postmodern / ili9341.c

Created 7 years ago

ILI9341

ili9341.c

```
#include "ili9341.h"

volatile uint16_t LCD_W=ILI9341_TFTWIDTH;
volatile uint16_t LCD_H=ILI9341_TFTHEIGHT;

void ili9341_hard_init(void)//init hardware
{
        rstddr=0xFF;//output for reset
        rstport |=(1<<rst);//pull high for normal operation
        controlddr|=(1<<dc);//D/C as output
}

void ili9341_spi_init(void)//set spi speed and settings
{
        DDRB |=(1<<1)|(1<<2)|(1<<3)|(1<<5);//CS,SS,MOSI,SCK as output(although SS will be unuse
        SPCR=(1<<SPE)|(1<<MSTR);//mode 0,fosc/4
        SPSR |=(1<<SPI2X);//doubling spi speed.i.e final spi speed-fosc/2
        PORTB |=(1<<1);//cs off during startup
}

void ili9341_spi_send(unsigned char spi_data)//send spi data to display
{
        SPDR=spi_data;//move data into spdr
        while(!(SPSR & (1<<SPIF)));//wait till the transmission is finished
}

void ili9341_writecommand8(uint8_t com)//command write
{
        controlport &=~((1<<dc)|(1<<cs));//dc and cs both low to send command
        _delay_us(5);//little delay
```

```
31          ili9341_spi_send(com);
32          controlport |=(1<<cs);//pull high cs
33  }
34
35  void ili9341_writedata8(uint8_t data)//data write
36  {
37          controlport |=(1<<dc);//set dc high for data
38          _delay_us(1);//delay
39          controlport &=~(1<<cs);//set cs low for operation
40          ili9341_spi_send(data);
41          controlport |=(1<<cs);
42  }
43
44  void ili9341_setaddress(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2)//set coordinate for pr
45  {
46          ili9341_writecommand8(0x2A);
47          ili9341_writedata8(x1>>8);
48          ili9341_writedata8(x1);
49          ili9341_writedata8(x2>>8);
50          ili9341_writedata8(x2);
51
52          ili9341_writecommand8(0x2B);
53          ili9341_writedata8(y1>>8);
54          ili9341_writedata8(y1);
55          ili9341_writedata8(y2);
56          ili9341_writedata8(y2);
57
58          ili9341_writecommand8(0x2C);//meory write
59  }
60
61  void ili9341_hard_reset(void)//hard reset display
62  {
63          rstport |=(1<<rst);//pull high if low previously
64          _delay_ms(200);
65          rstport &=~(1<<rst);//low for reset
66          _delay_ms(200);
67          rstport |=(1<<rst);//again pull high for normal operation
68          _delay_ms(200);
69  }
70
71  void ili9341_init(void)//set up display using predefined command sequence
72  {
73          ili9341_hard_init();
74          ili9341_spi_init();
75          ili9341_hard_reset();
76          ili9341_writecommand8(0x01);//soft reset
77          _delay_ms(1000);
78
```

```
        //power control A
        ili9341_writecommand8(0xCB);
        ili9341_writedata8(0x39);
        ili9341_writedata8(0x2C);
        ili9341_writedata8(0x00);
        ili9341_writedata8(0x34);
        ili9341_writedata8(0x02);

        //power control B
        ili9341_writecommand8(0xCF);
        ili9341_writedata8(0x00);
        ili9341_writedata8(0xC1);
        ili9341_writedata8(0x30);

        //driver timing control A
        ili9341_writecommand8(0xE8);
        ili9341_writedata8(0x85);
        ili9341_writedata8(0x00);
        ili9341_writedata8(0x78);

        //driver timing control B
        ili9341_writecommand8(0xEA);
        ili9341_writedata8(0x00);
        ili9341_writedata8(0x00);

        //power on sequence control
        ili9341_writecommand8(0xED);
        ili9341_writedata8(0x64);
        ili9341_writedata8(0x03);
        ili9341_writedata8(0x12);
        ili9341_writedata8(0x81);

        //pump ratio control
        ili9341_writecommand8(0xF7);
        ili9341_writedata8(0x20);

        //power control,VRH[5:0]
        ili9341_writecommand8(0xC0);
        ili9341_writedata8(0x23);

        //Power control,SAP[2:0];BT[3:0]
        ili9341_writecommand8(0xC1);
        ili9341_writedata8(0x10);

        //vcm control
        ili9341_writecommand8(0xC5);
        ili9341_writedata8(0x3E);
        ili9341_writedata8(0x28);
```

```
127
128         //vcm control 2
129         ili9341_writecommand8(0xC7);
130         ili9341_writedata8(0x86);
131
132         //memory access control
133         ili9341_writecommand8(0x36);
134         ili9341_writedata8(0x48);
135
136         //pixel format
137         ili9341_writecommand8(0x3A);
138         ili9341_writedata8(0x55);
139
140         //frameration control,normal mode full colours
141         ili9341_writecommand8(0xB1);
142         ili9341_writedata8(0x00);
143         ili9341_writedata8(0x18);
144
145         //display function control
146         ili9341_writecommand8(0xB6);
147         ili9341_writedata8(0x08);
148         ili9341_writedata8(0x82);
149         ili9341_writedata8(0x27);
150
151         //3gamma function disable
152         ili9341_writecommand8(0xF2);
153         ili9341_writedata8(0x00);
154
155         //gamma curve selected
156         ili9341_writecommand8(0x26);
157         ili9341_writedata8(0x01);
158
159         //set positive gamma correction
160         ili9341_writecommand8(0xE0);
161         ili9341_writedata8(0x0F);
162         ili9341_writedata8(0x31);
163         ili9341_writedata8(0x2B);
164         ili9341_writedata8(0x0C);
165         ili9341_writedata8(0x0E);
166         ili9341_writedata8(0x08);
167         ili9341_writedata8(0x4E);
168         ili9341_writedata8(0xF1);
169         ili9341_writedata8(0x37);
170         ili9341_writedata8(0x07);
171         ili9341_writedata8(0x10);
172         ili9341_writedata8(0x03);
173         ili9341_writedata8(0x0E);
174         ili9341_writedata8(0x09);
```

```
175         ili9341_writedata8(0x00);
176 
177         //set negative gamma correction
178         ili9341_writecommand8(0xE1);
179         ili9341_writedata8(0x00);
180         ili9341_writedata8(0x0E);
181         ili9341_writedata8(0x14);
182         ili9341_writedata8(0x03);
183         ili9341_writedata8(0x11);
184         ili9341_writedata8(0x07);
185         ili9341_writedata8(0x31);
186         ili9341_writedata8(0xC1);
187         ili9341_writedata8(0x48);
188         ili9341_writedata8(0x08);
189         ili9341_writedata8(0x0F);
190         ili9341_writedata8(0x0C);
191         ili9341_writedata8(0x31);
192         ili9341_writedata8(0x36);
193         ili9341_writedata8(0x0F);
194 
195         //exit sleep
196         ili9341_writecommand8(0x11);
197         _delay_ms(120);
198         //display on
199         ili9341_writecommand8(0x29);
200 }
201 
202 //set colour for drawing
203 void ili9341_pushcolour(uint16_t colour)
204 {
205         ili9341_writedata8(colour>>8);
206         ili9341_writedata8(colour);
207 }
208 
209 //clear lcd and fill with colour
210 void ili9341_clear(uint16_t colour)
211 {
212         uint16_t i,j;
213         ili9341_setaddress(0,0,LCD_W-1,LCD_H-1);
214 
215         for(i=0;i<LCD_W;i++)
216         {
217                 for(j=0;j<LCD_H;j++)
218                 {
219                         ili9341_pushcolour(colour);
220                 }
221         }
222 }
```

```
223
224 //draw pixel
225 void ili9341_drawpixel(uint16_t x3,uint16_t y3,uint16_t colour1) //pixels will always be counte
226 {
227         if ((x3 < 0) ||(x3 >= LCD_W) || (y3 < 0) || (y3 >= LCD_H))
228         {
229                 return;
230         }
231
232         ili9341_setaddress(x3,y3,x3+1,y3+1);
233         ili9341_pushcolour(colour1);
234 }
235
236 //draw vertical line
237 void ili9341_drawvline(uint16_t x,uint16_t y,uint16_t h,uint16_t colour)//basically we will see
238 {
239         if ((x >= LCD_W) || (y >= LCD_H))
240         {
241                 return;
242         }
243
244         if((y+h-1) >= LCD_H)
245         {
246                 h=LCD_H-y;
247         }
248
249         ili9341_setaddress(x,y,x,y+h-1);
250
251         while (h--)
252         {
253                 ili9341_pushcolour(colour);
254         }
255 }
256
257
258 //draw horizental line
259
260 void ili9341_drawhline(uint16_t x,uint16_t y,uint16_t w,uint16_t colour)
261 {
262         if((x >=LCD_W) || (y >=LCD_H))
263         {
264                 return;
265         }
266
267         if((x+w-1) >= LCD_W)
268         {
269                 w=LCD_W-x;
270         }
```

```
271
272         ili9341_setaddress(x,y,x+w-1,y);
273
274         while(w--)
275         {
276                 ili9341_pushcolour(colour);
277         }
278 }
279
280
281 //draw colour filled rectangle
282 void ili9341_fillrect(uint16_t x,uint16_t y,uint16_t w,uint16_t h,uint16_t colour)
283 {
284         if ((x >= LCD_W) || (y >= LCD_H))
285         {
286                 return;
287         }
288
289         if ((x+w-1) >= LCD_W)
290         {
291                 w = LCD_W-x;
292         }
293
294         if ((y+h-1) >= LCD_H)
295         {
296                 h = LCD_H-y;
297         }
298
299         ili9341_setaddress(x, y, x+w-1, y+h-1);
300
301         for(y=h; y>0; y--)
302         {
303                 for(x=w; x>0; x--)
304                 {
305                         ili9341_pushcolour(colour);
306                 }
307         }
308 }
309 //rotate screen at desired orientation
310 void ili9341_setRotation(uint8_t m)
311 {
312         uint8_t rotation;
313
314         ili9341_writecommand8(0x36);
315         rotation=m%4;
316
317         switch (rotation)
318         {
```

```
                case 0:
                        ili9341_writedata8(0x40|0x08);
                        LCD_W = 240;
                        LCD_H = 320;
                        break;
                case 1:
                        ili9341_writedata8(0x20|0x08);
                        LCD_W  = 320;
                        LCD_H = 240;
                        break;
                case 2:
                        ili9341_writedata8(0x80|0x08);
                        LCD_W  = 240;
                        LCD_H = 320;
                        break;
                case 3:
                        ili9341_writedata8(0x40|0x80|0x20|0x08);
                        LCD_W  = 320;
                        LCD_H = 240;
                        break;
        }
}
```

ili9341.h

```
#ifndef _ILI9341_H_
#define _ILI9341_H_

#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <limits.h>
#include <inttypes.h>

#define controlport PORTB
#define controlddr DDRB
#define controlpin PINB
#define rstport PORTD
#define rstddr DDRD
#define rstpin PIND
#define dc 0
#define cs 1
#define rst 7
#define ILI9341_TFTHEIGHT 240
#define ILI9341_TFTWIDTH 320

#define BLACK       0x0000
#define NAVY        0x000F
```

```
#define DARKGREEN   0x03E0
#define DARKCYAN    0x03EF
#define MAROON      0x7800
#define PURPLE      0x780F
#define OLIVE       0x7BE0
#define LIGHTGREY   0xC618
#define DARKGREY    0x7BEF
#define BLUE        0x001F
#define GREEN       0x07E0
#define CYAN        0x07FF
#define RED         0xF800
#define MAGENTA     0xF81F
#define YELLOW      0xFFE0
#define WHITE       0xFFFF
#define ORANGE      0xFD20
#define GREENYELLOW 0xAFE5
#define PINK        0xF81F

void ili9341_hard_init(void);
void ili9341_spi_init(void);
void ili9341_spi_send(unsigned char spi_data);
void ili9341_writecommand8(uint8_t com);
void ili9341_writedata8(uint8_t data);
void ili9341_setaddress(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2);
void ili9341_hard_reset(void);
void ili9341_init(void);
void ili9341_pushcolour(uint16_t colour);
void ili9341_clear(uint16_t colour);
void ili9341_drawpixel(uint16_t x3,uint16_t y3,uint16_t colour1);
void ili9341_drawvline(uint16_t x,uint16_t y,uint16_t h,uint16_t colour);
void ili9341_drawhline(uint16_t x,uint16_t y,uint16_t w,uint16_t colour);
void ili9341_fillrect(uint16_t x,uint16_t y,uint16_t w,uint16_t h,uint16_t colour);
void ili9341_setRotation(uint8_t x);

#endif
```

ili9341gfx.c

```
#include "ili9341.h"
#include "ili9341gfx.h"

volatile uint16_t cursor_x;
volatile uint16_t cursor_y;
volatile uint16_t textcolour;
volatile uint16_t textbgcolour;
volatile uint8_t textsize;
uint16_t vsetx,vsety,vactualx,vactualy,isetx,isety,iactualx,iactualy;

```

```
11  static FILE mydata = FDEV_SETUP_STREAM(ili9341_putchar_printf, NULL, _FDEV_SETUP_WRITE);//mydat
12
13  int16_t ili9341_putchar_printf(char var, FILE *stream)//this function will be called whenever p
14  {
15          ili9341_write(var);
16          return 0;
17  }
18
19  void backuplocationvset(void)//backing up vset data start location to print next vset data in e
20  {
21          vsetx=cursor_x;
22          vsety=cursor_y;
23  }
24
25  void backuplocationvactual(void)//backing up vactual data start location to print next vactual
26  {
27          vactualx=cursor_x;
28          vactualy=cursor_y;
29  }
30
31  void backuplocationiset(void)//backing up iset data start location to print next iset data in e
32  {
33          isetx=cursor_x;
34          isety=cursor_y;
35  }
36
37  void backuplocationiactual(void)//backing up iactual data start location to print next iactual
38  {
39          iactualx=cursor_x;
40          iactualy=cursor_y;
41  }
42
43  //array for font
44  static const unsigned char font[] PROGMEM = {
45          0x00, 0x00, 0x00, 0x00, 0x00,
46          0x3E, 0x5B, 0x4F, 0x5B, 0x3E,
47          0x3E, 0x6B, 0x4F, 0x6B, 0x3E,
48          0x1C, 0x3E, 0x7C, 0x3E, 0x1C,
49          0x18, 0x3C, 0x7E, 0x3C, 0x18,
50          0x1C, 0x57, 0x7D, 0x57, 0x1C,
51          0x1C, 0x5E, 0x7F, 0x5E, 0x1C,
52          0x00, 0x18, 0x3C, 0x18, 0x00,
53          0xFF, 0xE7, 0xC3, 0xE7, 0xFF,
54          0x00, 0x18, 0x24, 0x18, 0x00,
55          0xFF, 0xE7, 0xDB, 0xE7, 0xFF,
56          0x30, 0x48, 0x3A, 0x06, 0x0E,
57          0x26, 0x29, 0x79, 0x29, 0x26,
58          0x40, 0x7F, 0x05, 0x05, 0x07,
```

```
        0x40, 0x7F, 0x05, 0x25, 0x3F,
        0x5A, 0x3C, 0xE7, 0x3C, 0x5A,
        0x7F, 0x3E, 0x1C, 0x1C, 0x08,
        0x08, 0x1C, 0x1C, 0x3E, 0x7F,
        0x14, 0x22, 0x7F, 0x22, 0x14,
        0x5F, 0x5F, 0x00, 0x5F, 0x5F,
        0x06, 0x09, 0x7F, 0x01, 0x7F,
        0x00, 0x66, 0x89, 0x95, 0x6A,
        0x60, 0x60, 0x60, 0x60, 0x60,
        0x94, 0xA2, 0xFF, 0xA2, 0x94,
        0x08, 0x04, 0x7E, 0x04, 0x08,
        0x10, 0x20, 0x7E, 0x20, 0x10,
        0x08, 0x08, 0x2A, 0x1C, 0x08,
        0x08, 0x1C, 0x2A, 0x08, 0x08,
        0x1E, 0x10, 0x10, 0x10, 0x10,
        0x0C, 0x1E, 0x0C, 0x1E, 0x0C,
        0x30, 0x38, 0x3E, 0x38, 0x30,
        0x06, 0x0E, 0x3E, 0x0E, 0x06,
        0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x5F, 0x00, 0x00,
        0x00, 0x07, 0x00, 0x07, 0x00,
        0x14, 0x7F, 0x14, 0x7F, 0x14,
        0x24, 0x2A, 0x7F, 0x2A, 0x12,
        0x23, 0x13, 0x08, 0x64, 0x62,
        0x36, 0x49, 0x56, 0x20, 0x50,
        0x00, 0x08, 0x07, 0x03, 0x00,
        0x00, 0x1C, 0x22, 0x41, 0x00,
        0x00, 0x41, 0x22, 0x1C, 0x00,
        0x2A, 0x1C, 0x7F, 0x1C, 0x2A,
        0x08, 0x08, 0x3E, 0x08, 0x08,
        0x00, 0x80, 0x70, 0x30, 0x00,
        0x08, 0x08, 0x08, 0x08, 0x08,
        0x00, 0x00, 0x60, 0x60, 0x00,
        0x20, 0x10, 0x08, 0x04, 0x02,
        0x3E, 0x51, 0x49, 0x45, 0x3E,
        0x00, 0x42, 0x7F, 0x40, 0x00,
        0x72, 0x49, 0x49, 0x49, 0x46,
        0x21, 0x41, 0x49, 0x4D, 0x33,
        0x18, 0x14, 0x12, 0x7F, 0x10,
        0x27, 0x45, 0x45, 0x45, 0x39,
        0x3C, 0x4A, 0x49, 0x49, 0x31,
        0x41, 0x21, 0x11, 0x09, 0x07,
        0x36, 0x49, 0x49, 0x49, 0x36,
        0x46, 0x49, 0x49, 0x29, 0x1E,
        0x00, 0x00, 0x14, 0x00, 0x00,
        0x00, 0x40, 0x34, 0x00, 0x00,
        0x00, 0x08, 0x14, 0x22, 0x41,
        0x14, 0x14, 0x14, 0x14, 0x14,
```

```
        0x00, 0x41, 0x22, 0x14, 0x08,
        0x02, 0x01, 0x59, 0x09, 0x06,
        0x3E, 0x41, 0x5D, 0x59, 0x4E,
        0x7C, 0x12, 0x11, 0x12, 0x7C,
        0x7F, 0x49, 0x49, 0x49, 0x36,
        0x3E, 0x41, 0x41, 0x41, 0x22,
        0x7F, 0x41, 0x41, 0x41, 0x3E,
        0x7F, 0x49, 0x49, 0x49, 0x41,
        0x7F, 0x09, 0x09, 0x09, 0x01,
        0x3E, 0x41, 0x41, 0x51, 0x73,
        0x7F, 0x08, 0x08, 0x08, 0x7F,
        0x00, 0x41, 0x7F, 0x41, 0x00,
        0x20, 0x40, 0x41, 0x3F, 0x01,
        0x7F, 0x08, 0x14, 0x22, 0x41,
        0x7F, 0x40, 0x40, 0x40, 0x40,
        0x7F, 0x02, 0x1C, 0x02, 0x7F,
        0x7F, 0x04, 0x08, 0x10, 0x7F,
        0x3E, 0x41, 0x41, 0x41, 0x3E,
        0x7F, 0x09, 0x09, 0x09, 0x06,
        0x3E, 0x41, 0x51, 0x21, 0x5E,
        0x7F, 0x09, 0x19, 0x29, 0x46,
        0x26, 0x49, 0x49, 0x49, 0x32,
        0x03, 0x01, 0x7F, 0x01, 0x03,
        0x3F, 0x40, 0x40, 0x40, 0x3F,
        0x1F, 0x20, 0x40, 0x20, 0x1F,
        0x3F, 0x40, 0x38, 0x40, 0x3F,
        0x63, 0x14, 0x08, 0x14, 0x63,
        0x03, 0x04, 0x78, 0x04, 0x03,
        0x61, 0x59, 0x49, 0x4D, 0x43,
        0x00, 0x7F, 0x41, 0x41, 0x41,
        0x02, 0x04, 0x08, 0x10, 0x20,
        0x00, 0x41, 0x41, 0x41, 0x7F,
        0x04, 0x02, 0x01, 0x02, 0x04,
        0x40, 0x40, 0x40, 0x40, 0x40,
        0x00, 0x03, 0x07, 0x08, 0x00,
        0x20, 0x54, 0x54, 0x78, 0x40,
        0x7F, 0x28, 0x44, 0x44, 0x38,
        0x38, 0x44, 0x44, 0x44, 0x28,
        0x38, 0x44, 0x44, 0x28, 0x7F,
        0x38, 0x54, 0x54, 0x54, 0x18,
        0x00, 0x08, 0x7E, 0x09, 0x02,
        0x18, 0xA4, 0xA4, 0x9C, 0x78,
        0x7F, 0x08, 0x04, 0x04, 0x78,
        0x00, 0x44, 0x7D, 0x40, 0x00,
        0x20, 0x40, 0x40, 0x3D, 0x00,
        0x7F, 0x10, 0x28, 0x44, 0x00,
        0x00, 0x41, 0x7F, 0x40, 0x00,
        0x7C, 0x04, 0x78, 0x04, 0x78,
```

```
155         0x7C, 0x08, 0x04, 0x04, 0x78,
156         0x38, 0x44, 0x44, 0x44, 0x38,
157         0xFC, 0x18, 0x24, 0x24, 0x18,
158         0x18, 0x24, 0x24, 0x18, 0xFC,
159         0x7C, 0x08, 0x04, 0x04, 0x08,
160         0x48, 0x54, 0x54, 0x54, 0x24,
161         0x04, 0x04, 0x3F, 0x44, 0x24,
162         0x3C, 0x40, 0x40, 0x20, 0x7C,
163         0x1C, 0x20, 0x40, 0x20, 0x1C,
164         0x3C, 0x40, 0x30, 0x40, 0x3C,
165         0x44, 0x28, 0x10, 0x28, 0x44,
166         0x4C, 0x90, 0x90, 0x90, 0x7C,
167         0x44, 0x64, 0x54, 0x4C, 0x44,
168         0x00, 0x08, 0x36, 0x41, 0x00,
169         0x00, 0x00, 0x77, 0x00, 0x00,
170         0x00, 0x41, 0x36, 0x08, 0x00,
171         0x02, 0x01, 0x02, 0x04, 0x02,
172         0x3C, 0x26, 0x23, 0x26, 0x3C,
173         0x1E, 0xA1, 0xA1, 0x61, 0x12,
174         0x3A, 0x40, 0x40, 0x20, 0x7A,
175         0x38, 0x54, 0x54, 0x55, 0x59,
176         0x21, 0x55, 0x55, 0x79, 0x41,
177         0x22, 0x54, 0x54, 0x78, 0x42, // a-umlaut
178         0x21, 0x55, 0x54, 0x78, 0x40,
179         0x20, 0x54, 0x55, 0x79, 0x40,
180         0x0C, 0x1E, 0x52, 0x72, 0x12,
181         0x39, 0x55, 0x55, 0x55, 0x59,
182         0x39, 0x54, 0x54, 0x54, 0x59,
183         0x39, 0x55, 0x54, 0x54, 0x58,
184         0x00, 0x00, 0x45, 0x7C, 0x41,
185         0x00, 0x02, 0x45, 0x7D, 0x42,
186         0x00, 0x01, 0x45, 0x7C, 0x40,
187         0x7D, 0x12, 0x11, 0x12, 0x7D, // A-umlaut
188         0xF0, 0x28, 0x25, 0x28, 0xF0,
189         0x7C, 0x54, 0x55, 0x45, 0x00,
190         0x20, 0x54, 0x54, 0x7C, 0x54,
191         0x7C, 0x0A, 0x09, 0x7F, 0x49,
192         0x32, 0x49, 0x49, 0x49, 0x32,
193         0x3A, 0x44, 0x44, 0x44, 0x3A, // o-umlaut
194         0x32, 0x4A, 0x48, 0x48, 0x30,
195         0x3A, 0x41, 0x41, 0x21, 0x7A,
196         0x3A, 0x42, 0x40, 0x20, 0x78,
197         0x00, 0x9D, 0xA0, 0xA0, 0x7D,
198         0x3D, 0x42, 0x42, 0x42, 0x3D, // O-umlaut
199         0x3D, 0x40, 0x40, 0x40, 0x3D,
200         0x3C, 0x24, 0xFF, 0x24, 0x24,
201         0x48, 0x7E, 0x49, 0x43, 0x66,
202         0x2B, 0x2F, 0xFC, 0x2F, 0x2B,
```

```
        0xFF, 0x09, 0x29, 0xF6, 0x20,
        0xC0, 0x88, 0x7E, 0x09, 0x03,
        0x20, 0x54, 0x54, 0x79, 0x41,
        0x00, 0x00, 0x44, 0x7D, 0x41,
        0x30, 0x48, 0x48, 0x4A, 0x32,
        0x38, 0x40, 0x40, 0x22, 0x7A,
        0x00, 0x7A, 0x0A, 0x0A, 0x72,
        0x7D, 0x0D, 0x19, 0x31, 0x7D,
        0x26, 0x29, 0x29, 0x2F, 0x28,
        0x26, 0x29, 0x29, 0x29, 0x26,
        0x30, 0x48, 0x4D, 0x40, 0x20,
        0x38, 0x08, 0x08, 0x08, 0x08,
        0x08, 0x08, 0x08, 0x08, 0x38,
        0x2F, 0x10, 0xC8, 0xAC, 0xBA,
        0x2F, 0x10, 0x28, 0x34, 0xFA,
        0x00, 0x00, 0x7B, 0x00, 0x00,
        0x08, 0x14, 0x2A, 0x14, 0x22,
        0x22, 0x14, 0x2A, 0x14, 0x08,
        0xAA, 0x00, 0x55, 0x00, 0xAA,
        0xAA, 0x55, 0xAA, 0x55, 0xAA,
        0x00, 0x00, 0x00, 0xFF, 0x00,
        0x10, 0x10, 0x10, 0xFF, 0x00,
        0x14, 0x14, 0x14, 0xFF, 0x00,
        0x10, 0x10, 0xFF, 0x00, 0xFF,
        0x10, 0x10, 0xF0, 0x10, 0xF0,
        0x14, 0x14, 0x14, 0xFC, 0x00,
        0x14, 0x14, 0xF7, 0x00, 0xFF,
        0x00, 0x00, 0xFF, 0x00, 0xFF,
        0x14, 0x14, 0xF4, 0x04, 0xFC,
        0x14, 0x14, 0x17, 0x10, 0x1F,
        0x10, 0x10, 0x1F, 0x10, 0x1F,
        0x14, 0x14, 0x14, 0x1F, 0x00,
        0x10, 0x10, 0x10, 0xF0, 0x00,
        0x00, 0x00, 0x00, 0x1F, 0x10,
        0x10, 0x10, 0x10, 0x1F, 0x10,
        0x10, 0x10, 0x10, 0xF0, 0x10,
        0x00, 0x00, 0x00, 0xFF, 0x10,
        0x10, 0x10, 0x10, 0x10, 0x10,
        0x10, 0x10, 0x10, 0xFF, 0x10,
        0x00, 0x00, 0x00, 0xFF, 0x14,
        0x00, 0x00, 0xFF, 0x00, 0xFF,
        0x00, 0x00, 0x1F, 0x10, 0x17,
        0x00, 0x00, 0xFC, 0x04, 0xF4,
        0x14, 0x14, 0x17, 0x10, 0x17,
        0x14, 0x14, 0xF4, 0x04, 0xF4,
        0x00, 0x00, 0xFF, 0x00, 0xF7,
        0x14, 0x14, 0x14, 0x14, 0x14,
        0x14, 0x14, 0xF7, 0x00, 0xF7,
```

```
        0x14, 0x14, 0x14, 0x17, 0x14,
        0x10, 0x10, 0x1F, 0x10, 0x1F,
        0x14, 0x14, 0x14, 0xF4, 0x14,
        0x10, 0x10, 0xF0, 0x10, 0xF0,
        0x00, 0x00, 0x1F, 0x10, 0x1F,
        0x00, 0x00, 0x00, 0x1F, 0x14,
        0x00, 0x00, 0x00, 0xFC, 0x14,
        0x00, 0x00, 0xF0, 0x10, 0xF0,
        0x10, 0x10, 0xFF, 0x10, 0xFF,
        0x14, 0x14, 0x14, 0xFF, 0x14,
        0x10, 0x10, 0x10, 0x1F, 0x00,
        0x00, 0x00, 0x00, 0xF0, 0x10,
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
        0xFF, 0xFF, 0xFF, 0x00, 0x00,
        0x00, 0x00, 0x00, 0xFF, 0xFF,
        0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
        0x38, 0x44, 0x44, 0x38, 0x44,
        0xFC, 0x4A, 0x4A, 0x4A, 0x34, // sharp-s or beta
        0x7E, 0x02, 0x02, 0x06, 0x06,
        0x02, 0x7E, 0x02, 0x7E, 0x02,
        0x63, 0x55, 0x49, 0x41, 0x63,
        0x38, 0x44, 0x44, 0x3C, 0x04,
        0x40, 0x7E, 0x20, 0x1E, 0x20,
        0x06, 0x02, 0x7E, 0x02, 0x02,
        0x99, 0xA5, 0xE7, 0xA5, 0x99,
        0x1C, 0x2A, 0x49, 0x2A, 0x1C,
        0x4C, 0x72, 0x01, 0x72, 0x4C,
        0x30, 0x4A, 0x4D, 0x4D, 0x30,
        0x30, 0x48, 0x78, 0x48, 0x30,
        0xBC, 0x62, 0x5A, 0x46, 0x3D,
        0x3E, 0x49, 0x49, 0x49, 0x00,
        0x7E, 0x01, 0x01, 0x01, 0x7E,
        0x2A, 0x2A, 0x2A, 0x2A, 0x2A,
        0x44, 0x44, 0x5F, 0x44, 0x44,
        0x40, 0x51, 0x4A, 0x44, 0x40,
        0x40, 0x44, 0x4A, 0x51, 0x40,
        0x00, 0x00, 0xFF, 0x01, 0x03,
        0xE0, 0x80, 0xFF, 0x00, 0x00,
        0x08, 0x08, 0x6B, 0x6B, 0x08,
        0x36, 0x12, 0x36, 0x24, 0x36,
        0x06, 0x0F, 0x09, 0x0F, 0x06,
        0x00, 0x00, 0x18, 0x18, 0x00,
        0x00, 0x00, 0x10, 0x10, 0x00,
        0x30, 0x40, 0xFF, 0x01, 0x01,
        0x00, 0x1F, 0x01, 0x01, 0x1E,
        0x00, 0x19, 0x1D, 0x17, 0x12,
        0x00, 0x3C, 0x3C, 0x3C, 0x3C,
```

```
299         0x00, 0x00, 0x00, 0x00, 0x00
300 };
301
302 extern uint16_t LCD_W,LCD_H;
303
304 void ili9341_drawchar(int16_t x, int16_t y, unsigned char c,uint16_t color, uint16_t bg, uint8_
305 {
306
307         if ((x >=LCD_W) || // Clip right
308             (y >=LCD_H)           || // Clip bottom
309             ((x + 6 * size - 1) < 0) || // Clip left
310             ((y + 8 * size - 1) < 0))   // Clip top
311         {
312                 return;
313         }
314
315         for (int8_t i=0; i<6; i++ )
316         {
317                 uint8_t line;
318
319                 if (i == 5)
320                 {
321                         line = 0x0;
322                 }
323                 else
324                 {
325                         line = pgm_read_byte(font+(c*5)+i);
326                 }
327
328                 for (int8_t j = 0; j<8; j++)
329                 {
330                         if (line & 0x1)
331                         {
332                                 if (size == 1) // default size
333                                 {
334                                         ili9341_drawpixel(x+i, y+j, color);
335                                 }
336                                 else {  // big size
337                                         ili9341_fillrect(x+(i*size), y+(j*size), size, size, co
338                                 }
339                         } else if (bg != color)
340                         {
341                                 if (size == 1) // default size
342                                 {
343                                         ili9341_drawpixel(x+i, y+j, bg);
344                                 }
345                                 else
346                                 {  // big size
```

```
347                                     ili9341_fillrect(x+i*size, y+j*size, size, size, bg);
348                                 }
349                             }
350 
351                             line >>= 1;
352                     }
353             }
354 }
355 
356 void ili9341_setcursor(uint16_t x,uint16_t y)//set cursor at desired location to print data
357 {
358         cursor_x=x;
359         cursor_y=y;
360 }
361 
362 void ili9341_settextcolour(uint16_t x,uint16_t y)//set text colour and text background colour
363 {
364         textcolour=x;
365         textbgcolour=y;
366 }
367 
368 void ili9341_settextsize(uint8_t s)
369 {
370         if(s>8) return;
371         textsize=(s>0) ? s: 1;//this operation means if s0 greater than 0,then s=s,else s=1
372 }
373 
374 void ili9341_write(uint8_t c)//write a character at setted coordinates after setting location a
375 {
376         if (c == '\n')
377         {
378                 cursor_y += textsize*8;
379                 cursor_x  = 0;
380         }
381         else if (c == '\r')
382         {
383                 // skip em
384         }
385         else
386         {
387                 ili9341_drawchar(cursor_x, cursor_y, c, textcolour, textbgcolour, textsize);
388                 cursor_x += textsize*6;
389         }
390 }
391 
392 
393 void display_init(void)//display initial data regarding my power supply
394 {
```

```
395         stdout = & mydata;//it is used for printf function and must be declared locally
396         ili9341_setcursor(4,4);
397         _delay_ms(2);
398         ili9341_settextcolour(GREEN,BLACK);
399         _delay_ms(2);
400         ili9341_settextsize(2);
401         _delay_ms(2);
402         printf("mode - ");
403         _delay_ms(2);
404         ili9341_settextcolour(RED,BLACK);
405         _delay_ms(2);
406         ili9341_settextsize(2);
407         _delay_ms(2);
408         printf("constant voltage");
409         _delay_ms(2);
410         ili9341_setcursor(4,40);
411         _delay_ms(2);
412         ili9341_settextcolour(CYAN,BLACK);
413         _delay_ms(2);
414         ili9341_settextsize(4);
415         _delay_ms(2);
416         ili9341_write('V');
417         _delay_ms(2);
418         cursor_y=cursor_y+6;
419         ili9341_settextsize(3);
420         _delay_ms(2);
421         printf("set\n");
422         _delay_ms(2);
423         cursor_y=cursor_y+12;
424         backuplocationvset();
425         printf("00.00v");
426         _delay_ms(2);
427         ili9341_setcursor(4,120);
428         _delay_ms(2);
429         ili9341_settextsize(4);
430         _delay_ms(2);
431         ili9341_write('V');
432         _delay_ms(2);
433         cursor_y=cursor_y+6;
434         ili9341_settextsize(3);
435         _delay_ms(2);
436         printf("actual\n\n");
437         _delay_ms(2);
438         backuplocationvactual();
439         ili9341_settextsize(5);
440         printf("00.00");
441         _delay_ms(2);
442
```

```
443         ili9341_setcursor(164,40);
444         _delay_ms(2);
445         ili9341_settextcolour(YELLOW,BLACK);
446         _delay_ms(2);
447         ili9341_settextsize(4);
448         _delay_ms(2);
449         ili9341_write('I');
450         _delay_ms(2);
451         cursor_y=cursor_y+6;
452         ili9341_settextsize(3);
453         _delay_ms(2);
454         printf("set");
455         _delay_ms(2);
456         cursor_x=164;
457         cursor_y=(cursor_y+36);
458         backuplocationiset();
459         ili9341_settextsize(3);
460         printf("00.00a");
461         _delay_ms(2);
462         ili9341_setcursor(164,120);
463         _delay_ms(2);
464         ili9341_settextsize(4);
465         _delay_ms(2);
466         ili9341_write('I');
467         _delay_ms(2);
468         cursor_y=cursor_y+6;
469         ili9341_settextsize(3);
470         _delay_ms(2);
471         printf("actual");
472         _delay_ms(2);
473         cursor_x=164;
474         backuplocationiactual();
475         cursor_y=cursor_y+48;
476         ili9341_settextsize(5);
477         printf("00.00");
478         _delay_ms(2000);
479 }
```

ili9341gfx.h

```
1  #ifndef _ILI9341GFX_H_
2  #define _ILI9341GFX_H_
3
4  #include <avr/io.h>
5  #include <avr/pgmspace.h>
6  #include <stdio.h>
7  #include <stdbool.h>
8  #include <inttypes.h>
```

```
9
10 int16_t ili9341_putchar_printf(char var, FILE *stream);
11 void ili9341_drawchar(int16_t x, int16_t y, unsigned char c,uint16_t color, uint16_t bg, uint8_
12 void ili9341_setcursor(uint16_t x,uint16_t y);
13 void ili9341_settextcolour(uint16_t x,uint16_t y);
14 void ili9341_settextsize(uint8_t s);
15 void ili9341_write(uint8_t c);
16 void backuplocationvset(void);
17 void backuplocationvactual(void);
18 void backuplocationiset(void);
19 void backuplocationiactual(void);
20 void display_init(void);
21
22 #endif
```

main.c

```
1  #include <avr/io.h>
2  #include <util/delay.h>
3
4  #include "ili9341.h"
5  #include "ili9341gfx.h"
6
7  extern uint16_t vsetx,vsety,vactualx,vactualy,isetx,isety,iactualx,iactualy;
8
9  static FILE mydata = FDEV_SETUP_STREAM(ili9341_putchar_printf, NULL, _FDEV_SETUP_WRITE);
10
11 int main()
12 {
13         stdout = & mydata;
14
15         ili9341_init(); //initial driver setup to drive ili9341
16         ili9341_clear(BLACK); //fill screen with black colour
17         _delay_ms(1000);
18         ili9341_setRotation(3); //rotate screen
19         _delay_ms(2);
20         display_init(); //display initial data
21
22         while (1)
23         {
24                 ili9341_settextcolour(CYAN,BLACK);
25
26                 ili9341_setcursor(vsetx,vsety);
27                 _delay_ms(2);
28                 ili9341_settextsize(3);
29                 ili9341_write('1');
30                 _delay_ms(2);
31                 ili9341_write('0');
```

```
32                  _delay_ms(2);
33                  ili9341_write('.');
34                  _delay_ms(2);
35                  ili9341_write('2');
36                  _delay_ms(2);
37                  ili9341_write('3');
38                  _delay_ms(2);
39
40
41
42                  ili9341_setcursor(vactualx,vactualy);
43                  _delay_ms(2);
44                  ili9341_settextsize(5);
45                  ili9341_write('1');
46                  _delay_ms(2);
47                  ili9341_write('0');
48                  _delay_ms(2);
49                  ili9341_write('.');
50                  _delay_ms(2);
51                  ili9341_write('2');
52                  _delay_ms(2);
53                  ili9341_write('3');
54                  _delay_ms(2);
55
56                  _delay_ms(2000);
57
58                  ili9341_setcursor(vsetx,vsety);
59                  _delay_ms(2);
60                  ili9341_settextsize(3);
61                  ili9341_write('9');
62                  _delay_ms(2);
63                  ili9341_write('0');
64                  _delay_ms(2);
65                  ili9341_write('.');
66                  _delay_ms(2);
67                  ili9341_write('4');
68                  _delay_ms(2);
69                  ili9341_write('5');
70                  _delay_ms(2);
71
72
73
74                  ili9341_setcursor(vactualx,vactualy);
75                  _delay_ms(2);
76                  ili9341_settextsize(5);
77                  ili9341_write('9');
78                  _delay_ms(2);
79                  ili9341_write('0');
```

```
80                  _delay_ms(2);
81                  ili9341_write('.');
82                  _delay_ms(2);
83                  ili9341_write('4');
84                  _delay_ms(2);
85                  ili9341_write('5');
86                  _delay_ms(2);
87
88                  _delay_ms(2000);
89          }
90  }
```