電子工作

M5StickCのDisplay周り解析

Posted on 2019年6月17日 by たなかまさゆき / 0件のコメント

※現時点の情報ですので、最新情報はM5StickC非公式日本語リファレンスを確認してください。

サンプルスケッチにあるものだけ抜き出しています。すべての関数はAPIリファレンスなどで確かめてください。

https://lang-ship.com/reference/M5StickC/latest/class_m5_display.html

見出

代表的な関数列挙

カテゴリ	関数名	日本語
画面	M5.Lcd.height()	ディスプレイの縦幅を返す(回転に追従)
画面	M5.Lcd.width()	ディスプレイの横幅を返す(回転に追従)
画面	M5.Lcd.setRotation()	ディスプレイの向きを設定(0:M5が下, 1:電源ボタンが下, 2:上下反転, 3:RSTボタンが下)

画面	M5.Lcd.setCursor()	カーソルの座標とフォントを指定
色	M5.Lcd.color565()	RGBデータを16ビットカラーに変換
テキスト	M5.Lcd.drawString()	座標を指定して文字列を描画
テキスト	M5.Lcd.drawCentreString()	座標を指定して文字列を中央寄せで描画
テキスト	M5.Lcd.drawRightString()	座標を指定して文字列を右寄せで描画
テキスト	M5.Lcd.print()	カーソル位置に文字列を描画
テキスト	M5.Lcd.println()	カーソル位置に文字列を改行付きで描画
テキスト	M5.Lcd.printf()	カーソル位置に文字列をprintf書式で描画
描画	M5.Lcd.fillScreen()	画面全体を塗りつぶす
描画	M5.Lcd.drawPixel()	点を描画
描画	M5.Lcd.drawLine()	線を描画
描画	M5.Lcd.drawCircle()	塗りつぶしの無い円を描画
描画	M5.Lcd.drawRect()	塗りつぶしの無い四角を描画
描画	M5.Lcd.fillCircle()	塗りつぶしの円を描画
描画	M5.Lcd.fillRect()	塗りつぶしの四角を描画
描画	M5.Lcd.fillTriangle()	塗りつぶしの三角形を描画
描画	M5.Lcd.drawBitmap()	Bitmap(BMP)形式の画像を描画
描画	M5.Lcd.drawXBitmap()	X Bitmap(XBM)形式の画像を描画
フォント	M5.Lcd.highlight()	テキストハイライトの設定(true:ハイライト色, false:フォント背景色 or 塗りつぶし無し)
フォント	M5.Lcd.setHighlightColor()	テキストハイライト色の設定
フォント	M5.Lcd.setTextColor()	フォント色の指定
フォント	M5.Lcd.setTextDatum()	テキスト描画開始座標指定(0:TopLeft, 1:TopCenter, 2:TopRight, 3...8)
フォント	M5.Lcd.setTextFont()	テキストフォントの設定(1-8)
フォント	M5.Lcd.setTextSize()	テキストサイズの設定(1-7)
テキスト	M5.Lcd.drawChar()	Adafruit GLCDフォントで1文字描画
テキスト	M5.Lcd.drawNumber()	7桁までのlong intの数値を描画
テキスト	M5.Lcd.drawFloat()	7桁までのfloatの数値を描画
システム	M5.Lcd.write()	テキストを実際に書き込む
システム	M5.Lcd.writecommand()	ディスプレイにコマンド送信
システム	M5.Lcd.writedata()	ディスプレイにデータ送信

サンプルスケッチで使われている関数を拾い出してきました。ただし中国語フォント関係(HZK16)は抜いてあります。

画面サイズ

- int16_t M5.Lcd.height()
- int16_t M5.Lcd.width()

画面サイズを返却するだけの関数です。

画面の向き

- M5.Lcd.setRotation(r)

M5Stackの画面方向と同じようです。M5StickCしか触ったことないので、実験しないとわかりませんでした。

画面カーソル位置

- M5.Lcd.setCursor(x, y)
- M5.Lcd.setCursor(x, y, font)

カーソルの場所とフォントを設定します。

色

- uint16_t c = M5.Lcd.color565(r, g, b)

RGBから2バイトの色に変換します。

座標指定テキスト描画

- M5.Lcd.drawString(string, x, y)
- M5.Lcd.drawString(string, x, y, font)
- M5.Lcd.drawCentreString(...)
- M5.Lcd.drawRightString(...)

標準的なテキスト描画ですが、次のカーソルを利用した方がサンプルでは多用していました。

カーソル利用テキスト描画

- M5.Lcd.print(string)
- M5.Lcd.println(string)
- M5.Lcd.printf("%d", i)

printとprintlnは改行無し、有りの出力形式で、printfは書式指定の出力です。両方ESP32のPrintクラスで定義されている関数で、Serial.println()などと同じように使うことができます。

内部的には文字列操作してから M5.Lcd.write()関数で描画しているのですが、drawString()系とは別関数で描画しています。

画面全体塗りつぶし

- M5.Lcd.fillScreen(color)

画面全体を単色で塗りつぶします。色の定義は以下でされていました。

```
#define TFT_BLACK       0x0000      /*   0,   0,   0 */
#define TFT_NAVY        0x000F      /*   0,   0, 128 */
#define TFT_DARKGREEN   0x03E0      /*   0, 128,   0 */
#define TFT_DARKCYAN    0x03EF      /*   0, 128, 128 */
#define TFT_MAROON      0x7800      /* 128,   0,   0 */
#define TFT_PURPLE      0x780F      /* 128,   0, 128 */
#define TFT_OLIVE       0x7BE0      /* 128, 128,   0 */
#define TFT_LIGHTGREY   0xC618      /* 192, 192, 192 */
#define TFT_DARKGREY    0x7BEF      /* 128, 128, 128 */
#define TFT_BLUE        0x001F      /*   0,   0, 255 */
#define TFT_GREEN       0x07E0      /*   0, 255,   0 */
#define TFT_CYAN        0x07FF      /*   0, 255, 255 */
#define TFT_RED         0xF800      /* 255,   0,   0 */
#define TFT_MAGENTA     0xF81F      /* 255,   0, 255 */
#define TFT_YELLOW      0xFFE0      /* 255, 255,   0 */
#define TFT_WHITE       0xFFFF      /* 255, 255, 255 */
#define TFT_ORANGE      0xFDA0      /* 255, 180,   0 */
#define TFT_GREENYELLOW 0xB7E0      /* 180, 255,   0 */
#define TFT_PINK        0xFC9F

// Next is a special 16 bit colour value that encodes to 8 bits
// and will then decode back to the same 16 bit value.
// Convenient for 8 bit and 16 bit transparent sprites.
#define TFT_TRANSPARENT 0x0120
```

描画

- M5.Lcd.drawPixel(x, y, color)
- M5.Lcd.drawLine(x0, y0, x1, y1, color)
- M5.Lcd.drawCircle(x0, y0, r, color)
- M5.Lcd.drawRect(x, y, w, h, color)
- M5.Lcd.fillCircle(x0, y0, r, color)
- M5.Lcd.fillRect(x0, y0, x1, y1, color)
- M5.Lcd.fillTriangle(x0, y0, x1, y1, x2, y2, color)

一般的な描画関数もあります。この他にもdrawEllipse()とかfillCircle()とかありましたが、サンプルスケッチに無かったのでここでは取り上げません。

https://lang-ship.com/reference/M5StickC/latest/class_t_f_t__e_s_p_i.html

上記APIリファレンスとかで詳しくはみてください。

画像描画

- M5.Lcd.drawBitmap(x, y, w, h, data)
- M5.Lcd.drawXBitmap(x, y, bitmap, w, h, color)

drawBitmap()は2バイト単位の色データが並んでいる画像形式で、drawXBitmap()は1ビット1ピクセルの画像で、color色で塗りつぶします。

ハイライト

- M5.Lcd.highlight(bool)
- M5.Lcd.setHighlightColor(color)

ハイライトをtrueに設定すると、指定している背景色じゃなくて、ハイライト色で塗りつぶします。背景色を都度指定すればいい気がするのですが、よくわかりません。

フォント色

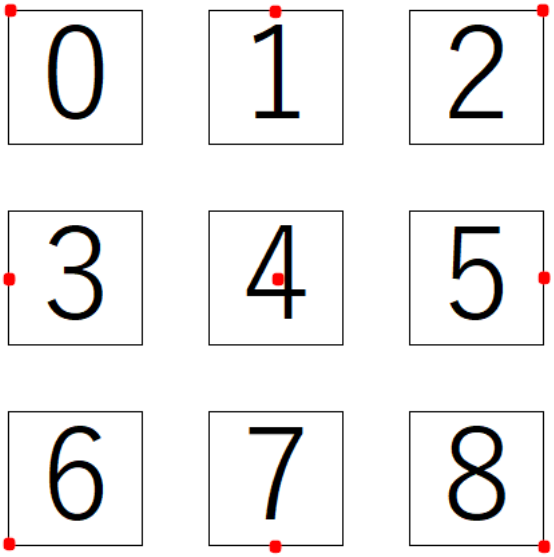
- M5.Lcd.setTextColor(color)
- M5.Lcd.setTextColor(color, bgcolor)

色の定義は以下です。TFTのと一緒な気がします。。。

```
#define BLACK             0x0000      /*   0,   0,   0 */
#define NAVY              0x000F      /*   0,   0, 128 */
#define DARKGREEN         0x03E0      /*   0, 128,   0 */
#define DARKCYAN          0x03EF      /*   0, 128, 128 */
#define MAROON            0x7800      /* 128,   0,   0 */
#define PURPLE            0x780F      /* 128,   0, 128 */
#define OLIVE             0x7BE0      /* 128, 128,   0 */
#define LIGHTGREY         0xC618      /* 192, 192, 192 */
#define DARKGREY          0x7BEF      /* 128, 128, 128 */
#define BLUE              0x001F      /*   0,   0, 255 */
#define GREEN             0x07E0      /*   0, 255,   0 */
#define CYAN              0x07FF      /*   0, 255, 255 */
#define RED               0xF800      /* 255,   0,   0 */
#define MAGENTA           0xF81F      /* 255,   0, 255 */
#define YELLOW            0xFFE0      /* 255, 255,   0 */
#define WHITE             0xFFFF      /* 255, 255, 255 */
#define ORANGE            0xFD20      /* 255, 165,   0 */
#define GREENYELLOW       0xAFE5      /* 173, 255,  47 */
#define PINK              0xF81F
```

テキスト座標

- M5.Lcd.setTextDatum(datum)



テキストの原点をどこに設定するか指定します。Advanced/Display/TFT_String_Alignのサンプルスケッチを参考にしてください。

フォント指定

- M5.Lcd.setTextFont(font)

1	Adafruit 8ピクセルASCIIフォント
2	16ピクセルASCIIフォント
3	未設定
4	26ピクセルASCIIフォント
5	未設定
6	26ピクセル数字フォント
7	48ピクセル7セグ風フォント
8	75ピクセル数字フォント

フォントの種類ごとに使える文字と、大きさが違うので注意してください。

フォントサイズ

- M5.Lcd.setTextSize(size)

1-7まで指定できて、元の大きさを何倍にするかを指定します。

その他

- M5.Lcd.drawChar()
- M5.Lcd.drawNumber()
- M5.Lcd.drawFloat()
- M5.Lcd.write()
- M5.Lcd.writecommand()
- M5.Lcd.writedata()

いろいろありますが、内部的に利用していて直接呼び出すことはあまりないと思います。

まとめ

もうちょっとフォント周りと画像周りは検証したいですが、よく使う関数は列挙できたと思います。あとでもう少し見やすい形にまとめたいと思います。

続編

M5StickCのDisplay周り解析 その2

※現時点の情報ですので、最新情報はM5StickC非公式日本語リファレンスを確認してください。 概要 M5StackCのライブラリが更新されたので、描画周りの動きを再度説明したいと思います。基本的な動作は前回の記事と変わ … 続きを読む

Lang-ship 0

たなかまさゆき

たなかまさゆき の投稿をすべて表示