BMduino-Shield

2.8" TFT-LCD 显示扩充板

# BMD58T280
# Arduino Library V1.0.3 说明

版本：V1.30　日期：2024-03-19

www.bestmodulescorp.com

# 目录

# 简介

BMD58T280 是倍创推出的一款 2.8 寸 TFT-LCD 显示扩充板，使用 SPI 与 EBI 通信方式。本文档对 BMD58T280 的 Arduino Lib 函数、Arduino Lib 安装方式进行说明；范例演示 TFT 显示。

# Arduino Lib 函数

| Arduino Lib 名称：BMD58T280 | | Lib 版本：V1.0.3 |
|---|---|---|
| Class 名称：BMD58T280 | | |
| **构造函数 & 初始化** | | |
| 1 | BMD58T280() | |
| | 描述 | 构造函数，配置 EBI 接口 |
| | 参数 | — |
| | 返回值 | — |
| | 备注 | Arduino UNO 不支持此接口 |
| 2 | BMD58T280(SPIClass *spiClass) | |
| | 描述 | 构造函数，配置 SPI 接口 |
| | 参数 | *spiClass：SPI 参数 |
| | 返回值 | — |
| | 备注 | — |
| 3 | void begin(uint32_t freq=0) | |
| | 描述 | 模块初始化，配置 SPI 频率 |
| | 参数 | freq：SPI 频率。当 freq=0 时，SPI 频率为 $f_{cpu}$/2Hz ( 默认 )。 |
| | 返回值 | void |
| | 备注 | — |
| **功能函数** | | |
| 4 | void image(ImageInf & img, uint16_t x, uint16_t y) | |
| | 描述 | 显示 SD 中的 BMP 图档 |
| | 参数 | & img：图档名称<br>x：显示的起始位置 x 坐标<br>y：显示的起始位置 y 坐标 |
| | 返回值 | void |
| | 备注 | — |
| 5 | void scrollTo(uint16_t y) | |
| | 描述 | 移动图像 |
| | 参数 | y：往左移动 y 个像素点 |
| | 返回值 | void |
| | 备注 | 将图像沿 y 轴往左移动 y 个像素点 |

| | | |
|---|---|---|
| 6 | void drawPixels(int16_t x, int16_t y, int16_t w, int16_t h, const uint8_t *pColor) | |
| | 描述 | 在设定起始位置 (x, y)，宽长 (w, h) 范围内的视窗里，画多个像素 pColor |
| | 参数 | x：起始的 x 坐标<br>y：起始的 y 坐标<br>w：宽度<br>h：高度<br>*pColor：像素点颜色数组，一个像素点的颜色由 2-bytes 组成，最多可写 w×h 个像素点 |
| | 返回值 | void |
| | 备注 | 一个像素点的颜色：<br>Bit 11~Bit 8：RED 颜色参数<br>Bit 7~Bit 4：GREEN 颜色参数<br>Bit 3~Bit 0：BLUE 颜色参数<br>视窗内无填写颜色时，默认无颜色 |
| 7 | int16_t height(void) | |
| | 描述 | 取得屏幕高度 |
| | 参数 | void |
| | 返回值 | 屏幕高度 |
| | 备注 | 此扩充板屏幕高度依转动位置，可为 320 或 240 |
| 8 | int16_t width(void) | |
| | 描述 | 取得屏幕宽度 |
| | 参数 | void |
| | 返回值 | 屏幕宽度 |
| | 备注 | 此扩充板屏幕宽度依转动位置，可为 240 或 320 |
| 9 | uint8_t getRotation(void) | |
| | 描述 | 取得 LCD 屏幕坐标顺时针转动值 (2) |
| | 参数 | void |
| | 返回值 | r：顺时针转动值<br>0：0 度<br>1：90 度<br>2：180 度<br>3：270 度 |
| | 备注 | 坐标变动详情见注 2 |
| 10 | void text(const char *text, int16_t x, int16_t y) | |
| | 描述 | 在 (x, y) 位置上显示字符串 |
| | 参数 | *text：字符串<br>x：x 坐标<br>y：y 坐标 |
| | 返回值 | void |
| | 备注 | 字符的大小：可通过 setTextSize 函数设置<br>字符的颜色：可通过 stroke 函数设置 |

| | | |
|---|---|---|
| 11 | void text(const char * text, int16_t x, int16_t y, uint16_t textcolor, uint16_t textbgcolor) | |
| | 描述 | 在 (x，y) 位置上显示字符串 |
| | 参数 | *text：字符串<br>x：x 坐标<br>y：y 坐标<br>textcolor：字符串颜色 (1)<br>textbgcolor：字符串背景颜色 (1) |
| | 返回值 | void |
| | 备注 | 直接设置的字符串颜色和字符串背景颜色，跟“stroke”和“background”函数设置无关 |
| 12 | void textWrap(const char *text, int16_t x, int16_t y) | |
| | 描述 | 在 (x, y) 位置上显示字符串，自动换行 |
| | 参数 | *text：字符串<br>x：x 坐标<br>y：y 坐标 |
| | 返回值 | void |
| | 备注 | 字符的大小：可通过 setTextSize 函数设置<br>字符的颜色：可通过 stroke 函数设置 |
| 13 | void textWrap(const char * text, int16_t x, int16_t y, uint16_t textcolor, uint16_t textbgcolor) | |
| | 描述 | 在 (x，y) 位置上显示字符串，自动换行 |
| | 参数 | *text：字符串<br>x：x 坐标<br>y：y 坐标<br>textcolor：字符串颜色 (1)<br>textbgcolor：字符串背景颜色 (1) |
| | 返回值 | void |
| | 备注 | 直接设置的字符串颜色和字符串背景颜色，跟“stroke”和“background”函数设置无关 |
| 14 | void circle(int16_t x, int16_t y, int16_t r) | |
| | 描述 | 画圆 |
| | 参数 | x：圆心 x 坐标<br>y：圆心 y 坐标<br>r：半径 |
| | 返回值 | void |
| | 备注 | 圆框的颜色：可通过 stroke 函数设置<br>圆内的颜色：可通过 fill 函数设置 |
| 15 | void point(int16_t x, int16_t y) | |
| | 描述 | 画点 |
| | 参数 | x：画点的 x 坐标<br>y：画点的 y 坐标 |
| | 返回值 | — |
| | 备注 | 点的颜色：可通过 stroke 函数设置 |

| | | |
|---|---|---|
| 16 | void line(int16_t x1, int16_t y1, int16_t x2, int16_t y2) | |
| | 描述 | 画线 |
| | 参数 | x1：起始点位的 x 坐标<br>y1：起始点位的 y 坐标<br>x2：末尾点位的 x 坐标<br>y2：末尾点位的 y 坐标 |
| | 返回值 | void |
| | 备注 | (x1, y1) 到 (x2, y2) 画线<br>线的颜色：可通过 stroke 函数设置 |
| 17 | void rect(int16_t x, int16_t y, int16_t width, int16_t height) | |
| | 描述 | 画矩形 |
| | 参数 | x：起始位置的 x 坐标<br>y：起始位置的 y 坐标<br>width：宽度<br>height：高度 |
| | 返回值 | void |
| | 备注 | 矩形框的颜色：可通过 stroke 函数设置<br>矩形内的颜色：可通过 fill 函数设置 |
| 18 | void rect(int16_t x, int16_t y, int16_t width, int16_t height, int16_t radius) | |
| | 描述 | 画圆角矩形 |
| | 参数 | x：起始的 x 坐标<br>y：起始的 y 坐标<br>width：宽度<br>height：高度<br>radius：圆角半径 |
| | 返回值 | void |
| | 备注 | 矩形框的颜色：可通过 stroke 函数设置<br>矩形内的颜色：可通过 fill 函数设置 |
| 19 | void triangle(int16_t x1, int16_t y1, int16_t x2, int16_t y2, int16_t x3, int16_t y3) | |
| | 描述 | 画三角形 |
| | 参数 | x1：点 1 的 x 坐标<br>y1：点 1 的 y 坐标<br>x2：点 2 的 x 坐标<br>y2：点 2 的 y 坐标<br>x3：点 3 的 x 坐标<br>y3：点 3 的 y 坐标 |
| | 返回值 | void |
| | 备注 | 三角形框的颜色：可通过 stroke 函数设置<br>三角形内的颜色：可通过 fill 函数设置 |
| **参数配置** | | |
| 20 | void setScrollMargins(uint16_t top, uint16_t bottom) | |
| | 描述 | 设置滚动边距 |
| | 参数 | top：顶部<br>bottom：底部 |
| | 返回值 | void |
| | 备注 | — |

| | | |
|---|---|---|
| 21 | void setAddrWindow(uint16_t x, uint16_t y, uint16_t w, uint16_t h) | |
| | 描述 | 设定视窗 |
| | 参数 | x：起始的 x 坐标<br>y：起始的 y 坐标<br>w：宽度<br>h：高度 |
| | 返回值 | void |
| | 备注 | 在屏幕中选择特定范围，可插入 BMP 文件或 C-array<br>插入 BMP：image 函数<br>插入 C-array：drawPixels 函数 |
| 22 | void setRotation(uint8_t r) | |
| | 描述 | 设设定 LCD 坐标顺时针转动 (2) |
| | 参数 | r：顺时针转动的角度<br>0：0 度<br>1：90 度<br>2：180 度<br>3：270 度 |
| | 返回值 | void |
| | 备注 | 坐标变动详情见注 2 |
| 23 | void fill(uint8_t red, uint8_t green, uint8_t blue) | |
| | 描述 | 设定填满颜色，可以填圆形、三角形等封闭式图形 |
| | 参数 | red：R 颜色<br>green：G 颜色<br>blue：B 颜色 |
| | 返回值 | void |
| | 备注 | R 颜色、G 颜色、B 颜色组成 RGB 颜色 |
| 24 | void fill(uint16_t c) | |
| | 描述 | 设定填满颜色 |
| | 参数 | c：颜色常数 ( 见颜色常数表 (1) ) |
| | 返回值 | void |
| | 备注 | — |
| 25 | void noFill() | |
| | 描述 | 设定无填满效果 |
| | 参数 | — |
| | 返回值 | void |
| | 备注 | 透明 |
| 26 | void stroke(uint8_t red, uint8_t green, uint8_t blue) | |
| | 描述 | 设定字体、外框颜色 |
| | 参数 | red：R 颜色<br>green：G 颜色<br>blue：B 颜色 |
| | 返回值 | void |
| | 备注 | R 颜色、G 颜色、B 颜色组成 RGB 颜色 |

| | | |
|---|---|---|
| 27 | void stroke(uint16_t c) | |
| | 描述 | 设定字体、外框颜色 |
| | 参数 | c：颜色常数 ( 见颜色常数表 (1) ) |
| | 返回值 | void |
| | 备注 | — |
| 28 | void noStroke() | |
| | 描述 | 设定无字体、外框 |
| | 参数 | — |
| | 返回值 | void |
| | 备注 | — |
| 29 | void background(uint8_t red, uint8_t green, uint8_t blue) | |
| | 描述 | 设定背景颜色 |
| | 参数 | red：R 颜色<br>green：G 颜色<br>blue：B 颜色 |
| | 返回值 | void |
| | 备注 | R 颜色、G 颜色、B 颜色组成 RGB 颜色 |
| 30 | void background(uint16_t c) | |
| | 描述 | 设定背景颜色 |
| | 参数 | c：颜色常数 ( 见颜色常数表 (1) ) |
| | 返回值 | void |
| | 备注 | — |
| 31 | void setTextSize(uint8_t s) | |
| | 描述 | 设定字体大小 |
| | 参数 | s：字体大小，范围 1~255，预设为 1 |
| | 返回值 | void |
| | 备注 | — |
| Class 名称：BmpImage | | |
| 构造函数 & 初始化 | | |
| 1 | BmpImage() | |
| | 描述 | BMP 图档，构造函数 |
| | 参数 | — |
| | 返回值 | — |
| | 备注 | — |
| 功能函数 | | |
| 2 | operator bool() | |
| | 描述 | 判断档案是否存在 |
| | 参数 | — |
| | 返回值 | 存在情况：<br>true：档案存在<br>false：档案不存在 |
| | 备注 | — |

| | | |
|---|---|---|
| 3 | int width() | |
| | 描述 | 取得图片宽度 |
| | 参数 | — |
| | 返回值 | 图片宽度 |
| | 备注 | — |
| 4 | int height() | |
| | 描述 | 取得图片高度 |
| | 参数 | — |
| | 返回值 | 图片高度 |
| | 备注 | — |
| **参数配置** | | |
| 5 | bool loadImage(const char *fileName, HardwareSerial *pSerial) | |
| | 描述 | 从 SD 载入 BMP 档 |
| | 参数 | *fileName：BMP 名称<br>*pSerial：预设 NULL，当 pSerial 为 NULL 不会显示 SD 连线相关讯息 |
| | 返回值 | 档案是否存在：<br>true：档案存在<br>false：档案不存在 |
| | 备注 | — |
| Class 名称：BM_XPT2046 | | |
| **构造函数 & 初始化** | | |
| 1 | BM_XPT2046(uint8_t cspin cspin=18, uint8_t tirq=19) | |
| | 描述 | 构造函数，配置 CS 和 TP IRQ 引脚 |
| | 参数 | cspin：CS 引脚<br>tirq：TP IRQ 引脚 |
| | 返回值 | — |
| | 备注 | — |
| 2 | void begin(SPIClass &wspi) | |
| | 描述 | 初始化触摸 IC |
| | 参数 | &wspi：配置 SPI 接口 |
| | 返回值 | void |
| | 备注 | — |
| **功能函數** | | |
| 3 | bool operator==(TS_Point p) | |
| | 描述 | 坐标与手指压力点是否相同 |
| | 参数 | — |
| | 返回值 | 执行情况：<br>false：相同<br>true：不相同 |
| | 备注 | — |

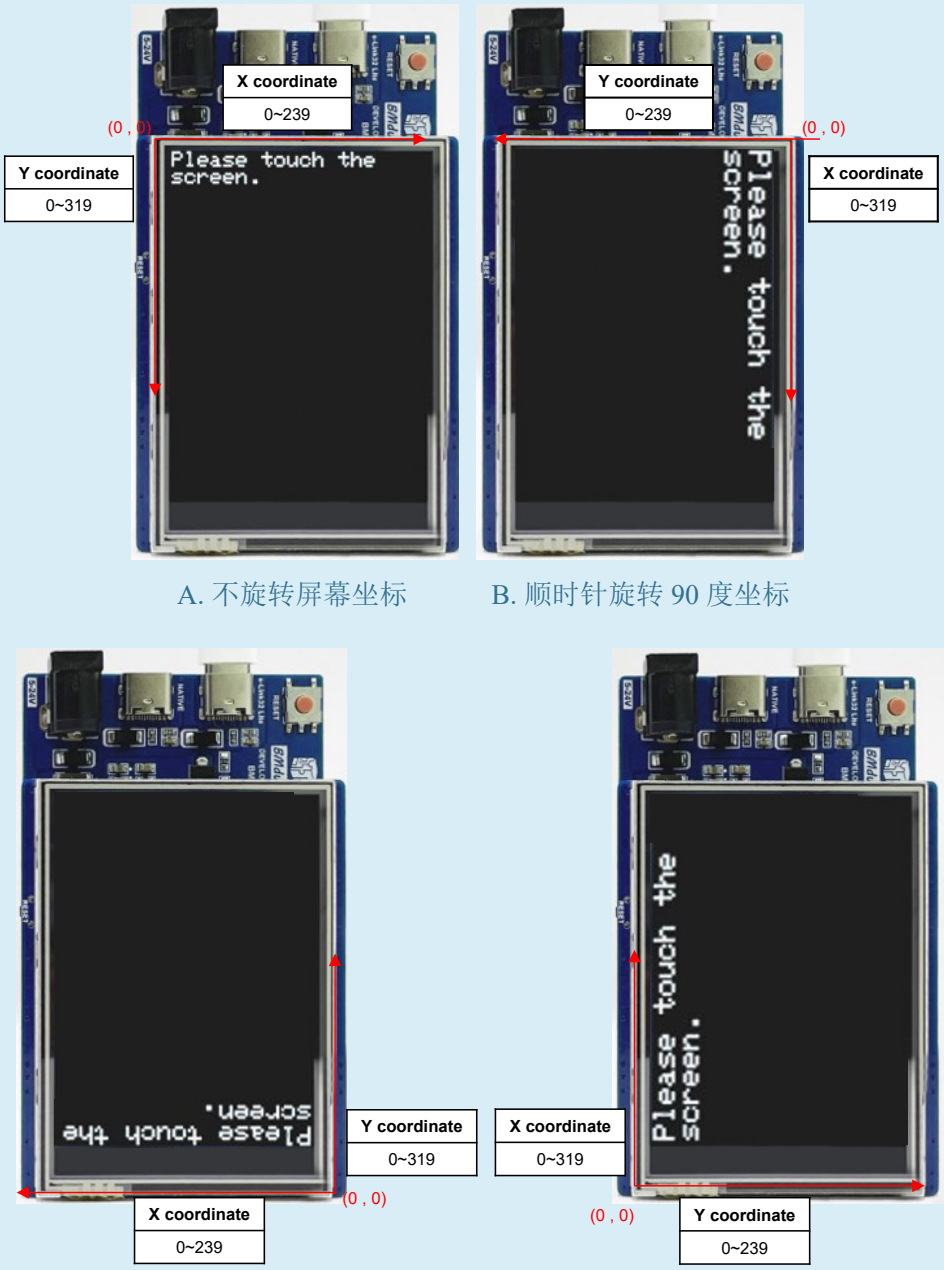
| | | |
|---|---|---|
| 4 | bool operator!=(TS_Point p) | |
| | 描述 | 坐标与压力是否不相同 |
| | 参数 | — |
| | 返回值 | 执行情况：<br>false：不相同<br>true：相同 |
| | 备注 | — |
| 5 | TS_Point getPoint() | |
| | 描述 | 回传手指压力点坐标 |
| | 参数 | — |
| | 返回值 | _xraw：x 坐标<br>_yraw：y 坐标<br>_zraw：z 坐标 |
| | 备注 | 返回值存储在 TS_Point 类中，获取压力点坐标时需要调用类 TS_Poin (int16_t x, int16_t y, int16_t z)，具体可参考范例 |
| 6 | bool touched() | |
| | 描述 | LCD 是否被触碰 |
| | 参数 | — |
| | 返回值 | LCD 是否被触碰情况：<br>true：LCD 被触碰<br>false：LCD 未被触碰 |
| | 备注 | — |
| **参数配置** | | |
| 7 | void setRotation(uint8_t n) | |
| | 描述 | 触摸屏坐标顺时针旋转 (2) |
| | 参数 | n：顺时针旋转角度<br>0：0 度<br>1：90 度<br>2：180 度<br>3：270 度 |
| | 返回值 | void |
| | 备注 | |
| Class 名称：SDReadRawData | | |
| **构造函数 & 初始化** | | |
| 1 | SDReadRawData(void) | |
| | 描述 | 创建结构函数 |
| | 参数 | — |
| | 返回值 | — |
| | 备注 | Arduino UNO 不支援 |

| | | |
|---|---|---|
| 2 | uint8_t init(uint8_t chipSelectPin) | |
| | 描述 | 初始化 SD 卡 |
| | 参数 | chipSelectPin：SPI CS 引脚 |
| | 返回值 | 执行情况：<br>true：设置成功<br>false：设置失败 |
| | 备注 | Arduino UNO 不支援 |
| 功能函数 | | |
| 3 | uint8_t readBlock(uint32_t block, uint16_t count, uint8_t *dst) | |
| | 描述 | 读取 SD 的 block 数据 |
| | 参数 | block：数据块索引<br>count：数据长度<br>*dst：终点 |
| | 返回值 | — |
| | 备注 | Arduino UNO 不支援 |
| 4 | void waitDmaFinish() | |
| | 描述 | 等待数据读取完成 |
| | 参数 | — |
| | 返回值 | void |
| | 备注 | Arduino UNO 不支援 |

注 1：颜色常数表

| |
|---|
| BM_ILI9341::NAVY – 海军蓝 |
| BM_ILI9341:: BLACK – 黑色 |
| BM_ILI9341:: DARKGREEN – 深绿色 |
| BM_ILI9341::DARKCYAN – 深青色 |
| BM_ILI9341::MAROON – 栗色 |
| BM_ILI9341::PURPLE – 紫色 |
| BM_ILI9341::OLIVE – 橄榄绿 |
| BM_ILI9341::LIGHTGREY – 浅灰色 |
| BM_ILI9341::DARKGREY – 深灰色 |
| BM_ILI9341::BLUE – 蓝色 |
| BM_ILI9341::GREEN – 绿色 |
| BM_ILI9341::CYAN – 蓝绿色 |
| BM_ILI9341::RED – 红色 |
| BM_ILI9341::MAGENTA – 洋红色 |
| BM_ILI9341::YELLOW – 黄色 |
| BM_ILI9341::WHITE – 白色 |
| BM_ILI9341::ORANGE – 橘黄色 |
| BM_ILI9341::GREENYELLOW – 黄绿色 |
| BM_ILI9341::PINK – 粉红色 |

注 2：TFT 屏显示使用 x、y 轴坐标




A. 不旋转屏幕坐标　　B. 顺时针旋转 90 度坐标

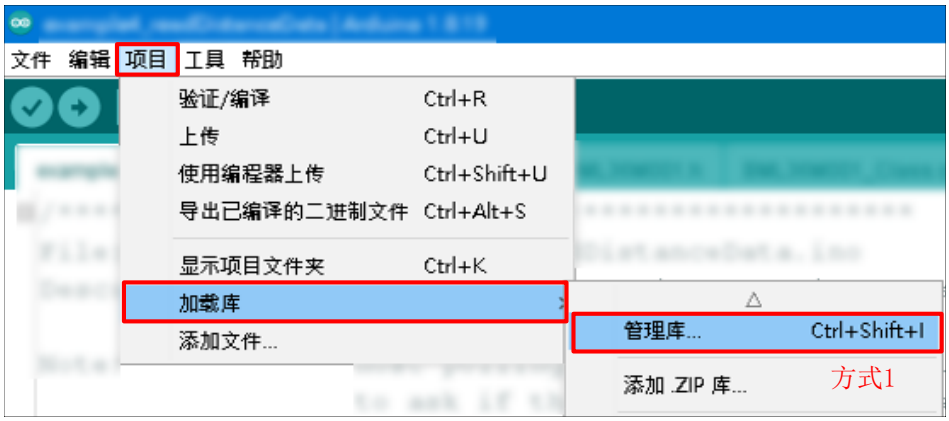
C. 顺时针旋转 180 度坐标　　D. 顺时针旋转 270 度坐标

# Arduino Lib 下载及安装

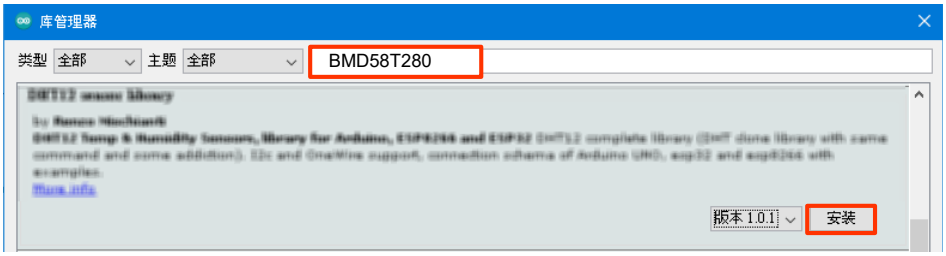
BMD58T280 Library：可参考下面两种方法安装 BMD58T280 的 Arduino Library。

## 方式 1：搜索安装

搜索安装：Arduino IDE → 项目 → 加载库 → 管理库 → 搜索 BMD58T280 → 安装




搜索安装流程 **1**

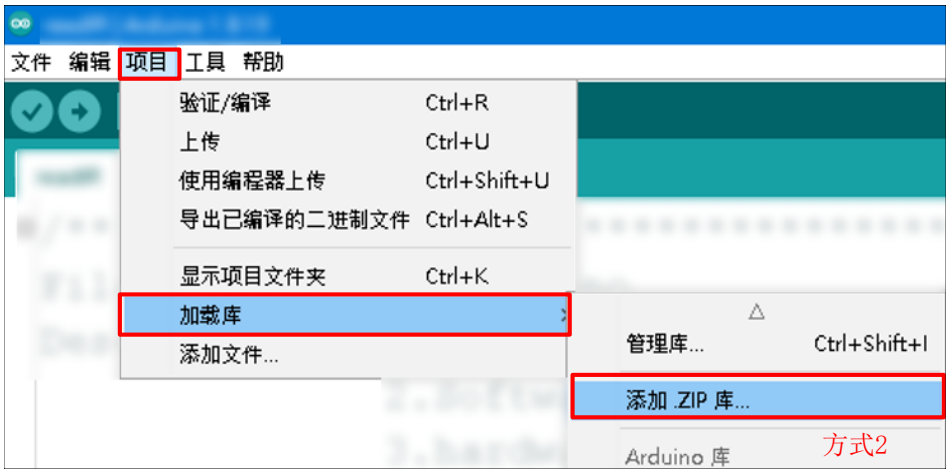



搜索安装流程 **2**

## 方式 2：添加 .ZIP 库，需提前下载 .ZIP 库

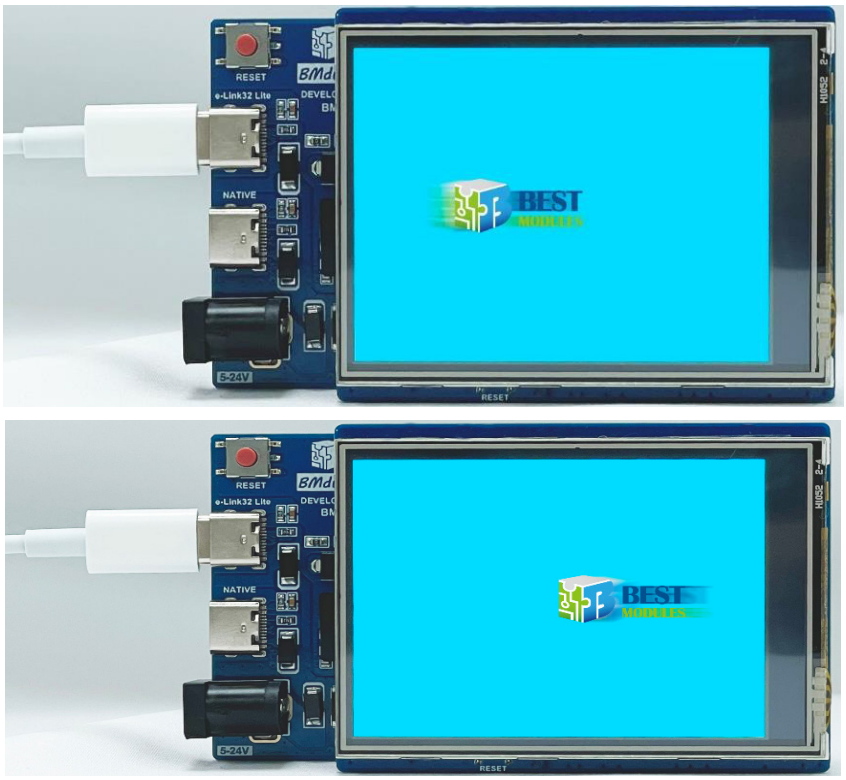
下载方法：打开倍创官网 (https://www.bestmodulescorp.com/bmd58t280.html) 文件目录下的 Arduino 范例程序 (BMD28T280 Library)。

添加 .ZIP 库：Arduino IDE → 项目 → 加载库 → 添加 .ZIP 库 ...



# Arduino 范例

## 范例 1：TFTBitmapScroll




实物连接示意图

范例实现功能：将 SD 卡内的 bmp 档显示在 LCD 上，并通过滚动特效让图片左右移动，展示 background 的功能、print 的功能、刷 BMP 图档、scrollTo 的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTBitmapScroll)

2. 环境设置：

   - 需准备 micro SD 卡并且文件系统格式需为 FAT32。
   - 通过 PC 将范例文件夹 (TFTBitmapScroll) 内的“Logo.bmp”复制到 SD 卡内。
   - 将 SD 插到 BMD58T280。

3. 示例说明：

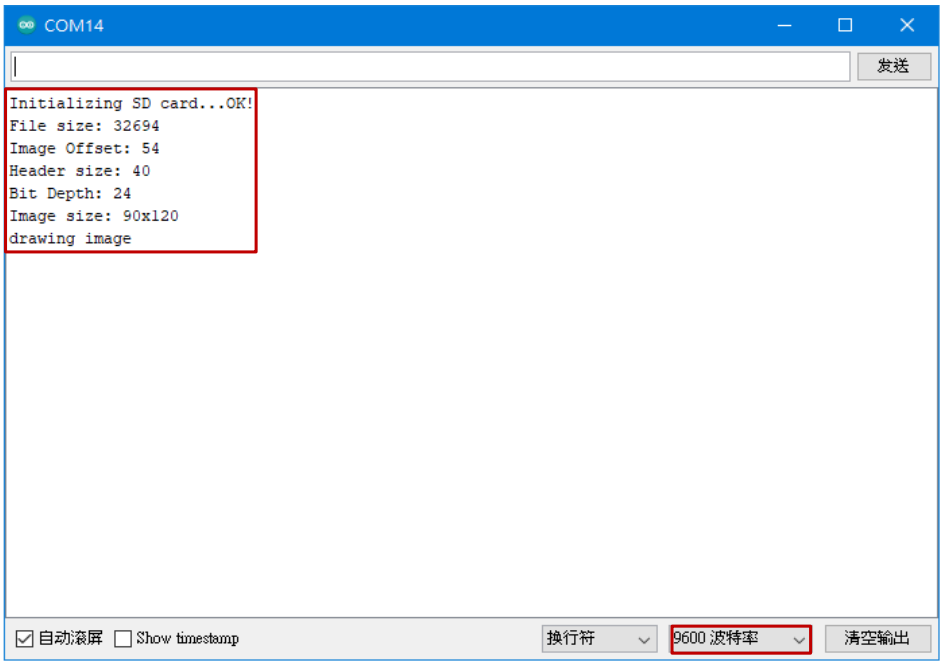
a. 构建 & 初始化对象

```
#include <SD.h>
#include <BMD58T280.h>
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)   // BMduino
BMD58T280 TFTscreen;            // 创建液晶显示对象
#else                           // Arduino
BMD58T280 TFTscreen(&SPI);      // 创建液晶显示对象
#endif
#define sd_cs  10               // SD 的引脚定义
BmpImage image1;                // 此变量表示要在屏幕上绘制的图像
void setup()
{
  // 初始化液晶屏并显示消息
  TFTscreen.begin();
  TFTscreen.background(255, 255, 255);
  TFTscreen.setTextSize (1);
  TFTscreen.stroke(0, 0, 255);
  TFTscreen.println();
  TFTscreen.println(F("BMduino TFT Bitmap Example And Scroll"));
  TFTscreen.stroke(0, 0, 0);
  TFTscreen.println(F("Open serial monitor"));
  TFTscreen.println(F("to run the sketch"));
  delay(3000);
  Serial.begin(9600);           // 初始化串口
  while(!Serial)
  {
    // 等待串口连接。仅本机 USB 端口需要
  }
  TFTscreen.background(0, 0xd8, 0xFF); // 启动前清除 GLCD 屏幕
  // 尝试访问 SD 卡。如果失败（例如不存在卡），设置过程将停止。
  Serial.print(F("Initializing SD card..."));
  if (!SD.begin(sd_cs))
  {
    Serial.println(F("failed!"));
    return;
  }
  Serial.println(F("OK!"));
  // SD 卡可以访问，尝试加载 image 文件
  image1.loadImage("Logo.bmp", &Serial);
  if (!image1)
  {
    Serial.println(F("error while loading Logo.bmp"));
  }
  Serial.println(F("drawing image"));
  TFTscreen.image(image1, (TFTWIDTH - image1.width())/2, 0); // 将图像
  // 绘制到屏幕上，图像的位置是 x 轴的中心
}
```

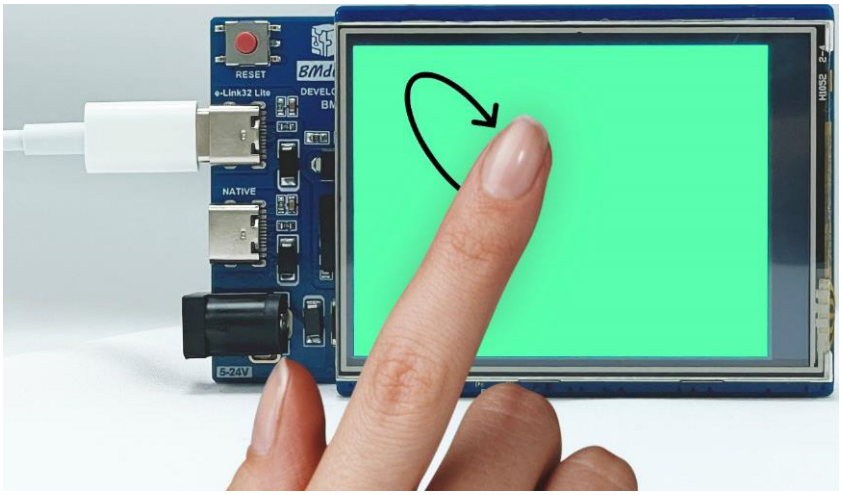
b. loop

```
void loop()
{
  // 图像从左向右移动
  for(int i= TFTscreen.height() ;i>image1.height();i--)
  {
    TFTscreen.scrollTo(i);
    delay(4);
  }
  // 图像从右向左移动
  for(int i=image1.height();i<TFTscreen.height();i++)
  {
    TFTscreen.scrollTo(i);
    delay(4);
  }
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 2：TFTColorPicker




实物连接示意图

范例实现功能：通过触摸的 x，y，z 值作为 LCD 屏的 RGB，展示触摸屏的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTColorPicker)

2. 互动方法：请任意滑动屏幕，可发现屏幕颜色一直变化。

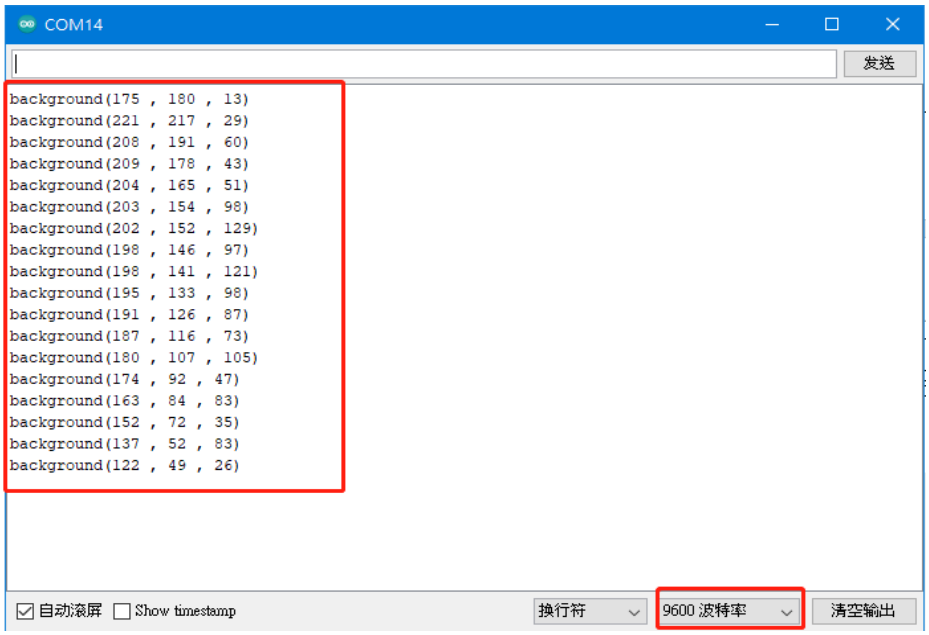
3. 示例说明：

   a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)   // For BMduino
BMD58T280 TFTscreen;            // 创建液晶显示对象
#else                           // For Arduino UNO
BMD58T280 TFTscreen(&SPI);      // 创建液晶显示对象
#endif
BM_XPT2046 touchScreen;         // 创建触摸屏对象
void setup()
{
  // 初始化串口
  Serial.begin(9600);
  // 设定初值
  TFTscreen.begin();
  touchScreen.begin();
  // 将背景设置为白色
  TFTscreen.background(255, 255, 255);
}
void loop()
{
  if (touchScreen.touched())
  {
    // 触摸屏幕，读取触摸屏的 (x, y, z)
    BM_XPT2046::TS_Point p = touchScreen.getPoint();
    p.x = map(p.x, 400, 3600, 0, 255);   if(p.x<0) p.x = 0;
    p.y = map(p.y, 400, 3600, 0, 255);   if(p.y<0) p.y = 0;
```

```
    p.z = map(p.z, 400, 2800, 0, 255);
    // 根据映射的值绘制背景
    TFTscreen.background(p.x, p.y, p.z);
    // 将值发送到串行监视器
    Serial.print("background(");
    Serial.print(p.x);
    Serial.print(" , ");
    Serial.print(p.y);
    Serial.print(" , ");
    Serial.print(p.z);
    Serial.println(")");
  }
  delay(20);
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 3：TFTDisplayText


实物连接示意图

范例实现功能：LCD 显示触摸屏的 x，y 坐标，展示触摸屏搭配 text 与 setTextSize 的功能。

1. 范例打开方式：

   Arduino IDE→ 文件 → 示例 → Lib 选择 → 选择范例 (TFTDisplayText)

2. 互动方法：请任意滑动屏幕，并观察坐标的变化。

3. 示例说明

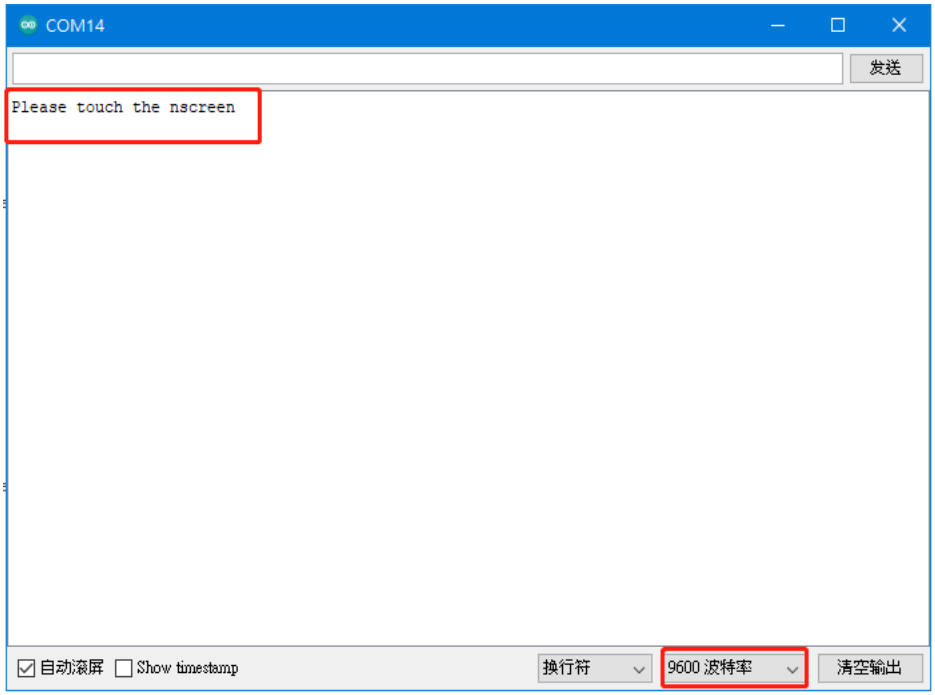
   a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)   // For BMduino
BMD58T280 TFTscreen;            // 创建液晶显示对象
#else                           // For Arduino UNO
BMD58T280 TFTscreen(&SPI);      // 创建液晶显示对象
#endif
BM_XPT2046 touchScreen;         // 创建触摸屏对象
char AxisPrintout[20];          // 要打印到屏幕的字符数组
void setup()
{
  TFTscreen.begin();
  touchScreen.begin();          // 触摸屏初始化
  Serial.begin(9600);
  TFTscreen.background(0, 0, 0);     // 清除屏幕
  // 将静态文本写入屏幕
  TFTscreen.stroke(255, 255, 255);  // 字体颜色设置为白色
  TFTscreen.setTextSize(2);    // 设置字体大小
  TFTscreen.text("Please touch the\nscreen.", 0, 0);  // 将文本写入屏幕
                                                      // 的左上角
  TFTscreen.setTextSize(5);    // 设置大字体用于 loop
  Serial.println("Please touch the nscreen");          // 串口输出
}
```

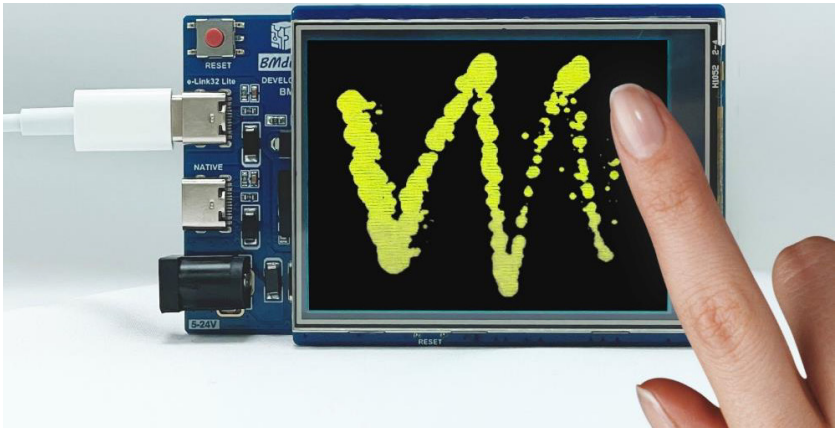
b. loop

```
void loop()
{
  if (touchScreen.touched())
  {
    // 触摸屏幕，读取触摸屏的 x 和 y
    BM_XPT2046::TS_Point p = touchScreen.getPoint();
    p.x = map(p.x, 400, 3600, 0, TFTscreen.width());   if(p.x<0) p.x
    = 0;
    p.y = map(p.y, 400, 3600, 0, TFTscreen.height());  if(p.y<0) p.y
    = 0;
    String strXaxis = String(p.x);
    String strYaxis = String(p.y);
    String strResult = strXaxis + "," + strYaxis + "    ";
    strResult.toCharArray(AxisPrintout, 20); // 将读数转换为字符数组
    TFTscreen.text(AxisPrintout, 0, 40 , BM_ILI9341::WHITE,
    BM_ILI9341::BLACK);          // 打印传感器值
    delay(250);                  // 等待一会
  }
  else
  {
    TFTscreen.text("                    ", 0, 40 , BM_ILI9341::WHITE,
    BM_ILI9341::BLACK); // 擦除文本
  }
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 4：TFTEtchASketch



实物连接示意图

范例实现功能：实现绘画板的功能，展示触摸屏搭配 fill 与 circle 的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTEtchASketch)

2. 互动方法：请在屏幕画图

3. 示例说明：

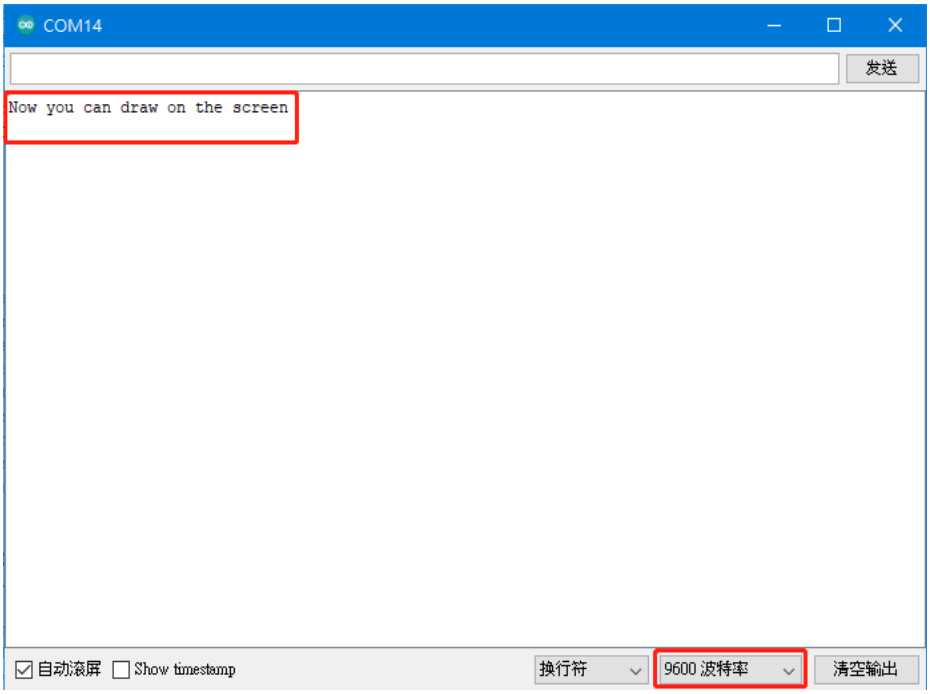
   a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)   // For BMduino
BMD58T280 TFTscreen;
#else                           // For UNO
BMD58T280 TFTscreen(&SPI);      // 创建液晶显示对象
#endif
BM_XPT2046 touchScreen;         // 创建液晶显示对象
void setup()
{
  // 初始化屏幕
  TFTscreen.begin();
  touchScreen.begin();
  Serial.begin(9600);
  TFTscreen.background(0, 0, 0);      // 背景黑色设置
  TFTscreen.fill(BM_ILI9341::YELLOW); // 设置黄色填充
  Serial.println(«Now you can draw on the screen»); // 串口输出
}
```

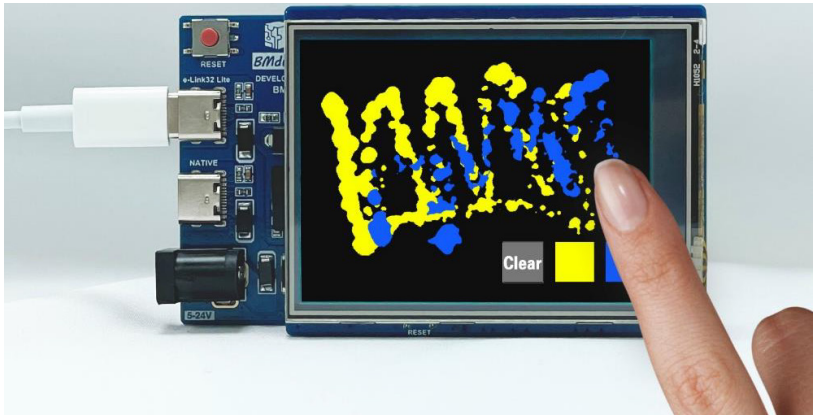
b. loop

```
void loop()
{
  if (touchScreen.touched())
  {
    BM_XPT2046::TS_Point p = touchScreen.getPoint();// 触摸屏幕，读取触
                                                    // 摸屏的(x, y, z)
    p.x = map(p.x, 400, 3600, 0, TFTscreen.width());
    p.y = map(p.y, 400, 3600, 0, TFTscreen.height());
    p.z = map(p.z, 400, 2800, 0, 20);
    TFTscreen.circle(p.x, p.y, p.z);  // 画一个圆
  }
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 5：TFTEtchASketchAddIcon




实物连接示意图

范例实现功能：于 TFTEtchASketch 的基础加上颜色选择与清除画面，展示触摸屏搭配 fill、circle、rect、drawPixels 的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTEtchASketchAddIcon)

2. 互动方法：请在屏幕画图

3. 示例说明：

   a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include "icon.h"                // “清除”图标的原始数据
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)    // For BMduino
BMD58T280 TFTscreen;             // 创建液晶显示对象
#else                            // For UNO
BMD58T280 TFTscreen(&SPI);       // 创建液晶显示对象
#endif
BM_XPT2046 touchScreen;          // 创建触摸屏对象
#define X_SATART          12     // 图标 x 轴起始位置
#define Y_SATART          195    // 图标 y 轴起始位置
#define ICON_W            35     // 图标宽度
#define ICON_H            36     // 图标长度
#define ICON_OFFSET       10     // 每个图标的距离
uint16_t color = (uint16_t)BM_ILI9341::YELLOW;
void setup()
{
  // 初始化屏幕
  TFTscreen.begin();
  touchScreen.begin();
  Serial.begin(9600);
  TFTscreen.background(0, 0, 0);      // 背景设置为黑色
  TFTscreen.fill(BM_ILI9341::YELLOW); // 设置黄色填充
  TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET + ICON_W +
                 ICON_OFFSET,ICON_W,ICON_H); // 绘制一个黄色图标
  TFTscreen.fill(BM_ILI9341::BLUE);           // 设置蓝色填充
  TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET + ICON_W +
                 ICON_OFFSET,ICON_W,ICON_H); // 绘制一个黄色图标
```

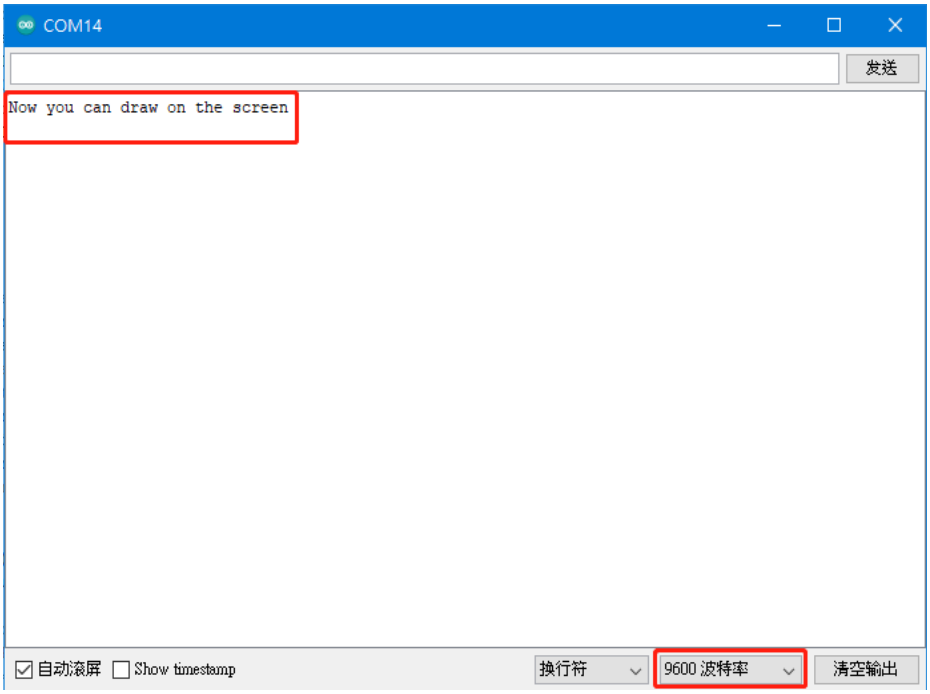
```
  TFTscreen.drawPixels(X_SATART,Y_SATART,ICON_W,ICON_H,
                       clearIcon);     // 绘制一个“清除”图标
  Serial.println("Now you can draw on the screen");  // 串口输出
}
```

b. loop

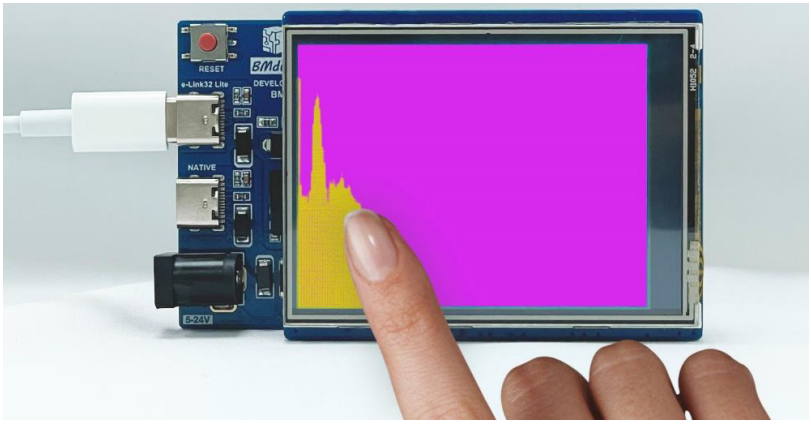
```
void loop()
{
  uint16_t isTouchIcon;
  if (touchScreen.touched())
  {
    BM_XPT2046::TS_Point p = touchScreen.getPoint();// 触摸屏幕。读取触
                                                    // 摸屏的(x, y, z)
    p.x = map(p.x, 400, 3600, 0, TFTWIDTH);
    p.y = map(p.y, 400, 3600, 0, TFTHEIGHT);
    p.z = map(p.z, 400, 2800, 0, 20);
    isTouchIcon = 0;
    if(p.x>X_SATART & p.x<(ICON_H + X_SATART))
    {
      if((p.y>Y_SATART &&
      p.y <(Y_SATART +ICON_W) ))
      {
        // 用户触摸“清除”图标。需要重新设置屏幕布局
        isTouchIcon = 1;
        TFTscreen.background(0, 0, 0);      // 背景设置为黑色
        TFTscreen.fill(BM_ILI9341::YELLOW); // 设置黄色填充
        TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET +
                       ICON_W + ICON_OFFSET,ICON_W,ICON_H);// 绘制一个
                                                           // 黄色图标
        TFTscreen.fill(BM_ILI9341::BLUE);   // 设置蓝色填充
        TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET +
                       ICON_W + ICON_OFFSET,ICON_W,ICON_H);// 绘制一个
                                                           // 黄色图标
        TFTscreen.drawPixels(X_SATART,Y_SATART,ICON_W,ICON_H,
                             clearIcon);    // 绘制一个“清除”图标
        TFTscreen.fill(color);              // 设置画笔颜色
      }
      else if((p.y>(Y_SATART + ICON_W + ICON_OFFSET) &&
      p.y<(Y_SATART + ICON_W + ICON_OFFSET + ICON_W) ))
      {
        // 用户触摸“黄色”图标。重新绘制黄色图标
        isTouchIcon = 1;
        TFTscreen.fill(BM_ILI9341::YELLOW); // 设置黄色填充
        TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET,
                       ICON_W,ICON_H);      // 绘制一个黄色图标
        color = (uint16_t)BM_ILI9341::YELLOW;
        TFTscreen.fill(color);              // 设置画笔颜色
      }
      else if((p.y>(Y_SATART + ICON_W + ICON_OFFSET + ICON_W +
               ICON_OFFSET) &&
      p.y<(Y_SATART + ICON_W + ICON_OFFSET + ICON_W + ICON_OFFSET
           +ICON_W) ))
      {
        // 用户触摸“蓝色”图标。重新绘制蓝色图标
        isTouchIcon = 1;
        TFTscreen.fill(BM_ILI9341::BLUE);   // 设置蓝色填充
```

```
        TFTscreen.rect(X_SATART,Y_SATART + ICON_W + ICON_OFFSET +
                       ICON_W + ICON_OFFSET,ICON_W,ICON_H);// 画一个蓝
                                                           // 色的图标
        color = (uint16_t)BM_ILI9341::BLUE;
        TFTscreen.fill(color);                  // 设置画笔颜色
      }
    }
    if(isTouchIcon == 0)
    {
      TFTscreen.circle(p.x, p.y, p.z);          // 没有触摸图标，开始绘制圆圈
    }
  }
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 6：TFTGraph



实物连接示意图

范例实现功能：x 轴的变化图，展示触摸屏搭配 stroke、line、setRotation 的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTGraph)

2. 互动方法：请在上下屏幕滑动，并观察屏幕曲线变化。

3. 示例目标：

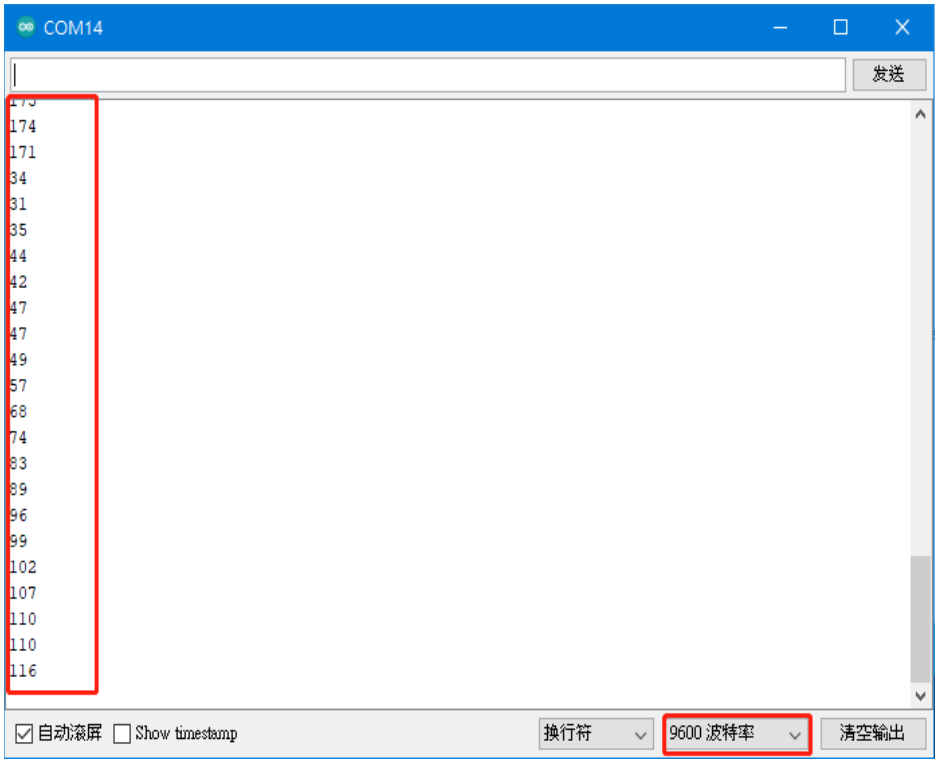
   a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include <SPI.h>
// 选择通信接口
#if !defined(ARDUINO_AVR_UNO)   // For BMduino
BMD58T280 TFTscreen;            // 创建液晶显示对象
#else                           // For UNO
BMD58T280 TFTscreen(&SPI);      // 创建液晶显示对象
#endif
BM_XPT2046 touchScreen;         // 创建触摸屏对象
int xPos = 0;                   // 线条位置
void setup()
{
  Serial.begin(9600);           // 串口初始化
  // 显示初始化
  TFTscreen.begin();
  touchScreen.begin();
  TFTscreen.setRotation(1);
  // 用漂亮的颜色清除屏幕
  TFTscreen.background(250, 16, 200);
}
```

b. loop

```
void loop()
{
  // 读取传数据并将其映射到屏幕高度
  if (touchScreen.touched())
  {
    // 触摸屏幕，读取触摸屏的 x 轴
    BM_XPT2046::TS_Point p = touchScreen.getPoint();
    int drawHeight = map(p.x, 400, 3600, 0, TFTscreen.height());
    Serial.println(drawHeight);  // 将高度打印到串行监视器
    // 画一条线
    TFTscreen.stroke(250, 180, 10);
    TFTscreen.line(xPos, TFTscreen.height() - drawHeight, xPos,
                   TFTscreen.height());
    // 如果图形已到达屏幕边缘擦除屏幕并重新开始
    if (xPos >= 360)
    {
      xPos = 0;
      TFTscreen.background(250, 16, 200);
    }
    else
    {
      xPos++;  // 增加水平位置
      delay(16);
    }
  }
}
```

4. 打开串口监视器，波特率选择 9600；串口监视器显示如下：



## 范例 7：TFTLogoVideo


实物连接示意图

范例实现功能：通过读取 SD 卡内的 Raw data，并通过底层的控制实现快速刷图的功能到达动画的效果，实现绘画板的功能。

1. 范例打开方式：

   Arduino IDE → 文件 → 示例 → Lib 选择 → 选择范例 (TFTLogoVideo)

2. 环境设置：

   - 需准备 micro SD 卡。
   - 下载 win32-disk-imager (https://win32diskimager.download/download-win32-disk-imager/)，下载与使用方法请直接参考上方连接。
   - 通过 win32-disk-imager 将范例文件夹 (TFTLogoVideo) 内“Video.img”写到 SD 卡内。
   - 将 SD 插到 BMD58T280。

3. 产生新的 Video.img ( 160×192 像素 ) 方法请直接参考档案 how_to_create_img_file.txt (TFTLogoVideo\Image_src\how_to_create_img_file.txt) 内的说明。

注：此范例仅支持 BMduino 的 EBI 模式。

4. 示例说明：

a. 构建 & 初始化对象

```
#include <BMD58T280.h>
#include <SD.h>
#include <SDReadRawData.h>
SDReadRawData sdcard;  // 创建 SD 对象
BMD58T280 TFTscreen;   // 创建 SD 对象
#define sd_cs  10       // SD 的引脚定义
#define IMAGE_HEIGHT       160
#define IMAGE_WIDTH        192
#if defined(ARDUINO_AVR_UNO)
#ERROR "This example don't support UNO. Please use the BMduino
        development board instead."
#endif
void setup()
{
  Serial.begin(9600);  // 串口初始化
  // 尝试访问 SD 卡。如果失败（例如，没有卡存在），设置过程将停止
  if (!sdcard.init(sd_cs))
  {
    Serial.println(F("failed!"));
    return;
  }
  Serial.println(F("succeeded!"));
  TFTscreen.begin();   // 初始化 LCD
}
```

b. loop

```
void loop()
{
  // 显示第一个视频（背景蓝色），启动前清除 GLCD 屏幕
  TFTscreen.background(0, 216, 255);     // 蓝色背景
  // 在 LCD 上显示 50 张图像
  for(int i = 0;i<50;i++)
  {
    // 显示大小 160×192 像素的图像，起始坐标 (40,69)
    isplayImage(40,69,IMAGE_HEIGHT,IMAGE_WIDTH,IMAGE_HEIGHT*IMAGE_
                WIDTH*2*i);
  }
  // 显示第二个视频（灰色背景）
  // 开始前清除 GLCD 屏幕
  TFTscreen.background(135, 144, 146);  // 灰色背景
  // 在 LCD 上显示 25 张图像
  for(int i =50;i<75;i++)
  {
    // 显示大小 160×192 像素的图像，起始坐标 (40,69)
    isplayImage(40,69,IMAGE_HEIGHT,IMAGE_WIDTH,IMAGE_HEIGHT*IMAGE_
                WIDTH*2*i);
  }
}
#define SD_BLOCK_BYTE_SIZE              512
#define SD_BLOCK_PIXEL_SIZE             (SD_BLOCK_BYTE_SIZE / 2)
#define SD_BLOCK_BYTE_SIZE_WITH_CRC     (SD_BLOCK_BYTE_SIZE + 2)
#define HALF_ADDRESS_INDEX              516
// 图像缓冲区
u8 dataBuffer[1032];
```

```
// 图像缓冲区开始地址
u8 *pStartDatabuffer = &(dataBuffer[0]);
// 图像缓冲区半地址
u8 *pHalfDatabuffer = &(dataBuffer[HALF_ADDRESS_INDEX]);
void displayImage(uint16_t x, uint16_t y, uint16_t w, uint16_t h,
                  uint32_t SDOffset)
{
  SDOffset = SDOffset/SD_BLOCK_BYTE_SIZE; // 获取 SD 的块索引
  TFTscreen.setAddrWindow(x, y, w, h);  // 设置图像的起始位置、宽度和长度
/* 为了加快速度，请先读取 SD 卡中的 514 字节。读取 SD 的块数据，512（数据）+
   2(CRC)=514 字节。将数据存储在缓冲区的开头 */
  sdcard.readBlock(SDOffset, SD_BLOCK_BYTE_SIZE_WITH_CRC,
                   pStartDatabuffer);
  sdcard.waitDmaFinish(); // 等待读取 514 字节完成。（缓冲区开始处的数据）
  for(int i=1;i<(w*h/SD_BLOCK_PIXEL_SIZE);i++)
  {
/* 读取 SD 的块数据，512(data)+2(CRC)=514 字节。将数据存储在缓冲区的一半 */
    sdcard.readBlock(SDOffset + i, SD_BLOCK_BYTE_SIZE_WITH_CRC,
                     pHalfDatabuffer);
    // 在 LCD 上显示 512 字节的数据，并跳过 2 字节的 CRC。（缓冲区开头的数据）
    for(int j=0;j<SD_BLOCK_BYTE_SIZE;j+=2)
    {
      // 将原始数据转换为颜色
      u16 _color = (u16)((dataBuffer[j+1]<<8) | dataBuffer[j]);
      // 通过底层 EBI 来加快数据的写入速度
      TFTscreen.lowLevel->writeData16(_color);
    }
    sdcard.waitDmaFinish();  // 等待读取 514 字节完成。（缓冲区一半的数据）
    i++;
    // 确认图像数据是否从 SD 中完全读取
    if(i < (w*h/SD_BLOCK_PIXEL_SIZE))
    {
/* 读取 SD 块的数据，512（数据）+2(CRC)=514 字节。将数据存储在缓冲区的开头。（缓
冲区开头的数据）*/
      sdcard.readBlock(SDOffset + i, SD_BLOCK_BYTE_SIZE_WITH_CRC,
                       pStartDatabuffer);
    }
/*LCD 显示 512 字节的数据，跳过 2 字节的 CRC（缓冲区一半的数据）*/
    for(int j=0;j<SD_BLOCK_BYTE_SIZE;j+=2)
    {
      u16_color = (u16)((dataBuffer[j+HALF_ADDRESS_INDEX+1]<<8) |
                        dataBuffer[j+HALF_ADDRESS_INDEX]);
      TFTscreen.lowLevel->writeData16(_color);
    }
    sdcard.waitDmaFinish();  // 等待读取 514 字节完成
  }
}
```

5. 打开串口监视器，波特率选择 9600；串口监视器显示如下：