|  |  |  |
| --- | --- | --- |
| 函数名 | 参数 | 用法 |
| TFT\_eSPI(int16\_t \_W = TFT\_WIDTH, int16\_t \_H = TFT\_HEIGHT) | \_W：显示屏宽度  \_H：显示屏高度 | 创建TFT\_eSPI对象 |
| init(uint8\_t tc = TAB\_COLOUR), begin(uint8\_t tc = TAB\_COLOUR) | tc：标签颜色 | 初始化TFT显示屏 |
| drawPixel(int32\_t x, int32\_t y, uint32\_t color) | x：像素点的x坐标  y：像素点的y坐标  color：像素点的颜色 | 绘制单个像素点 |
| drawChar(int32\_t x, int32\_t y, uint16\_t c, uint32\_t color, uint32\_t bg, uint8\_t size) | x：字符左上角的x坐标  y：字符左上角的y坐标  c：要绘制的字符  color：字符颜色  bg：背景颜色  size：字符大小 | 绘制字符 |
| drawLine(int32\_t xs, int32\_t ys, int32\_t xe, int32\_t ye, uint32\_t color) | xs：起始点的x坐标  ys：起始点的y坐标  xe：结束点的x坐标  ye：结束点的y坐标  color：线的颜色 | 绘制直线 |
| drawFastVLine(int32\_t x, int32\_t y, int32\_t h, uint32\_t color) | x：垂直线的起始点x坐标  y：垂直线的起始点y坐标  h：垂直线的高度  color：线的颜色 | 绘制垂直线段 |
| drawFastHLine(int32\_t x, int32\_t y, int32\_t w, uint32\_t color) | x：水平线的起始点x坐标  y：水平线的起始点y坐标  w：水平线的宽度  color：线的颜色 | 绘制水平线段 |
| fillRect(int32\_t x, int32\_t y, int32\_t w, int32\_t h, uint32\_t color) | x：矩形左上角的x坐标  y：矩形左上角的y坐标  w：矩形的宽度  h：矩形的高度  color：填充颜色 | 填充矩形 |
| drawChar(uint16\_t uniCode, int32\_t x, int32\_t y, uint8\_t font) | uniCode：要绘制的字符编码  x：字符左上角的x坐标  y：字符左上角的y坐标  font：字体 | 绘制字符 |
| drawChar(uint16\_t uniCode, int32\_t x, int32\_t y) | uniCode：要绘制的字符编码  x：字符左上角的x坐标  y：字符左上角的y坐标 | 绘制字符 |
| height(void) | 无 | 返回显示屏高度 |
| width(void) | 无 | 返回显示屏宽度 |
| readPixel(int32\_t x, int32\_t y) | x：像素点的x坐标  y：像素点的y坐标 | 读取指定位置像素点的颜色 |
| setWindow(int32\_t xs, int32\_t ys, int32\_t xe, int32\_t ye) | xs：窗口的起始点x坐标  ys：窗口的起始点y坐标  xe：窗口的结束点x坐标  ye：窗口的结束点y坐标 | 设置绘图窗口的范围 |
| pushColor(uint16\_t color) | color：要写入TFT的颜色 | 将颜色值写入TFT |
| begin\_nin\_write() | 无 | 启动非内联写入模式 |
| end\_nin\_write() | 无 | 结束非内联写入模式 |
| setRotation(uint8\_t r) | r：图像方向（0、1、2或3） | 设置显示图像的方向 |
| getRotation(void) | 无 | 读取当前的显示图像方向 |
| setOrigin(int32\_t x, int32\_t y) | x：新的原点x坐标  y：新的原点y坐标 | 更改原点位置 |
| getOriginX(void) | 无 | 获取当前原点的x坐标 |
| getOriginY(void) | 无 | 获取当前原点的y坐标 |
| invertDisplay(bool i) | i：是否反转显示颜色 | 反转显示颜色 |
| setAddrWindow(int32\_t xs, int32\_t ys, int32\_t w, int32\_t h) | xs：窗口的起始点x坐标  ys：窗口的起始点y坐标  w：窗口的宽度  h：窗口的高度 | 设置地址窗口范围 |
| setViewport(int32\_t x, int32\_t y, int32\_t w, int32\_t h, bool vpDatum = true) | x：视口的起始点x坐标  y：视口的起始点y坐标  w：视口的宽度  h：视口的高度  vpDatum：视口参考点（默认为true） | 设置视口范围 |
| checkViewport(int32\_t x, int32\_t y, int32\_t w, int32\_t h) | x：视口的起始点x坐标  y：视口的起始点y坐标  w：视口的宽度  h：视口的高度 | 检查视口范围是否有效 |
| getViewportX(void) | 无 | 获取当前视口的起始点x坐标 |
| getViewportY(void) | 无 | 获取当前视口的起始点y坐标 |
| getViewportWidth(void) | 无 | 获取当前视口的宽度 |
| getViewportHeight(void) | 无 | 获取当前视口的高度 |
| getViewportDatum(void) | 无 | 获取当前视口的参考点 |
| frameViewport(uint16\_t color, int32\_t w) | color：边框颜色  w：边框宽度 | 在视口周围绘制边框 |
| resetViewport(void) | 无 | 重置视口到整个屏幕的范围 |
| clipAddrWindow(int32\_t\* x, int32\_t\* y, int32\_t\* w, int32\_t\* h) | x：窗口的起始点x坐标（输入/输出参数）  y：窗口的起始点y坐标（输入/输出参数）  w：窗口的宽度（输入/输出参数）  h：窗口的高度（输入/输出参数） | 将输入的窗口裁剪到视口范围内 |
| clipWindow(int32\_t\* xs, int32\_t\* ys, int32\_t\* xe, int32\_t\* ye) | xs：窗口的起始点x坐标（输入/输出参数）  ys：窗口的起始点y坐标（输入/输出参数）  xe：窗口的结束点x坐标（输入/输出参数）  ye：窗口的结束点y坐标（输入/输出参数） | 将输入的窗口区域裁剪到视口范围内 |
| pushColor(uint16\_t color, uint32\_t len) | color：要写入TFT的颜色  len：写入像素的数量 | 将指定数量的颜色值写入TFT |
| pushColors(uint16\_t \*data, uint32\_t len, bool swap = true) | data：颜色数据数组的指针  len：颜色数据数组的长度  swap：是否交换字节顺序（默认为true） | 将颜色数据数组写入TFT |
| pushColors(uint8\_t \*data, uint32\_t len) | data：颜色数据数组的指针  len：颜色数据数组的长度 | 将颜色数据数组写入TFT |
| pushColors(const uint8\_t \*data, uint32\_t len, uint8\_t bpp) | data：颜色数据数组的指针  len：颜色数据数组的长度  bpp：每个像素的位深度 | 将指定位深度的颜色数据数组写入TFT |
| pushColors(const uint16\_t \*data, uint32\_t len, bool swap = true) | data：颜色数据数组的指针  len：颜色数据数组的长度  swap：是否交换字节顺序（默认为true） | 将颜色数据数组写入TFT |
| pushImage(int32\_t x, int32\_t y, int32\_t w, int32\_t h, const uint16\_t \*data) | x：图像的起始点x坐标  y：图像的起始点y坐标  w：图像的宽度  h：图像的高度  data：图像数据数组的指针 | 将图像数据数组绘制到TFT上 |
| drawPixel(int32\_t x, int32\_t y, uint32\_t color, uint8\_t alpha, uint32\_t bg\_color = 0x00FFFFFF) | x：像素的x坐标  y：像素的y坐标  color：绘制的像素颜色  alpha：绘制像素的透明度  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一个与背景像素颜色混合的像素，返回混合的颜色值 |
| drawSmoothArc(int32\_t x, int32\_t y, int32\_t r, int32\_t ir, uint32\_t startAngle, uint32\_t endAngle, uint32\_t fg\_color, uint32\_t bg\_color, bool roundEnds = false) | x：圆心的x坐标  y：圆心的y坐标  r：外圆半径  ir：内圆半径  startAngle：起始角度  endAngle：结束角度  fg\_color：前景颜色  bg\_color：背景颜色  roundEnds：是否使用圆形结束（可选，默认为false） | 绘制一个平滑的弧线，支持设置起始角度和结束角度，可以选择圆形结束 |
| drawArc(int32\_t x, int32\_t y, int32\_t r, int32\_t ir, uint32\_t startAngle, uint32\_t endAngle, uint32\_t fg\_color, uint32\_t bg\_color, bool smoothArc = true) | x：圆心的x坐标  y：圆心的y坐标  r：外圆半径  ir：内圆半径  startAngle：起始角度  endAngle：结束角度  fg\_color：前景颜色  bg\_color：背景颜色  smoothArc：是否平滑绘制弧线（可选，默认为true） | 绘制一个弧线，支持设置起始角度和结束角度，可以选择平滑绘制 |
| drawSmoothCircle(int32\_t x, int32\_t y, int32\_t r, uint32\_t fg\_color, uint32\_t bg\_color) | x：圆心的x坐标  y：圆心的y坐标  r：半径  fg\_color：前景颜色  bg\_color：背景颜色 | 绘制一个平滑的填充圆形 |
| fillSmoothCircle(int32\_t x, int32\_t y, int32\_t r, uint32\_t color, uint32\_t bg\_color = 0x00FFFFFF) | x：圆心的x坐标  y：圆心的y坐标  r：半径  color：填充颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一个填充圆形，支持设置填充颜色和背景颜色 |
| drawSmoothRoundRect(int32\_t x, int32\_t y, int32\_t r, int32\_t ir, int32\_t w, int32\_t h, uint32\_t fg\_color, uint32\_t bg\_color = 0x00FFFFFF, uint8\_t quadrants = 0xF) | x：矩形左上角的x坐标  y：矩形左上角的y坐标  r：外圆角半径  ir：内圆角半径  w：矩形的宽度  h：矩形的高度  fg\_color：前景颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF）  quadrants：控制圆角绘制的象限（可选，默认为0xF） | 绘制一个具有圆角的矩形，支持设置外圆角半径、内圆角半径、前景颜色和背景颜色 |
| fillSmoothRoundRect(int32\_t x, int32\_t y, int32\_t w, int32\_t h, int32\_t radius, uint32\_t color, uint32\_t bg\_color = 0x00FFFFFF) | x：矩形左上角的x坐标  y：矩形左上角的y坐标  w：矩形的宽度  h：矩形的高度  radius：圆角半径  color：填充颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一个填充圆角矩形，支持设置圆角半径、填充颜色和背景颜色 |
| drawSpot(float ax, float ay, float r, uint32\_t fg\_color, uint32\_t bg\_color = 0x00FFFFFF) | ax：圆心的x坐标  ay：圆心的y坐标  r：半径  fg\_color：前景颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一个小的填充圆形，支持设置半径、前景颜色和背景颜色 |
| drawWideLine(float ax, float ay, float bx, float by, float wd, uint32\_t fg\_color, uint32\_t bg\_color = 0x00FFFFFF) | ax：起点的x坐标  ay：起点的y坐标  bx：终点的x坐标  by：终点的y坐标  wd：线宽  fg\_color：前景颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一条宽线段，支持设置起点、终点、线宽、前景颜色和背景颜色 |
| fillTriangle(int32\_t x0, int32\_t y0, int32\_t x1, int32\_t y1, int32\_t x2, int32\_t y2, uint32\_t color, uint32\_t bg\_color = 0x00FFFFFF) | x0：第一个顶点的x坐标  y0：第一个顶点的y坐标  x1：第二个顶点的x坐标  y1：第二个顶点的y坐标  x2：第三个顶点的x坐标  y2：第三个顶点的y坐标  color：填充颜色  bg\_color：背景颜色（可选，默认为0x00FFFFFF） | 绘制一个填充三角形，支持设置三个顶点坐标、填充颜色和背景颜色 |
|  |  |  |
| readRectRGB(int32\_t x, int32\_t y, int32\_t w, int32\_t h, uint8\_t \*data) | x: 矩形区域的左上角 x 坐标  y: 矩形区域的左上角 y 坐标  w: 矩形区域的宽度  h: 矩形区域的高度  data: 像素颜色值缓冲区指针 | 读取屏幕上指定区域的像素颜色值，将每个像素的RGB颜色值存储在缓冲区中 |
| drawNumber() |  | 根据指定的字体，将整数以文本形式绘制在屏幕上 |
| drawFloat() |  | 根据指定的字体，将浮点数以文本形式绘制在屏幕上 |
| drawString() |  | 根据指定的字体，将字符串以文本形式绘制在屏幕上 |
| drawCentreString() |  | 已弃用的函数，使用setTextDatum()和drawString()代替 |
| drawRightString() |  |  |
| setCursor() |  | 设置tft.print()函数的光标位置 |
| getTextCursor() |  | 获取当前光标位置 |
| setTextColor() |  | 设置字符（字形）颜色 |
| setTextSize() |  | 设置字符的大小倍数 |
| setTextWrap() |  | 设置文本是否自动换行 |
| setTextDatum() |  | 设置文本对齐方式 |
| getTextDatum() |  | 获取当前文本对齐方式 |
| setTextPadding() |  | 设置文本填充宽度，用于清除之前显示的值 |
| getTextPadding() |  | 获取文本填充宽度 |
| setFreeFont() |  | 设置要使用的字体 |
| setTextFont() |  |  |
| textWidth() |  | 计算指定文本在指定字体下的像素宽度 |
| fontHeight() |  | 获取指定字体的像素高度 |
| decodeUTF8() |  | 解码UTF-8编码的字符串，将Unicode码点转换为UTF-8编码 |
| write() |  | 通过打印流传递的字符数组写入函数 |
| setCallback() |  | 设置获取抗锯齿字体像素颜色的回调函数 |
| fontsLoaded() |  | 返回已加载的字体类型的位表示 |