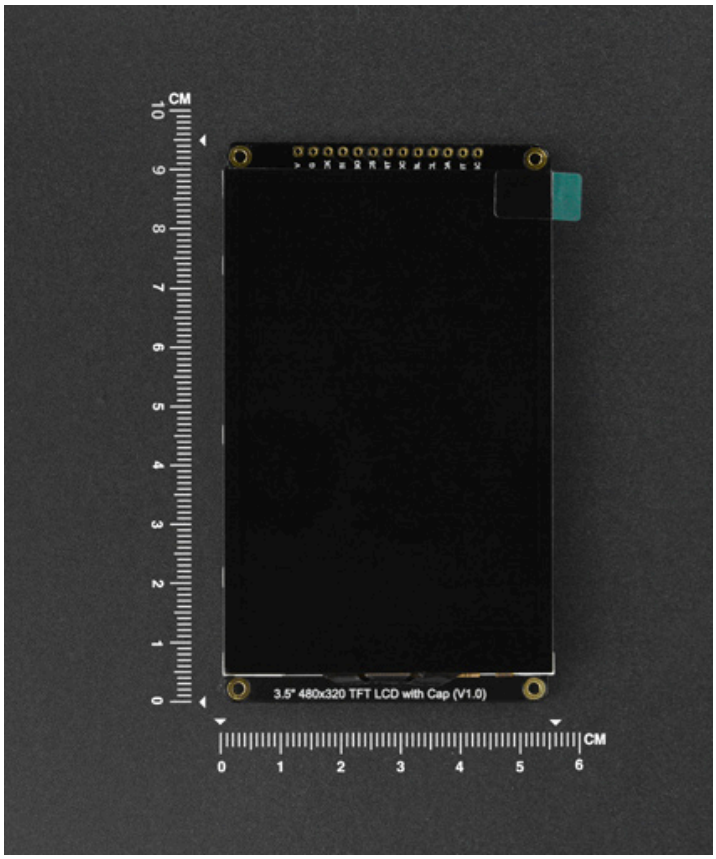
SKU:DFR0669 (https://www.dfrobot.com/product-2107.html)



(https://www.dfrobot.com/product-2107.html)

Introduction

This is a 3.5” IPS capacitive Touchscreen Display. The module, with a resolution of 480x320, adopts ILI9488 as driver IC and SPI (4-line) communication mode. The board integrates touch chip GT911, employing I2C communication to realize multiple touchpoints controlling. The module also integrates an SD card slot allowing you to easily read the full-color bitmap. There are two modes of wiring supplied, normal pin header wiring and GDI. The latter one requires to work with a

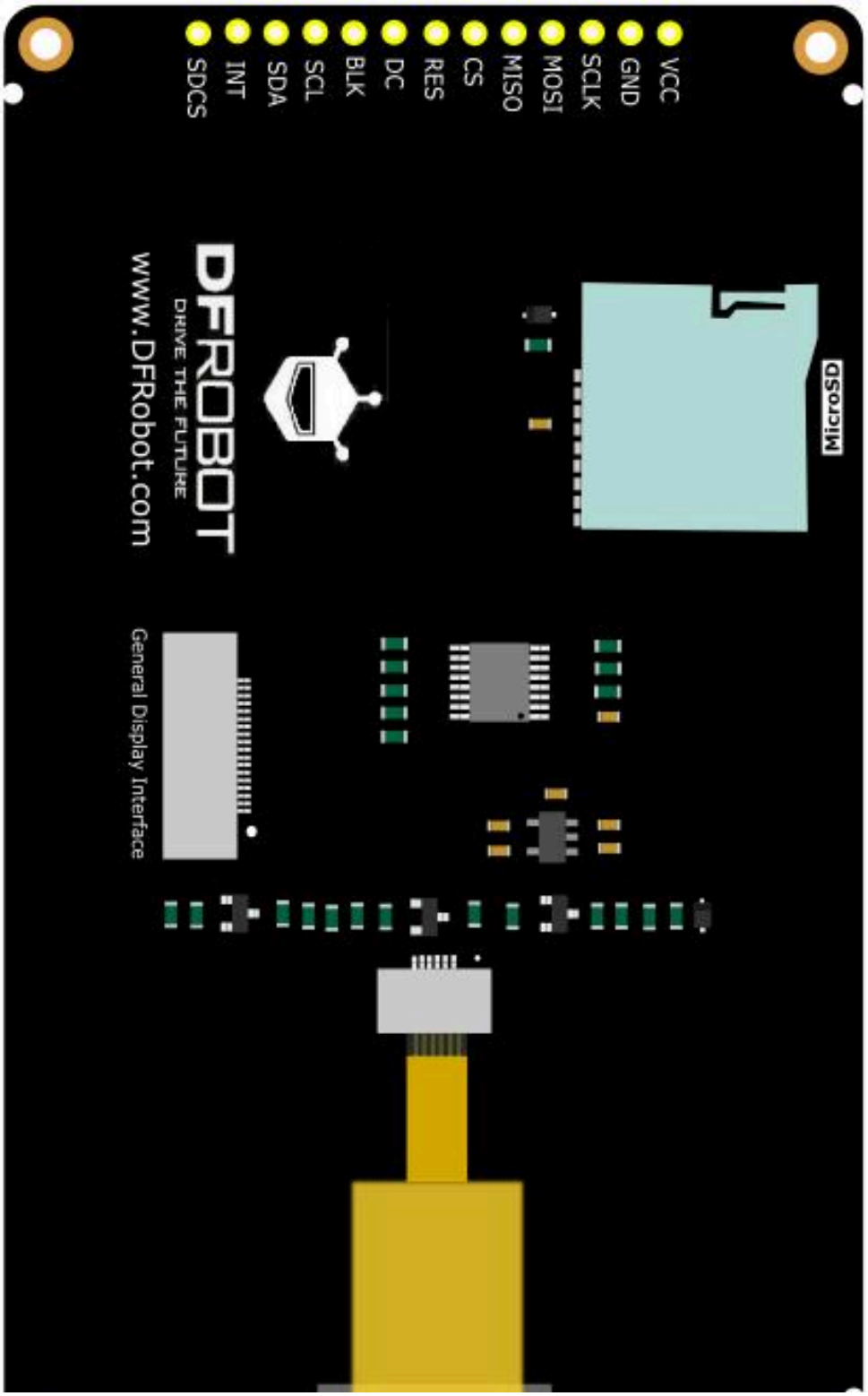
main controller board with a GDI interface (e.g. FireBeetle-M0). You can use it with only one FPC line plugging in, which reduces the complexity of the wiring.

Furthermore, it features high resolution, wide viewing angle, and simple wiring, which can be used in all sorts of display applications, such as, IoT controlling device, game console, desktop event notifier, touch interface, etc.

Specification

- Operating Voltage: 3.3V-5.5 V
- IPS View Angle: 120 degrees
- Color Depth: 16-bit(RGB565)
- Driver Chip: ILI9488
- Touch Chip: GT911
- Pixels: 480x320
- Communication Mode: SPI
- Brightness (cd/m2): 300(TYP)
- Full-screen Consumption: 48.1mA *5V/3.3V （Typ）
- Operating Temperature: -20℃~+70℃
- Display Area: 73.44×48.96 mm
- Module Size: 55.50x95.00 mm
- Mounting Hole Diameter: 2 mm
- Weight:

Board Overview



Num	Label	Description
1	VCC	Positive
2	GND	Negative
3	SCLK	Clock
4	MOSI	Data (Master send; Slave receive)
5	MISO	Data (Master receive; Slave send)
6	CS	Screen Chip Select
7	RES	Reset
8	DC	Data/Command
9	BL	Backlight The backlight has been set to a default value, and can be turned on without connecting the backlight pin. When the backlight pin is connected, input High level(1) to turn the backlight brightness to maximum; Input Low level to turn off backlight.
10	SCL	Touch Clock
11	SDA	Touch Data
12	INT	Touch Interrupt
13	SDCS	SD card chip select

Dimension Diagram

- Dimension: 55.50x95.00mm

- Mounting Hole Pitch: 50mm, 90mm
- Mounting Hole Size: 2.0mm

Tutorial

The product is a Breakout module. It adopts SPI communication and has onboard GDI interface, which reduces the complexity of wiring and can easily display the contents read from SD card.

Note
1. The GDI interface should be used in conjunction with the main controller board with GDI interface
2. It is recommended to use Arduino 1.8.10 and above
3. If the SD card slot is in poor contact, it may fail to initialize. Please try to replug it.

Requirements

- **Hardware**
  - DFRduino UNO R3 (https://www.dfrobot.com/product-838.html) (or similar) x 1
  - 3.5”480x320 TFT LCD with Capacitive Touchscreen x1
  - Wires

- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - DFRobot_GDL Library Files (https://codeload.github.com/DFRobot/DFRobot_GDL/zip/master)
  - About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0)
  - DFRobot_GDL API Functions, click to find more details (https://github.com/DFRobot/DFRobot_GDL/wiki/%E4%B8%AD%E6%96%87-WIKI)

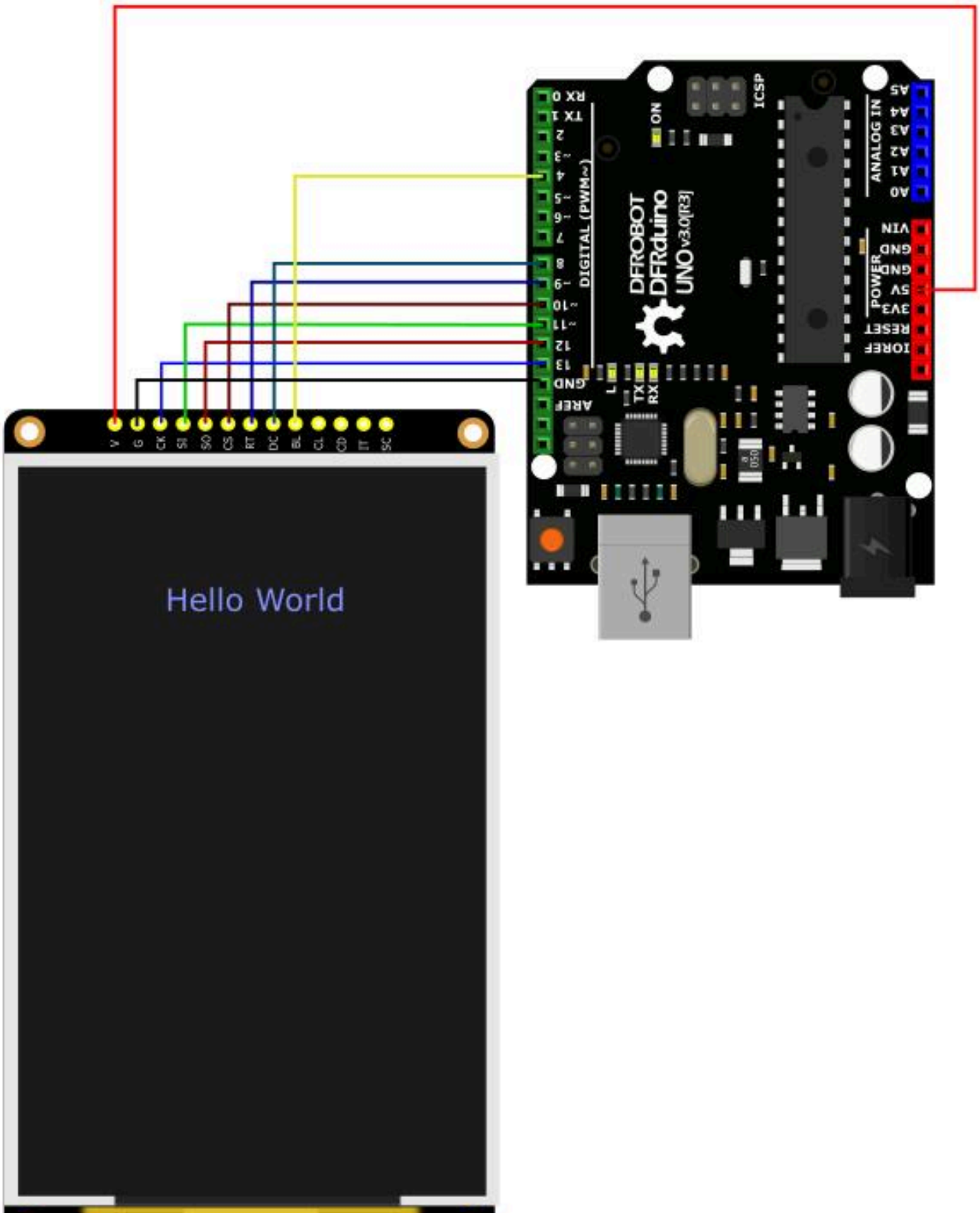
Note

Note

1. All the demos of this product are all stored in the DFRobot_GDL->example->basic file

2. Before burning the demo, please open the corresponding materialized function (DFRobot_ST7789_240x320_HW_SPI)

Connection Diagram



Sample Code 1- Basic Test

This is a basic display sample, including drawing points, lines, circles, rectangles and etc.

```
/*!
 * @file basicTest.ino
 * @brief Demonstrate various graphic painting effects
 * @n This demo supports Arduino Uno, Leonardo, Mega2560, FireBeetle-ESP32, Fi
 * @copyright Copyright (c) 2010 DFRobot Co. Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [LuoYufeng] (yufeng.luo@dfrobot.com)
 * @version V0.1
 * @date 2020-01-07
 * @url https://github.com/DFRobot/DFRobot_GDL
 */
#include "DFRobot_GDL.h"
/*M0*/
#if defined ARDUINO_SAM_ZERO
#define TFT_DC  7
#define TFT_CS  5
#define TFT_RST 6
/*ESP32 and ESP8266*/
#elif defined(ESP32) || defined(ESP8266)
#define TFT_DC  D2
#define TFT_CS  D6
#define TFT_RST D3
/*AVR series mainboard*/
#else
#define TFT_DC  2
#define TFT_CS  3
#define TFT_RST 4
#endif

/**
 * @brief Constructor Constructor of hardware SPI communication
 * @param dc Command/data line pin for SPI communication
 * @param cs Chip select pin for SPI communication
 * @param rst reset pin of the screen
 */
//DFRobot_ST7789_240x240_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT
```

```
//DFRobot_ST7789_240x320_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT
//DFRobot_ILI9341_240x320_HW_SPI  screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/T
//DFRobot_ILI9488_320x480_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TF
/* M0 mainboard DMA transfer */
//DFRobot_ST7789_240x240_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TF
//DFRobot_ST7789_240x320_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TF
//DFRobot_ILI9341_240x320_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/T
//DFRobot_ILI9488_320x480_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/T


/*
 *User-selectable macro definition color
 *COLOR_RGB565_BLACK   COLOR_RGB565_NAVY    COLOR_RGB565_DGREEN   COLOR_RGB565
 *COLOR_RGB565_MAROON  COLOR_RGB565_PURPLE  COLOR_RGB565_OLIVE    COLOR_RGB565
 *COLOR_RGB565_DGRAY   COLOR_RGB565_BLUE    COLOR_RGB565_GREEN    COLOR_RGB565
 *COLOR_RGB565_RED     COLOR_RGB565_MAGENTA COLOR_RGB565_YELLOW   COLOR_RGB565
 *COLOR_RGB565_WHITE
 */

void setup() {
  Serial.begin(115200);
  screen.begin();
}

void loop(){
    testDrawPixel();
    testLine();
    testFastLines(COLOR_RGB565_PURPLE,COLOR_RGB565_YELLOW);
    testRects(COLOR_RGB565_BLACK,COLOR_RGB565_WHITE);
    testRoundRects();
    testCircles(24,COLOR_RGB565_BLUE);
    testTriangles(COLOR_RGB565_YELLOW);
    testPrint();

}

/* Test to draw a pixel*/
void testDrawPixel() {
  //Clear screen
  screen.fillScreen(COLOR_RGB565_BLACK);
  int x = 0;
  int y = screen.height();
```

```
  for(int i = 0; i <= screen.width()/2; i += 10){
    for (x = screen.width() - i; x >= i; x-=10 ){
      /*
       * @ brief draw a pixel
       * @ param x coordinate
       *         y coordinate
       * c pixel color
       */
      screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
      delay(10);
    }

    for (y = screen.height() - i; y >= i; y-=10){
      screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
      delay(10);
    }

    for (x = i; x <= screen.width() - i + 1; x+=10 ){
      screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
      delay(10);
    }

    for (y = i; y <= screen.height() - i + 1; y+=10){
      screen.drawPixel(x, y, COLOR_RGB565_ORANGE);
      delay(10);
    }
  }
}

/* Test to draw a line*/
void testLine(){
// 0x00FF is the color data in the format of RGB565
  uint16_t color = 0x00FF;
  screen.fillScreen(COLOR_RGB565_BLACK);
  for (int16_t x=0; x < screen.width(); x+=6) {
    /*
     * @ brief draw a line
     * @ param x0 The x-coordinate of the first vertex
     *         y0 The y-coordinate of the first vertex
     *         x1 The x-coordinate of the second vertex
     *         y1 The y-coordinate of the second vertex
     *         c line color
```

```
     */
    screen.drawLine(/*x0=*/screen.width()/*Screen width*//2, /*y0=*/screen.hei
  }
  for (int16_t y=0; y < screen.height(); y+=6) {
    screen.drawLine(screen.width()/2, screen.height()/2, screen.width(), y, co
  }

  for (int16_t x = screen.width(); x >= 0; x-=6) {
    screen.drawLine(screen.width()/2, screen.height()/2, x,screen.height(), co
  }

  for (int16_t y = screen.height(); y >= 0; y-=6) {
    screen.drawLine(screen.width()/2, screen.height()/2, 0, y, color+=0x0700);
  }
}

/* Test to fast draw line(need to set delay), only horizontal line and vertica
void testFastLines(uint16_t color1, uint16_t color2) {
  for (int16_t y=0; y < screen.height(); y+=4) {
    /*
     * @ brief draw a line
     * @ param x The x-coordinate of the first vertex
     *         y The y-coordinate of the first vertex
     *         w Length of line segment
     *         c line color
     */
    screen.drawFastHLine(/*x=*/0, /*y=*/y, /*w=*/screen.width(),/*c=*/color2);
    delay(10);
  }

  for(int16_t x=0; x < screen.width(); x+=3) {
    /*
     * @ brief draw a line
     * @ param x The x-coordinate of the first vertex
     *         y The y-coordinate of the first vertex
     *         h length of line segment
     *         c line color
     */
    screen.drawFastVLine(/*x=*/x, /*y=*/0, /*h=*/screen.height(), /*c=*/color1
    delay(10);
  }
}
```

```
/* Test to draw a rectangle*/
void testRects(uint16_t color1, uint16_t color2) {
    screen.fillScreen(COLOR_RGB565_BLACK);
    int16_t x=screen.width()-12;
    for (; x > 100; x-=screen.width()/40) {
      /*
       * @ brief draw a hollow rectangle
       * @ param x The x-coordinate of the vertex
       * @ param y The y-coordinate of the vertex
       * @ param w horizontal side length
       * @ param h longitudinal side length
       * @ param color Fill color, RGB color with 565 structure
       */
      screen.drawRect(/*x=*/screen.width()/2 -x/2, /*y=*/screen.height()/2 -x/
      delay(100);
    }

    /*
     * @ brief draw a filled rectangle
     * @ param x The x-coordinate of the vertex
     * @ param y The y-coordinate of the vertex
     * @ param w horizontal side length
     * @ param h longitudinal side length
     * @ param color Fill color, RGB color with 565 structure
     */
    screen.fillRect(/*x=*/screen.width()/2 -x/2, /*y=*/screen.height()/2 -x/2
    delay(100);
    for(; x > 6; x-=screen.width()/40){
      screen.drawRect(screen.width()/2 -x/2, screen.height()/2 -x/2 , x, x, co
      delay(100);
    }
}

/* Test to draw a rounded rectangle */
void testRoundRects() {
  screen.fillScreen(COLOR_RGB565_BLACK);
// 0xF00F is the color data in the format of RGB565
  int color = 0xF00F;
  int i;
  int x = 0;
  int y = 0;
```

```
  int w = screen.width()-3;
  int h = screen.height()-3;
  for(i = 0 ; i <= 16; i+=2) {
    /*
     * @ brief Draw a hollow rounded rectangle
     * @ param x0 The x-coordinate of the start vertex
     * @ param y0 The y-coordinate of the start vertex
     * @ param w horizontal side length
     * @ param h longitudinal side length
     * @ param radius Round corner radius
     * @ param color border color, 565 structure RGB color
     */
    screen.drawRoundRect(/*x0=*/x, /*y0=*/y, /*w=*/w, /*h=*/h, /*radius=*/20,
    x+=5;
    y+=5;
    w-=10;
    h-=10;
    color+=0x0100;
    delay(50);
  }
  for(i = 0 ; i <= 16; i+=2) {
    /*
     * @ brief Draw a filled and rounded rectangle
     * @ param x0 The x-coordinate of the start vertex
     * @ param y0 The y-coordinate of the start vertex
     * @ param w horizontal side length
     * @ param h longitudinal side length
     * @ param radius Round corner radius
     * @ param color Fill color, RGB color with 565 structure
     */
    screen.fillRoundRect(/*x0=*/x, /*y0=*/y, /*w=*/w, /*h=*/h, /*radius=*/10,
    x+=5;
    y+=5;
    w-=10;
    h-=10;
    color+=0x0500;
    delay(50);
  }
}

/* Test to draw a circle */
void testCircles(uint8_t radius, uint16_t color) {
```

```
  screen.fillScreen(COLOR_RGB565_BLACK);
  for (int16_t x=radius; x <=screen.width()-radius; x+=radius*2) {
    for (int16_t y=radius; y <=screen.height()-radius; y+=radius*2) {
      /*
       * @ brief Draw a hollow circle
       * @ param x0 The x-coordinate of the center point
       * @ param y0 The y-coordinate of the center point
       * @ param r radius
       * @ param color Circle color, RGB color with 565 structure
       */
      screen.drawCircle(/*x0=*/x, /*y0=*/y, /*r=*/radius, /*color=*/color);
        if(x == y ||x == -y ||x == y + 2*radius)
          /*
           * @ brief Draw a filled circle
           * @ param x0 The x-coordinate of the center point
           * @ param y0 The y-coordinate of the center point
           * @ param r radius
           * @ param color Fill color, RGB color with 565 structure
           */
          screen.fillCircle(/*x0=*/x, /*y0=*/y, /*r=*/radius, /*color=*/color)
       color += 800;
       delay(100);
    }
  }
}

/* Test to draw a triangle */
void testTriangles(uint16_t color){
  screen.fillScreen(COLOR_RGB565_BLACK);

  for (int16_t i=0; i <=screen.width(); i+=24)
    /*
     * @ brief Draw a hollow triangle
     * @ param x0 The x-coordinate of the start vertex
     * @ param y0 The y-coordinate of the start vertex
     * @ param x1 The x-coordinate of the second vertex
     * @ param y1 The y-coordinate of the second vertex
     * @ param x2 The x-coordinate of the third vertex
     * @ param y2 The y-coordinate of the third vertex
     * @ param color border color, 565 structure RGB color
     */
    screen.drawTriangle(/*x0=*/i,/*y0=*/0,/*x1=*/0,/*y1=*/screen.height()-i,/*
```

```
  for (int16_t i=0; i <screen.width(); i+=24)
    screen.drawTriangle(screen.width(),i*4/3,0,screen.height()-i*4/3,i,0, colo

  for (int16_t i=0; i <screen.width(); i+=24)
    screen.drawTriangle(screen.width(),i*4/3,i,0,screen.width()-i,screen.heigh

  color = COLOR_RGB565_RED;
  for (int16_t i=0; i <=screen.width(); i+=24)
     /*
      * @ brief Draw a filled triangle
      * @ param x0 The x-coordinate of the start vertex
      * @ param y0 The y-coordinate of the start vertex
      * @ param x1 The x-coordinate of the second vertex
      * @ param y1 The y-coordinate of the second vertex
      * @ param x2 The x-coordinate of the third vertex
      * @ param y2 The y-coordinate of the third vertex
      * @ param color Fill color, RGB color with 565 structure
      */
    screen.fillTriangle(/*x0=*/i,/*y0=*/0,/*x1=*/0,/*y1=*/screen.height()-i,/*

  for (int16_t i=0; i <screen.width(); i+=24)
    screen.fillTriangle(screen.width(),i*4/3,0,screen.height()-i*4/3,i,0, colo

  for (int16_t i=0; i <screen.width(); i+=24)
    screen.fillTriangle(screen.width(),i*4/3,i,0,screen.width()-i,screen.heigh
}

void testPrint() {
  // 0x00FF is the color data in the format of RGB565
  int16_t color = 0x00FF;
   // Set text wrapping mode
   // true = Text word wrap, false = No word wrap
  screen.setTextWrap(false);
  //Fill color, RGB color with 565 structure
  screen.fillScreen(COLOR_RGB565_BLACK);

  //Set the coordinate position x = 0, y = 50
  screen.setCursor(0, 50);
  //Set the text color; this is a changeable value
  screen.setTextColor(color+=0x3000);
  //Set text size to 0
```

```
    screen.setTextSize(0);
    //Output text
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 1
    screen.setTextSize(1);
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 2
    screen.setTextSize(2);
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 3
    screen.setTextSize(3);
    screen.println("Hello World!");

    screen.setTextColor(color+=0x3000);
    //Set text size to 4
    screen.setTextSize(4);
    screen.println("Hello!");
    //Set text size to 5
    screen.setTextSize(5);
    screen.print("Hello!");
    delay(2000);

    //Set coordinate position x = 0, y = 0
    screen.setCursor(0, 0);
    //Fill color, RGB color with 565 structure
    screen.fillScreen(COLOR_RGB565_BLACK);
    screen.setTextSize(2);
    screen.setTextColor(color+=0x3000);
    screen.print("a = ");

    screen.setTextColor(color+=0x3000);
    int a = 1234;
    screen.println(a, 1);
    screen.setTextColor(color+=0x3000);
    screen.print(8675309, HEX);
    screen.println("this is HEX!");
```

```
  screen.println("");

  screen.setTextColor(color+=0x0F00);
  screen.println("running for: ");
  screen.setTextColor(color+=0x0F00);
  //Output time in millisecond
  screen.print(millis());
  screen.setTextColor(color+=0x0F00);
  screen.println("/1000 seconds.");

  char *text = "Hi DFRobot!";
  screen.setTextColor(color+=0x0F00);
  screen.setTextWrap(true);
  screen.setTextSize(3);
  screen.println(text);
  //screen.setFonts((const gdl_Font_t *)SIMKAIFont18ptBitmaps);
  screen.println(text);
  delay(2000);
}
```

Expected Result 1

Sample Code 2 - Gesture Font

This is UI control demo -- digital keyboard. At the start, click the textbox, and click the number after the cursor shows in the textbox. The corresponding number will be shown in the textbox. The "x" at the bottom right corner is used to delete the context in the textbox.

```
/*!
 * @file gettureFont.ino
 * @brief Control the words in the center of the screen by zooming in, zooming
 * @n The demo supports Arduino Uno, Mega2560, FireBeetle-ESP32, FireBeetle-ES
 *
 * @copyright Copyright (c) 2010 DFRobot Co. Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [fengli] (li.feng@dfrobot.com)
 * @version V1.0
 * @date 2019-12-6
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/DFRobot_GDL/src/DFRpbot_UI
*/
#include "DFRobot_UI.h"
#include "Arduino.h"
#include "DFRobot_GDL.h"
#include "DFRobot_Touch.h"
/*M0*/
#if defined ARDUINO_SAM_ZERO
#define TFT_DC  7
#define TFT_CS  5
#define TFT_RST 6
/*ESP32 and ESP8266*/
#elif defined(ESP32) || defined(ESP8266)
#define TFT_DC  D2
#define TFT_CS  D6
#define TFT_RST D3
/* AVR series mainboard */
#else
#define TFT_DC  2
#define TFT_CS  3
#define TFT_RST 4
#endif

/**
   @brief Constructor  When the touch uses the gt series chip, you can call th
```

```
*/
DFRobot_Touch_GT911 touch;


/**
   @brief Constructor When the screen uses hardware SPI communication, the dri
   @param dc Command/data line pin for SPI communication
   @param cs Chip select pin for SPI communication
   @param rst Reset pin of the screen
*/
DFRobot_ILI9488_320x480_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_
/* M0 mainboard DMA transfer */
//DFRobot_ILI9488_320x480_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/T


/**
   @brief Constructor
   @param gdl Screen object
   @param touch Touch object
*/
DFRobot_UI ui(&screen, &touch);

uint8_t fontSize = 2; //Font size
uint16_t fontColor = 0xf800; //Font color
uint8_t fontLen = 4 * fontSize * 8; //String width
char * font = "DFRC"; // String
uint8_t fontHeight = fontSize * 8;//String height
uint16_t posx =  screen.width() / 2 ; //The x-coordinate of the starting posit
uint16_t posy =  screen.height()  / 2;//the y-coordinate of the beginning of t

void setup()
{

  Serial.begin(9600);
  //UI initialization
  ui.begin();
  //Draw the initial string
  ui.drawString(posx - fontLen / 2-8, posy - fontHeight / 2, font, fontColor ,


}
```

```
void loop()
{
  refresh();
//Serial.println(touch.scan());
}

void refresh() {
  DFRobot_UI:: eGesture_t gesture = ui.getGestures();
  /*if(gesture != ui.NONE){
    Serial.println(gesture);
    }
  */
  switch (gesture) {
    //Gesture detected as zoom out
    case ui.SHRINK : {
        screen.fillRect(posx - fontLen / 2 -8,  posy - fontHeight / 2, 4 * for
        //fontSize decrease
             fontSize--;
        if (fontSize <= 1) fontSize = 1;

        fontHeight = fontSize * 8;
        fontLen = 4 * fontSize * 8;
        ui.drawString(posx - fontLen / 2-8 ,  posy - fontHeight / 2,  font, fo
      }; break;
    //Gesture detected as zoom in
    case ui.MAGNIFY : {
        screen.fillRect(posx - fontLen / 2-8 ,  posy - fontHeight / 2, 4 * for
        //fontSize increase
            fontSize++;
        if (fontSize >= 7) fontSize = 7;
        fontHeight = fontSize * 8;
        fontLen = 4 * fontSize * 8;
        ui.drawString(posx - fontLen / 2-8 , posy - fontHeight / 2, font, font
      } break;
    //Gesture detected as swiping up with two fingers
    case ui.DUPGLIDE : {
        screen.fillRect(posx - fontLen / 2-8 ,   posy - fontHeight / 2, 4 * fo
        posy -= 10 ;
        fontHeight = fontSize * 8;
        fontLen = 4 * fontSize * 8;
        ui.drawString(posx - fontLen / 2 - 8,  posy - fontHeight / 2, font, fo
```

```
      } break;
    //Gesture detected as swiping down with two fingers
    case ui.DDOWNGLIDE : {
        screen.fillRect(posx - fontLen / 2 - 8,  posy - fontHeight / 2, 4 * fo
        posy += 10;
        fontHeight = fontSize * 8;
        fontLen = 4 * fontSize * 8;
        ui.drawString(posx - fontLen / 2 - 8,  posy - fontHeight / 2, font, fo
      } break;
    case ui.NONE: {
        return;
      }
  }
}
```

Result 2

Sample Code 3- Rotate

Use two fingers to operate: clockwise rotation, counterclockwise rotation.

```
/*!
 * @file rotate.ino
 * @brief Two fingers rotate left or right to change the display direction
 * @n The demo supports Mega2560, FireBeetle-ESP32, FireBeetle-ESP8266, FireBe
 * @copyright Copyright (c) 2010 DFRobot Co. Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [fengli] (li.feng@dfrobot.com)
 * @version V1.0
 * @date 2019-12-6
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/DFRobot_GDL/src/DFRpbot_UI
*/
#include "DFRobot_UI.h"
#include "Arduino.h"
#include "DFRobot_GDL.h"
#include "DFRobot_Touch.h"
#include "GrayscaleBitmap1.h"
/*M0*/
#if defined ARDUINO_SAM_ZERO
#define TFT_DC  7
#define TFT_CS  5
#define TFT_RST 6
/*ESP32 and ESP8266*/
#elif defined(ESP32) || defined(ESP8266)
#define TFT_DC  D2
#define TFT_CS  D6
#define TFT_RST D3
/* AVR series motherboard */
#else
#define TFT_DC  2
#define TFT_CS  3
#define TFT_RST 4
#endif

/**
   @brief Constructor   When the touch uses the gt series chip, you can call t
```

```
*/
DFRobot_Touch_GT911 touch;

/**
   @brief Constructor When the screen uses hardware SPI communication, the dri
   @param dc Command/data line pin for SPI communication
   @param cs Chip select pin for SPI communication
   @param rst Reset pin of the screen
*/
DFRobot_ILI9488_320x480_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_
/* M0 mainboard DMA transfer */
//DFRobot_ILI9488_320x480_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/T



/**
   @brief Constructor
   @param gdl Screen object
   @param touch Touch object
*/
DFRobot_UI ui(&screen, &touch);
int8_t rotate =0;
void setup()
{

  Serial.begin(9600);
  //UI initialization
  ui.begin();
  //Draw picture
  screen.drawRGBBitmap(/*x=*/(screen.width()-100)/2,/*y=*/(screen.height()-100
}


void loop()
{

    //getGestures(): Recognize gestures
    DFRobot_UI:: eGesture_t gesture = ui.getGestures();

   switch (gesture) {
    //Clockwise rotation
    case ui.DWROTATE : {
```

```
      rotate++;
      if(rotate>3) rotate=0;
      //Set screen display orientation
       screen.setRotation(rotate);
       screen.drawRGBBitmap(/*x=*/(screen.width()-100)/2,/*y=*/(screen.height(

      } break;
    //Anticlockwise rotation
    case ui.DCWROTATE : {
      if(rotate<0) {rotate=3;}
      else{
        rotate--;
      }
       screen.setRotation(rotate);
       screen.drawRGBBitmap(/*x=*/(screen.width()-100)/2,/*y=*/(screen.height(


      } break;
    case ui.NONE: {
        return;
      }
  }


}
```

Compatibility Test

MCU	Pass	Failed	Not Test	Remarks
FireBeetle-ESP32	√			
FireBeetle-ESP8266	√			
Arduino Uno	√			
Leonardo	√			
Mega2560	√			

Arduino M0	√			

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/).

More Documents

- [[DFR0669]Schematic Diagram](3.5” 480x320 TFT LCD with Cap-Schematic.pdf (https://dfimg.dfrobot.com/nobody/wiki/52bc3cb510e527902cb5edc9c3a8c966.pdf) "[DFR0669]Schematic Diagram")
- [[DFR0669]Dimension Diagram](3.5” 480x320 TFT LCD with Cap-Dimensions.pdf (https://dfimg.dfrobot.com/nobody/wiki/12a15c3f3ccd3c3106ece9d5b80ad355.pdf) "[DFR0669]Dimension Diagram")
- [[DFR0669]Chip Manual](电容触控芯片GT911 Datasheet_20130108.pdf (https://dfimg.dfrobot.com/nobody/wiki/7851e6cd4275cf3fb6706dc2a8085a72.pdf) "[DFR0669]Chip Manual")