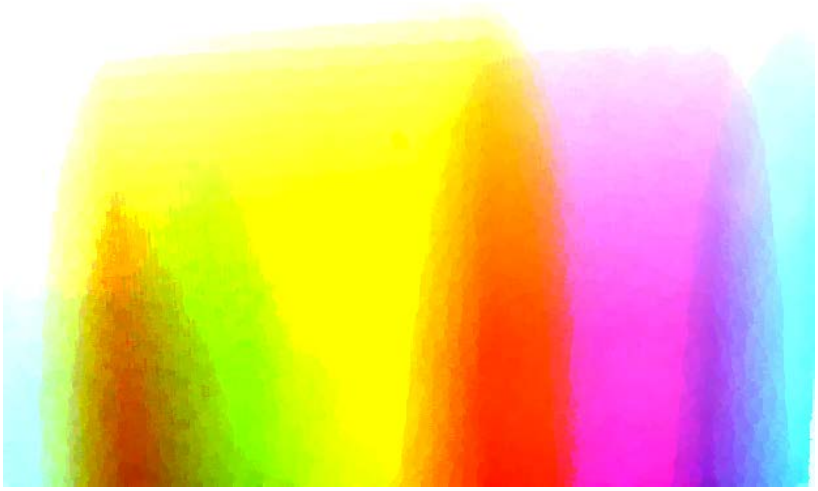
MCB32 APP
Zeige Farbraum auf Display



MCB32 - Embedded Programmierung

Version: 1.01A

Bitte beachten. Diese Unterlagen können ohne Vorankündigung jederzeit angepasst, verbessert und erweitert werden. Wir bitten Sie Wünsche und auch Fehler zu melden. (info@mcb32.ch)

2 Einleitung Display

Das Display hat eine Auflösung von 240 * 320 Punkten mit einer Farbdarstellung von 16Bit pro Pixel. Der Farbcode hat 32 ROT-, 32 Blau- und 64 Grünanteile.

1. Bitaufteilung des Farbcodes

5 Bit **R**, 6 Bit **G**, 5 Bit **B** also: | **RRRR'RGGG'GGGB'BBBB**

Beispiel: maximales grün

`long maxgruen = 1'1111B <<5 =` | **0000'0111'1110'0000**

Beispiel: grau

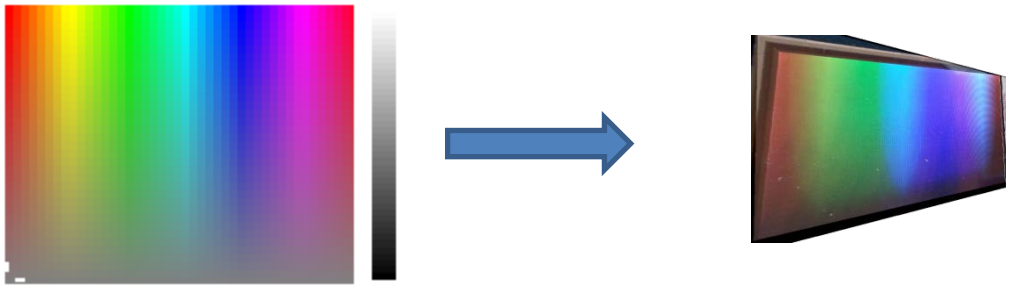
`long grauton_7 = 7<<11 + 14<<5 + 7 =` | **0011'1001'1100'0111**

Folgende Farben sind vordefiniert:

BLACK	WHITE		DARK_GRAY	LIGHT_GRAY
OLIVE	GREEN	BRIGHT_GREEN	DARK_GREEN	LIGHT_GREEN
BROWN	BLUE	BRIGHT_BLUE		LIGHT_BLUE
	YELLOW	BRIGHT_YELLOW	DARK_YELLOW	
	CYAN	BRIGHT_CYAN		LIGHT_CYAN
	RED	BRIGHT_RED		LIGHT_RED
	MAGENTA	BRIGHT_MAGENTA		LIGHT_MAGENTA

3 Mustercode für Farbraumdarstellung

Neben den verschiedene Möglichkeiten Daten auf dem Bildschirm darzustellen interessiert auch die Ansteuerung der Farben.



Wie einfach ist es dieses Muster auf den Bildschirm zu bringen. Im Prinzip sind, wenn nur die Hauptfarben angenommen werden 8 Kombinationen möglich. Von 000 bis RGB. 000 ist Schwarz und RGB (maximum) ist WEISS. Wenn 000 als Schwarz, also Abwesenheit von Ansteuerung, als Farbe definiert wird, so bleiben noch folgende Sequenzen übrig:

000 – R00 – RG0 – 0G0 – 0GB – 00B –R0B. Damit sind alle Farben gegeneinander variiert worden. Wenn nun wir noch WEISS dazu nehmen und den Farbkreis schliessen sieht die Sequenz wie folgt aus: 000 – R00 – RG0 – 0G0 – 0GB – 00B –R0B – RGB – R00

000 bis jeweils maximales R,G,B wird auf der X-Achse dargestellt. Das heisst die Intensität der Farbe wird damit variiert. Die restlichen Sequenzen werden auf der Y-Achse dargestellt.

In einer ersten Version werden die Sequenzen hintereinander abgearbeitet. Ein Loop variiert die X-Achse und der zweite Loop berechnet die Farben entlang der Y-Achse.

Das Struktogramm für die erste Sequenz sieht wie folgt aus:

```
void zeichne_farbraum_rechteckig(void)  // Zeichne den Farbraum über den ganzen Bildschirmuchar Linie_ON
                                                    /* Init Sequenz 1 */
for( uc_va2=wStartGuiyV;uc_va2<=wEndGuiyV;uc_va2++ )
    // 1ter äusserer Loop um Farbe darzustellen  Y-Achse
    for( uc_va1=StartGuixV;uc_va1<=EndGuixV;uc_va1++ )
        // innerer Loop um Farbsättigung darzustellen  (X-Achse)
        wRED     =    ((wEndValueRED*uc_va1)/EndGuixV);                    // setze Farben zusammen
        wGREEN   =    (wEndValueGREEN*ucFarbRasterPoint*uc_va1)/(FarbRasterPoints*EndGuixV);
        wBLUE    =    0;
        plotDot(uc_va1,uc_va2,(((wRED)<<11)|((wGREEN)<<5)|(wBLUE)));        // zeichne nun die aktuelle Farbe
    printdebug2;                // Macro aus Debugumgebung. Kein C-Code
    ucFarbRasterPoint++;        // Erhöhe FarbrasterPoint -> damit wird die Intensität der neu dazukommenden Farbe erhöht. (0 .. 100% innerhalb der Sequenz)
if ( ucZeichneLiniezwSequenz==ON )
then
    // Zeichne Linie 1 (Position xxx). Nur wenn der Paramter ZeichneLinie = ON ist
    printAt(8, "uc_va2");printDec (4, uc_va2); line(1,uc_va2,319,uc_va2,1,WHITE);
else

/* ------------------------------ Erste Sequenz Fertig  Wir haben nun RG0 erreicht.  */
```

Im Abschnitt „InitSequenz 1“ werden die Parameter gesetzt. Die Ausgestaltung des Programmes erlaubt in diesem Abschnitt einiges an Logik zu implementieren. Dazu gehören auch die Berechnung der Start- und Endpunkte des folgenden Loops.

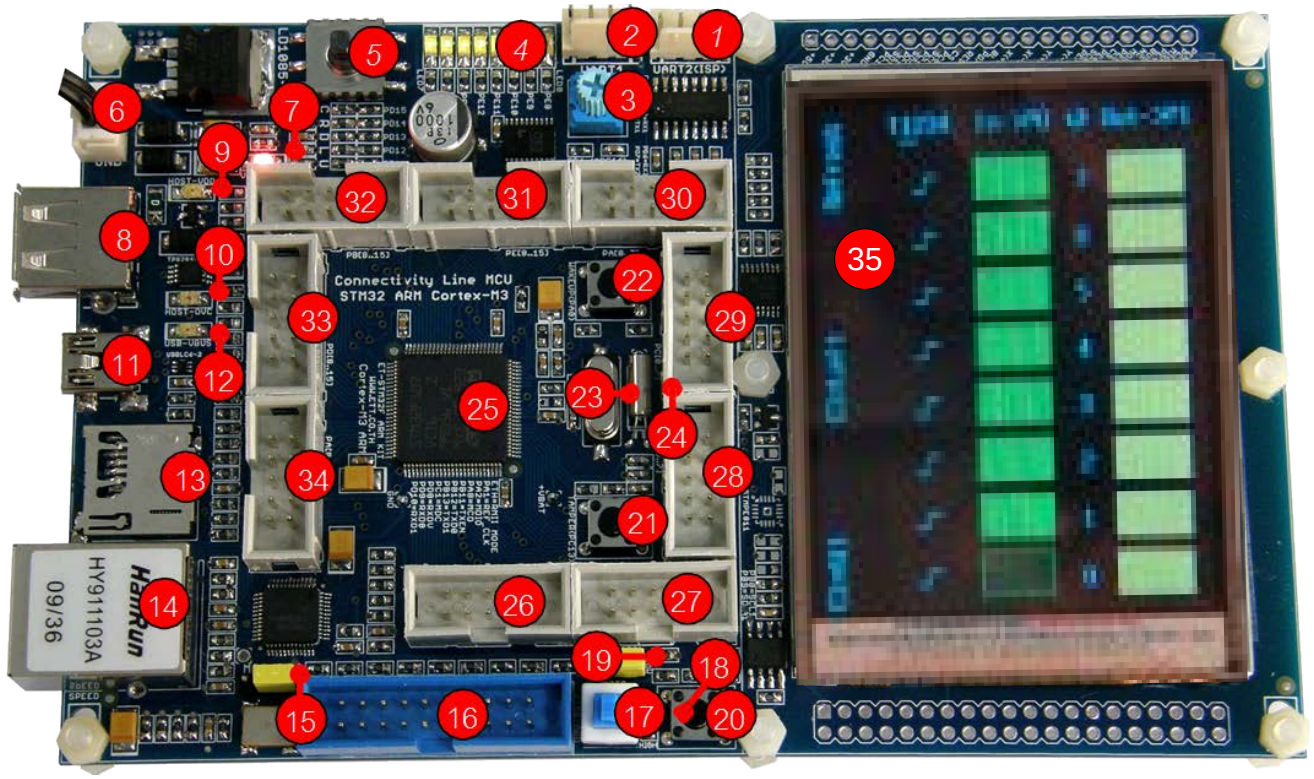
Der Abschnitt Zeichne Linie ist eine Spielerei um am Ende der Sequenz eine Linie zu zeichnen.

Der Obige Code wiederholt sich mit unterschiedlichen Inits und Berechnungen der Farben bis alle Sequenzen gezeichnet sind.

Die 3 Grundfarben werden einzelnen von 0 bis Maximum variiert.

Für mehr Informationen siehe Demo Programm.

4 Anhang Übersicht Board



4.1 Tabelle mit Beschreibung der MCB32-Funktionen

#	Beschreibung der markierten Position (i)
1	RS232-2 (Uart2) Stecker. Empfängt HEX File für Bootloader
2	RS232-1 (Uart1) Stecker.
3	0…3,3Volt für A/D Wandler Test an (PC4/ADC14)
4	LED[0..7] angesteuert via PE[8..15].
5	5 Richtungs (Joy) Schalter
6	Stecker für +5Volt Speisung für das ganze Board
7	LED (+VDD: +3.3V) Status
8	USB Host
9	LED zeigt Status von USB-Host VDD.
10	LED zeigt Status von "Host Over Current"
11	Stecker "USB Device/OTG"
12	LED zeigt Status von "USB VBUS"
13	Micro SD Card Sockel
14	RJ45 Ethernet LAN.
15	Jumper(MCO/OSC) für die Auswahl des Clock-Signals für DP83848V.
16	JTAG Stecker für Real Time Debugging
17	Switch BOOT0 welcher zusammen mit Jumper BOOT1 den Boot-Modus der MCU bestimmt: Boot Loader (BOOT0=1, BOOT1=0) und Run (BOOT0=0,BOOT1=0).
18	LED zeigt Logic Status von BOOT0 = 1 (ON=Boot Loader, OFF=Run).

19	Jumper BOOT1(PB2); Ist im Normalfall auf LOW
20	Switch RESET.
21	Switch Tamper (PC13).
22	Switch Wakeup (PA0).
23	Quarz 25MHz für die Zeitbasis der MCU
24	Quarz 32.768KHz für die Zeitbasis der RTC (Real Time Clock)
25	MCU No.STM32F107VC T6
26	Stecker GPIO PD[0..7].
27	Stecker GPIO PB[0..7].
28	Stecker GPIO PE[0..7].
29	Stecker GPIO PC[0..7].
30	Stecker GPIO PA[0..7].
31	Stecker GPIO PE[8..15].
32	Stecker GPIO PB[8..15].
33	Stecker GPIO PD[8..15].
34	Stecker GPIO PA[8..15].
35	320x240 Dot TFT LCD mit Touch Screen Sensor. REV C oder REV D