# Projektarbeit

# Laufschriftpanel

## - 5 X 7 Dot Matrix –

2009

Sebastian Lintermann
Berufsbildende Schule Bingen
Informationselektroniker
Photo Porst

## Inhaltsverzeichnis

## Vorwort

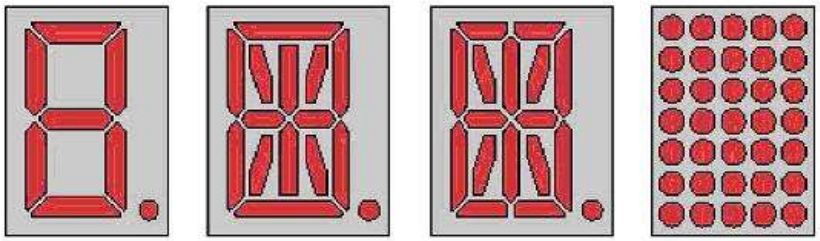
Schon zu Beginn des Jahres machte ich mir Gedanken darüber, welches Thema ich als Projektarbeit zur Gesellenprüfung wählen sollte. Da ich kein rein praktisches Thema ausarbeiten wollte und ich mich insgeheim sowieso für die LED Technik sowie die Entstehung visueller Effekte interessiere, entschied ich mich das Thema „Laufschrift“ als Projektarbeit zu wählen.

Mein Ziel war es im eigentlichen Sinne, eine vom PC aus über USB gesteuerte LED Anzeige zu entwickeln, mit der Schriftzüge und sogar definierbare Effekte wie „blinken“ und ähnliche dargestellt werden können. Leider bemerkte ich schnell, dass dieses Ziel für einen Anfänger ohne große Vorkenntnisse der Softwareentwicklung und praktischer Erfahrung in dem zur Verfügung stehenden Zeitrahmen schwer zu realisieren war. Also beschränkte ich mich auf ein wesentliches Ziel:

**Das grundlegende Verständnis sowie die Funktion und Realisierung einer steuerbaren Laufschrift.**

## Einführung

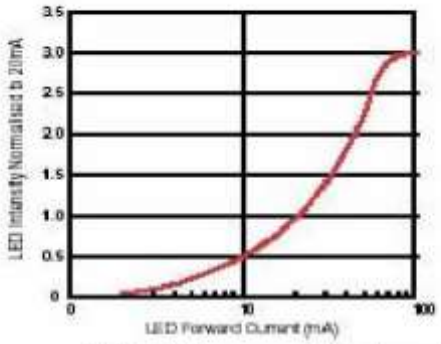
LED Anzeigen sind überall anzutreffen und werden überall dort eingesetzt wo, es gilt, auf etwas aufmerksam zu machen bzw. Informationen anzuzeigen. Dies wäre zum Beispiel als Werbezweck oder Sicherheitshinweis sowie zur reinen Unterhaltung.
Hier gibt es mittlerweile sehr große Anzeigen mit denen sich komplexe Grafiken in allen Farben darstellen lassen.
Nicht zu vergessen sind neuentwickelte RGB Leinwände wie sie in Fernsehstudios und Konzerten eingesetzt werden und eine enorme Energieersparnis im Vergleich zu herkömmlichen Leuchtmitteln, neue Möglichkeiten im Bereich visueller Effekte sowie brillantere Farben mit sich bringen.
Die einfachste Art einer LED Anzeige ist wohl die Siebensegmentanzeige. Mit ihr lassen sich aber nur Zahlen darstellen.



## Die Idee

### Wie steuert man 665 Leuchtdioden an?

Der einfachste Weg, Mehrfach - LED - Anzeigen wie Anzeigensegmente anzusteuern ist, jede einzelne LED über einen Vorwiderstand oder mit einer Stromquelle einzeln zu treiben. Man spricht von statischer Ansteuerung, weil der Diodenstrom kontinuierlich fließt. Diese Methode bietet sich zum Ansteuern von wenigen Leuchtdioden an. Eine kritische Grenze liegt im Bereich von zweistelligen Siebensegment-Anzeigen. LEDs mit hohem Wirkungsgrad erreichen eine gute Helligkeit bereits bei einem Strom von etwa 2 mA und können direkt von einem Mikrocontroller angesteuert werden. Sobald viele Segmente angesteuert werden müssen, erfordert die statische Ansteuerung eine unökonomisch hohe Anzahl von Treiberausgängen, nämlich einen pro LED!



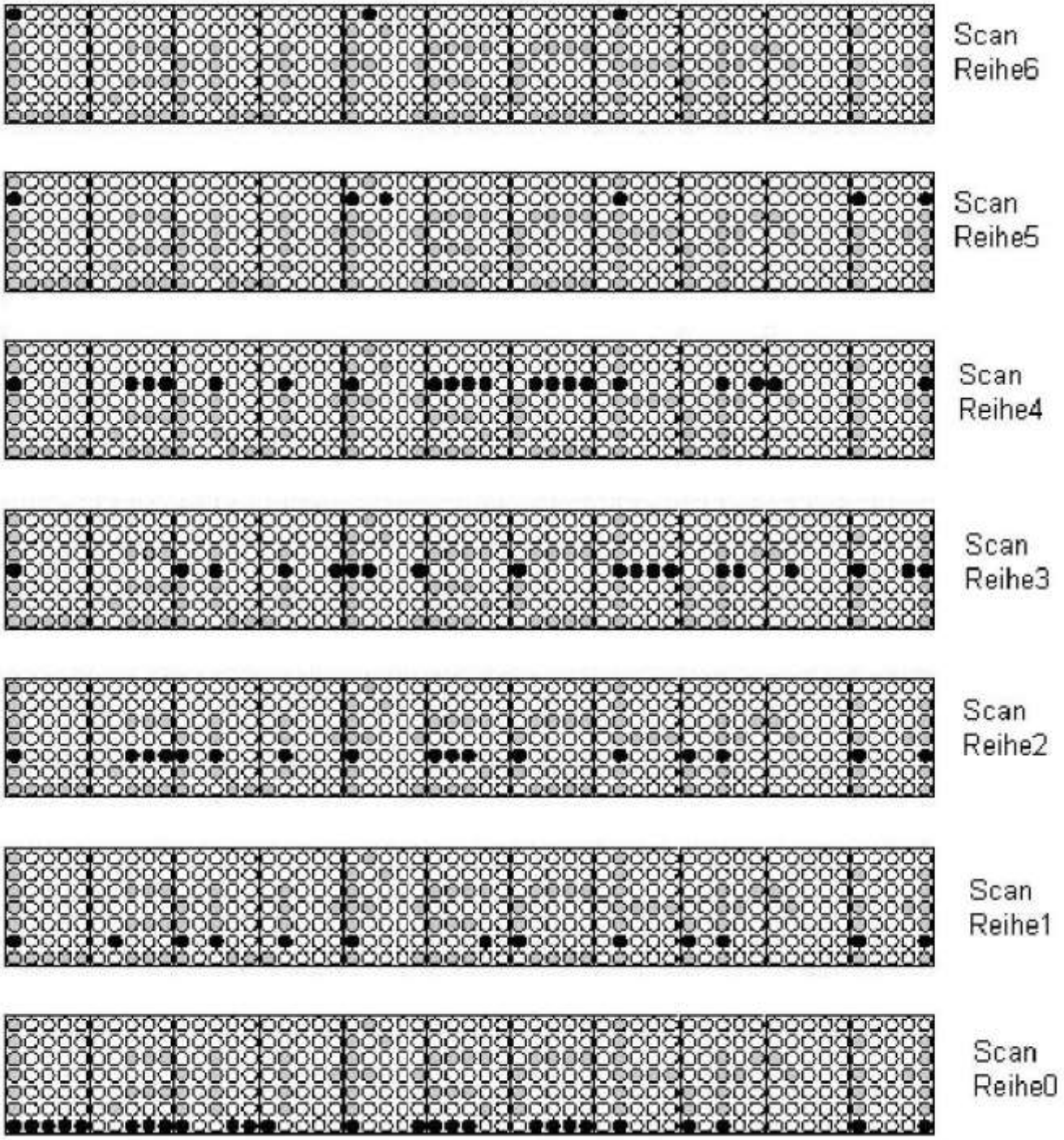

**Mit höherem Diodenstrom erhöht sich auch die Lichtintensität der LED. Bei hohen Strömen wird eine Sättigung erreicht.**

Eine gemultiplexte oder gepulste Ansteuerung reduziert die Anzahl von Anschlüssen, indem nur eine geringe Anzahl von Segmenten, üblicherweise nur genau eines, mittels Auftastimpulsen angesteuert werden. Das Auftasten geschieht bei einer Wiederholrate (Bekannt von Bildröhre 50 / 100 Hz Technik), die das menschliche Auge bereits als kontinuierliche Ausleuchtung wahrnimmt.
Die LEDs müssen jedoch mit einem höheren Strom betrieben werden, um das reduzierte Tastverhältnis zu kompensieren. Den Vorteil, dass das menschliche Auge teilweise als Integrator und teilweise als Spitzenwert lesendes Photometer arbeitet, macht sich die gepulste Ansteuerung zunutze. Als Ergebnis nimmt das Auge schnell gepulstes Licht irgendwo zwischen maximaler und durchschnittlicher Helligkeit auf. Ein intensiver Puls mit niedrigem Tastverhältnis erscheint heller als ein dem Durchschnitt des gepulsten Signals entsprechendes Gleichstromsignal.
Daher ist die Verbesserung der Displayintensität bei gemultiplexter Ansteuerung ein Vorteil bei einer gegebenen, durchschnittlichen Leistungsaufnahme.

## Die Idee

### Allgemeine Funktionserklärung

Die Anzeige wird zeilenweise gemultiplext. Es wird immer eine komplette Zeile angezeigt, bis die nächste folgt.



## Die Idee

### Allgemeine Funktionserklärung

Um die Illusion eines komplett dargestellten Schriftzuges zu erhalten braucht es zwei Anforderungen. Erstens, eine Übersteuerung der Leuchtdioden mit einem größeren Strom als im Normalbetrieb. Macht man dies nicht, kann es sein, dass die LEDs nur schwach leuchten. Dieser Dimmeffekt tritt deshalb auf, da jede Reihe nur 1/7 der Zeit leuchtet.
Zweitens müssen die Zeilen oft genug geupdatet werden, sodass jede Zeile ungefähr 35 - 50 mal pro Sekunde angezeigt wird. Dies reduziert ein mögliches Flackern des Displays. Mithilfe dieser Informationen kann bestimmt werden, mit welcher Frequenz der Zeilenwechsel stattzufinden hat:

35 Hz x 7 Zeilen = 245 Hz

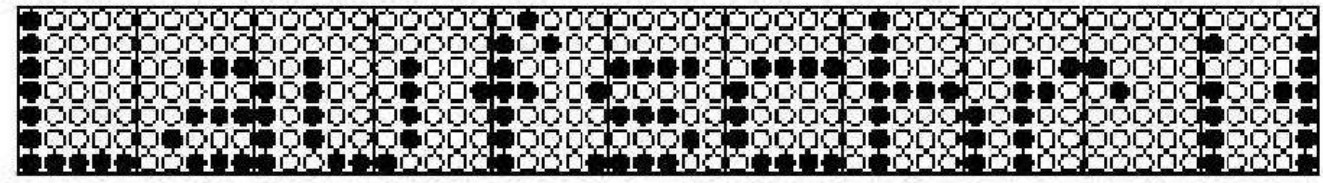
mit Hilfe der Frequenzformel ( $F = \frac{1}{T}$ ) errechnen wir eine Zeitperiode: ( $T = \frac{1}{245Hz}$ ) = 4,08 mS

Anders gesagt bedeutet dies, dass jede Zeile 4Millisekunden angezeigt wird, bevor die Nächste dargestellt wird. Da wir jedoch über höhere Kapazitäten des Atmega 16 verfügen und um eine Flimmerfreiheit zu gewährleisten werden diese mindestwerte überschritten:

500 Hz x 7 Zeilen = 3500 Hz

mit Hilfe der Frequenzformel ( $F = \frac{1}{T}$ ) errechnen wir eine Zeitperiode: ( $T = \frac{1}{3500Hz}$ ) = 285,7 µS

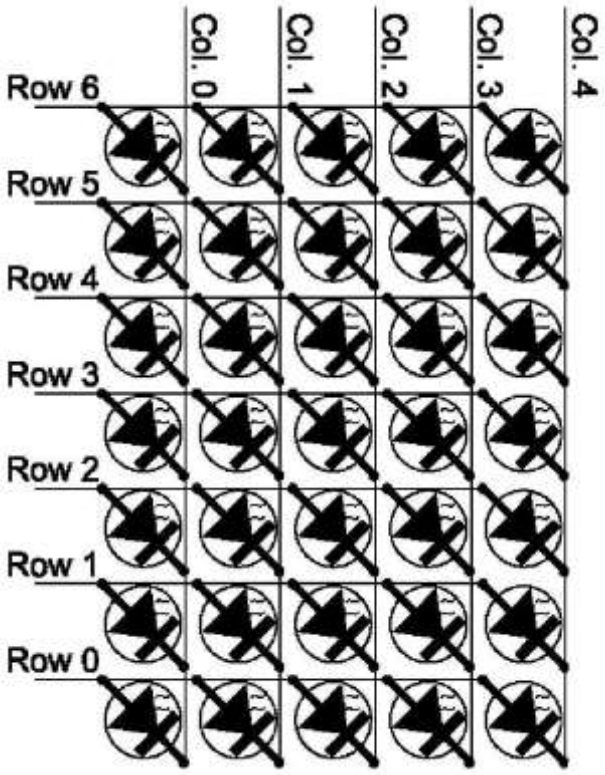
Wir erhalten also durch das Multiplex Verfahren eine stehende, flimmerfreie Anzeige.



## Anzeigeeinheit

### Verschaltung der Leuchtdioden

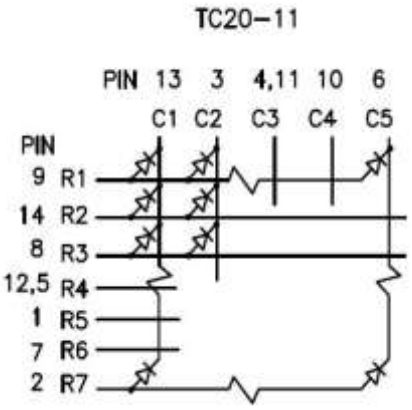
Um die Ansteuerung der LEDs zu erleichtern, wurden alle Anoden einer Reihe zu einem Pin zusammengefasst. Auch die Kathoden der Leuchtdioden jeder einzelnen Spalte sind miteinander verbunden. Dies hat den Vorteil, dass man mit relativ wenig Steuerleitungen eine große Anzahl von LEDs steuern kann.




Die Anzeige hat eine Höhe von 7 Leuchtdioden und eine Länge von 95 Leuchtdioden. Dies ergibt eine Gesamtanzahl von 665 LEDs, die angesteuert werden müssen. Um eine solche Menge an LEDs kostengünstig und effektiv (nicht jede LED eines Bautyps ist exakt gleich) zu realisieren, wurden 19 „Sieben - Segment - Anzeigen“, sogenannte „5 X 7 Dot - Matrix - Anzeigen“ verwendet.

## Anzeigeeinheit

### Beschreibung der LED Bausteine




Hersteller: Kingbright

Modell: TC20-11HWA
Common Cathode
5 X 7 Dot Matrix Display
5 mm Punktgröße

Farbe: rot
$\lambda$ = 625nm

Abmessungen: 38 mm (B) X 53,5 mm (H) X 8,5 mm (T)
.

### Berechnung der Vorwiderstände

| Parameter | Bright Red | Units |
|---|---|---|
| Power dissipation | 120 | mW |
| DC Forward Current | 25 | mA |
| Peak Forward Current [1] | 150 | mA |
| Reverse Voltage | 5 | V |
| Operating/Storage Temperature | -40°C To +85°C | |
| Lead Soldering Temperature [2] | 260 °C For 5 Seconds | |

Der Strom der durch die Leuchtdioden fließt wird mit den Widerständen R1 – R95 begrenzt.
Aus den Herstellerangaben geht hervor, dass die LEDs zur optimalen Lichtausbeute
einen Strom von 150 mA bei 5V benötigen.

$$R = \frac{U_{Led}}{I_{Led}} = \frac{5V}{150mA} = 33\ \Omega$$

## Treibereinheit

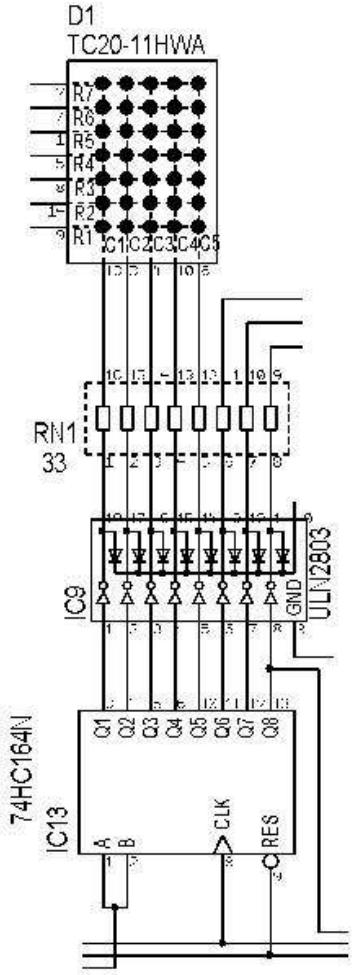
### Allgemeine Beschreibung

Die Treibereinheit der Anzeige übernimmt die Aufgabe, die Anzeigen anzusteuern.
Hauptbestandteile der Treiberschaltungen sind:

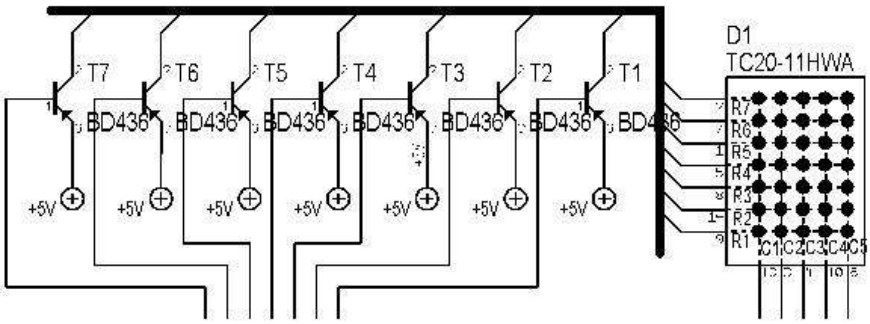
- Leistungstransistoren, mit denen die Ansteuerung der Zeilen durchgeführt wird

- Schieberegister, mit ihnen werden die Spalten und Zeilentransistoren angesteuert

### Schaltplan Treiberansteuerung einer Matrix




### Schaltplan Treiberansteuerung der Zeilentransistoren



## Treibereinheit

### Transistoren


Die sieben Zeilen der Anzeige werden mithilfe von 7 PNP - Leistungstransistoren gesteuert, sie haben die Aufgabe die Anoden, welche in jeder Reihe miteinander verbunden sind, frei zu schalten.
Die Transistoren schalten die positive Betriebsspannung $U_b$ an die Zeilen der LED - Matrix.
Pro Zeile leuchten maximal 95 LEDs. Im Impulsbetrieb werden die LEDs mit 25 mA Strom betrieben. Es ergibt sich der folgende Strom, den die Transistoren aushalten müssen:

25 mA • 95 LEDs = 2,4 A

Die wichtigsten Daten des Transistors sind:

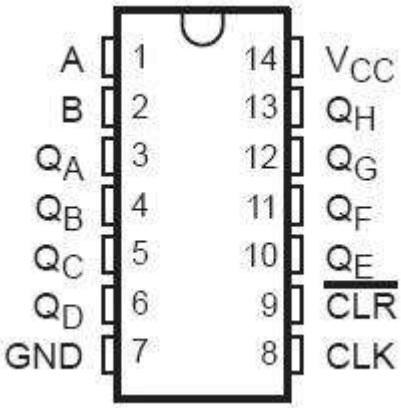
| Symbol | Parameter | Value | Units |
|---|---|---|---|
| $V_{CBO}$ | Collector-Base Voltage : BD434 | - 22 | V |
| $V_{CES}$ | Collector-Emitter Voltage : BD434 | - 22 | V |
| $V_{CEO}$ | Collector-Emitter Voltage : BD434 | - 22 | V |
| $V_{EBO}$ | Emitter-Base Voltage | - 5 | V |
| $I_C$ | Collector Current (DC) | - 4 | A |
| $I_{CP}$ | *Collector Current (Pulse) | - 7 | A |
| $I_B$ | Base Current | - 1 | A |
| $P_C$ | Collector Dissipation ($T_C$=25°C) | 36 | W |
| $T_J$ | Junction Temperature | 150 | °C |
| $T_{STG}$ | Storage Temperature | - 65 ~ 150 | °C |

## Treibereinheit

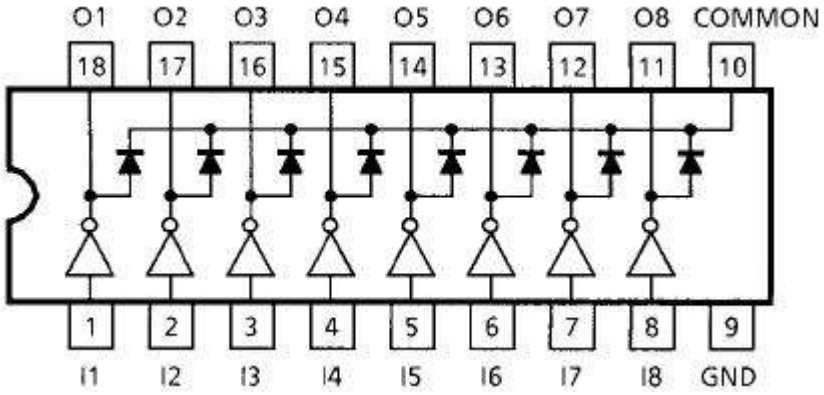
### Schieberegister

Die 95 Spalten werden mit zwölf 8 – Bit Schieberegister gesteuert. Da diese jedoch keinen eigenen Latch haben werden die Ausgänge des Registers, welches zur Steuerung der Transistoren benutzt wird mit einem D – Latch verbunden. Zusätzlich werden noch zwölf Darlingtontreiber benötigt um den fließenden Strom, den jede Zeile benötigt, gegen Masse abzutragen.

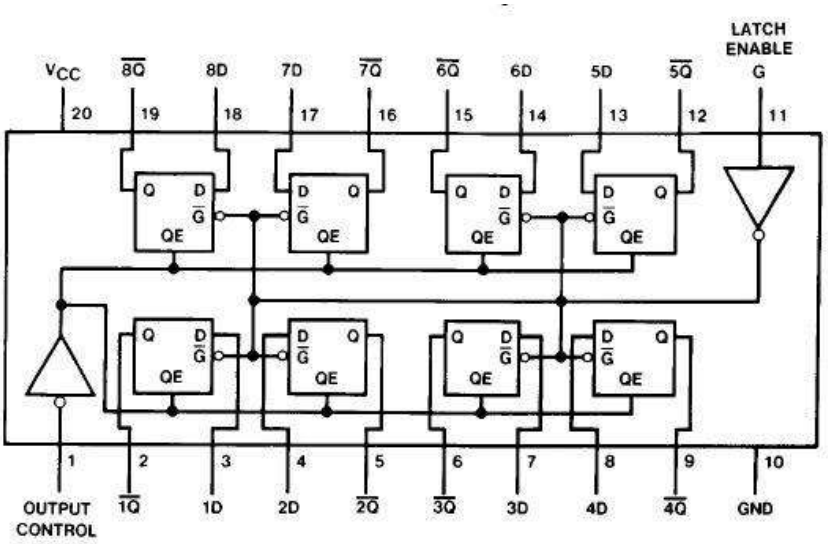
8 - Bit Schieberegister




Darlington Treiber




D - Latch




In der Regel wären jedoch drei 32 - Bit Schieberegister, welche den Latch sowie bipolare NPN Darlingtontreiber schon integriert haben kostengünstiger gewesen. Ebenso hätten Sie das Layout sowie Bauteilkosten erheblich minimiert.

## Treibereinheit

### Schieberegister

Für die Anzeige benötigt man also 12 Schieberegister. Sie wurden mit dem letzten Serial Data Out des ersten Schiebregisters und dem Serial Data In des zweiten Schiebregisters miteinander verbunden.

Ein weiteres Schieberegister steuert die Leistungstransistoren der Zeilen.
Es ist jeweils immer nur ein Transistor aktiviert. Man schiebt deshalb für jenen Transistor, der aktiviert werden soll, eine logische „1“ ins Schieberegister.

Zur Veranschaulichung

| | Spaltensteuerung | Transistorsteuerung |
|---|---|---|
| Zeile 1 aktivieren: | 1 0 1 0 1 0 1 0 1 0 1 | 0 0 0 0 0 0 1 |
| Zeile 2 aktivieren: | 1 0 1 0 1 0 1 0 1 0 1 | 0 0 0 0 0 1 0 |
| Zeile 3 aktivieren: | 1 0 1 0 1 0 1 0 1 0 1 | 0 0 0 0 1 0 0 |

### Zusammenfassung Anzeigeeinheit

Die Anzeigeeinheit benötigt zur Ansteuerung lediglich 10 Steuerleitungen.

- die Clock - Leitung
- die Serial - In Leitung (Datenleitung)
- die Reset - Leitung (Löscht das Schieberegister)
- Sieben Steuerleitungen zum Schalten der Zeilentransistoren

Neben den Steuerleitungen benötigt man eine Spannungsversorgung von mindestens 6 V bis maximal 12 V. Die Spannungsquelle muss dabei einen maximalen Strom von 3 A liefern können, dies deshalb weil jede LED einer Spalte im Impulsbetrieb 25 mA benötigt und im Extremfall 95 LEDs einer Zeile leuchten können.

## Rechnereinheit

### Allgemeine Beschreibung

Die Rechnereinheit der Anzeige dient dazu, die Daten für die Steuerung zu verarbeiten bzw. bereitzustellen.

Als µC wählte ich den im AVR - Kid vorhandenen Atmega16 von Atmel. Ich integrierte das ganze AVR - Kid, da es neben der Rechnereinheit weitere Funktionen wie Taster, LEDs, Piezo, UART sowie eine direkte Stromversorgung bereitstellt.
Ich entschied mich für die Eingabe des Anzeigetextes im Programmspeicher d.h. der Pic muss jedes mal neu programmiert werden, um einen anderen Text anzeigen zu können. Als Schnittstellenbaustein wurde der ISP Programmer verwendet.
Die Taster sah ich zum Wechsel verschiedener Funktionen wie festgelegte Texte oder Uhrzeitanzeige sowie dem Wechsel von statischer Textanzeige und Laufschrift vor.

Als Spannungsregler dient ein externes Netzteil von 5 V.
Die Spannungsversorgung für die Anzeigeeinheit muss getrennt erfolgen da der Atmega Baustein keinen ausreichenden Strom zur Versorgung der LEDs zur Verfügung stellen kann.

### Atmel Atmega 16

40 Pin PDip Gehäuse

High Performance RISC CPU

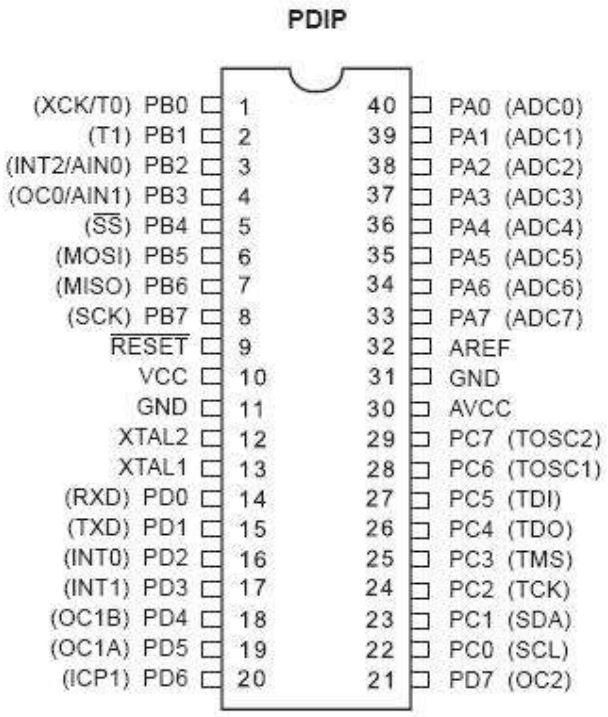
32 Arbeitsregister
32 I/O Ports

Taktfrequenz bis zu 16 MHz

16 KByte Flash
512 Byte EEPROM
1 KByte SRAM

Zwei 8 Bit Timer
Ein 16 Bit Timer

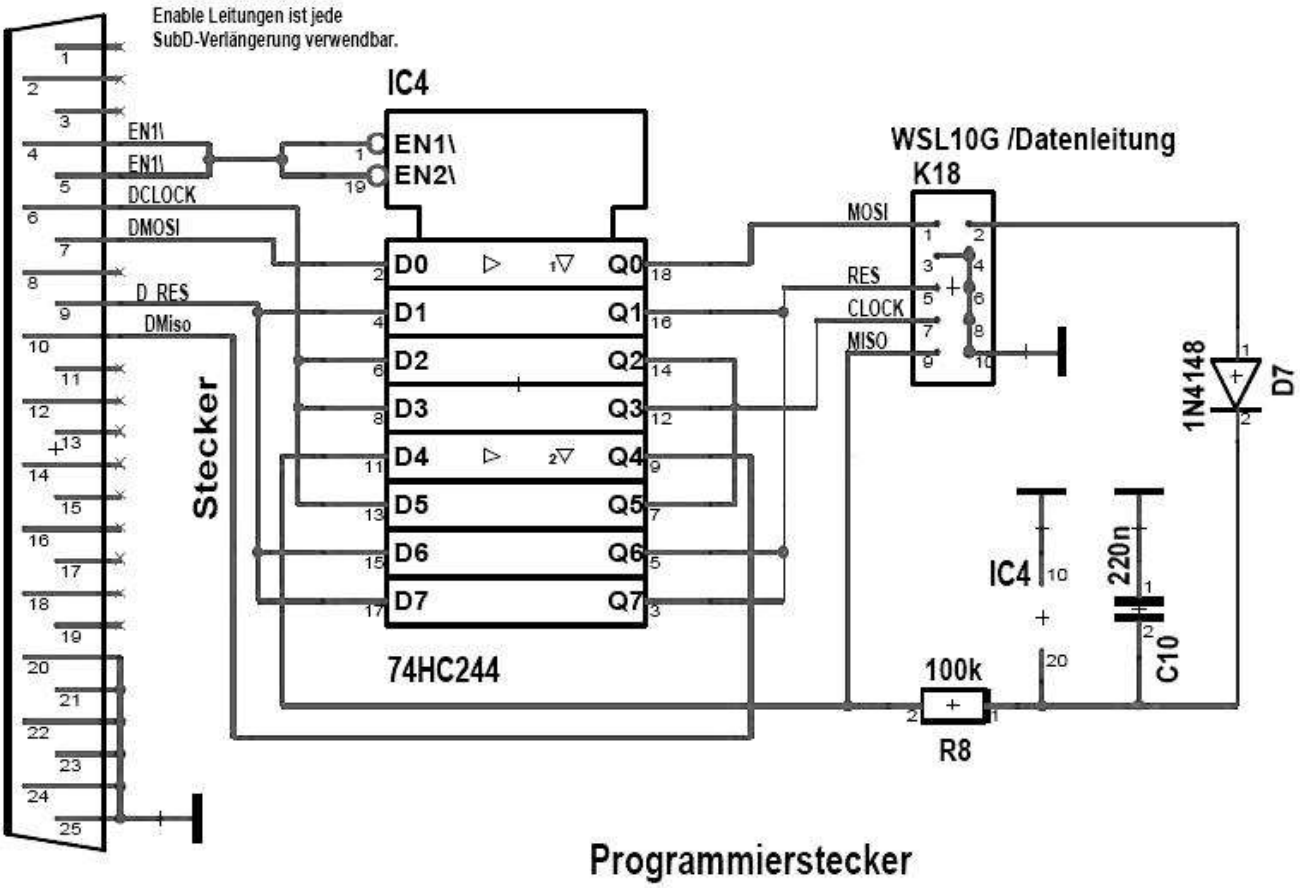
Niedriger Stromverbrauch (< 1 mA im Normalbetrieb)



## Rechnereinheit

### Programmierschnittstelle ISP

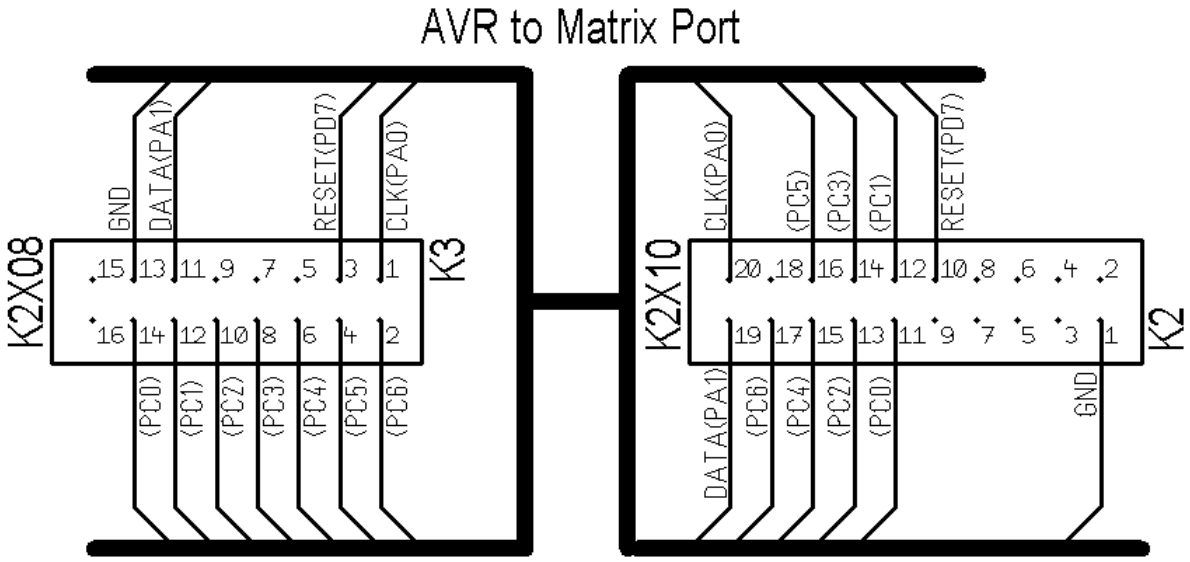
Die Schaltung für den Programmieradapter besteht aus einem $I^2C$ Bus zur Programmierung des Atmega16 und einem 74HC244 IC welches ein nicht invertierender Treiber ist. Zum Anschluss am PC wurde hier nicht wie üblich ein Comport genutzt, sondern eine SUB D25 Buchse zum Anschluss an den Parallelport des Computers. Der RS 232 Anschluss des Comports ist im AVR Kid zwar auch integriert, er erfüllt jedoch eine andere Funktion.



## Rechnereinheit

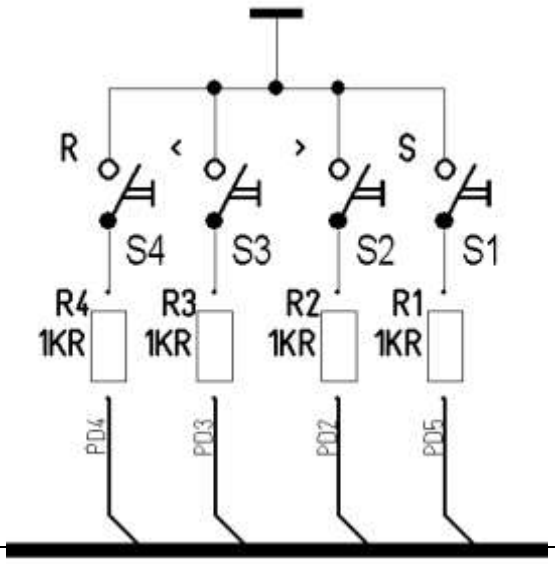
### Bus Schnittstelle Displayport

Das AVR Kid verfügt über einen 20 poligen Port zum Anschließen eines LCD Displays. Da die Ansteuerung dieses Ports jedoch direkt über die I/O Ports des µC abläuft, können wir diesen Port auch für X Beliebige Dinge verwenden. Wir benutzen Ihn hier zur Ansteuerung des Lauflicht Panels über die 10 Steuerleitungen.




### Tastatur

Die Tastatur besteht aus 4 Tasten. Sie sind an Port D des µControllers angeschlossen. Wenn eine Taste gedrückt wird, wird die Taste mit Masse verbunden. Die Taster sind über vier 1kΩ Pull Up Widerstände geschaltet. Der µC verfügt jedoch auch wahlweise über interne Pull Up Widerstände. Besonders hervorzuheben ist der Taster S4, welcher durch einen gesetzten Jumper einen kompletten Reset durchführen lassen kann. Der Taster S1 wurde mit einem Wechsel zwischen statischer und bewegter Anzeige belegt. Nicht zu vergessen ist der Softwaremäßige Debounce (Entprellung), um einen zeitkontinuirlichen Tastendruck zu ermöglichen.



## Software

### Entwicklungssoftware BASCOM

Als Entwicklungssoftware verwendete ich die BASCOM AVR der Firma MCS Electronics. Diese kann kostenlos unter der Internetseite www.mcselec.com heruntergeladen werden.
Das Programm vereint einen Editor, einen Assembler, einen Simulator sowie einen Pic Programmer und einige andere Features. So kann der Quellcode, welcher auf der Programmiersprache BASIC fundiert direkt auf den Chip gebrannt werden. Zur Übertragung nutzen wir den schon genannten ISP Programmier Adapter. Hier kommt auch die RS 232 Schnittstelle zum Einsatz, welche eine Verbindung zwischen Simulator und AVR Kid herstellt. So kann einfach und effektiv aus dem Quelltext heraus das Programm während seines Ablaufs geprüft und gestestet werden.
Zu beachten ist jedoch, dass BASCOM, vor Inbetriebnahme mit dem AVR KID, richtig konfiguriert werden muss. Diese Voreinstellungen können aus der Gesamtbeschreibung des Entwicklers unter www.informationselektroniker.de.tf entnommen werden.

### Softwareentwicklung

Bevor sie diesen Teil lesen möchte ich vorausschicken, dass die Entwicklung der Software für die Anzeige noch nicht vollständig abgeschlossen ist.

Die Anzeige sollte möglichst viele Zeichen darstellen können. Um aber in einer realistischen Grenze zu bleiben wählte ich den ASCII Zeichensatz als guten Kompromiss zwischen Anzahl der Zeichen und benötigtem Speicherplatz. Der ASCII Zeichensatz ist auch deshalb von Vorteil, da sich so das Programm des Pic vereinfachte. Außerdem ist der ASCII Zeichensatz Universell festgelegt, was bedeutet, dass sich die Verarbeitung zwischen Ein- und Ausgabe vereinfacht, da die Codierung eins zu eins übertragen werden kann.

Die Zeichen die dargestellt werden können entsprechen jenen wie sie in der ASCII Tabelle festgelegt sind. Ergo sind es 127 Datensätze wovon jedoch 34 Befehlssätze wie „Space“ oder „Backspace“ wegfallen, da sie ja keine Darstellung bieten. Theoretisch könnten diese Datensätze jedoch auch frei definiert werden, da die gesamte Tabelle im Quellcode abgelegt werden muss.

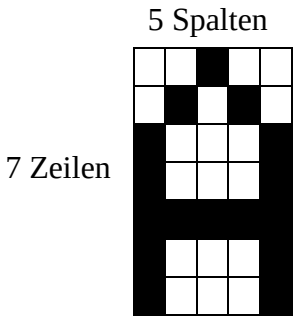
## Software

### ASCII und Darstellung

Im Folgenden ist die Darstellung der Zeichen auf der Matrixanzeige im Vergleich zur ASCII Tabelle zu sehen.

| Decimal | Hex | ASCII | Decimal | Hex | ASCII | Decimal | Hex | ASCII | Decimal | Hex | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | NUL | 32 | 20 | (blank) | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 01 | SOH | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 02 | STX | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 03 | ETX | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 04 | EOT | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 05 | ENQ | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 06 | ACK | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 07 | BEL | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 08 | BS | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 09 | HT | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | 0A | LF | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | 0B | VT | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | 0C | FF | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | 0D | CR | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | 0E | SO | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | 0F | SI | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | DLE | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | DC1 | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | DC2 | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | DC3 | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | DC4 | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | NAK | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | SYN | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | ETB | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | CAN | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | EM | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | SUB | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | ESC | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | FS | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | GS | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | RS | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | US | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | (delete) |

Nehmen wir als Beispiel den Großbuchstaben „A“ mit der Wertigkeit 65. Wenn wir also ein „A“ darstellen lassen wollen, legen wir im Quellcode eine Tabelle an, welche die Wertigkeit sowie die passende Bitcodierung für unsere Anzeige beinhaltet. Da die Anzeige aus sieben Zeilen besteht lag es nah, die Zeilen als 7 Bit Register festzulegen. Da wir eine Matrix aus 5 X 7 Dots verwenden, „zerschneiden“ wir den Buchstaben „A“ in fünf Spalten.



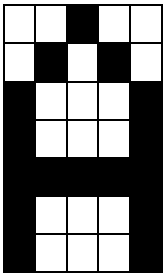
## Software

### ASCII und Darstellung

Da wir die Zeilen als 7 Bit Register festegelegt haben, erhält jede Zeile eine Binäre Wertigkeit.

5 Spalten

Bit 7 ( Wert 64 )
Bit 6 ( Wert 32 )
Bit 5 ( Wert 16 )
Bit 4 ( Wert 8 )
Bit 3 ( Wert 4 )
Bit 2 ( Wert 2 )
Bit 1 ( Wert 1)



Anhand der Wertigkeit der Zeilen, können wir nun für den Buchstaben „A“ fünf Spalten definieren, in denen die Wertigkeiten abgelegt sind. Mit dem Wert definieren wir spaltenweise
welche LED leuchtet ( schwarz dargestellt ) und welche LED aus bleibt ( weiß dargestellt ).
So erhalten wir folgenden Tabellen Eintrag für den Quellcode.

**65 = 31 , 36, 68, 36, 31 = „A“**

| ASCII Wert | | Spalte 1 | Spalte 2 | Spalte 3 | Spalte 4 | Spalte 5 | | Buchstabe |
|---|---|---|---|---|---|---|---|---|
| 65 | = | 31 | 36 | 68 | 36 | 31 | = | A |

Die Spalten definieren sich durch die Addition der Bits ( leuchtenden LEDs )

Für Spalte 1 also 1 + 2 + 4 + 8 + 16 = 31

Wir können also selbst definieren wie unser „A“ auf der Anzeige auszusehen hat, da wir die Wertigkeiten beliebig verändern können. Wichtig ist nur, dass die Darstellung mit dem im ASCII Zeichensatz abgespeicherten Wert übereinstimmt, sonst würden die Buchstaben beim Programmablauf vertauscht werden oder gar nicht erst Angezeigt werden.

## Software

### Aufbau und Struktur

Das Programm lässt sich in zwei Teile gliedern

Hauptroutine

Dieser Teil des Programms führt den Multiplexbetrieb durch. Es steuert die Anzeige und holt sich die Information, wie ein Buchstabe aufgebaut ist aus dem Speicher, in dem sich der Zeichensatz befindet. Dieser Teil arbeitet wie eine Endlosschleife. Etwas umständlich dabei ist, dass die Schrift von „hinten nach vorne“ in das Schieberegister geschrieben werden muss, damit sie wieder lesbar dargestellt wird. Gelöst wird dieses Problem durch einen Reset und eine anschließende Neubeschreibung der Register. Die Register werden also prinzipiell nach jedem Multiplexdurchlauf ( das Zeilenweise schalten der sieben Zeilen ) gelöscht und neu beschrieben. Neben der Information, die die Schrift behandelt übernimmt dieser Teil auch den zeitlichen Zeilenwechsel und die Ansteuerung der Transistoren.

Zeichensatz

Dieser Teil besteht aus einer Tabelle mit ASCII Datensätzen. In jedem Datensatz der Tabelle sind die fünf 7 - Bitwerte für die jeweiligen Spalten eines Buchstabens abgelegt.
( Wir erinnern uns an das obige „A“ und die Errechnung der Wertigkeit )

Jeder Bitwert steht für eine Zeile der Anzeige. Wenn nun im Hauptprogramm ein Buchstabe eingegeben wird, sucht das Programm den ASCII Wert in der Tabelle und gibt die jeweiligen Bitfolgen auf die Schieberegister aus.

## Software

### Programmablauf

Das Hauptprogramm steuert das Multiplexverfahren. Es wird durch „Sub Graf“ aufgerufen.

Sub Graf

| | |
|---|---|
| Call 7bitgeber | Unterprogramm zum schalten der sieben Zeilen |
| Yspalt = 0 | Alle Spalten ausschalten |
| Call Xreset | Schieberegister löschen |
| Call Xfont | Unterprogramm zum Auswählen, Decodieren und Füllen der Schieberegister mit den jeweiligen Spalten Bits |
| Yspalt = 7bit | Y - Spalte wieder einschalten |
| Waitms 1 | Beschreibt die Leuchtdauer, da nach jedem Takt das Schieberegister neu beschrieben werden muss und in der Zeit die LEDs aus sind |

End Sub

Der „7bitgeber“ wird über einen „Sub“ aus der Hauptroutine aufgerufen. Dieses Unterprogramm steuert die sieben Zeilen in dem es die Variable „7bit“ immer um eine Wertigkeit erhöht.

Sub 7bitgeber

| | |
|---|---|
| Rotate 7bit , Left , 1 | „7bit“ ist eine als Byte definierte Variable |
| If 7bit < 1 Then 7bit = 1 | Einmalige Funktion, damit der Zähler bei 1 statt 0 beginnt |
| If 7bit > 64 Then 7bit = 1 | Funktion zum Rücksetzen des Zählers bei 64 statt 128 |

End Sub

Im Unterprogramm „Xfont“ wird der Text aus dem EEProm gelesen und die Pixelzuordnung mit Hilfe von „Xdata“ dekodiert um sie anschließend in den Schieberegistern abzulegen.

Sub Xfont

| | |
|---|---|
| Lsze = Lsz + 94 | Definiert die Anzahl der zu lesenden Spalten im sichtbaren Bereich des Displays |
| For Xi1 = Lsz To Lsze | Definiert den sichtbaren Bereich ( 95 LEDs )selbst |
| Readeeprom Xb , Xi1 | Binärdaten aus dem EEProm auslesen. |
| Call Xdata | Zum Unterprogramm „Xdata“ springen um Binärcode aus „Xb“ zu decodieren |
| Next Xi1 | Springe zu „Xi1“ |

End Sub

## Software

### Programmablauf

„Xdata“ errechnet durch eine logische UND Verknüpfung von „Xb“ und „7bit“ die darzustellenden Pixel ( High / Low oder LED an / LED aus ) einer Spalte. Diese Funktion ist in ihrer Arbeitsweise grundlegend einfach und ideal, da sie große, komplexe Rechenzyklen verhindert und so effektiv die Daten filtert.

```
Sub Xdata

    Clk = 0                     Schieberegister auf Low setzen
    Db = Xb And 7bit            Pixel entsprechend der Wertigkeit filtern
    If Db >= 1 Then             Wenn eine Wertigkeit vorhanden ist High,
    Dat = 1                     wenn nicht Low am Schieberegister setzen
    Else Dat = 0
    End If

    Clk = 1                     Übernahme ins Schieberegister

End Sub
```

Bei Aufruf von „Xreset“ wird das Schieberegister vollständig gelöscht.

```
Sub Xreset

    Res = 0 : Waitus 1 : Res = 1        Es wird eine µs vor dem Neustart gewartet

End Sub
```

## Software

### Programmablauf

Der Fontgenerator holt sich entsprechend dem Text das Buchstabenabbild aus dem Font ( Data - Restore - Anweisung ) und legt die Zuordnung der LEDs zeilenweise im EEProm ab. Der Darzustellende Text wird also einmalig im EEProm abgespeichert.

Sub Fontgenerator

| | |
|---|---|
| Xgl = Len(text) | Gesamtlänge X wird durch die Länge des Textes definiert |
| Xip = 1 | Interpolationszähler |
| For Xi1 = 1 To Xgl | Interpolationszähler 1 wird definiert von eins zu Xgl |
| Xtxt = Mid(text , Xi1 , 1) | Definition von „Xtxt“ über den eigentlichen Text |
| Xb = Asc(xtxt) | Den ASCII Wert eines Zeichens ermitteln ( Beispiel "A" = 65 ) |
| Xb = Xb - 29 | Von dem ermittelten Wert 29 abzählen, da wie oben schon genannt die ersten 30 Datensätze nicht darstellbare Tastaturbefehle beschreiben. |
| Xp = Xb * 5 | Definiert die Breite eines Buchstabens, also 5 Datensätze und Durchläuft die Wertigkeiten der ASCII Tabelle bis der entsprechende Buchstabe gefunden ist. |
| Xp = Xp - 5 | Springt zur ersten Wertigkeit des Buchstabens zurück, damit nach der For Next Schleife die für den Buchstaben interessanten Daten einzeln ausgelesen und sofort zur Ausgabe gesendet werden können. |
| Restore Dta1 | |
| For Xi2 = 1 To Xp | Interpolationszähler 2 wird mit dem eben ermittelten Byte gefüttert |
| Read Xb | Lese alle Byte ( also fünf Stück ) des Buchstabens |
| Next Xi2 | |
| Read Xb : Xpreb = Xb | Hier wird das erstes Byte ausgegeben ( linkes Pixel ) |
| Writeeeprom Xb , Xip : Incr Xip | Beschreibe das EEProm mit der Information |
| Read Xb | Wir verfahren genauso mit dem zweiten und dem dritten |
| If Xpreb > 0 Then | Byte |
| Xpreb = Xb : Xi2 = 0 | |
| End If | |
| Writeeeprom Xb , Xip : Incr Xip | |
| If Xb > 0 Then | Diese Abfrage bricht die Ausgabe ab, wenn nach den drei |
| Xb = 0 | Durchläufen keine Daten mehr folgen. |
| Writeeeprom Xb , Xip : Incr Xip | |
| End If | |
| Naext: | |
| Next Xi1 | Beginne erneut mit dem nächsten Buchstaben |

End Sub

**Software**

**Zusammenfassung**

Die Zeile X wird durch 95 CLK Impulse bestimmt
Die Spalte Y wird durch 7 Bits bestimmt

| | |
|---|---|
| „Graf“ | steuert das Multiplexverfahren |
| „7bitgeber“ | Erstellt eine globale Variable 7Bit für die Zeilen und die Spaltenerrechnung |
| „Xreset“ | löscht alle Schieberegister |
| „Xfont“ | ermittelt entsprechendes darzustellendes Zeichen |
| „Xdata“ | ermittelt welche LED an bzw. aus ist |
| „Fontgenerator“ | legt den kompletten Text formatiert in EEProm und fügt nach jedem Zeichenende ein Leerzeichen ein, da zum Beispiel ein „Komma“ nur 1 von 5 Pixel breit ist |

Diese Programmabläufe sind mit Absicht in Unterprogramme ( Subs ) unterteilt, da sie

a) Eine bessere Übersicht des gesamten Quelltextes mit sich ziehen und
b) Das beliebige Aufrufen aus einem anderen Unterprogramm ermöglichen

## Software

### Quelltext

```
'#################################################################################
'
'Projektarbeit:         Gesellenstück
'
'Projektname:           7 X 5 DOT MATRIX LED Panel mit 19 Segmenten
'
'Pixel Gesamt:          665
'
'Pixel Reihe:           95
'
'Ansteuerung:           Zeilenweise Multiplex
'
'                       Flimmerfreiheit gewährleistet ab 700Hz
'
'                       7 Zeilen X 100Hz = 700Hz
'
'CPU:                   Atmega16
'
'Speicher:              1 kByte SRAM
'                     512  Byte EEPROM
'
'Version:               1.2
'
'Datum:                 25.06.09
'
'Autor:                 Sebastian Lintermann
'
'#################################################################################

    $regfile = "m16def.dat"
    $crystal = 8000000
    $baud = 38400

    Config Timer1 = Timer , Prescale = 8
    On Timer1 Sprungtim1
    Enable Interrupts

'--------------------------------------------------------------------------------
```

## Software

### Quelltext

```
    Ddrc = &B01111111               'Spalten als Ausgang
    Portc = &B0000000               'Pullup ein

    Ddra = &B00000011               'Schaltsignal clk(0) und Datensignal(1)
    Porta = &B00000010              'Pullup ein

    Ddrd = &B10000000               'Reset als Ausgang / Tasten als Eingang
    Portd = &B10111100              'Pullup ein

    Ddrb = &B11110000                'LED Ausgang
    Portb = &B11110000               'Pullup ein

'--------------------------------------------------------------------------

    S1 Alias Pind.5                 'Taste [S]
    S2 Alias Pind.2                 'Taste [2]
    S3 Alias Pind.3                 'Taste [3]
    S4 Alias Pind.4                 'Taste [R]
    V7 Alias Portb.4                'LED links
    V6 Alias Portb.5                'LED halblinks
    V5 Alias Portb.6                'LED halbrechts
    V4 Alias Portb.7                'LED rechts
    Clk Alias Porta.0               'Schaltsignal
    Dat Alias Porta.1               'Dateneingang Schieberegister
    Res Alias Portd.7               'RESET setzt Schieberegister auf 0
    Yspalt Alias Portc              'Spaltenkanal wird Binär angesteuert.

'--------------------------------------------------------------------------

    Declare Sub Graf                'Multiplexansteuerung
    Declare Sub 7bitgeber           'globale Variable 7Bit
    Declare Sub Xreset              'Resetet Schieberegister
    Declare Sub Xfont               'ermittelt Zeichen aus Font
    Declare Sub Xdata               'ermittelt aus XB Data
    Declare Sub Fontgenerator       'legt Text formatiert in EE²
```

## Software

### Quelltext

```
'------------------------------Dimensionierung---------------------------------

    Dim I As Integer
    Dim W As Word
    Dim Text As String * 350          'Stringvariablen wird die Länge
    Dim Xgl As Word                   'Txt-Länge in Sub XFont
    Dim Xi1 As Word                   '1.For-Next in Sub XFont
    Dim Xi2 As Word                   '2.ForNext in Sub XFont
    Dim Xp As Word                    'Interpolation in Sub XFont
    Dim Xb As Byte                    'Restore/Read Inhalt in Sub XFont
    Dim Xpreb As Byte                 'vorhergehender Read in Sub XFont
    Dim Xtxt As String * 1            'gefilteter Text in Sub XFont
    Dim Xip As Word                   'Interpolationszähler in Sub XFont
    Dim Lsz As Word                   'Laufschriftzähler
    Dim Lsze As Word                  'Laufschriftzähler-Ende
    Dim 7bit As Byte                  '7Bit Zähler von 1 bis 64 (Binär)
    Dim Db As Byte                    'Datenbit fürs S-Reg. in Sub XData

'-------------------------------TEXT EINGABE-----------------------------------

    Text = "0123456789"

    Xgl = Len(text)
    Call Fontgenerator
    Lsz = 1

'-------------------------------Hauptroutine-----------------------------------

Do

    Debounce S1 , 0 , Tast1 , Sub      'S1 ["S"]
    Debounce S2 , 0 , Tast2 , Sub      'S2 ["2"]
    Debounce S3 , 0 , Tast3 , Sub      'S3 ["3"]
    Debounce S4 , 0 , Tast4 , Sub      'S4 ["R"]

    Call Graf

Loop

'-----------------------------Tiemerroutine------------------------------------

Sprungtim1:

    Incr Lsz                           'Wenn Laufschriftzähler = Zählerende,
    If Lsz = Lsze Then Lsz = 1         'dann Beginne von vorne

Return
```

## Software

### Quelltext

```
'--------------------------------Taster---------------------------------------

Tast1:

    V7 = 1 : Clk = V7
    If Timsk.toie1 = 0 Then
    Timsk.toie1 = 1
    Else
    Timsk.toie1 = 0
    V4 = 0 : V5 = 0 : V6 = 0 : V7 = 0
    End If

    V7 = 0 : Clk = V7

Return

'-----------------------------------------------------------------------------

Tast2:

    V6 = Not V6

    Call Graf

Return

'-----------------------------------------------------------------------------

Tast3:

    V5 = 0
    Call Graf

Return

'-----------------------------------------------------------------------------

Tast4:

    V4 = 1
    Portd.7 = 0                     'RESET an
    Waitms 200
    Portc = 1                       'Y zurück setzen
    Dat = 1                         'Data auf H
    Portd.7 = 1                     'RESET wieder aus
    V4 = 0
```

Return

**Software**

**Quelltext**

```
'----------------------------Sub Routinen-------------------------------------

'   Aufgabe:        Wird durch Takt aufgerufen und erhöht die Variable 7Bit im Binärcode
'                   immer um eine Wertigkeit von 1 bis 64, also 1-2-4-8-16-32-64
'   Hinweis:        7Bit ist eine globale Variable. Sie hat zwei Aufgaben:
'                1) aus dem Font je nach Wertigkeit das Data zu filtern
'                2) die Y- Achse anzusteuern (PortC).

Sub 7bitgeber

    Rotate 7bit , Left , 1
    If 7bit < 1 Then 7bit = 1
    If 7bit > 64 Then 7bit = 1

End Sub

'--------------------------------------------------------------------------------

'   Aufgabe:        Bei Aufruf von XReset wird das Schieberegister vollständig gelöscht.

Sub Xreset

    Res = 0 : Waitus 1 : Res = 1

End Sub

'--------------------------------------------------------------------------------

'   Aufgabe:        der Binärwert aus XB wird je nach 7Bit (Wertigkeit-Spalte 1-7) ge-
'                   filtert und betreffenden Data (H/L) daraus gemacht und ins
'                   SchiebeRegister geschrieben.

Sub Xdata

    Clk = 0                         'SR Eingang auf Low setzen
    Db = Xb And 7bit                'Pixel entsprechend der Wertigkeit filtern
    If Db >= 1 Then                 'daraus ein H oder L Data machen
    Dat = 1                         'Dateneingang Schieberegister
    Else
    Dat = 0                         'Dateneingang Schieberegister
    End If
    Clk = 1                         'Übernahme ins Schieberegister (Pos.Flanke)

End Sub

'--------------------------------------------------------------------------------
```

## Software

### Quelltext

```
Sub Graf

  Call 7bitgeber                  'Rotierer für Spalte 1-7 Umschaltung
  Yspalt = 0                      'alle Spalten ausschalten
  Call Xreset                     'Schieberegister löschen
  Call Xfont                      'Font auswählen, Decodieren, SR-füllen
  Yspalt = 7bit                   'Y-Spalte einschalten.

  Waitms 1                        'das ist die Leuchtdauer, weil nach jedem
                                  'Takt das Schieberegister neu beschrieben
End Sub                           'werden muss und in der Zeit die LEDs aus sind.

'--------------------------------------------------------------------------

'   Aufgabe:        Text aus EEProm hlen und die Pixelzuordnung dekodieren
'   Hinweis:        zu jedem Read gehört dann die Ablage ins Schieberegister, was von
'                   XData erledigt wird.

Sub Xfont

    Lsze = Lsz + 94               'Laufschriftzähler-Ende für Laufschrift
    For Xi1 = Lsz To Lsze         'Läuft Sichbaren Bereich (95) durch
    Readeeprom Xb , Xi1
    Call Xdata
    Next Xi1

End Sub
```

## Software

### Quelltext

```
'-------------------------------------------------------------------------------

'   Aufgabe:        Der Fontgenerator holt sich entsprechend dem Text das Schriftbild
'                   aus dem Font (Data-Restore- Anweisung) und legt die Zuordnung der
'                   LEDs zeilenmäßig im EEProm ab.
'   Hinweis:        Dieser Vorgang ist nur einmal erforderlich um das Schriftbild im
'                   EEProm zu speichern.

Sub Fontgenerator

    Xgl = Len(text)
    Xip = 1
    For Xi1 = 1 To Xgl
    Xtxt = Mid(text , Xi1 , 1)
    Xb = Asc(xtxt)                  'den ASCII Wert des Zeichens ermitteln (Beispiel "T"=84)

    Select Case Xb                  'ASCII Codewandler wegen Erweitertem ASCII Code

        Case 252 : Xb = 129         'ü
        Case 228 : Xb = 132         'ä
        Case 246 : Xb = 148         'ö
        Case 220 : Xb = 154         'Ü
        Case 196 : Xb = 142         'Ä
        Case 214 : Xb = 153         'Ö

    End Select

    Xb = Xb - 29                    'davon 29 abzählen weil Data Liste erst bei 30 anfängt
    Xp = Xb * 5                     'jedes Datafeld hat  Schritte, die Read durchlaufen muss
    Xp = Xp - 5                     'wir machen aber 5 Schritte weniger, weil die ersten

    Restore Dta1                    'durch diesen Read Befehl leer durchgezogen werden, damit
    For Xi2 = 1 To Xp
    Read Xb                         'nach der For-Next Schleife die uns interessierenden Daten
    Next Xi2                        'einzeln ausgelesen werden und sofort zur Ausgabe kommen.
    Read Xb : Xpreb = Xb            'erstes Byte (li.Pixel v.unten nach oben)
    Writeeeprom Xb , Xip : Incr Xip
    Read Xb                         'zweites Byte (linke Pixel von unten nach oben)
    If Xpreb > 0 Then
    Xpreb = Xb : Xi2 = 0
    End If
    Writeeeprom Xb , Xip : Incr Xip
    Read Xb                         'die ersten drei Bytes werden grundsätzlich ausgegeben.
```

## Software

### Quelltext

```
    If Xi2 = 0 And Xpreb = 0 Then Goto Naext
    Xpreb = Xb
    Writeeeprom Xb , Xip : Incr Xip          'wenn danach nur noch Leerzeichen kommen, wird
    Read Xb                                  'Ausgabe abgebrochen.
    If Xb = 0 And Xpreb = 0 Then Goto Naext
    Xpreb = Xb
    Writeeeprom Xb , Xip : Incr Xip
    Read Xb
    If Xb = 0 And Xpreb = 0 Then Goto Naext
    Xpreb = Xb
    Writeeeprom Xb , Xip : Incr Xip
    If Xb > 0 Then                           'falls kein Leerzeichen vorhanden
    Xb = 0                                   'wird eins noch gesetzt.
    Writeeeprom Xb , Xip : Incr Xip
    End If
    Naext:
    Next Xi1

End Sub
```

## Software

### Quelltext

```
'---------------------------------Die ASCII Tabelle--------------------------

Dta1:                                   'nur Grundlegende Werte definiert

    Data 127 , 127 , 127 , 127 , 127    '030-
    Data 8 , 8 , 8 , 8 , 8              '031-
    Data 0 , 0 , 0 , 0 , 0              '032-
    Data 125 , 0 , 0 , 0 , 0            '033-!
    Data 96 , 0 , 96 , 0 , 0            '034 -"
    Data 20 , 127 , 20 , 127 , 20       '035-#
    Data 58 , 42 , 127 , 42 , 46        '036-$
    Data 3 , 36 , 8 , 18 , 96           '037-%
    Data 54 , 73 , 85 , 34 , 5          '038-&
    Data 96 , 0 , 0 , 0 , 0             '039-'
    Data 28 , 99 , 0 , 0 , 0            '040-(
    Data 99 , 28 , 0 , 0 , 0            '041-)
    Data 32 , 112 , 32 , 0 , 0          '042-*
    Data 8 , 28 , 8 , 0 , 0             '043-+
    Data 3 , 0 , 0 , 0 , 0              '044-,
    Data 8 , 8 , 8 , 0 , 0              '045--
    Data 1 , 0 , 0 , 0 , 0              '046-.
    Data 2 , 4 , 8 , 16 , 32            '047-/
    Data 62 , 65 , 65 , 65 , 62         '048-0
    Data 16 , 32 , 127 , 0 , 0          '049-1
    Data 6 , 73 , 73 , 73 , 48          '050-2
    Data 34 , 65 , 73 , 73 , 54         '051-3
    Data 120 , 8 , 8 , 127 , 0          '052-4
    Data 56 , 73 , 73 , 73 , 6          '053-5
    Data 54 , 73 , 73 , 73 , 6          '054-6
    Data 64 , 64 , 71 , 88 , 96         '055-7
    Data 54 , 73 , 73 , 73 , 54         '056-8
    Data 48 , 73 , 73 , 73 , 54         '057-9
    Data 20 , 0 , 0 , 0 , 0             '058-:
    Data 11 , 0 , 0 , 0 , 0             '059-;
    Data 8 , 20 , 34 , 65 , 0           '060-<
    Data 20 , 20 , 20 , 20 , 0          '061-=
    Data 65 , 34 , 20 , 8 , 0           '062->
    Data 32 , 64 , 77 , 72 , 48         '063-?
    Data 62 , 65 , 93 , 85 , 61         '064-@
    Data 55 , 72 , 72 , 72 , 55         '065-A
    Data 127 , 73 , 73 , 73 , 54        '066-B
    Data 62 , 65 , 65 , 65 , 34         '067-C
    Data 127 , 65 , 65 , 65 , 62        '068-D
    Data 127 , 73 , 73 , 73 , 0         '069-E
    Data 127 , 72 , 72 , 64 , 0         '070-F
    Data 62 , 65 , 73 , 73 , 46         '071-G
    Data 127 , 8 , 8 , 8 , 127          '072-H
```

```
    Data 127 , 0 , 0 , 0 , 0          '073-I
```

**Software**

**Quelltext**

```
    Data 6 , 1 , 1 , 1 , 126          '074-J
    Data 127 , 8 , 20 , 34 , 65       '075-K
    Data 127 , 1 , 1 , 1 , 1          '076-L
    Data 127 , 32 , 16 , 32 , 127     '077-M
    Data 127 , 48 , 8 , 6 , 127       '078-N
    Data 62 , 65 , 65 , 62 , 0        '079-O
    Data 127 , 72 , 72 , 48 , 0       '080-P
    Data 62 , 65 , 69 , 62 , 3        '081-Q
    Data 127 , 72 , 72 , 55 , 0       '082-R
    Data 50 , 73 , 73 , 38 , 0        '083-S
    Data 64 , 64 , 127 , 64 , 64      '084-T
    Data 126 , 1 , 1 , 126 , 0        '085-U
    Data 124 , 2 , 1 , 2 , 124        '086-V
    Data 124 , 3 , 28 , 3 , 124       '087-W
    Data 99 , 20 , 8 , 20 , 99        '088-X
    Data 112 , 8 , 7 , 8 , 112        '089-Y
    Data 67 , 69 , 73 , 81 , 97       '090-Z
    Data 127 , 65 , 0 , 0 , 0         '091-[
    Data 64 , 48 , 8 , 6 , 1          '092-\
    Data 65 , 127 , 0 , 0 , 0         '093-]
    Data 32 , 64 , 32 , 0 , 0         '094-^
    Data 1 , 1 , 1 , 1 , 1            '095-_
    Data 64 , 32 , 0 , 0 , 0          '096-`
    Data 23 , 21 , 21 , 15 , 0        '097-a
    Data 127 , 17 , 17 , 14 , 0       '098-b
    Data 14 , 17 , 17 , 10 , 0        '099-c
    Data 14 , 17 , 17 , 127 , 0       '100-d
    Data 14 , 21 , 21 , 13 , 0        '101-e
    Data 8 , 63 , 72 , 64 , 0         '102-f
    Data 9 , 21 , 21 , 14 , 0         '103-g
    Data 127 , 16 , 16 , 15 , 0       '104-h
    Data 79 , 0 , 0 , 0 , 0           '105-i
    Data 1 , 46 , 0 , 0 , 0           '106-j
    Data 127 , 4 , 10 , 17 , 0        '107-k
    Data 126 , 1 , 1 , 0 , 0          '108-l
    Data 15 , 16 , 15 , 16 , 15       '109-m
    Data 15 , 16 , 16 , 15 , 0        '110-n
    Data 14 , 17 , 17 , 14 , 0        '111-o
    Data 31 , 18 , 18 , 12 , 0        '112-p
    Data 12 , 18 , 18 , 31 , 0        '113-q
    Data 31 , 16 , 16 , 24 , 0        '114-r
    Data 9 , 21 , 21 , 18 , 0         '115-s
    Data 16 , 127 , 16 , 0 , 0        '116-t
    Data 31 , 1 , 1 , 31 , 0          '117-u
    Data 30 , 1 , 1 , 30 , 0          '118-v
    Data 30 , 1 , 30 , 1 , 30         '119-w
    Data 27 , 4 , 4 , 27 , 0          '120-x
```

```
    Data 24 , 7 , 7 , 24 , 0          '121-y
```

**Software**

**Quelltext**

```
    Data 19 , 21 , 21 , 25 , 0        '122-z
    Data 8 , 54 , 65 , 0 , 0          '123-{
    Data 127 , 0 , 0 , 0 , 0          '124-|
    Data 65 , 54 , 8 , 0 , 0          '125-}
    Data 8 , 16 , 8 , 4 , 8           '126-~
    Data 0 , 0 , 0 , 0 , 0            '127- •
    Data 20 , 62 , 85 , 65 , 34       '128-€
    Data 94 , 1 , 1 , 94 , 0          '129- •
    Data 3 , 0 , 0 , 0 , 0            '130-‚
    Data 2 , 1 , 62 , 64 , 32         '131-ƒ
    Data 87 , 21 , 21 , 79 , 0        '132-„
    Data 1 , 0 , 1 , 0 , 1            '133-…
    Data 16 , 16 , 127 , 16 , 16      '134-†
    Data 20 , 20 , 127 , 20 , 20      '135-‡
    Data 32 , 64 , 32 , 0 , 0         '136-ˆ
    Data 3 , 36 , 10 , 16 , 98        '137-‰
    Data 0 , 0 , 0 , 0 , 0            '138-Š
    Data 0 , 0 , 0 , 0 , 0            '139-‹
    Data 0 , 0 , 0 , 0 , 0            '140-Œ
    Data 0 , 0 , 0 , 0 , 0            '141- •
    Data 79 , 20 , 36 , 20 , 79       '142-Ž
    Data 0 , 0 , 0 , 0 , 0            '143- •
    Data 0 , 0 , 0 , 0 , 0            '144- •
    Data 0 , 0 , 0 , 0 , 0            '145-‘
    Data 0 , 0 , 0 , 0 , 0            '146-’
    Data 0 , 0 , 0 , 0 , 0            '147-“
    Data 78 , 17 , 17 , 78 , 0        '148-”
    Data 0 , 0 , 0 , 0 , 0            '149-•
    Data 0 , 0 , 0 , 0 , 0            '150-–
    Data 0 , 0 , 0 , 0 , 0            '151-—
    Data 0 , 0 , 0 , 0 , 0            '152-˜
    Data 94 , 33 , 33 , 94 , 0        '153-™
    Data 94 , 1 , 1 , 94 , 0          '154-š
    Data 0 , 0 , 0 , 0 , 0            '155-›
    Data 0 , 0 , 0 , 0 , 0            '156-œ
    Data 0 , 0 , 0 , 0 , 0            '157- •
    Data 0 , 0 , 0 , 0 , 0            '158-ž
    Data 0 , 0 , 0 , 0 , 0            '159-Ÿ
    Data 0 , 0 , 0 , 0 , 0            '160-
    Data 0 , 0 , 0 , 0 , 0            '161-¡
    Data 0 , 0 , 0 , 0 , 0            '162-¢
    Data 0 , 0 , 0 , 0 , 0            '163-£
    Data 0 , 0 , 0 , 0 , 0            '164-¤
    Data 0 , 0 , 0 , 0 , 0            '165-¥
    Data 24 , 36 , 18 , 36 , 24       '166-¦
    Data 0 , 0 , 0 , 0 , 0            '167-§
```

'--------------------------------------------------------------------------

## Schaltpläne

### ACR / Laufschrift Panel

Alle Schaltpläne wurden mit TARGET 3000 erstellt. Sie werden dem Anhang ohne Seitenangaben beigelegt. Bedauerlicherweise war es mir nicht möglich ein Platinen Layout herzustellen.

## Abschluss

### Spannungsversorgung

Das AVR Kid wurde in das Laufschrift Panel integriert, was im Nachhinein nicht die idealste Lösung ist. Das AVR Kid wird separat zum Laufschrift Panel gespeist sodass zwei externe Netzteile benötigt werden. Man hätte jedoch auch das im AVR Kid integrierte Netzteil zur allgemeinen Spannungsversorgung nutzen können. Die RS 232 Schnittstelle dient in erster Linie lediglich zur Simulation in BASCOM.
Die USB Schnittstelle ist zum einen Vorgesehen, den µC wahlweise auch über USB mit Spannung zu Versorgen und zum anderen eine Schnittstelle zwischen PC als Eingabemedium und µC als Steuereinheit herzustellen.

Daten:

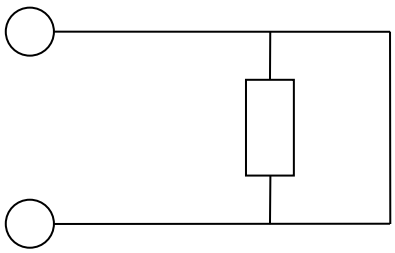
| | | | |
|---|---|---|---|
| AVR KID | 5V | Gleichspannung / Stabilisiert | |
| Laufschrift Panel | 5V | Gleichspannung / Stabilisiert | max. 3A |

### Probleme

Im Großen und Ganzen ergaben sich soweit keine größeren Probleme. Ein wesentlicher Nachteil bei der Programmierung und Erstellung des Quellcodes waren die mir fehlenden Grundkenntnisse. Jedoch konnte ich dank meines Ehrgeizes und meiner Motivation an meinem Ziel festhalten.

### Letzter Eindruck

Ich bin froh, dass ich mich für ein zwar recht komplexes, jedoch für mich persönlich sehr interessantes Projekt entschieden habe. Die letzten Wochen waren in meinem Bestreben mein Ziel zu erreichen in jeglicher Hinsicht sehr effektiv. Ich habe vor meine neu erworbenen Kenntnisse zu festigen und mir weiterhin auf dem Gebiet der Regeltechnik sowie Programmiertechnik Kenntnisse anzueignen.
Zum Abschluss möchte ich noch meinen herzlichsten Dank an Herrn Dieter Bork ausrichten, welcher mir bei der Lösung komplexerer Aufgaben mit Rat und Tat zur Seite stand.



**Wi(e)derstand ist Zwecklos**

## Abschluss

### Quellenverzeichnis

Internet:

www.microcontroller.net

www.roboternetz.org

www.wikipedia.de

www.informationselektroniker.de.tf

www.Datasheeds.com

Literatur:

Basic & C++ Programmierungshandbuch

BASCOM Index

u. w.

---