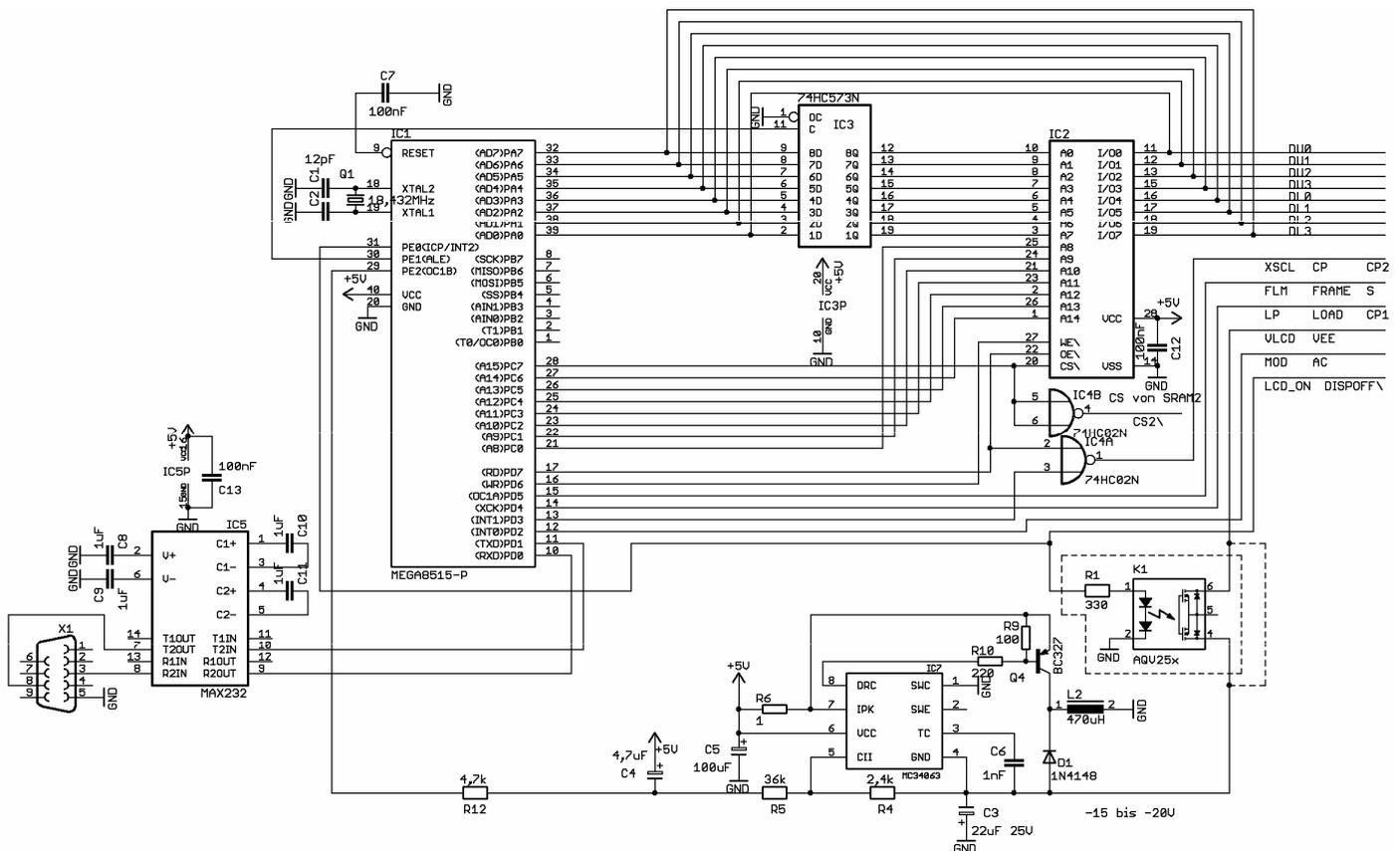


640x480 LCD Controller

- LCD Controller für ein 640x480 Dualscan LCD.
- Schnittstelle: UART, 115200Baud (max. etwa 300kBaud)
- vollständiger 8x12 Zeichensatz (bis zu 8 Zeichensätze in externem Flash)
- Grafikfunktion wie Linie, Kreise, Rechtecke, Bilder
- Bis zu 16 Bildschirmseiten (abhängig von der SRAM Größe)

Befehl	Parameter	Beschreibung
0		NOP (nichts machen)
1		Cursor auf (0,0) setzen
2		LCD (Kontrast Spannung) aus
3		LCD (Kontrast Spannung) an
8		Backspace
9	X Low X High Y Low Y High Farbe	Set Pixel (x,y)
10		Cursor eine Zeile nach unten
11		Zeichen an Cursorposition löschen
12	Farbe	Komplettes Bild löschen bzw. mit der Farbe beschreiben
13		Cursor an Zeilenanfang setzen
15	Kontrast	Kontrast einstellen (0-255)
16	170 X Y Low Y High XS YS Low YS High Daten...	Bild an Position (x,y) mit Auflösung (xs,ys) laden. Auf diesen Befehl folgen xs*ys Bytes mit Bilddaten. x zählt die Nibbles und hat daher den Bereich 0-159, xs zählt Bytes und hat daher den Bereich 1-80. 170 dient nur als zusätzliche Sicherheit
17	X Y Low Y High	Textcursor auf (x,y) setzen. X zählt Nibbles, Y Zeilen.
18	X1 Low X1 High X2 Low X2 High Farbe	Linie von (x1,y1) nach (x2,y2) zeichnen
19	X1 Low X1 High X2 Low X2 High Radius Farbe 1 Farbe 2	Kreis mit Mittelpunkt (x,y) und Radius R zeichnen. Farbe 1 ist die Füllfarbe. Farbe 2 ist die Rahmenfarbe. Hinweis: Der Radius darf nicht größer sein als x oder y, ebenso darf er nicht 0 sein, ansonsten wird der Befehl ignoriert.
20	Schriftart	Wählt die aktive Schriftart aus (nur mit externem FlashROM)
25	Nummer Zeile 1 Zeile 2 ... Zeile 8	Benutzerdefiniertes Zeichen an die entsprechende Position laden laden. Jedes Zeichen besteht aus 8 Zeilen (Bytes) zu je 8 Pixeln (Bits). Insgesamt sind 16 Zeichen verfügbar (Nummer 0-15)
27	Seite	Setzt die Seite des Grafikspeichers in den geschrieben wird
28	Seite	Setzt die Seite des Grafikspeichers die angezeigt wird

29	X1 Low X1 High Y1 Low Y1 High X2 Low X2 High Y2 Low Y2 High Farbe 1 Farbe 2	Rechteck mit Eckkoordinaten (x1,y1, x2,y2) zeichnen. Farbe 1 ist die Füllfarbe. Farbe 2 ist die Rahmenfarbe.
30	Ziel Wert	Zeichen aus erweitertem Zeichensatz schreiben. Ziel = 0: Benutzerdefiniertes Zeichen 0-7 schreiben Ziel > 0: ASCII Zeichen 0-255 schreiben.
32- 255		Buchstaben an Cursorposition zeichnen, Cursor erhöhen.



Der OptoMOSFET K1 (bzw. der gestrichelte Bereich) kann bei LCDs die über einen ONOFF oder DispOFF\ Pin verfügen entfallen und durch die gestrichelte Verbindung ersetzt werden. Falls der OptoMOSFET nicht erhältlich ist, kann dieser durch einen normalen Optokoppler ersetzt werden, wenn dieser genügend Strom für das LCD schalten kann (Stromverstärkung/Übertragungsverhältnis beachten!).

R5/R4 stellen die Spannung für das Display und somit den Kontrast ein. Der Kontrast lässt sich über einen kleinen Bereich per Software über PWM regeln. Sollte dieser Bereich für das LCD nicht passen, müssen R5/R4 angepasst werden.

IC4B generiert aus dem RD\ Impuls und dem Enable Signal (PD3) die Shiftclock Impulse für das LCD. IC4A invertiert das Chipselect Signal für den 2. 32kByte SRAM. Alternativ kann natürlich auch ein 64kByte SRAM verwendet werden, bei dem der Inverter entfallen kann, da A15 als normale Adressleitung genutzt wird. CS\ wird dann dauerhaft an GND gelegt.

Hinweise zur Ansteuerung:

Aufgrund der teilweise ziemlich rechenintensiven Grafikbefehle sollte der RTS/Busy Ausgang (PD1) vor dem Senden eines Befehls abgefragt werden. Der Controller verfügt zwar über einen 256 Byte Befehlspeicher, 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 64 Bytes senden, da RTS/Busy aktiviert wird, wenn der Puffer zu $\frac{3}{4}$ gefüllt ist. Man braucht also nicht vor jedem Byte RTS/Busy zu prüfen, sondern es reicht einmal vor jedem Befehl (falls dieser kürzer als 64 Bytes ist.)

Bei Verwendung eines vollständigen UARTs reicht es dafür das CTS Handshaking zu aktivieren, die Schnittstelle erledigt dann den Rest.

Die Fusebits müssen auf externen Quarz gestellt werden. Und da die Frequenz >8MHz ist, CKOPT nicht vergessen zu setzen! Um einen sicheren Betrieb zu gewährleisten sollte man auch die BOD auf 4V einschalten. Dann schaltet der AVR nämlich ab wenn die Spannung zu klein wird, was wiederum die Displayspannung über den Optomofet abschaltet. Das ganze sieht dann in etwa so aus:

Hinweis: Häkchen bedeutet 1, also unprogrammed!

Low	High
<input checked="" type="checkbox"/> 0: CKSELO	<input checked="" type="checkbox"/> 0: BOOTRST
<input checked="" type="checkbox"/> 1: CKSEL1	<input type="checkbox"/> 1: BOOTSZ0
<input checked="" type="checkbox"/> 2: CKSEL2	<input type="checkbox"/> 2: BOOTSZ1
<input checked="" type="checkbox"/> 3: CKSEL3	<input checked="" type="checkbox"/> 3: EESAVE
<input checked="" type="checkbox"/> 4: SUT0	<input type="checkbox"/> 4: CKOPT
<input checked="" type="checkbox"/> 5: SUT1	<input type="checkbox"/> 5: SPIEN
<input type="checkbox"/> 6: BODEN	<input checked="" type="checkbox"/> 6: WDTON/JTAGEN
<input type="checkbox"/> 7: BODLEVEL	<input checked="" type="checkbox"/> 7: RSTDIBL/OCDEN

Ein paar Worte zum SRAM Timing:

Das Timing hängt hauptsächlich von der "OE\ Access Time" des SRAMs ab:

Im Datenblatt vom ATmega8515, Seite 202, Tabelle 98, Punkt 10:

Read Low to Data valid: max $1.0 \cdot t_{clcl} - 50\text{ns}$. Bei 16MHz ist $t_{clcl} = 62,5\text{ns}$.

Das SRAM darf daher maximal 12,5ns brauchen zwischen dem Anlegen des RD Impulses bis die Daten stabil sein müssen.

Das von mir gerne verwendete IS61C256 Cache SRAM hat in der langsamsten 25ns Ausführung hier nur 9ns. Es ist also ausreichend schnell.

Bei der Low Power Variante IS62C256 dagegen, hat selbst die 45ns Variante hier 25ns. Dies kann funktionieren, da die Werte jeweils die garantierten Maximalwerte sind, muss aber nicht.

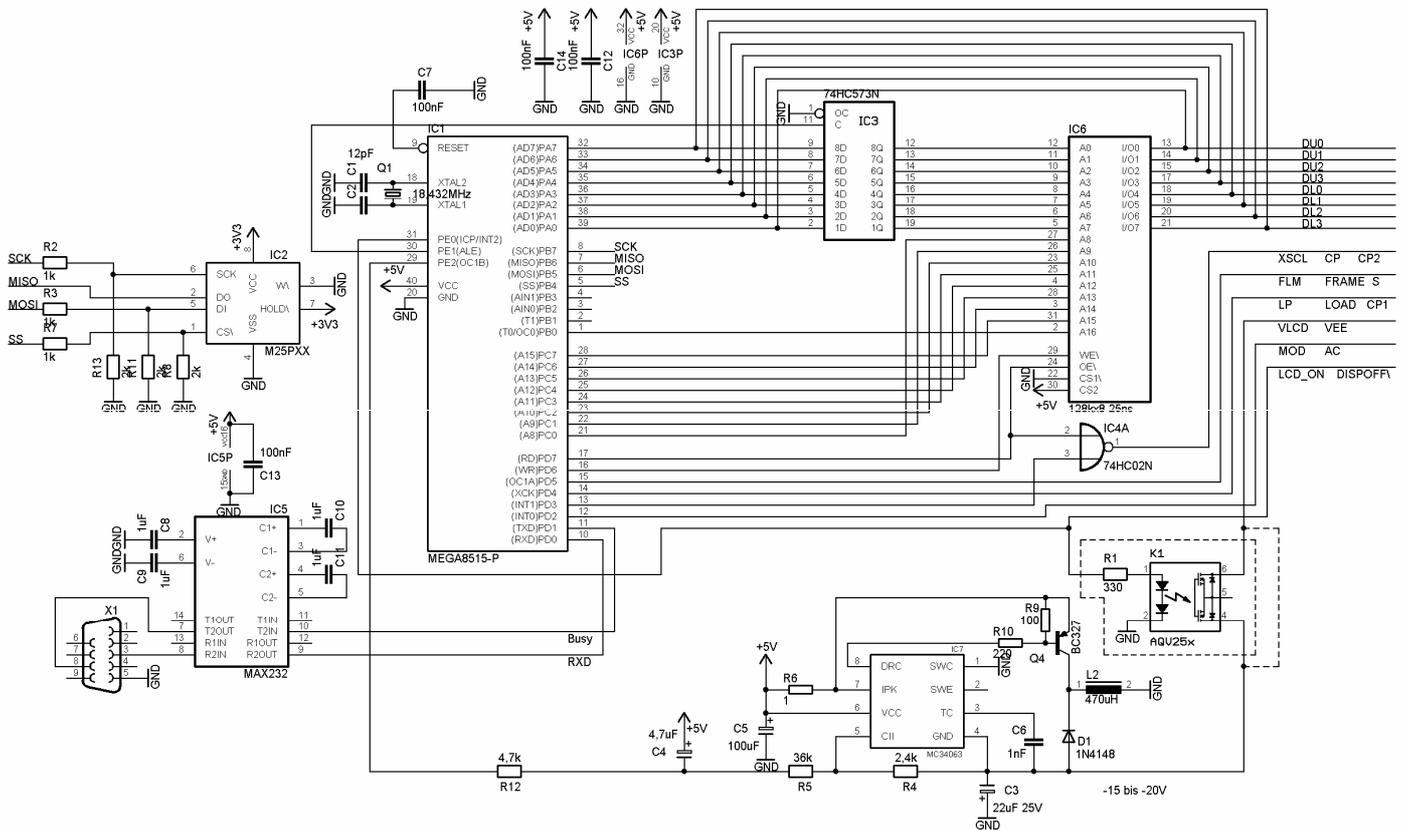
Zusätzlich ist natürlich noch die Zugriffszeit wichtig:

Das wäre Punkt 5 im Datenblatt: Address valid to RD Low: $1.0 \cdot t_{tcc} - 10\text{ns}$.

Insgesamt ergibt sich dadurch eine Zeit von $2.0 \cdot t_{tcc} - 60\text{ns}$, also 55ns zwischen Adresse gültig bis Daten stabil, was eigentlich alle SRAMs die obige Bedingung erfüllen, auch erfüllen.

Mit viel Glück funktionieren aber selbst 100ns SRAMs ohne waitstates. Trotzdem sollte man möglichst SRAMs mit 55ns oder weniger verwenden.

Das Latch ist da meiner Meinung nach unkritisch, da die Zugriffszeit im Vergleich zu der OE Zeit des Speichers extrem groß ist. Daher verwende ich auch nur HC02 und HC573 statt den AC Versionen. Bei langsam SRAMs sind allerdings die AC Versionen empfehlenswert.



Die Software unterstützt auch größere SRAMs (bis max. 1024k x8). Mit diesen sind dann mehrere Bildschirmseiten verfügbar die getrennt beschrieben und angezeigt werden können. Pro Seite werden 64kByte Speicher benötigt. Mit den im Schaltplan gezeigten 128kByte sind daher 2 Seiten verfügbar. Weitere Adressleitungen werden an PortB2 und 3 angeschlossen.

Um den großen SRAM sowie die Seiten Umschaltung zu aktivieren, muss die Zeile `#define LARGEMEM 512`

In der param.h eingefügt werden. Bei nur 64kByte Speicher muss diese Zeile auskommentiert werden. Der Wert hinter der Zeile ist die Speichergröße in kByte.

Ebenso können auch bis zu 8 zusätzliche Schriftarten über ein externes Flash hinzugefügt werden, denn große Zeichensätze benötigen sehr viel Speicherplatz. Aus Platzgründen findet dafür ein 25Pxx SPI Flash in SO8 Verwendung. Da diese nur mit 3,3V laufen ist eine Pegelanpassung in Form von Spannungsteilern notwendig. Dies muss in der param.h freigeschaltet werden.

Zur Fehlersuche, falls die Schaltung nicht läuft, hier ein paar Messwerte (bei 18,432MHz, 1/240 Duty und 65Hz Framerate):

- FLM: kurze, high aktive Peaks mit 640µs Breite und 65,2Hz
- LP: kurze, high aktive Peaks mit 540ns Breite und 15,7kHz
- XSCl: Burst aus 160 low aktiven Peaks, mit etwa 55ns Breite und etwa 5,5MHz (im Mittel sind es rund 2,5MHz wenn man mit einem Frequenzähler misst.)
- MOD: Rechteck mit 50% Tastverhältnis und 32,6Hz.
- PD3 (Enable Signal für die XSCl Impulse): Etwa 51% Tastverhältnis, 15,7kHz