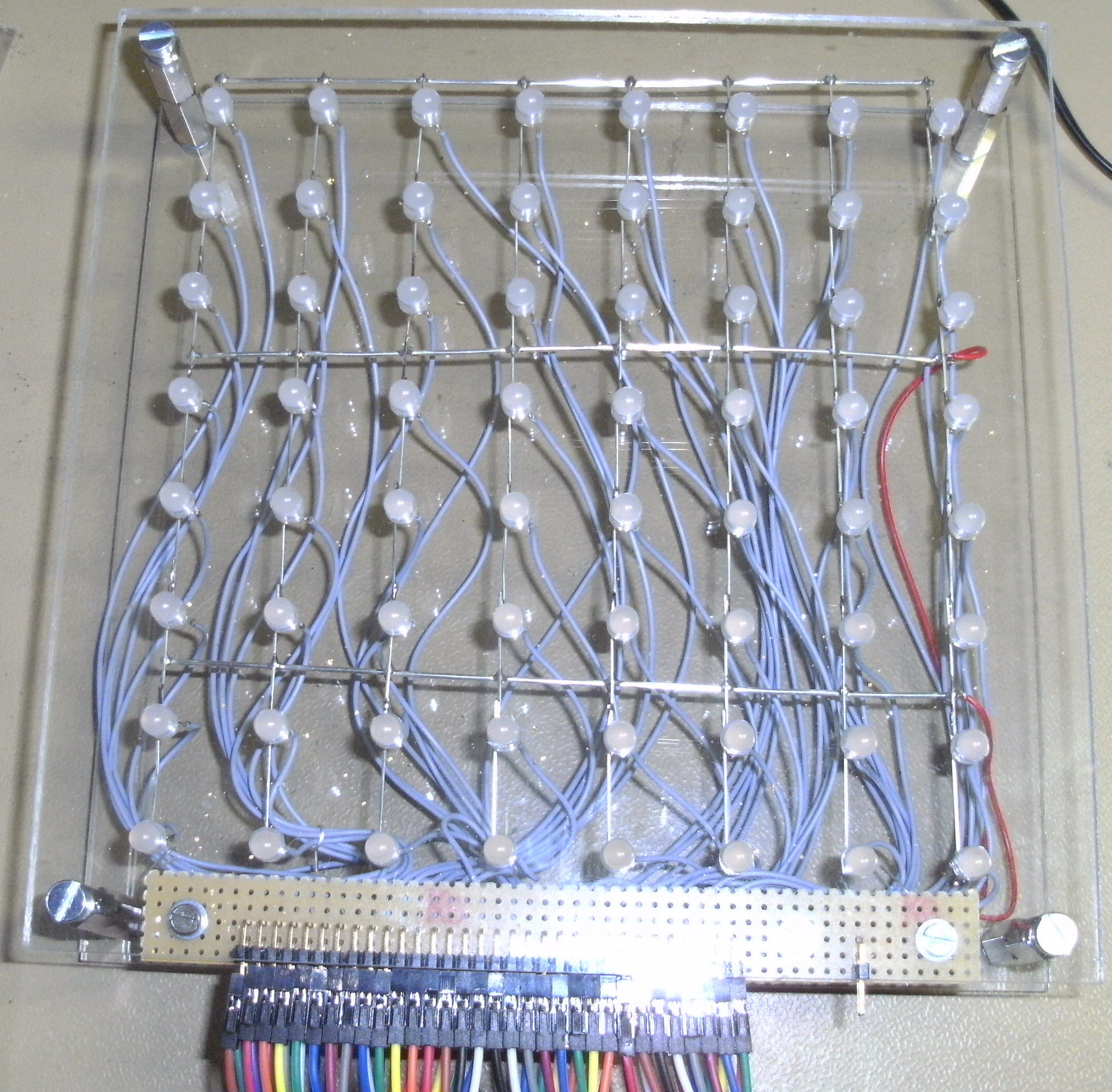
# Allgemeines

Gewünscht: 100 Frames per second

Deshalb: 10ms pro frame

10ms/8 Layer = 1,25 ms pro Layer

## Modell eines Layers



Kathoden der LEDs 1..64

Gemeinsame Anode

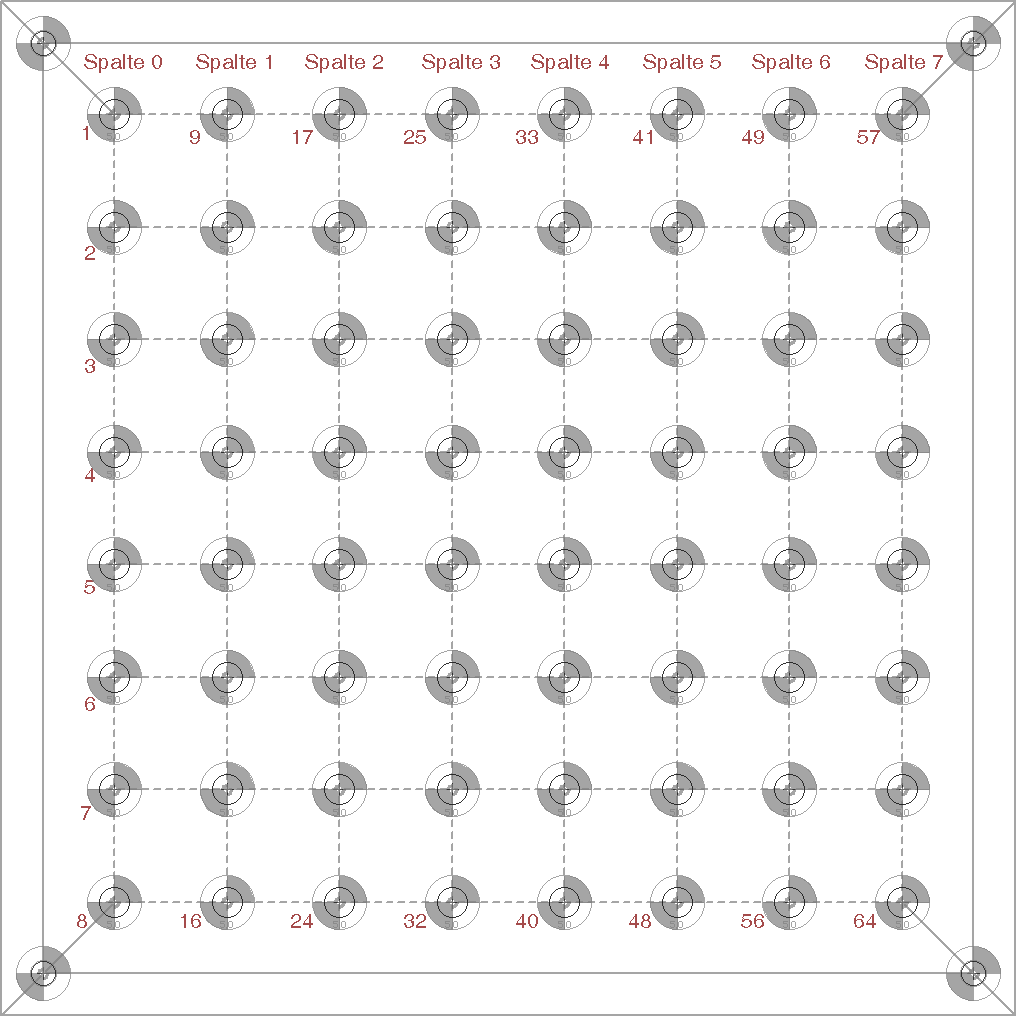
Die LEDs sind spaltenweise organisiert. Im nachfolgenden Diagramm ist die Zuordnung der LEDs zu den Anschlusspins unten dargestellt.

Die verbauten LEDs haben die folgenden Daten: Blau, diffus, IF=20mA, max. 30mA UF=3,02..3,11V

Ein Datenblatt liegt nicht vor.

Quelle: <https://www.amazon.de/dp/B0083S9O44/ref=pe_386171_37038021_TE_3p_dp_1>

Anordnung der LEDs



Anschlussfeld der 64 Kathoden:  
(obere Reihe der Tabelle ist die obere Reihe der Stiftleiste, von vorne auf die Stifte gesehen)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |

|  |  |  |  |
| --- | --- | --- | --- |
| Spalte 1 | Spalte 3 | Spalte 5 | Spalte 7 |
| Spalte 0 | Spalte 2 | Spalte 4 | Spalte6 |

Der einzelne Anschluß ganz rechts ist die gemeinsame Anode aller Dioden

Jede Spalte ist mit einem der 8-Bit-Latches verbunden.

Adressierung im Layer

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Spalten (1 Spalte = 1 Byte) | | | | | | | |  |
| Bits im Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Spalte col |
| 0 |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
|  | cube[layer][0]  cube[layer][7] | | | | | | | |  |
|  | Stiftreihe | | | | | | | |  |

Feld: cube[ layer ][ col ] row

Ein zweidimensionales Feld von Bytes beschreibt den LED-Würfel als Datenstruktur. Dabei gibt die erste Dimensin den Layer an, also die Ebene. Die zweite Dimension beschreibt die Spalte einer Ebene und damit eines der 8 Latches, während das Datenbyte die Zeilen einer Ebene in der Spalte angibt.

Jedes blaue Kästchen entspricht einer LED

Ein 8-Bit-Latch enthält die Daten einer Zeile.

Die 8 Bit eines jeden Latches sind in der üblichen Notation angeordnet

Eine solche Ebene wird über einen High-Side-Treiber ein- und ausgeschaltet (Schaltbild: Q2..Q9)

# Hardware

Der High-Side Switch mit bipolaren Transistoren funktioniert jetzt, Ansteuerschaltung läuft also.

## Die LEDs

Gekauft wurden blaue, diffuse, ultrahelle LEDs , 5mm

500 Stück: <https://www.amazon.de/dp/B0083S9O44?ref_=pe_1365651_54849071> 36,00 €

100 Stück: <https://www.amazon.de/dp/B007VPRR30?ref_=pe_1365651_54849071> 8,50 €

Ein Datenblatt gab es nicht, auf dem Beutel mit den LED standen folgende Angaben:

1,2 Lumen

I = 20mA (max. 30mA)

U = 3,02 – 3,11 V

Eine Messung ergab die folgenden Ergebnisse:

| I (in mA) | U (in V) |  |
| --- | --- | --- |
| 5 | 2,87 | Gut Erkennbares Leuchten |
| 7 | 2,94 | Ausreichende Helligkeit bei normales Tageslicht |
| 9,92 | 3,02 | Sehr hell |
| 14,3 | 3,11 | Keine erkennbar größere Helligkeit gegenüber vorher |
|  |  |  |

Bei 7,5mA leuchten die LEDs hell genug (getestet)

Bei 8mA pro LED und einem Multiplexverhältnis von 1:8 sollten also 64mA durch die LED fließen

Dazu den Spannungsabfall an der LED ausmessen bei realer Schaltung und Tastverhältnis von 1:8

Die LEDS muss man unbedingt testen, es funktionierten einige wenige im 500er-Beutel von Anfang an nicht. Auch ein „Einlaufen“ ist nicht schlecht vor dem Zusammenlöten, denn 4-5 gingen nach wenigen Stunden kaputt.

## LED-Vorwiderstände

Test mit 39Ω und UB=5V

Die LEDs leuchten ausreichend hell. Der Pulsstrom beträgt gemessene 23mA. Obwohl der mittlere Strom damit nur 1/8, also etwa 2,9 mA beträgt, leuchtet die Led schon ausreichend hell. Eine Steigerung von UB auf 7V führte zu einem LED-Strom von 80mA. Trotz des fast 4-fachen Stromes ergab sich nur eine mäßige (sichtbare) Helligkeitssteigergung.

Im endgültigen Projektaufbau sollte der Wert zu 100 Ω gewählt werden. Damit sollten sich bei UB=12V ganz brauchbare Ergebnisse einstellen. Der LED-Strom muss dann nicht durch den 5V-Regler des Übungssystems und es bleibt noch ein bisschen Luft für Experimente. Man kann dann auch eine separate Spannungsquelle nehmen

Die Stromversorgung für die LEDs ist von der Versorgung der Ansteuerlogik getrennt. Damit kann die Helligkeit der LEDS über die Betriebspannung beeinflusst werden.

Gewünscht sind UB=12V, als untere Grenze gelten 8V, damit der 5V-Regler noch genügend Eingangsspannung erhält

Bei einem LED-Strom von 10mA fallen 3V an der LED ab. Ca. 1 V gemessen (Datenblatt: 0,9..1,6V) beträgt der Spannungsabfall an UCE der Treiberstufe, sodaß etwa 12V-3V-1V = 8V am Vorwiderstand der LED abfallen müssen. R = 8V / 10mA = 800 Ohm. Bei einem Multiplex-Verhaltnis von 1:8

Die Angabe der LED zum Nennstrom lautet: 20mA (max. 30mA) bei einem gepulsten Betrieb mit Tastverhaltnis 1:10 sollte die LED den 5-fachen Nennstrom aushalten, macht also 100mA. Mit einem Rv = 100 Ohm und URV=8V ergibt sich ein Strom von etwa 80mA, allerdings ist das Tastverhältnis auch nur 1:8 und nicht 1:10.

Ergo: 100Ohm als RV gewählt und ggf. mit der Betriebsspannung etwas runtergehen sollte ok sein.

## High-Side-Treiber

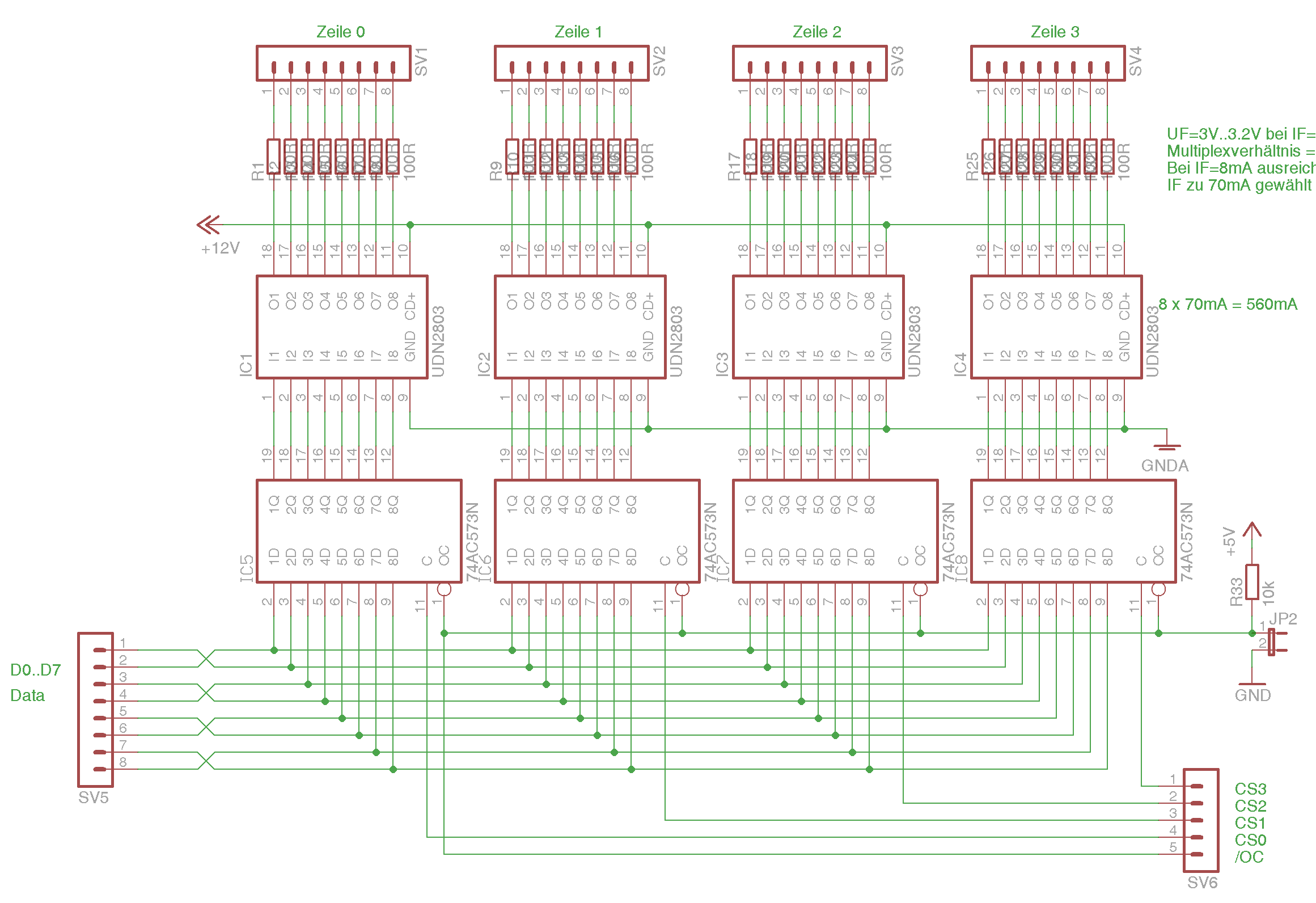
## Low-Side-Treiber

## Latches und Ansteuerung

### Parallel

Der Entwurd Muxer\_Rev\_0.sch stellt den ersten Entwurf dar. Diese Platine wurde in der Schule von Schülern hergestellt und bestückt. Das alleine erwies sich bereits als Schwierigkeit.

Parallel dazu habe ich die Muxer\_Rev\_1.sch entworfen. Es handelt sich um die gleiche Schaltung wie Rev\_0, sie wirde nur auf zwei kleinere Platinen verteilt. Die hier dargestellte Schaltung braucht man also 2x



#### Besonderheiten:

Als schwierig erwies sich die Ansteuerung der LatchEnable-Leitungen CS0..CS7. Der zu beobachtende Effekt war:

* Ein beliebiges Latch mit einem Wert beschreiben, bei dem nur eine 1 an beliebiger Stelle anthalten war 🡪 funktioniert
* Mehr als eine 1 im Datenwart hatte zur Folge, dass andere Latches, die gar nicht adressiert waren, auch Werte übernommen haben, aber scheinbar zufällige.

Getestet wurden:

74HC573 zustandsgesteuert, CMOS

74HCT573 zustandsgesteuert, TTL-kompatible Ein-/Ausgänge

74HC574 positiv flankengetriggert, CMOS

74HCT574 positiv flankengetriggert, TTL-kompatible Ein-/Ausgänge

Der Effekt konnte überall beobachtet werden.

Wahrscheinlich liegt das am ungünstigem Layout der Platine, die ein Übersprechen der schnellen Schaltsignale auf die benachbarten Leitungen begünstigt oder an der freifliegenden Verdrahtung zwischen der Muxerplatine und dem Controllerboard. Aus der Platine gibt es zwar eine Massefläche auf der Oberseite, reicht aber scheinbar nicht.

Behoben werden konnte das Problem durch folgende Schaltungsmaßnahme:

Direkt am Ausgang des Controllers werden in die Leitungen der LatchEnable-Signale der Latches Widerstände mit 39 Ohm eingeschleift.

Am anderen Ende, direkt am LatchEnable-Eingang jedes 573-Bausteines wird eine kleine Kapazität gegen Masse geschaltet. Der Wert der Kapazität wurde experimentell ermittelt. Begonnen mit 22pF, weil die noch vorhanden waren. Kein sichtbarer Effekt. Mit 0,47nF besser, aber immer nocht nicht weg. Aber immerhin konnte man jetzt etwa 6 Bit setzen, bevor der Effekt auftrat. Dann ein Test mit 4,7nF (auf die waren noch vorhanden). Damit klappt es jetzt, nur dürfen die Pulse nicht mehr zu kurz sein, sonst „verschluckt“ sie die Kapazität

ToDo:

Jetzt könnte man mal auf die flankengetriggerte Version zurückgehen und die Caps verkleinern. Der Dämpfungswiderstand alleine in Verbindung mit den flankengetriggerten D-Latches hat nicht genügt. Allerdings ist es jetzt auch schnell genug, trotz des zusätzlichen deelays.

CodeSchnipsel:

void drv\_clockToLatch( uint8\_t latch, uint8\_t value ) {

D573\_Data\_Port = value; //Datenwort an Eingänge der D-Latches anlegen

asm volatile ("nop");

D573\_Clock\_PORT = (1<<latch); //Clock-Leitung des betreffenden Latches aktivieren

//asm volatile ("nop"); //

\_delay\_us(1); //

D573\_Clock\_PORT = 0; //Clock-Leitung wieder auf Low setzen

asm volatile ("nop"); //warte, bis Signal Low-Pegel erreicht hat

\_delay\_us(20); //

//Das Datenwort an den Eingangen der D-Latches stehen lassen

}

Dabei stellte sich heraus, dass der delay zwischen Abschalten des LatchEnable-Signals (Zeile 8) und dem Abschalten des Bitmusters am Data-Port (hier bereits nicht mehr vorhanden, kam in Zeile 11) von Bedeutung ist.

Das ergibt auch sinn, denn die HCT-Typen haben eine untere Schwellenspannung VIL = 1,2V. Erst ab dort wird sicher Low erkannt (VIH = 1,6V). Wird also der Datenport verändert, solange das LatchEnable-Signal wegen des Zeitverzuges duch den Kondensator noch nicht unter 1,2V gesunken ist, werden diese Änderungen evtl. noch in die D-Flipflops übernommen.

τ= RC = 33R \* 4,7E-9F = 156 ns

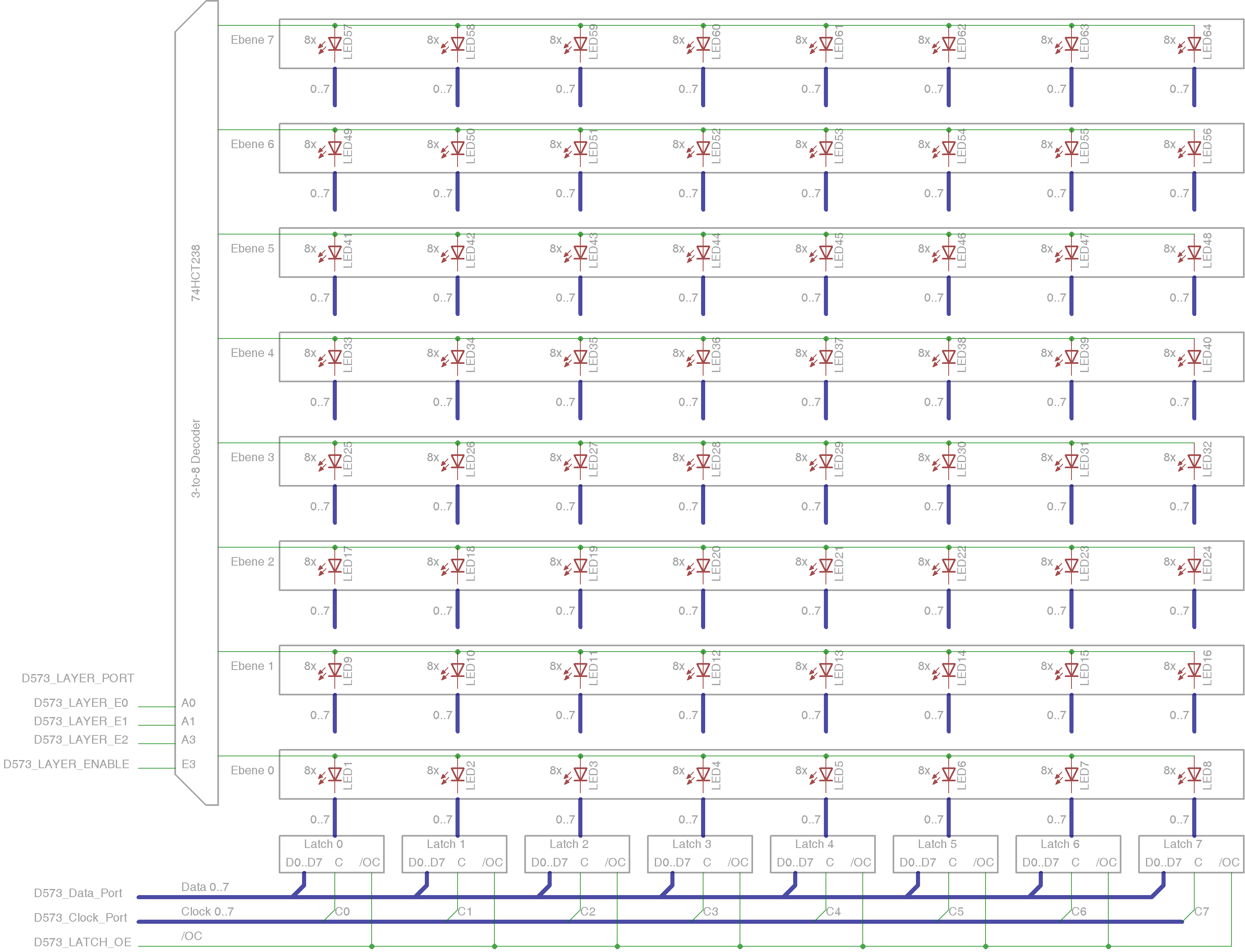
Bei ca. 1,5 τ, also etwa ab 230 ns sollte der Pegel niedrig genug sein. Klappt aber nicht. Mit einem Delay von 20µs klappt es noch. Weitere verkürzungen habe ich nicht mehr getestet. Naja 1µs, geht aber nicht.

Hier gibt es noch Optimierungspotential.

Werde ich noch mal in Verbindung mit der Schülerlösung betrachten...

### Seriell

In Muxer\_Rev\_2.sch wurde serielle Schrieberegister vom Typ 74HC595 verwendet. Diese Schaltung wurde bisher nicht aufgebaut.

Die Bezeichner links am Diagramm sind die gleichen wie sie auch im Treiber drv\_573.c/h verwendet werden

# Software

Belegung der Anschlüsse

|  |  |  |
| --- | --- | --- |
| Port | Bedeutung |  |
| Port A | Unbenutzt, bzw. Testsignale  PA0..PA3 = unbenutzt  PA4 = TEST\_SIG3  PA5 = TEST\_SIG2  PA5 = TEST\_SIG1  PA7 = TEST\_SIG0  TEST\_SIG0  liefert einen kurzen High-Impuls (ca. 200ns), welche zur Synchronisation eines Oszilloskopes verwendet werden kann. Er wird ausgelöst, wenn mit der Ausgabe der Lagen auf Layer 0 begonnen wird.  TEST\_SIG1  liefert einen Puls, welcher der Ausführungszeit der Funktion void drv\_drive() entspricht (ca. 30 µs)  Hinweis: Die 30 µs sind die Ausführungszeit, um alle 8 Latches mit daten zu beschreiben, allerdings ohne die bereits erwähnte Verzögerungszeit von derzeit gut 20µs pro Latch. Sie bildet also die kleinste erreichbare Zeit. Die derzeitige Ausführungszeit dürfte etwa bei gut 160 µs liegen. |  |
| Port B | Daten für die Latches : Data (D0..D7)  X-Koordinate  Treibt die Leitungs Data (D0..D7) diese liegen parallel an allen 8 D-Latches an. Das LatchEnable-Signal (ChipSelect) (CS0..CS7) bestimmt das D-Latch, welches die anliegenden Daten übernehmen soll |  |
| Port C | Multiplexer: LatchEnable / ChipSelect (CS0..CS7)  Y-Koordinate  Hier ist jeweils nur eine Leitung kurz auf 1. Dabei wird das Datenwort an den Eingängen des 74573, welches über Port B bereitgestellt wird, wird übernommen.  void drv\_clockToLatch( uint8\_t latch, uint8\_t value ) |  |
| Port D | Layer adressieren, Z-Koordinate  PD0..PD2 = unbenutzt  PD3 = OE  PD4 = E0  PD5 = E1  PD6 = E2  PD7 = Enable  **E2..E0** bestimmen als 3-Bit-Zahl, welche der 8 Ebenen adressiert wird. Eine 3-Bit-Zahl erlaubt 8 verschiedene Zustände, die Zahlen 0..7. Diese werden mit Hilfe des 74238 ausdekodiert. Der Chip sorgt dafür, dass in jedem Fall max. 1 Ebene eingeschaltet ist. Es können auch alle Ebenen abgeschaltet sein (Enable=0)  **Enable** erlaubt es, alle Layer auszuschalten  Enable = 0 🡪 alle Lagen aus, egal was in E2..E0 steht  Enable = 1 🡪 die durch E2..E0 angegebene Lage wird angesteuert  **OE** steuert die Ausgänge aller Latches zusammen.  OE = 1 🡪 Alle LEDs aus, egal was im Latch steht  OE = 0 🡪 Alle LEDs folgen den Einstellungen im Latch Der Eingang OE des Multiplexers kann auch einfach auf Low gelegt werden. |  |

Software-Layer

|  |  |
| --- | --- |
| LED\_Cube.c | Das Hauptprogramm.  Initialisiert den Treiber, Timer usw und startet den Musterablauf.  Nimmt Kommandos über die serielle Schnittstelle entgegen und wertet diese aus. |
| effect.c/h | Definiert eine Kontrollstruktur für den abaluf verschiedener Effekte und implementiert auch verschiedene Effekte.  Hier kann nach eigenem Gusto erweitert werden. |
| cube.c/h | Definiert die Datenstruktur für den LED-Würfel  uint8\_t volatile cube[CUBE\_MAX\_LAYERS][CUBE\_MAX\_COLS];  Hier finden siech auch Funktionen, welche verschiedene Operationen auf d erdatenstruktur ausführen. Die Funktionen arbeiten nur auf dem Array. Die Ausgabe übernimmt die Funktion drv\_drive() |
| drv\_573.c/h | Stellt die Funktionen für die ansteuerung der 8x8 D-Latches sowie der High-Side Treiber für die 8 Ebenen zur Verfügung  Die Funktion void drv\_drive() muss 1x pro Millisekunde aufgerufen werden. Sie übernimmt die eigentliche Multiplex-Ansteuerung des Cube.    Mit Hilfe der Funktion  void drv\_Register\_Callback( drv\_callback\_t cbf )  kann eine Callback-Funktion registriert werden. Diese kann die folgenden Ereignisse zurückgeben:  CB\_SYNC Es wird mit der Ausgabe des Layer 0 begonnen  CB\_DRIVE\_ENTER Der Treiber, der 1x je ms ausgeführt werdne muss, wurde betreten  CB\_DRIVE\_LEAVE Der Treiber wurde verlassen  CB\_NEW\_LAYER Tritt nicht auf  Das Programm LED\_Cube.c definiert eine solche Callback-Funktion  void cb\_syncFromDriver( TCB\_DRV573 cb )  Sie wird am Beginn von main() registriert.  Das Hauptprogramm sorgt auch für den regelmässigen aufruf von drv\_drive() |
|  |  |

## Testsignale

Über eine Callback-Funktion werden verschiedene Zustände an das Hauptprogramm gemeldet.

## Die Adressierung der einzelnen LEDs als Voxel

Im Gegensatz zu einem Pixel[[1]](#footnote-1), das typischerweise in einer Ebene mit 2 Koordinaten x,y adressiert wird, hat ein Voxel[[2]](#footnote-2) als Punkt im Raum drei Koordinaten x,y, und z. Da unser LEC-Cube dreidimensional ist, bilden die einzelnen LEDs also je ein Voxel.

Z

Y

X

# Testprogramm

## Allgemein

## Testschritte

Eine LED in einem Latch umlaufen lassen (Der Layer wird über einen anderen Befehl bestimmt)

1. <https://de.wikipedia.org/wiki/Pixel> [↑](#footnote-ref-1)
2. <https://de.wikipedia.org/wiki/Voxel> [↑](#footnote-ref-2)