID 0, PORT CHANNEL C, PORTS BIT POS 4, false)
    #define D C High PLIB PORTS PinWrite (PORTS ID 0, PORT CHANNEL C, PORTS BIT POS 4, true)
26
27
28
    #define TFT CS Low
    PLIB PORTS PinWrite (PORTS ID 0, PORT CHANNEL C, PORTS BIT POS 3, false)
29
    #define TFT CS High
    PLIB PORTS PinWrite (PORTS ID 0, PORT CHANNEL C, PORTS BIT POS 3, true)
30
31
    #define BCKL ON PLIB PORTS PinWrite(PORTS ID 0, PORT CHANNEL B, PORTS BIT POS 5, true)
    #define BCKL OFF PLIB PORTS PinWrite(PORTS ID 0, PORT CHANNEL B, PORTS BIT POS 5, false)
32
33
34
    #define RST ON PLIB PORTS PinWrite(PORTS ID 0, PORT CHANNEL B, PORTS BIT POS 6, false)
    #define RST_OFF PLIB_PORTS_PinWrite(PORTS_ID_0,PORT_CHANNEL_B,PORTS_BIT_POS_6,true)
35
36
37
    // Section: User Functions
38
39
    // Fonction qui permet d'envoyer des commandes
40
41
    void writecommand(char c)
42
    {
        // Datacommand low
43
44
        D C low;
        // Chip Select low
45
46
        TFT CS Low;
47
48
        // Envoi des données avec le SPI
49
        PLIB SPI BufferWrite(SPI_ID_1, c);
50
51
        do {
52
53
        } while (PLIB_SPI_IsBusy(SPI_ID_1));
54
5.5
        // Chip select high
56
        TFT_CS_High;
57
    }
58
59
    // Fonction qui permet d'envoyer des datas
60
    void writedata(char c)
61
62
        // Datacommand high
63
       D_C_High;
64
65
        // Chip Select low
66
        TFT CS Low;
67
68
        // Envoi des données avec le SPI
69
        PLIB SPI BufferWrite(SPI ID 1, c);
70
        do {
71
```

```
72
          } while (PLIB SPI IsBusy(SPI ID 1));
 73
 74
          // Chip select high
 75
          TFT CS High;
 76
      }
 77
 78
 79
 80
      // Initialisation de l'écran
 81
      void tft begin(void) {
 82
 83
          //RST ON;
 84
          RST OFF;
 85
          BCKL OFF;
 86
          BCKL ON;
 87
 88
          delay ms(500); // delay 500 ms
          BCKL OFF;
 89
          //RST OFF;
 90
 91
          TFT CS High; // Chip Select low
 92
          D C High; // Datacommand high
 93
          writecommand(0xEF);
 94
 95
          writedata (0x03);
 96
          writedata(0x80);
 97
          writedata (0x02);
 98
 99
          writecommand (0xCF);
100
          writedata (0x00);
101
          writedata(0XC1);
102
          writedata(0X30);
103
104
          writecommand(0xED);
105
          writedata (0x64);
106
          writedata (0x03);
107
          writedata(0X12);
108
          writedata(0X81);
109
110
          writecommand (0xE8);
111
          writedata (0x85);
112
          writedata (0x00);
113
          writedata (0x78);
114
115
          writecommand (0xCB);
116
          writedata (0x39);
117
          writedata (0x2C);
118
          writedata (0x00);
119
          writedata (0x34);
120
          writedata (0x02);
121
122
          writecommand (0xF7);
123
          writedata (0x20);
124
125
          writecommand (0xEA);
126
          writedata (0x00);
127
          writedata (0 \times 00);
128
129
          writecommand(ILI9341 PWCTR1);
                                            //Power control
130
          writedata (0x23);
                             //VRH[5:0]
131
132
          writecommand(ILI9341 PWCTR2);
                                            //Power control
133
          writedata(0x10); //SAP[2:0];BT[3:0]
134
135
          writecommand(ILI9341 VMCTR1);
                                            //VCM control
136
          writedata(0x3e);
137
          writedata (0x28);
138
139
          writecommand(ILI9341 VMCTR2);
                                            //VCM control2
140
          writedata (0x86);
141
142
          writecommand(ILI9341 MADCTL);
                                            // Memory Access Control
143
          writedata (0x48);
144
```

```
writecommand(ILI9341 PIXFMT);
145
146
          writedata (0x55);
147
148
          writecommand(ILI9341 FRMCTR1);
149
          writedata (0x00);
150
          writedata(0x18);
151
          writecommand(ILI9341 DFUNCTR); // Display Function Control
152
          writedata (0x08);
153
154
          writedata(0x82);
155
          writedata (0x27);
156
157
          writecommand(0xF2);  // 3Gamma Function Disable
158
          writedata (0x00);
159
          writecommand(ILI9341 GAMMASET); //Gamma curve selected
160
161
          writedata (0x01);
162
163
          writecommand(ILI9341 GMCTRP1); //Set Gamma
164
          writedata(0x0F);
165
          writedata(0x31);
166
          writedata(0x2B);
167
          writedata (0 \times 0 C);
168
          writedata (0 \times 0 E);
169
          writedata (0x08);
170
          writedata(0x4E);
171
          writedata(0xF1);
172
          writedata (0x37);
173
          writedata (0 \times 07);
174
          writedata(0x10);
175
          writedata(0x03);
176
          writedata (0 \times 0 E);
177
          writedata (0x09);
178
          writedata(0 \times 00);
179
180
          writecommand(ILI9341 GMCTRN1); //Set Gamma
181
          writedata(0x00);
182
          writedata(0x0E);
183
          writedata (0x14);
184
          writedata (0x03);
185
          writedata (0x11);
186
          writedata (0x07);
187
          writedata (0x31);
188
          writedata (0xC1);
189
          writedata (0x48);
190
          writedata (0x08);
191
          writedata(0x0F);
192
          writedata(0 \times 0 C);
193
          writedata(0x31);
194
          writedata(0x36);
195
          writedata(0x0F);
196
          writecommand(ILI9341 SLPOUT); //Exit Sleep
197
          delay ms(120);
198
          writecommand(ILI9341_DISPON); //Display on
199
200
           width = ILI9341 TFTWIDTH;
201
202
          height = ILI9341 TFTHEIGHT;
203
204
          // Backlight ON
205
206
          BCKL ON;
207
      }
208
209
210
211
      // Remplissage de l'écran avec une couleur
212
      void tft fillScreen(unsigned short color) {
213
      /* Rempli l'écran en entier avec une certaine couleur
214
       * Parameters: color: 16-bit color value
215
       * Returs: Nothing
216
217
          tft_fillRect(0, 0, _width , _height, color);
```

```
218
219
220
      // Dessine et rempli un rectangle avec une couleur
221
      void tft fillRect(short x, short y, short w, short h,
222
                         unsigned short color) {
223
      /* Desine un écran rempli avec une certaine couleur.
224
       * Commence top-left (en haut à gauche) et on peut définir
225
       * la taille et la hauteur
226
       * Parameters:
227
              x: x-coordinate of top-left vertex; top left of screen is x=0
228
                       and x increases to the right
229
                   y-coordinate of top-left vertex; top left of screen is y=0
230
                       and y increases to the bottom
231
                   width of rectangle
               w:
                  height of rectangle
              h:
233
              color:
                      16-bit color value
234
       * Returns:
                       Nothing
235
236
237
          // rudimentary clipping (drawChar w/big text requires this)
          if((x >= _width) || (y >= _height)) return;
if((x + w - 1) >= _width) w = _width - x;
if((y + h - 1) >= _height) h = _height - y;
238
239
240
241
242
          // Défini l'endroit ou écrire
243
          tft_setAddrWindow(x, y, x+w-1, y+h-1);
244
245
          // Masquage pour envoyer en 2x8bits
246
          uint8 t hi = color >> 8;
247
          uint8 t lo = color;
248
249
          //D C High;// Datacommand high
250
          D C low;
251
          TFT CS Low; // Chip Select low
252
253
          for(y=h; y>0; y--) {
254
            for (x=w; x>0; x--) {
255
               // MSB
256
               PLIB SPI BufferWrite (SPI ID 1, hi);
257
               do {
258
259
               } while (PLIB SPI IsBusy(SPI ID 1));
260
261
               // LSB
262
                   PLIB SPI BufferWrite (SPI ID 1, lo);
263
264
265
               } while (PLIB SPI IsBusy(SPI ID 1));
266
267
               }
268
          }
269
          TFT CS High; // Chip Select high
270
271
272
       // Fonction qui choisi l'endroit ou on veut écrire ou dessiner
273
      void tft setAddrWindow(unsigned short x0, unsigned short y0, unsigned short x1,
274
       unsigned short y1) {
275
276
          // x
277
          writecommand(ILI9341 CASET); // Column addr set
278
          writedata(x0 \gg 8);
279
          writedata(x0 & 0xFF);
                                      // XSTART
280
          writedata(x1 >> 8);
281
                                      // XEND
          writedata(x1 & 0xFF);
282
283
          // Y
284
          writecommand(ILI9341 PASET); // Row addr set
285
          writedata(y0>>8);
                               // YSTART
286
          writedata(y0);
287
          writedata(y1>>8);
288
                               // YEND
          writedata(y1);
289
290
          // Écrit dans la mémoire
```

```
291
          writecommand(ILI9341 RAMWR); // write to RAM
292
      }
293
294
      // Fonction qui permet de dessiner des Pixels
295
      void tft drawPixel(short x, short y, unsigned short color) {
296
      /* Dessine un pixel dans la localization voulu (x,y) avec une certaine couleur
297
       * Parameters:
298
              x: x-coordinate of pixel to draw; top left of screen is x=0
                       and x increases to the right
299
300
                  y-coordinate of pixel to draw; top left of screen is y=0
301
                       and y increases to the bottom
302
              color:
                       16-bit color value
303
       * Returns:
                       Nothing
304
305
306
          if((x < 0) \mid | (x >= width) \mid | (y < 0) \mid | (y >= height)) return;
307
308
          // Défini l'endroit ou écrire
309
          tft setAddrWindow(x,y,x+1,y+1);
310
311
          // Masquage pour envoyer en 2x8bits
312
          uint8 t hi = color >> 8;
313
          uint8 t lo = color;
314
315
          D C High; // Datacommand high
          TFT CS Low; // Chip Select low
316
317
318
          // MSB
319
          PLIB SPI BufferWrite (SPI ID 1, hi);
320
          do {
321
322
          } while (PLIB SPI IsBusy(SPI ID 1));
323
          // LSB
324
325
              PLIB SPI BufferWrite (SPI ID 1, lo);
326
          do {
327
328
          } while (PLIB SPI IsBusy(SPI ID 1));
329
330
          TFT CS High; // Chip Select high
331
      }
332
333
334
335
      /* DrawLine(Xa, Ya, Xb, Yb, color);
336
       * dessine une ligne entre 2 points A et B
337
       * http://www.brackeen.com/vga/shapes.html
338
339
      void DrawLine (short Xa, short Ya, short Xb, short Yb, unsigned short color)
340
341
          int dx,dy,sdx,sdy,px,py,dxabs,dyabs,i;
342
          float slope;
343
344
                          /* the horizontal distance of the line */
          dx=Xb-Xa;
345
                          /* the vertical distance of the line */
          dy=Yb-Ya;
346
          dxabs=abs(dx);
347
          dyabs=abs(dy);
348
          if(dx==0 ) {tft drawFastVLine(Xa,Ya,dy,color);return;}
349
          if(dy==0 ) {tft drawFastHLine(Xa,Ya,dx,color);return;}
350
          sdx = (dx > 0)? 1:-1;
351
          sdy=(dy>0)? 1:-1;
352
          if (dxabs>=dyabs) /* the line is more horizontal than vertical */
353
354
              slope=(float)dy / (float)dx;
355
              for (i=0;i!=dx;i+=sdx)
356
               {
357
                px=i+Xa;
358
                py=slope*i+Ya;
359
                 tft_drawPixel(px,py,color);
360
361
          }
362
          else /* the line is more vertical than horizontal */
363
```

```
364
              slope=(float)dx / (float)dy;
365
              for (i=0; i!=dy; i+=sdy)
366
367
                px=slope*i+Xa;
368
                py=i+Ya;
369
                tft drawPixel(px,py,color);
370
371
          }
372
373
374
      }
375
376
      // Fonction qui dessine des lignes horizantales
377
      void tft drawFastHLine(short x, short y, short w, unsigned short color)
378
      {
379
          // Rudimentary clipping
380
          if((x \ge width) || (y \ge height)) return;
381
          if((x+w-1)) >= width) w = width-x;
382
383
          // Défini l'endroit ou écrire
384
          tft setAddrWindow(x,y,x+w-1,y);
385
386
          D C High; // Datacommand high
387
          TFT CS Low; // Chip Select low
388
389
          // Masquage pour envoyer en 2x8bits
390
          uint8 t hi = color >> 8;
391
          uint8 t lo = color;
392
393
          while (w--) {
394
              writedata(hi); // MSB
395
              writedata(lo); // LSB
396
397
398
          TFT CS High; // Chip Select high
399
      }
400
401
      // Fonction qui dessine des lignes verticales
402
      void tft_drawFastVLine(short x, short y, short h, unsigned short color) {
403
404
          // Rudimentary clipping
405
          if((x >= _width) || (y >= _height)) return;
          if((y+h-1) >= height)
406
           h = height-y;
407
408
409
          // Défini l'endroit ou écrire
410
          tft setAddrWindow(x, y, x, y+h-1);
411
          D C High; // Datacommand high
412
413
          TFT CS Low;// Chip Select low
414
415
          // Masquage pour envoyer en 2x8bits
416
          uint8 t hi = color >> 8;
417
          uint8_t lo = color;
418
419
          while (h--) {
420
              writedata(hi); // MSB
421
              writedata(lo); // LSB
422
          }
423
424
          TFT CS High; // Chip Select high
425
      }
426
427
      // Fonction qui permet de dessiner un caractère
428
      void drawChar(short x, short y, unsigned char c,
429
              unsigned short fgcolor, unsigned short bgcolor, unsigned short size)
430
      {
431
                                        || // Clip right
          if((x \ge width))
432
             (y >= _height)
                                       // Clip bottom
433
             ((x + \frac{1}{6} * size - 1) < 0) || // Clip left TODO: is this correct?
434
             ((y + 8 * size - 1) < 0)) // Clip top
                                                         TODO: is this correct?
435
              return;
436
```

```
437
           if (fgcolor == bgcolor) {
438
               // This transparent approach is only about 20% faster
439
               if (size == 1) {
440
                   uint8 t mask = 0 \times 01;
441
                   int16 t xoff, yoff;
442
                   for (yoff=0; yoff < 8; yoff++) {</pre>
443
                       uint8 t line = 0;
                        for (xoff=0; xoff < 5; xoff++) {</pre>
444
445
                            if (font[c * 5 + xoff] & mask) line |= 1;
446
                            line <<= 1;
447
                        1
448
                       line >>= 1;
449
                       xoff = 0;
450
                       while (line) {
451
                            if (line == 0x1F) {
                                tft drawFastHLine(x + xoff, y + yoff, 5, fgcolor);
452
453
                            } else if (line == 0x1E) {
454
455
                                tft drawFastHLine(x + xoff, y + yoff, \frac{4}{2}, fgcolor);
456
                                break;
457
                            } else if ((line & 0x1C) == 0x1C) {
458
                                tft drawFastHLine(x + xoff, y + yoff, 3, fgcolor);
459
                                line <<= 4;
                                xoff += 4;
460
461
                            } else if ((line & 0x18) == 0x18) {
462
                                tft drawFastHLine(x + xoff, y + yoff, 2, fgcolor);
463
                                line <<= 3;
464
                                xoff += 3;
465
                            } else if ((line & 0x10) == 0x10) {
466
                                tft drawPixel(x + xoff, y + yoff, fgcolor);
467
                                line <<= 2;
468
                                xoff += 2;
469
                            } else {
470
                                line <<= 1;
                                xoff += 1;
471
472
                            }
473
                        }
474
                       mask = mask << 1;
475
                   }
               } else {
476
477
                   uint8 t mask = 0 \times 01;
478
                   int16 t xoff, yoff;
479
                   for (yoff=0; yoff < 8; yoff++) {</pre>
480
                       uint8 t line = 0;
481
                        for (xoff=0; xoff < 5; xoff++) {</pre>
482
                            if (font[c * 5 + xoff] & mask) line |= 1;
483
                            line <<= 1;
484
485
                       line >>= 1;
486
                       xoff = 0;
487
                       while (line) {
                            if (line == 0x1F) {
488
489
                                tft fillRect(x + xoff * size, y + yoff * size,
490
                                     5 * size, size, fgcolor);
491
                                break;
492
                            } else if (line == 0x1E) {
                                tft fillRect(x + xoff * size, y + yoff * size,
493
494
                                    4 * size, size, fgcolor);
495
                                break;
496
                            } else if ((line & 0x1C) == 0x1C) {
497
                                tft fillRect(x + xoff * size, y + yoff * size,
498
                                    3 * size, size, fgcolor);
499
                                line <<= 4;
500
                                xoff += 4;
501
                            } else if ((line & 0x18) == 0x18) {
502
                                tft_fillRect(x + xoff * size, y + yoff * size,
503
                                    2 * size, size, fgcolor);
504
                                line <<= 3;
505
                                xoff += 3;
506
                            } else if ((line & 0x10) == 0x10) {
                                tft fillRect(x + xoff * size, y + yoff * size,
507
                                    size, size, fgcolor);
508
509
                                line <<= 2;
```

```
510
                                xoff += 2;
511
                            } else {
512
                                line <<= 1;
513
                                xoff += 1;
514
                            }
515
                        }
516
                       mask = mask << 1;
517
                   }
518
               }
           } else {
519
               // This solid background approach is about 5 time faster
520
521
               tft setAddrWindow(x, y, x + 6 * size - 1, y + 8 * size - 1);
               uint8_t xr, yr;
               uint8 t mask = 0x01;
524
               uint16 t color;
               uint8 t hi;
525
               uint8 t lo ;
526
527
               uint8 t bhi = bgcolor >> 8;
528
               uint8 t blo = bgcolor;
529
               for (y=0; y < 8; y++) {
530
                   for (yr=0; yr < size; yr++) {</pre>
531
                        for (x=0; x < 5; x++) {
532
                            if (font[c * 5 + x] & mask) {
533
                                color = fgcolor;
534
                            } else {
535
                                color = bgcolor;
536
                            }
537
                            hi = color >> 8;
538
                            lo = color;
539
                            for (xr=0; xr < size; xr++) {</pre>
540
541
                                writedata(hi);
542
                                writedata(lo);
543
                            }
544
545
                        for (xr=0; xr < size; xr++) {</pre>
546
                            writedata(bhi);
547
                            writedata(blo);
548
                        }
549
                   }
550
                   mask = mask << 1;
551
               }
552
          }
553
      }
554
      // Fonction qui permet de faire une rotation de 90\,^{\circ} de l'écran
555
556
      void setRotation(short m) {
557
558
          writecommand(ILI9341 MADCTL);
559
          short rotation = m % 4; // can't be higher than 3
560
           switch (rotation) {
                   case 0: // Écran initial
561
                   writedata(MADCTL MX | MADCTL BGR);
562
563
                    width = ILI9341 TFTWIDTH;
                    height = ILI9341_TFTHEIGHT;
564
565
               break;
566
567
                   case 1: // Rotation de l'écran 90°
568
                   writedata (MADCTL MV | MADCTL BGR);
569
                    width = ILI9341 TFTHEIGHT;
570
                    height = ILI9341 TFTWIDTH;
571
               break;
572
                   case 2: // Rotation de l'écran 180°
573
574
                   writedata(MADCTL MY | MADCTL BGR);
                   _width = ILI934\overline{1}_TFTWIDTH;
575
576
                    height = ILI9341 TFTHEIGHT;
577
               break;
578
579
                   case 3: // Rotation de l'écran 270°
580
                   writedata (MADCTL MX | MADCTL MY | MADCTL MV | MADCTL BGR);
                   _width = ILI934\overline{1}_TFTHEIGHT;
581
582
                   _height = ILI9341_TFTWIDTH;
```

```
583
             break;
584
          }
585
      }
586
587
      // Defini l'emplacement du curseur
588
      void tft setCursor(short x, short y) {
589
      /* Set cursor for text to be printed
       * Parameters:
590
591
              x = x-coordinate of top-left of text starting
592
              y = y-coordinate of top-left of text starting
593
       * Returns: Nothing
594
595
596
          cursor x = x; // en haut à gauche
          cursor y = y; // en haut à gauche
597
598
599
600
      // Modifie la couleur de la police
601
      void tft setTextColor(unsigned short c) {
602
          // For 'transparent' background, we'll set the bg
603
          // to the same as fg instead of using a flag
604
          textcolor = textbgcolor = c;
605
606
      // Modifie la couleur ET le background de la police
607
      void tft setTextColor F(unsigned short fore, unsigned short Back)
608
      {
609
610
          textcolor = fore;
611
          textbgcolor = Back;
612
613
      // Fonction qui permet d'écrire une chaine de caractères
614
615
      void tft writeString(char* str){
616
      /* Call tft_setCursor(), tft_setTextColor(), tft setTextSize()
617
       * as necessary before printing
618
619
          while (*str) {
620
              tft write(*str++);
621
622
      }
623
624
      // Fonction qui permet d'écrire
625
     void tft_write(unsigned char c){
626
          if (c == '\n') {
627
              cursor_y += textsize*8;
628
              cursor x = 0;
              } else if (c == '\r') {
629
              // skip em
630
              } else if (c == '\t'){
631
632
              int new_x = cursor_x + 4;
              if (new x < width){</pre>
633
634
              cursor x = new x;
635
636
              } else {
637
              drawChar(cursor_x, cursor_y, c, textcolor, textbgcolor, textsize);
638
              cursor x += textsize*6;
639
              if (wrap && (cursor x > ( width - textsize*6))) {
640
              cursor_y += textsize*8;
641
              cursor_x = 0;
642
643
          }
644
645
646
      // Modifie la taille du text à afficher
647
      void tft setTextSize(unsigned char s) {
648
      /*Set size of text to be displayed
649
       * Parameters:
650
              s = text size (1 being smallest)
651
       * Returns: nothing
652
653
        textsize = (s > 0) ? s : 1;
654
655
```