Befehle für das Display des TTGO T-Display

1. Initialisierung

```
# SPI...
from machine import Pin, SPI
spi = SPI(2, baudrate=20000000, polarity=1, sck=Pin(18), mosi=Pin(19))

# Display...
import st7789
display = st7789.ST7789(spi, 135, 240,  reset=Pin(23, Pin.OUT), cs=Pin(5, Pin.OUT), dc=Pin(16, Pin.OUT), backlight=Pin(4, Pin.OUT), rotation=3) #landscape
display.init()  #Display initialisieren
display.fill(0) #Display löschen
```

Der erste Parameter von `st7789.ST7789` gibt an, welches SPI-Objekt von `display` benutzt werden soll; der zweite und dritte Parameter geben die Anzahl der Pixel in horizontaler und vertikaler Richtung (im Portrait-Modus, s. u.) an. Darauf folgen die Zuweisungen der Signalleitungen `reset`, `cs`, `dc` und `backlight`, s. o.). Der letzte Parameter gibt die Orientierung an (Drehung jeweils im Uhrzeigersinn):

0: Portrait-Modus (Drehung um 0˚)
1: Landscape-Modus (Drehung um 90˚)
2: Inverser Portrait-Modus (Drehung um 180˚)
3: Inverser Landscape-Modus (Drehung um 270˚)

2. Ausgabe von Texten

Zur Ausgabe eines Textes wird unserer Instanz *display* nur eine einzige Methode zur Verfügung gestellt: Mit

```
display.text(font, s, x, y[, fg, bg])
```

wird die Zeichenkette `s` mit dem angegebenen Font ausgegeben. Die linke obere Ecke des Text-Feldes wird durch die Koordinaten `x` und `y` vorgegeben. Die Parameter `fg` (foreground) und `bg` (background) sind optional; sie geben die Textfarbe bzw. die Hintergrundfarbe an. Lässt man sie weg, dann werden die Vorgabewerte WHITE (`fg`) bzw. BLACK (`bg`) benutzt.

Die von uns benutzte Firmware stellt bereits eine Reihe von Zeichensätzen zur Verfügung; sie müssen also nicht mehr auf dem ESP32 installiert werden. Die Zeichensätze, welche wir in unserem Programm einsetzen wollen, müssen allerdings importiert werden. Durch

```
import vga1_8x8 as font1
```

wird z. B. der Zeichensatz `vga1_8x8` (kleine Schriftgröße, keine Sonderzeichen) importiert und steht dann unter dem Namen font1 zur Verfügung.

Hier eine Liste der Zeichensätze: vga1_16x32, vga1_8x16, vga1_8x8, vga1_bold_16x16, vga1_bold_16x32, vga2_16x16, vga2_16x32, vga2_8x16, vga2_8x8, vga2_16x16, vga2_bold_16x16, vga2_bold_16x32.

Der Font vga2_bold_16x32 stellt z. B. Zeichen in Fett-Schrift mit einer Breite von 16 Pixeln und einer Höhe von 32 Pixeln dar; es handelt sich um einen erweiterten Zeichensatz.

Sonderzeichen: Für `s` kann in der `text`-Methode auch ein Byte-String eingesetzt werden; auf diese Weise lassen sich auch Sonderzeichen ausgeben.

3. Ausgabe von Grafiken

Unser `display`-Objekt besitzt folgende Grafik-Methoden:

• `fill(color)`
Füllt das gesamte Display mit der angegebenen Farbe

• `pixel(x, y, color)`
Gibt dem Pixel (x, y) den angegebenen Farbwert

• `line(x0, y0, x1, y1, color)`
Zeichnet eine Strecke mit der angegebenen Farbe von (x0, y0) nach (x1, y1)

• `hline(x, y, length, color)`
Zeichnet eine horizontale Strecke mit der angegebenen Farbe und Länge (in Pixeln), beginnend bei (x, y); schneller als `line`

• `vline(x, y, length, color)`
Zeichnet eine vertikale Strecke mit der angegebenen Farbe und Länge (in Pixeln), beginnend bei (x, y); schneller als `line`

• `rect(x, y, width, height, color)`
Zeichnet ein Rechteck mit den angegebenen Seitenlängen mit (x, y) als linker oberer Ecke

• `fill_rect(x, y, width, height, color)`
Füllt ein Rechteck mit der angegebenen Farbe

• `blit_buffer(buffer, x, y, width, height)`
Kopiert die Inhalte von Bytes() oder Bytearrays() in den internen Bildschirmspeichers. Beachten Sie: Jede Farbe erfordert 2 Bytes in dem Array.

4. Farben

Unser `display`-Objekt unterstützt nur 16-Bit-Farben im RGB565-Format. Das Modul st7786 stellt einige häufig benutzte RGB565-Farbwerte als Konstanten zur Verfügung: `st7789.BLACK`, `st7789.BLUE`, `st7789.CYAN`, `st7789.GREEN`, `st7789.MAGENTA`, `st7789.RED`, `st7789.WHITE` und `st7789.YELLOW`.

Daneben stellt dieses Modul auch eine Methode zur Umwandlung von RGB-Werten in das RGB565-Format zur Verfügung: `st7789.color565(r, g, b)`.