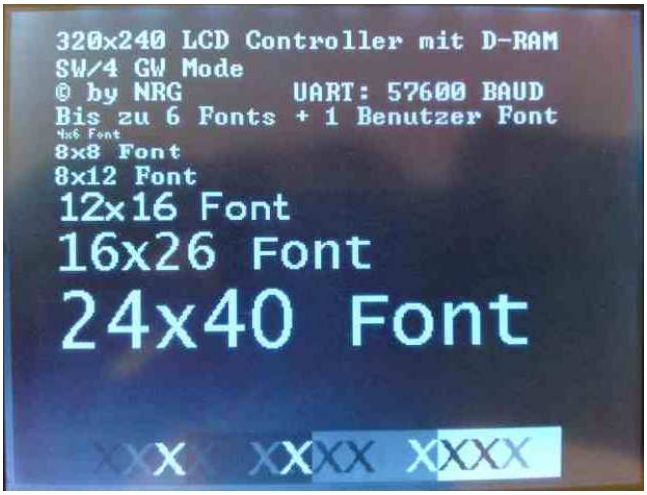
Display Controller für ein 320x240 4 Bit Display



Unterstützte Mikrocontroller

- Mega16/164/32/324/644

Schnittstelle: UART

- (Baudrate über 3 Jumper wählbar)
- 9600 (Default), 14400, 19200, 28800, 38400, 56000, 57600, 115200
- Befehlspuffer: 512 Byte

Grafikfunktionen

- Pixel Setzen/Löschen/Farbe
- Bilder 2/4 Grauwerte
- Linien
- Rechtecke
- Kreise
- Bereich/Rechteck invertieren

2/4 Graustufen

- (während der Laufzeit umschaltbar)
- virtuelle Auflösung von
    - 1024x240 @1bpp
    - 512x240 @2bpp

3 - 6 Fonts (Zeichensätze)

- 4x6 Font
- 8x8 Font
- 8x12 Font
- 12x16 Font (Nur Mega32/324/644)
- 16x26 Font (Nur Mega32/324/644) (reduzierter Zeichensatz 0-200)
- 24x40 Font (Nur Mega644)

Ein Benutzerzeichensatz (max. 240 Zeichen) (nur mit einem 256kx4 DRAM)

- maximale Größe je Zeichen 128 Byte
- Größe einstellbar
- X -> 4/8/12/16/20/24/(28/32/36/40)
- Y -> 1-42

Touchpanel Unterstützung

- Für ein four-wire Touchpanel
- Werte werden über UART übertragen
- umschaltbar zwischen AD-Werten und Pixelposition
- Kalibrierwerte für Pixelposition sind im EEPROM gespeichert
- Bis zu 400 Messungen pro Sekunde

2 PWM Ausgänge für Hintergrundbeleuchtung und Kontrast

Der Controller ist ca. 50-55% mit der Anzeige ausgelastet. Die restliche Rechenleistung steht zum Zeichnen der Fonts und Grafikfunktionen zur Verfügung.

Befehl	Parameter	Beschreibung
0		NOP (nichts machen)
1		Cursor auf (0,0) setzen
2		LCD (Kontrastspannung) aus
3		LCD (Kontrastspannung) an (Default)
4		LCD LED aus (Default)
5		LCD LED an
6	Mode (Werte)	Touchpanel Mode 0 -> Touchpanel aktivieren und AD-Werte übertragen Mode 1 -> Touchpanel aktivieren und Pixelposition übertragen Mode 2 -> (Anzahl der Messungen) (Default -> 4) (Wert -> 1,2,4,8,16,32,64) Mode 3 -> Kalibrierwert für Xmin (Xmin Low / Xmin High) Mode 4 -> Kalibrierwert für Xmax (Xmax Low / Xmax High) Mode 5 -> Kalibrierwert für Ymin (Ymin Low / Ymin High) Mode 6 -> Kalibrierwert für Ymax (Ymax Low / Ymax High) (Kalibrierwerte werden im EEPROM gespeichert) Übertragungsprotokoll: Touch gedrückt --> 01, X Low, X High, Y Low, Y High (wiederholt sich solange Touch gedrückt bleibt) Touch losgelassen --> ein mal 02
7		Touchpanel aus (Default)
8		Backspace
9	X Low X High Y Farbe	Set Pixel (x, y, Farbe(MSB))
10		Cursor eine Zeile nach unten
11		Zeichen an Cursorposition löschen
12		Komplettes Bild löschen
13		Cursor an Zeilenanfang setzen
14	Wert	LCD_LED_PWM-Wert (0-255) (Default->128)
15	Wert	Kontrast einstellen (0-255)
16	0xAA XY XS YS Modus Daten…	Bild an Position (x,y) mit Größe (xs,ys) laden. Auf diesen Befehl folgen xs*ys Bytes mit Bilddaten. X und xs zählen Bytes (ein Byte ist bei 1bpp daher 8 Pixel, bei 2bpp 4 Pixel). X hat daher den Bereich 0-127, xs den Bereich 1-128. y und ys zählen Zeilen. Y hat daher den Bereich 0-239, ys den Bereich 1-240. Modus schaltet zwischen 1bpp (0), 2bpp (1) um . 0xAA (170) dient nur als Sicherheitsbyte, damit nicht versehentlich dieser Befehl (z.B. aufgrund eines Übertragungsfehlers) ausgeführt wird.
17	X Y	Textcursor auf (x,y) setzen X zählt in 4 Pixel Schritten, Y in Zeilenschritten
18	X1 Low X1 High Y1 X2 Low X2 High Y2 Farbe	Linie von (x1,y1) nach (x2,y2) mit Farbe zeichnen

19	X1 Low X1 High Y1 R Farbe 1 Farbe 2	Kreis mit Mittelpunkt (x,y) und Radius R zeichnen. Farbe 1 ist die Füllfarbe (MSB). Farbe 2 ist die Rahmenfarbe (MSB). Hinweis: Der Radius darf nicht größer sein als x oder y, ebenso darf er nicht 0 sein, ansonsten wird der Befehl ignoriert.
20	X1 Low X1 High Y1 X2 Low X2 High Y2	Invertiere Bereich/Rechteck mit Eckkoordinaten (x1,y1, x2,y2).
21	Textfarbe Hintergrundfarbe	Setzt Textfarbe und Hintergrundfarbe (MSB) (Es wird nur B7 und B6 ausgewertet)
22		Nicht benutzt
23	Zeichensatz	Zeichensatz Auswahl 0-> Benutzer Zeichensatz (240 Zeichen mit maximal 128 Byte je Zeichen) 1-> 4x6 2-> 8x8 3-> 8x12 (Default) 4-> 12x16 (nur Mega32/324 und Mega644) 5-> 16x26 (nur Mega32/324 und Mega644) (verkleinerter Zeichensatz 0-200) 6-> 24x40 (nur Mega644)
24	Mode	Farbmodus 0=> SW Mode (virtueller Bildschirm 1024 Pixel) 1=> 4GW Mode (Default) (virtueller Bildschirm 512 Pixel)
25	Nummer Zeile 1 … Zeile n	Benutzerdefiniertes Zeichen an die entsprechende Position laden. Jedes Zeichen besteht aus n Zeilen (Bytes) zu je m Pixeln (Bits). Insgesamt sind 240 Zeichen verfügbar (Nummer 0-239)
26	X Y	Setze Größe des benutzerdefinierten Zeichensatzes. Erlaubte Werte: (Default: X=8 , Y=12) X -> 4/8/12/16/20/24/ (28/32/36/40) Y -> 1-42 X*Y max. 1024
27	X	Setzt den X-Offset für den Textcursor (für das Beschreiben des virtuellen Bildes). X zählt in 4 Pixel Schritten
28	X	Setzt die linke Startposition des virtuellen Bildes in 4 Pixel Schritten. Der Wertebereich liegt zwischen 0-175 bei 1bpp und 0-47 bei 2bpp.
29	X1 Low X1 High Y1 X2 Low X2 High Y2 Farbe 1 Farbe 2	Rechteck mit Eckkoordinaten (x1,y1, x2,y2) zeichnen. Farbe 1 ist die Füllfarbe (MSB). Farbe 2 ist die Rahmenfarbe (MSB).
30	Ziel Wert	Zeichen aus erweitertem Zeichensatz schreiben (nur für ASCII Codes <32 notwendig). Ziel = 0: Benutzerdefiniertes Zeichen 0-240 schreiben Ziel > 0: ASCII Zeichen aus dem ausgewählten Font schreiben. (0-255)
31	Font X Y Zeichen	Zeichen aus einem Zeichensatz an bestimmte Position schreiben. X-> in 4 Pixel Schritten (0-255) Y -> Zeile (0-240) Zeichen -> (0-255)
32-255		Buchstaben an Cursorposition zeichnen, Cursor erhöhen.

Das Sendeprotokoll unterscheidet sich zum Projekt von Benedikt K. nur um ein paar Befehle.

Änderungen:

- 17 -> X zählt in 4 statt 8 Pixelschritten (Curser auf (X,Y))
- 27 -> X zählt in 4 statt 8 Pixelschritten (Curser X-Offset)
- 28 -> X zählt in 4 statt 8 Pixelschritten (Bildschirm Startposition)

Neue Befehle:

- 04 -> LCD Backlight an
- 05 -> LCD Backlight aus
- 06 -> Touchpanel An/Mode/Kalibrieren
- 07 -> Touchpanel Aus
- 14 -> LCD-Backlight PWM-Wert
- 20 -> invertiere Bereich
- 23 -> Zeichensatz wählen
- 24 -> 2/4 Graustufen Mode
- 26 -> Benutzerzeichensatz Größe.
- 31 -> Zeichen aus Font an Position X/Y

.

Beispiele:

A B C (oder 65, 66, 67 als Binärwert)
schreibt die Zeichen A, B, C an die aktuelle Cursorposition

29 140 0 100 179 0 139 0 255
zeichnet ein 40x40 großes, schwarzes Rechteck, mit einem weißen Rand

19 160 0 120 15 64 128
zeichnet einen Kreis mit dem Mittelpunkt 160,120 und dem Radius 15, gefüllt
mit der dunkelsten Graustufe und einem etwas helleren Rand

16 170 2 16 1 8 0 1 2 4 8 16 32 64 128
zeichnet ein 8x8 Pixel großes Bild mit den Daten 1, 2, 4, 8, 16, 32, 64, 128
(=schräge Linie) an die Position 16, 16

16 170 4 16 2 8 1 0 3 0 12 0 48 0 192
zeichnet ein 8x8 Pixel großes Bild (schräge Linie) an die Position 16, 16, also 3 0 12 0 48 0 192 0
genauso wie vorher, nur im Graustufenmodus

Touchpanel:

Die Touchpanel-Unterstützung ist für ein four-wire Touch geschrieben.
Mit dem Befehl 6 kann das Touchpanel aktiviert oder konfiguriert werden.
Die Kalibrierwerte , um aus den AD-Werten die Pixel Positionen zu berechnen, werden im EEPROM des Controllers gespeichert und können über Mode 3-6 geändert werden.

Achtung: Nach dem Übertragen der Kalibrierwerte muss das Touch erneut aktiviert werden um die neuen Werte zu übernehmen.

Protokoll:
Bei Berührung wird „01, X Low, X High, Y Low, Y High“ gesendet.
Solange das Touch gedrückt bleibt werden die aktuellen Werte übertragen,
Sobald das Touch losgelassen wird, wird einmal „02“ gesendet

Mode 0 und 1 unterscheiden sich nur darin das statt AD-Werten die Pixelpositionen übertragen werden.

Benutzerzeichensatz:

Die Zeichen werden von links oben nach rechts unten übertragen.
Vor dem Senden des ersten Zeichens muss die Größe des Zeichensatzes definiert werden.
Bei Fonts bei denen X nicht durch 8 teilbar ist (4,12,20..) ist im letzten Byte von Zeile 1 , 4 Bit (B7-B4) von Zeile 1 und 4 Bit (B3-B0) von Zeile 2 enthalten.
Bei nicht gerader Zeilenanzahl und X nicht durch 8 teilbar ist (4,12,20..) , sind im letzten Byte die unteren 4 Bit (B3–B0) gleich 0 (werden nicht ausgewertet/gespeichert).

Hinweise zur Ansteuerung:

Aufgrund der teilweise ziemlich rechenintensiven Grafikbefehle sollte der RTS/Busy Ausgang vor dem Senden eines Befehls abgefragt werden. Der Controller verfügt zwar über einen 512 Byte Befehlspuffer, aber auch dieser ist irgendwann voll, wenn mehrere aufwendige Befehle ohne Pause gesendet werden! Solange der RTS/Busy Pin Low ist, kann man gefahrlos mindestens 128 Bytes senden, da RTS/Busy aktiviert wird, wenn der Puffer zu ¾ gefüllt ist. Man braucht also nicht vor jedem Byte RTS/Busy zu prüfen, sondern es reicht einmal vor jedem Befehl (falls dieser kürzer als 128 Bytes ist.)

Wer mag kann den Befehlspuffer in der uart.h auch erhöhen.

- Bei mega16/164 max. 512
- Bei mega32/324 max. 1536
- Bei mega644 max. 3584

UART Baud Auswahl:

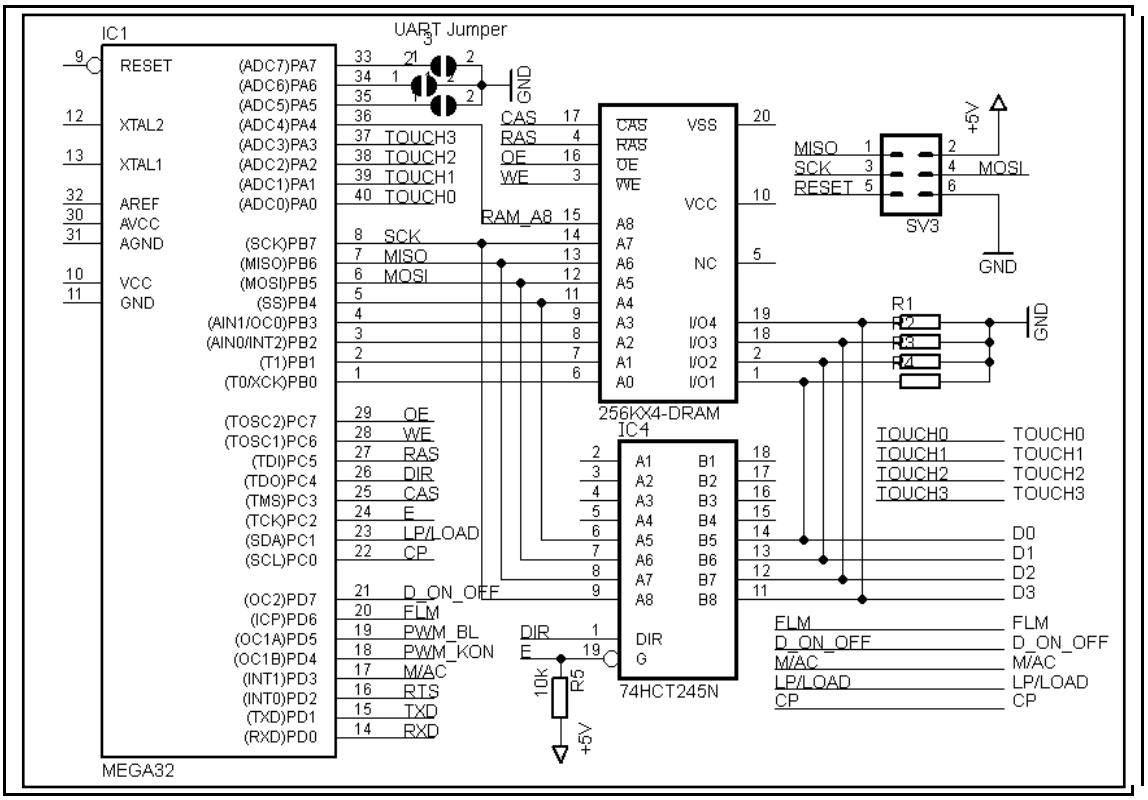
Interner Pull-Up aktiv

JP_2	JP_1	JP_0	BAUD
Low	Low	Low	115.200
Low	Low	High	57.600
Low	High	Low	56.000
Low	High	High	38.400
High	Low	Low	28.800
High	Low	High	19.200
High	High	Low	14.400
High	High	High	9.600

Hardware:

Bauteileliste:
1x ATMega 16/164/32/324/644
1x 74HC245 (oder HCT/F/LS)
1x D-Ram mit 256k x 4 Bit
+ einige Widerstände und Kondensatoren

Schaltplan:



Die meisten Pins können in der param.h angepasst werden.

R1-R5 sollten einen Wert von 10k - 100k haben.

PWM Ausgänge:
Die Hintergrundbeleuchtung kann über PD5(OC1A) gesteuert werden.
Die Kontrastspannung kann über PD4(OC1B) gesteuert werden

D-Ram:
Der D-Ram sollte bei 16MHz min 120ns haben, bei 20MHz min 100ns.
Für alle Funktionen wird ein 256kx4 Bit D-Ram benötigt.
Bei einem 64kx4 Bit D-Ram ist der Benutzerzeichensatz nicht verfügbar.
Alternativ kann man auch 4x 256kx1 (64kx1) Bit D-Rams verwenden.
(Gute Quellen für D-Rams sind alte Simm- und PS/2 Speichermodule)

Falls zusätzliche Grafikfunktionen in die Software eingebaut werden sollten, hier eine kurze Beschreibung der Schnittstelle:

```
void lcd_writechar (    unsigned char x,        // X Position in 4 Pixel Schritten, Bereich 0-255
                        unsigned char y,        // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char c,        // Zeichen, Bereich 0-255
                        unsigned char fontnr);  // Font (Zeichensatz) , Bereich 0-6
```
Schreibt einen Buchstaben auf das LCD an die angegebene Position (x,y).
Die Text- und Hintergrundfarbe sind in der globalen Variable charcol gespeichert.

```
void lcd_string(        unsigned char x,        // X Position in 4 Pixel Schritten, Bereich 0-255
                        unsigned char y,        // Y Position in 1 Pixel Schritten, Bereich 0-239
                        char *txt,              // Nullterminierter Textstring im Flash
                        unsigned char fontnr);  // Font (Zeichensatz) , Bereich 0-6
```
Schreibt einen Text aus dem Flash an Position (x,y) auf das LCD.

```
void lcd_writebyte (    unsigned char x,        // X Position in 8 Pixel Schritten, Bereich 0-63/0-127
                        unsigned char y,        // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char c);       // Datenbyte das 8 Pixel enthält
```
Schreibt 1 Byte Pixeldaten (8 Pixel) auf das LCD an die angegebene Position (x,y).

```
void lcd_writebyteg (   unsigned char x,        // X Position in 4 Pixel Schritten, Bereich 0-127/0-255
                        unsigned char y,        // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char c);       // Datenbyte das 4 Pixel enthält
```
Schreibt 1 Byte Pixeldaten (4 Pixel) mit 2bpp Farben auf das LCD an die angegebene Position (x,y).

```
void lcd_writebyteex (  unsigned char x,        // X Position in 8 Pixel Schritten, Bereich 0-63/0-127
                        unsigned char y,        // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char c,        // Datenbyte das 8 Pixel enthält
                        unsigned char textcol,  // Farbe die den 1en aus dem Byte zugewiesen
                        unsigned char backcol);// Farbe die den 0en aus dem Byte zugewiesen
```
Schreibt 1 Byte Pixeldaten (8 Pixel) auf das LCD und erweitert die Farben auf die angegeben Werte.

```
void lcd_setpixel (     unsigned short x,       // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y,       // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char c);       // Pixelfarbe (2bpp, an MSB ausgerichtet)
```
Zeichnet einen Pixel an (x,y) mit einer bestimmten Farbe.

```
void lcd_invpixel (     unsigned short x,       // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y);      // Y Position in 1 Pixel Schritten, Bereich 0-239
```
Invertiert ein Pixel an (x,y).

```
void lcd_invnibbel (    unsigned short x,       // X Position in 4 Pixel Schritten, Bereich 0-127/0-255
                        unsigned short y);      // Y Position in 1 Pixel Schritten, Bereich 0-239
```
Invertiert ein 4 Bit an (x,y).

```
void lcd_invblock (     unsigned short x1,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y1,      // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned short x2,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y2);     // Y Position in 1 Pixel Schritten, Bereich 0-239
```
Invertiert einen Bereich/Rechteck auf dem LCD.

```
void lcd_line (         unsigned short x1,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y1,      // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned short x2,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y2,      // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char color);   // Linienfarbe (2bpp, an MSB ausgerichtet)
```
Zeichnet eine Linie von (x1,y1) nach (x2,y2) mit einer bestimmten Farbe.

```
void lcd_block (        unsigned short x1,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y1,      // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned short x2,      // X Position in 1 Pixel Schritten, Bereich 0-511/0-1023
                        unsigned short y2,      // Y Position in 1 Pixel Schritten, Bereich 0-239
                        unsigned char color,    // Füllfarbe (2bpp, an MSB ausgerichtet)
                        unsigned char color2);  // Rahmenfarbe (2bpp, an MSB ausgerichtet)
```
Zeichnet ein Rechteck auf das LCD und löscht alles was vorher an dieser Stelle war.

```
void lcd_circle(        unsigned short xm,      // X Position Mittelp. in 1 P. Schritten, Bereich 0-511/0-1023
                        unsigned short ym,      // Y Position Mittelp. in 1 P. Schritten, Bereich 0-239
                        unsigned char r,        // Radius in 1 Pixel Schritten, Bereich 1-119
                        unsigned char color1,   // Füllfarbe (2bpp, an MSB ausgerichtet)
                        unsigned char color2);  // Rahmenfarbe (2bpp, an MSB ausgerichtet)
```
Zeichnet einen Kreis auf das LCD und löscht alles was vorher an dieser Stelle war.

```
void ram_writenibbel (  unsigned short x,       // Spalte (Col), Bereich 0-255 (Daten)
                        unsigned short y,       // Zeile (Row), Bereich 0-239 (Zeichen)
                        unsigned char c);       // Datenbyte das 4 Bit enthält (B7-B4)
```
Schreibt 4 Bit ins D-Ram für den Benutzerzeichensatz.

```
void lcd_clear(void);
```
Löscht das LCD (setzt alle Pixel auf 0)

```
void lcd_init(void);
```
Startet den LCD-Controller

```
void init_touch(void);
```
Berechnet die Tabelle für die Umrechnung von AD-Werten nach Pixelposition.

In der param.h sind einige Einstellmöglichkeiten wie z.B. die Anschlussbelegung, sowie einige Konstanten definiert:

#define F_CPU 16000000UL
Hier wird der Quarztakt eingestellt, falls F_CPU nicht global über das AVR Studio definiert wird. Dann kann diese Zeile auskommentiert werden.

#define Framerate 75
Damit wird die Bildwiederholfrequenz des LCDs eingestellt. Üblich sind etwa 50-80Hz. Wegen den Graustufen sollte man eher einen etwas höheren Wert einstellen, damit das Display nicht flimmert. Allerdings verursacht eine höhere Frequenz natürlich auch eine höhere Rechenleistung. Über etwa 120Hz sollte man daher nie gehen!

#define COMMAND_TIMEOUT 75
Um zu verhindern, dass bei einer Unterbrechung während des Ladens eines Bildes der Controller nicht endlos auf Daten wartet, bricht jede Pause größer als der eingestellte Wert das Laden eines Bildes ab. Die Werte gelten in Frames. Ein Wert von 75 würde also in diesem Fall eine Zeit von 1s ergeben.

#define FASTLOOP
Ist diese Zeile vorhanden/nicht auskommentiert, wird die Schleife mit der Datenausgabe entrollt, also anstelle der Schleife stehen die Befehle 80x im Code. Kostet Flash, ist aber rund 15% schneller.

Globale Variablen (main.c)

Über diese Variablen können die Defaults (Startwerte) geändert werden.

- viewstart
  - legt die linke Startposition des virtuellen Bildes fest (0-175 / 0-47)
- Graymode
  - 0 = 2 Graustufen / 1 = 4 Graustufen
- charcol
  - Text- Hintergrundfarbe (B7/6 Hintergrund Farbe – B5/4 Textfarbe)
- fontnr
  - Aktiver Font (0-6) (Achtung text_x / text_y auch anpassen)
- text_x
  - Aktiver Font X Größe in 4 Pixel Schritten
- text_y
  - Aktiver Font Y Größe
- textb_x
  - Benutzer Font X Größe in 4 Pixel Schritten
- textb_y
  - Benutzer Font Y Größe
- tprogpos
  - Touchpanel -> 0 = deaktiviert / 1= aktiviert / 2-10 wird in der ISR verwendet wenn aktiviert
- tmode
  - Touchpanel Mode -> 0 = AD-Werte übertragen / 1 = Pixelposition übertragen
- Touchcount
  - Anzahl der AD-Messungen (1,2,4,8,16,32,64)

Informationen zu Befehlsdauer/Rechenleistung:

Je nach Befehl benötigt der Controller unterschiedlich viele Takte zum Abarbeiten.

TIMER0_COMP_vect:

- Touch deaktiviert: 445 Takte
- Touch aktiviert : ca. 490 Takte

Die Routine wird 75*240 mal die Sekunde ausgeführt (sendet eine Zeile zum LCD)

lcd_writechar:

Font	SW Mode (Takte)	4GW Mode (Takte)
0	(((39*X/4)+7)*Y)+40	(((46*X/4)+7)*Y)+40
1	250	320
2	360	510
3	490	720
4	930	1430
5	1600	2600
6	3400	5920

lcd_clear:

- 377.900 Takte