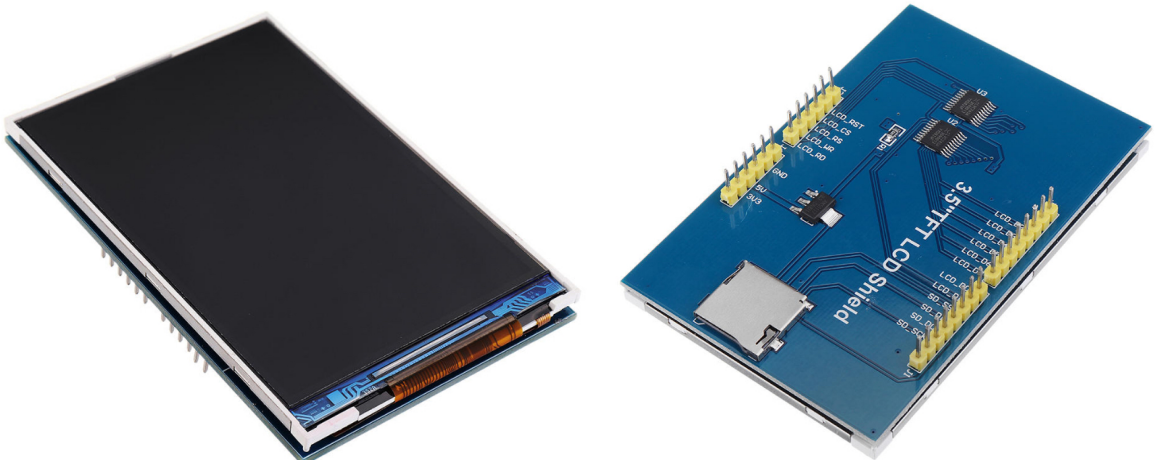
Using a 3.5" TFT colour display

This display is designed by MCUFRIEND.com and is a plug-in shield for the Arduino UNO or MEGA. If it is used on an UNO only 2 digital and 1 analog input are spare. Thus it's better used on the MEGA to provide many more digital and analog lines. A model is available from banggood.com.



There are a number of controllers and libraries for TFT displays. This one is supported by the Adafruit_GFX general graphics library which defines the primitive display functions and colour definitions. Plus a second library MCUFRIEND_kbv provides the interfacing to this specific display. Both are used together in sketches.

HEADER OUTLINE

To make using the display easier some higher level functions have been written in my own header file. The header file TFT.h provides the basics for using the GFX & MCUFRIEND libraries. It provides five fonts (9, 12, 18 & 24pt sans serif and one default system font) and 9 pre-defined colors. It provides ready-written functions for screen orientation and background colour, for drawing frames, lines and displaying messages and dedicated formats.

If a function name looks like this "`xxxF()`" then it uses a defined font, if not it uses the rather poor default system font. The system font can be "magnified" by 0, 1, 2, 3, 4 or 5 but anything larger than 2 gives a very chunky bit mapped display.

FUNCTIONS FORMAT

Throughout the general form of the functions in `TFT.h` is,

```
void name(position x y, COLOR, [&font,] parameters);
```

`x` is across and `y` is down from the top left, in whatever screen rotation is chosen. `COLOR` is a text name, `font` is a pointer to one of the many available fonts and the parameters are whatever the function needs. For example, to display a 24pt text message based on using one of the Sans Serif fonts,

```
void dispMsgF(int x, int y, uint16_t  c, const GFXfont *f,  char *m)
```

README.txt reference

This file summarises the functions of `TFT.h` for reference.

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <MCUFRIEND_kbv.h>   // HW specific library

#include <Fonts/FreeSans9pt7b.h> // fonts
#include <Fonts/FreeSans12pt7b.h>
#include <Fonts/FreeSans18pt7b.h>
#include <Fonts/FreeSans24pt7b.h>
#include <FreeDefaultFonts.h>

#define BLACK   0x0000       // basic colors
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF
#define GRAY    0x8410

#define TFT_BLACK       0x0000      /*   0,   0,   0 */ // alternative colors
#define TFT_NAVY        0x000F      /*   0,   0, 128 */ // CAN BE put in TFT.h
#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

MCUFRIEND_kbv tft;          // tft object

FUNCTIONS

void landscape(uint16_t c)
void dispFrame(int x, int y, int w, int h, uint16_t c)
void dispLineH(int x, int y, int l, uint16_t c)
void dispLineV(int x, int y, int l, uint16_t c)
void dispMsgS(int x, int y, uint16_t c, char *m)
void dispMsg(int x, int y, uint16_t c, char *m)
void dispMsgL(int x, int y, uint16_t c, char *m)
void dispMsgF(int x, int y, uint16_t c, const GFXfont *f,  char *m)
void dispNum(int x, int y, uint16_t c, double n, byte d)
void dispNumF(int x, int y, uint16_t c, const GFXfont *f,  double n, byte d)
void dispDateF(int x, int y, uint16_t c, const GFXfont *f, byte dw, byte da, byte
mo, byte yr)
```

```
void dispTimeF(int x,int y, uint16_t c, const GFXfont *f, byte h, byte m, byte s)
void dispFreqF(int x, int y, uint16_t c, const GFXfont *f, double hz, double chz,
byte d)
void dispBar(int x, int y,  uint16_t c, int w, byte h, byte b)
void dispStepF(int x, int y, uint16_t c, const GFXfont *f, int s)


Notes
1. xxxF() have fixed internal size = 1
2. color is defined as above names
3. fonts included are only those above, many more are available, see GFX
4. dispNum uses default font size 2, dispNumF can use any font included, 9-24pt
5. date, 0 = Sun, da = 1-31, mo = 1-12, yr = [20]xx
6. time h = 0-23, m = 0-59, s = 0-59
7. freq hz = Hz, chz = cHz, d = decimal places (max 3)
8. bar width, height, 0-100% fill
9. step (Hz) 10, 100, 1000... 1000000
```

TFT functions

The header file TFT.h provides the basics for using the GFX & MCUFRIEND libraries. It provides four fonts and 9 pre-defined colors. It also provides ready-written functions for screen orientation and color, frames, lines and messages using either system default or defined fonts.

```
// TFT.h 10-8-19
// for 3.5" 28 pin MCUFRIEND TFT

// libraries
#include <Adafruit_GFX.h>    // Core graphics library
#include <MCUFRIEND_kbv.h>   // HW specific library

// fonts
#include <Fonts/FreeSans9pt7b.h> // include only if required
#include <Fonts/FreeSans12pt7b.h>
#include <Fonts/FreeSans18pt7b.h>
#include <Fonts/FreeSans24pt7b.h>
#include <FreeDefaultFonts.h>

// colors
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF
#define GRAY    0x8410

// alternative colors, above, can be defined here, if needed

// object
MCUFRIEND_kbv tft;
```

```
// FUNCTIONS ==========================================

// screen orientation and color
void landscape(uint16_t c) {
  tft.fillScreen(c);
  tft.setRotation(1);
}


// display frame x, y, w, h, color
void dispFrame(uint16_t x, uint16_t y, uint16_t w, uint16_t h, uint16_t c) {
  tft.drawRect(x, y, w, h, c);
}

// display horiz line across x, y, l,  color
void dispLineH(uint16_t x, uint16_t y, uint16_t l, uint16_t c) {
  tft.drawFastHLine(x, y, l, c);
}

// display vert line down x, y, l,  color (start top)
void dispLineV(uint16_t x, uint16_t y, uint16_t l, uint16_t c) {
  tft.drawFastVLine(x, y, l, c);
}

// display small message at x, y, color, message
void dispMsgS(uint16_t x, uint16_t y, uint16_t c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(1);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}

// display message at x, y, color, message
void dispMsg(uint16_t x, uint16_t y, uint16_t c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(2);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}
// display large message at x, y, color, message
void dispMsgL(uint16_t x, uint16_t y, uint16_t c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(3);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}

// display font message at x, y, color, &font, message
void dispMsgF(uint16_t x, uint16_t y, uint16_t c, const GFXfont *f,  char *m) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  tft.print(m);
}
```

```
// display number at x, y, color, number (double), decimal places
void dispNum(uint16_t x, uint16_t y, uint16_t c, double n, byte d) {
  tft.setFont(NULL);
  tft.setTextSize(2);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(n, d);
}


// display font number at x, y, color, &font, number, decimalplaces
void dispNumF(uint16_t x, uint16_t y, uint16_t c, const GFXfont *f,  double n,
byte d) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  tft.print(n, d);
}

// display freq at x, y, color, font, Hz, cHz, d decimal places
void dispFreqF(uint16_t x, uint16_t y, uint16_t c, const GFXfont *f, double hz,
double chz, byte d) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);

  hz = hz / 1000.0;
  chz = chz / 100000.0;

  tft.print(hz + chz, d);
  tft.print("kHz");
}

// display step at x, y, color, font, s (Hz)
void dispStepF(uint16_t x, uint16_t y, uint16_t c, const GFXfont *f, uint16_t s) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);

  switch (s) // display freqStep
  {
    case 10:
      tft.print(" 10Hz");
      break;
    case 100:
      tft.print("100Hz");
      break;
    case 1000:
      tft.print(" 1kHz");
      break;
    case 10000:
      tft.print(" 10kHz");
      break;
    case 100000:
      tft.print("100kHz");
      break;
```

```
    case 1000000:
      tft.print(" 1MHz");
      break;
    case 10000000:
      tft.print("10MHz");
      break;
  }
}

// display at x, y, color, font, dw, da, mo, yr [20]xx
void dispDateF(uint16_t x, uint16_t y, uint16_t c, const GFXfont *f, byte dw, byte
da, byte mo, byte yr) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
   switch (dw) {
    case 1:
      tft.print("Mon");
      break;
    case 2:
      tft.print("Tue");
      break;
    case 3:
      tft.print("Wed");
      break;
    case 4:
      tft.print("Thu");
      break;
    case 5:
      tft.print("Fri");
      break;
    case 6:
      tft.print("Sat");
      break;
    case 7:
      tft.print("Sun");
      break;
  }

  tft.print(" ");
  tft.print(da);

  tft.print(" ");
  switch (mo)
  {
    case 1:
      tft.print("Jan");
      break;
    case 2:
      tft.print("Feb");
      break;
    case 3:
      tft.print("Mar");
      break;
    case 4:
      tft.print("Apr");
      break;
    case 5:
```

```
      tft.print("May");
      break;
    case 6:
      tft.print("Jun");
      break;
    case 7:
      tft.print("Jul");
      break;
    case 8:
      tft.print("Aug");
      break;
    case 9:
      tft.print("Sep");
      break;
    case 10:
      tft.print("Oct");
      break;
    case 11:
      tft.print("Nov");
      break;
    case 12:
      tft.print("Dec");
      break;
  }
  tft.print(" ");
  tft.print("20");
  tft.print(yr);
}

// display time HH:MM:SS at x, y, color, font, h, m , s
void dispTimeF(uint16_t x,uint16_t y, uint16_t c, const GFXfont *f, byte h, byte
m, byte s) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  if (h < 10)
    tft.print("0");
  tft.print(h);
  tft.print(":");
  if (m < 10)
    tft.print("0");
  tft.print(m);
  tft.print(":");
  if (s < 10)
    tft.print("0");
  tft.print(s);
}

// display bar at x, y, color, w, h, b (0-100%)
void dispBar(uint16_t x, uint16_t y,  uint16_t c, uint16_t w, uint16_t h, uint16_t
b) {
  tft.drawRect(x, y, w, h, c);
  tft.fillRect(x, y, (w * b)/100, h, c);
}
```

TFT_TEST example
This is a sketch that exercises all the functions, no attempt has been made to create a pretty layout, just to demo each function.

```
// TFT_TEST

#include "TFT.h"

void setup() {
  uint16_t ID;

  ID = tft.readID();   // get display ID and initialise
  tft.begin(ID);
  landscape(BLACK);    // screen format & background COLOR
}

void loop() {
  uint16_t b;             // bar 0-100%
// ===== drawing
  dispFrame(10, 10, 450, 300, WHITE); // frame
  dispLineH(10, 150, 450, BLUE);      // horiz line
  dispLineV(225, 10, 300, BLUE);      // vert line
// ===== default fonts
  dispMsgS(15, 15, YELLOW, "Small");  // default font text, num
  dispMsg(80, 15, YELLOW, "Medium");
  dispMsgL(200, 15, YELLOW, "Large");
  dispNum(350, 15, YELLOW, 129.77, 2);
// ===== fonts
  dispNumF(15, 70,GREEN, &FreeSans18pt7b, 129.77, 2); // font number
  dispMsgF(15, 105, GREEN, &FreeSans18pt7b, "Message 18pt"); // font message, num
  dispDateF(15, 140, CYAN, &FreeSans18pt7b, 0, 1, 7, 19); // font date
  dispTimeF(15, 180, CYAN, &FreeSans18pt7b, 13, 55, 32); // font time
  dispFreqF(15, 220, MAGENTA, &FreeSans18pt7b, 14500000, 5000, 2); // font freq
  dispStepF(15, 260, MAGENTA, &FreeSans18pt7b, 100); // font step
// graphics
  for (b = 0; b <= 100; b++) {
   dispBar(40, 280, WHITE, 400, 20, b); // bar
   delay(20);
  }

  while(1);
}
```

TFT.h header file
This is the header itself. Some functions are dedicated to use in an Amateur Radio signal generator or VFO application and date and time display (RTC or GPS use)

```
// TFT.h 10-8-19
// for 3.5" 28 pin MCUFRIEND TFT

// libraries
#include <Adafruit_GFX.h>    // Core graphics library
#include <MCUFRIEND_kbv.h>   // HW specific library

// fonts
#include <Fonts/FreeSans9pt7b.h> // include only if required
#include <Fonts/FreeSans12pt7b.h>
```

```
#include <Fonts/FreeSans18pt7b.h>
#include <Fonts/FreeSans24pt7b.h>
#include <FreeDefaultFonts.h>

// colors
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF
#define GRAY    0x8410

// object
MCUFRIEND_kbv tft;

// FUNCTIONS ==========================================

// screen orientation and color
void landscape(uint16_t  c) {
  tft.fillScreen(c);
  tft.setRotation(1);
}


// display frame x, y, w, h, color
void dispFrame(uint16_t x, uint16_t y, uint16_t w, uint16_t h, uint16_t  c) {
  tft.drawRect(x, y, w, h, c);
}

// display horiz line across x, y, l,  color
void dispLineH(uint16_t x, uint16_t y, uint16_t l, uint16_t  c) {
  tft.drawFastHLine(x, y, l, c);
}

// display vert line down x, y, l,  color (start top)
void dispLineV(uint16_t x, uint16_t y, uint16_t l, uint16_t  c) {
  tft.drawFastVLine(x, y, l, c);
}

// display small message at x, y, color, message
void dispMsgS(uint16_t x, uint16_t y, uint16_t  c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(1);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}

// display message at x, y, color, message
void dispMsg(uint16_t x, uint16_t y, uint16_t  c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(2);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}
```

```
// display large message at x, y, color, message
void dispMsgL(uint16_t x, uint16_t y, uint16_t  c, char *m) {
  tft.setFont(NULL);
  tft.setTextSize(3);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(m);
}

// display font message at x, y, color, &font, message
void dispMsgF(uint16_t x, uint16_t y, uint16_t  c, const GFXfont *f,  char *m) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  tft.print(m);
}

// display number at x, y, color, number (double), decimal places
void dispNum(uint16_t x, uint16_t y, uint16_t  c, double n, byte d) {
  tft.setFont(NULL);
  tft.setTextSize(2);
  tft.setTextColor(c);
  tft.setCursor(x, y);
  tft.print(n, d);
}


// display font number at x, y, color, &font, number, decimalplaces
void dispNumF(uint16_t x, uint16_t y, uint16_t  c, const GFXfont *f,  double n,
byte d) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  tft.print(n, d);
}

// display freq at x, y, color, font, Hz, cHz, d decimal places
void dispFreqF(uint16_t x, uint16_t y, uint16_t  c, const GFXfont *f, double hz,
double chz, byte d) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);

  hz = hz / 1000.0;
  chz = chz / 100000.0;

  tft.print(hz + chz, d);
  tft.print("kHz");
}

// display step at x, y, color, font, s (Hz)
void dispStepF(uint16_t x, uint16_t y, uint16_t  c, const GFXfont *f, uint16_t s)
{
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
```

```
  tft.setTextSize(1);

  switch (s) // display freqStep
  {
    case 10:
      tft.print(" 10Hz");
      break;
    case 100:
      tft.print("100Hz");
      break;
    case 1000:
      tft.print(" 1kHz");
      break;
    case 10000:
      tft.print(" 10kHz");
      break;
    case 100000:
      tft.print("100kHz");
      break;
    case 1000000:
      tft.print(" 1MHz");
      break;
    case 10000000:
      tft.print("10MHz");
      break;
  }
}

// display at x, y, color, font, dw, da, mo, yr [20]xx
void dispDateF(uint16_t x, uint16_t y, uint16_t  c, const GFXfont *f, byte dw,
byte da, byte mo, byte yr) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
   switch (dw) {
    case 1:
      tft.print("Mon");
      break;
    case 2:
      tft.print("Tue");
      break;
    case 3:
      tft.print("Wed");
      break;
    case 4:
      tft.print("Thu");
      break;
    case 5:
      tft.print("Fri");
      break;
    case 6:
      tft.print("Sat");
      break;
    case 7:
      tft.print("Sun");
      break;
  }
```

```
  tft.print(" ");
  tft.print(da);

  tft.print(" ");
  switch (mo)
  {
    case 1:
      tft.print("Jan");
      break;
    case 2:
      tft.print("Feb");
      break;
    case 3:
      tft.print("Mar");
      break;
    case 4:
      tft.print("Apr");
      break;
    case 5:
      tft.print("May");
      break;
    case 6:
      tft.print("Jun");
      break;
    case 7:
      tft.print("Jul");
      break;
    case 8:
      tft.print("Aug");
      break;
    case 9:
      tft.print("Sep");
      break;
    case 10:
      tft.print("Oct");
      break;
    case 11:
      tft.print("Nov");
      break;
    case 12:
      tft.print("Dec");
      break;
  }
  tft.print(" ");
  tft.print("20");
  tft.print(yr);
}

// display time HH:MM:SS at x, y, color, font, h, m , s
void dispTimeF(uint16_t x,uint16_t y, uint16_t  c, const GFXfont *f, byte h, byte
m, byte s) {
  tft.setFont(f);
  tft.setCursor(x, y);
  tft.setTextColor(c);
  tft.setTextSize(1);
  if (h < 10)
    tft.print("0");
  tft.print(h);
  tft.print(":");
```

```
  if (m < 10)
    tft.print("0");
  tft.print(m);
  tft.print(":");
  if (s < 10)
    tft.print("0");
  tft.print(s);
}

// display bar at x, y, color, w, h, b (0-100%)
void dispBar(uint16_t x, uint16_t y,  uint16_t  c, uint16_t w, uint16_t h,
uint16_t b) {
  tft.drawRect(x, y, w, h, c);
  tft.fillRect(x, y, (w * b)/100, h, c);
}
```

Addendum, basic GFX

Basic functions available in GFX library, from which the TFT.h functions are created. The GFX library manual should be consulted for their use.

```
void drawPixel(uint16_t x, uint16_t y, uint16_t color);
void drawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t color);
void drawFastVLine(uint16_t x0, uint16_t y0, uint16_t length, uint16_t color);
void drawFastHLine(uint8_t x0, uint8_t y0, uint8_t length, uint16_t color);
void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
void fillRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
void drawCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color);
void fillCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color);
void drawRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t
radius, uint16_t color);
void fillRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t
radius, uint16_t color);
void drawTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2,
uint16_t y2, uint16_t  color);;
void fillTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2,
uint16_t y2, uint16_t color);
void drawChar(uint16_t x, uint16_t y, char c, uint16_t color, uint16_t bg, uint8_t
size);

void setCursor(uint16_t x0, uint16_t y0);
void setTextColor(uint16_t color);
void setTextColor(uint16_t color, uint16_t backgroundcolor);
void setTextSize(uint8_t size);
void setTextWrap(boolean w);

void drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h,
uint16_t color);

void fillScreen(uint16_t color);
```