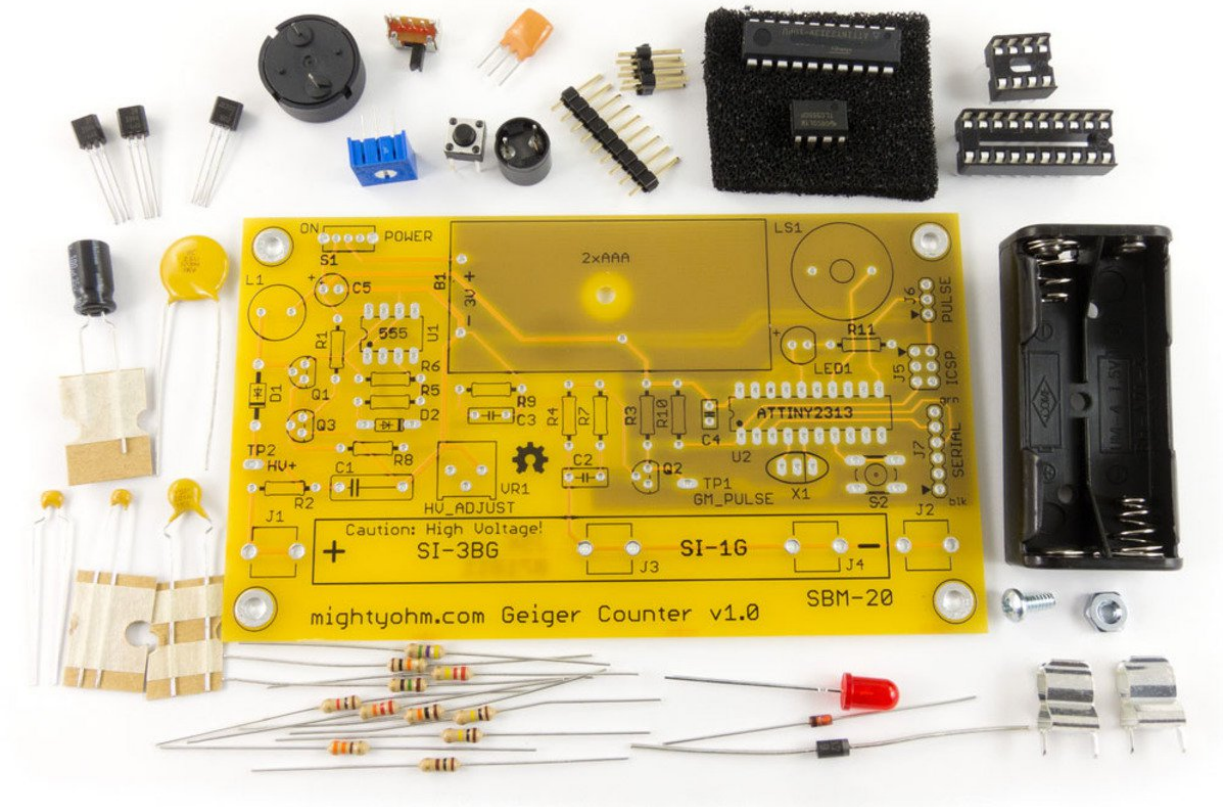


Geigerzähler „myGeigerle“

Version 1.0



Geiger-Müller-Zähler mit Auswertelogik und Anzeigedisplay

Lonsee im Februar 2019

Markus Fulde

Finkenweg 3

D-89173 Lonsee

Telefon +49 (7336) 92 11 89

Fax +49 (7336) 92 10 68

Mobil +49 (160) 84 54 314

Email Markus.Fulde@t-online.de

1 Inhaltsverzeichnis

1	Inhaltsverzeichnis.....	3
2	Abbildungsverzeichnis.....	5
3	Tabellenverzeichnis.....	6
4	Schaltbilderverzeichnis.....	7
5	Softwareverzeichnis.....	8
7	Sicherheitshinweis	10
8	Allgemeines.....	11
8.1	Die Entwicklungsumgebung	11
8.2	Allgemeine Beschreibung und Idee zur Realisierung.....	11
8.3	Leistungsumfang.....	13
8.4	Funktionskomponenten	14
9	Elektronische Grundlagen	15
9.1	Wichtiges und Basics zum MightyOhm Geiger Counter	15
9.1.1	Kommunikationsprotokoll des MightyOhm Geigers.....	15
9.1.2	Der Schaltplan des MightyOhm Geiger Counter v1.0	16
9.1.3	Das Layout des MightyOhm Geiger Counter v1.0.....	17
9.2	Mikrokontroller ATmega8L.....	18
9.3	Ressourcenzuordnung ATmega8L im Projekt myGeigerle.....	19
9.4	Interrupt-Vektor-Tabelle ATmega8L.....	21
9.5	Basisbeschaltung eines ATmega8L inkl. Display	22
	Kurzhubtaster 6x6mm, Höhe: 4,3mm, 12V, vertikal	25
9.6	ISP-Schnittstelle / ISP-Programmierung	25
9.7	Traceschnittstelle.....	27
9.7.1	RS232-Traceschnittstelle.....	27
9.7.4	FTDI Friend + extra – v1.0 UART-USB-Bridge	29
9.7.5	FTDI Serial TTL-232 USB Cable	32
9.8	AVR Fusebits Tutorial.....	33
9.9	AVR Fuse Konfiguration ATmega8L	37
9.9.1	Die Fuse-Konfiguration von myGeigerle	39
9.10	Grundlagen zur Spannung 5V, Vcc und VDD	41
10	Elektronische Teilkomponenten	42
10.1	Hauptschalter	42
10.2	Spannungsversorgung	43
10.2.1	Beschaltung des Festspannungsreglers:	43
10.3	LCD-Grafik-Display	44
10.3.1	Beschaltung des LCD-Display	47
10.3.2	Befehlstabellen	49
10.3.3	BASCOM Beispielcode für die Displayansteuerung (Funktionsbibliothek)	52
10.3.4	Prototyp LCD-Display-Ansteuerung	59
10.4	LED-Zuordnung Schaltplan.....	59
11	Mechanik.....	61
12	Bauteile und Bauteilbeschaffung	62
13	Hardware.....	64
13.1	Festlegung von Netzklassen im Projekt	64
13.2	Die PCB zum Projekt myGeigerle.....	66
13.2.1	Schematic.....	66
13.2.2	Layout, Layer und Bestückung	68
13.2.3	Bestückungen in Groß	70
13.2.4	Eagle-BOM	71
13.2.5	Das Board	72
13.3	Die fertige Hardware	73
14	Software.....	74
14.1	Systemfestlegungen und Definitionen	74
14.1.1	Timerfestlegungen	74
14.2	Verwendete SW	74

14.3	Der Source-Code zum Projekt myGeigerle	76
15.1	Bücher und Literatur.....	86
15.2	Internet.....	86
15.2.1	Firmen und Foren.....	86
15.2.2	ATmega SW und HW-Lösungen	88
15.2.3	Foren	88
16	Entwicklungsbegleitende Notizen und Informationen	90
16.1	Projektcheckliste für AVR Systemdesigns.....	90
16.1.1	Abblockkondensator(en) ordnungsgemäß installiert?.....	90
16.1.2	Spannungsversorgung richtig angeschlossen?	90
16.1.3	Reset-Pin korrekt beschaltet?	90
16.1.4	Alle Ground-Anschlüsse beschaltet?.....	91
16.2	Datenblätter	91

2 Abbildungsverzeichnis

Abbildung 1: Entwicklungsumgebung für das Projekt myGeigerle.....	11
Abbildung 2: Der MightyOhm Geiger Counter ... die Inspiration	12
Abbildung 3: Das Display für das Projekt myGeigerle in Betrieb	13
Abbildung 4: Das Kommunikationsprotokoll des MightyOhm Geiger Counters	15
Abbildung 5: Das Layout des MightyOhm Geiger Counter v1.0	17
Abbildung 6: Erstinbetriebnahme des Displays mit ATmega8L auf einem Breadboard	18
Abbildung 7: PINOut ATmega8L PDIP	19
Abbildung 8: PINOut ATmega8L TQFP	19
Abbildung 9: PINOut ATmega8L PDIP	19
Abbildung 10: USB-ISP-Programmer ERFOS AVR-ISP	26
Abbildung 11: RS232-Traceadapter	29
Abbildung 12: FTDI Friend + Extras - v1.0	30
Abbildung 13: Layout des FTDI Friend	31
Abbildung 14: FTDI Serial TTL-232 USB Cable	32
Abbildung 15: FTDI Serial TTL-232 USB Cable Datenblatt.....	32
Abbildung 16: FTDI Serial TTL-232 USB Cable Detailansichten	33
Abbildung 17: Fusebits im AVR Studio.....	37
Abbildung 18: Hauptschalter	42
Abbildung 19: LCD EA DOGM128E-6 128x64 Pixel	44
Abbildung 20: LED Hintergrundbeleuchtung	45
Abbildung 21: LCD-Display EA DOG-M 128x-6.....	45
Abbildung 22: Abmessungen und technische Daten DOGM LCD-Display	46
Abbildung 23: ST7565 – SPI Pinout.....	49
Abbildung 24: Page-Organisation des Displays.....	49
Abbildung 25: Prototyp LCD-Display-Ansteuerung mit Breadboard	59
Abbildung 26: Platine des myGeigerle mit Träger für MightyOhm Geiger	61
Abbildung 27: MightyOhm Geiger auf myGeigerle montiert	61
Abbildung 28: Definition der Netzklassen	64
Abbildung 29: Demoboard Netzklassen	65
Abbildung 30: PCB myGeigerle – Layout gesamt.....	68
Abbildung 31: PCB myGeigerle – Top Layer.....	68
Abbildung 32: PCB myGeigerle – Bottom Layer.....	68
Abbildung 33: PCB myGeigerle – Bestückung Top Layer	69
Abbildung 34: PCB myGeigerle – Pads und Vias	69
Abbildung 35: PCB myGeigerle – Restricted Areas	69
Abbildung 36: PCB myGeigerle – Bestückung.....	70
Abbildung 37: PCB myGeigerle TOP.....	72
Abbildung 38: PCB myGeigerle BOTTOM.....	72
Abbildung 39: PCB TOP fertig bestückt.....	72
Abbildung 40: PCB BOTTOM fertig bestückt.....	73
Abbildung 41: Die fertige Platine.....	73

3 Tabellenverzeichnis

Tabelle 1: Historie	9
Tabelle 2: Stückliste CPU-Unit ATmega8L.....	19
Tabelle 3: Ressourcenzuordnung ATmega8L.....	20
Tabelle 4: Interrupt-Vektor-Tabelle ATmega8L	21
Tabelle 5: Interrupt-Vektor-Tabelle ATmega8L	21
Tabelle 6: Stückliste Basisbeschaltung ATmega8L.....	23
Tabelle 7: Stückliste Basisbeschaltung ATmega8L.....	24
Tabelle 8: Kurzhubtaster für RESET	25
Tabelle 9: ISP connection Pinout	25
Tabelle 10: PIN-Belegung des seriellen RS232-Ports	28
Tabelle 11: PIN-Belegung der 9-poligen RS232 Stecker/Buchse	28
Tabelle 12: Stückliste RS232-Adapter zwischen PIN-Header und D-SUB9 Buchse PC.....	28
Tabelle 13: PIN-Belegung RS232-Pfostensteckers	29
Tabelle 14: Ressourcenzuordnung SW-RS232 für den ATmega8L.....	29
Tabelle 15: Fuse High Byte ATmega8L.....	39
Tabelle 16: Extended Fuse Byte ATmega8L.....	39
Tabelle 17: AVR-Studio – Interface settings	39
Tabelle 18: AVR-Studio – Tool information	39
Tabelle 19: AVR-Studio – Device information.....	40
Tabelle 20: AVR-Studio – Oscillator calibration	40
Tabelle 21: AVR-Studio - Memories.....	40
Tabelle 22: AVR-Studio – Fuses	40
Tabelle 23: AVR-Studio – Lock bits	40
Tabelle 24: AVR-Studio – Production file.....	40
Tabelle 25: Vorgeschriebene Namensgebung für Spannungsversorgungen	41
Tabelle 26: Stückliste Beschaltung Festspannungsregler	43
Tabelle 27: LED-Hintergrundbeleuchtung für LCD-Display.....	47
Tabelle 28: Stückliste Basisbeschaltung ATmega8L.....	48
Tabelle 29: Ressourcenzuordnung ATmega8L für LCD-Display	48
Tabelle 31: Befehlstabelle des Grafik-Displays EA-DOGM128x-6.....	50
Tabelle 32: LCD-Display EA DOGM081 Initialisierungsbeispiel (Single power supply)	51
Tabelle 33: LCD-Display EA DOGM081 Initialisierungsbeispiel (dual power supply).....	51
Tabelle 34: LED-Zuordnung Schaltplan	59
Tabelle 35: Bauteile für LED-Ansteuerung.....	60
Tabelle 36: Ressourcenzuordnung für LED-Ansteuerung	60
Tabelle 37: Bauelemente Reichelt Elektronik.....	62
Tabelle 38: Bauelemente Conrad Electronic.....	62
Tabelle 39: Bauelemente Amazon.....	63
Tabelle 40: Bauelemente Conrad Electronic.....	63
Tabelle 41: Eagle BOM für das Projekt myGeigerle.....	71
Tabelle 42: Weitere Bauteile für das Projekt myGeigerle	71

4 Schaltbilderverzeichnis

Schaltbild 1: Der Schaltplan des MightyOhm Geiger Counter v1.0	16
Schaltbild 2: Exemplarische Basisbeschaltung ATmega 8L inklusive LCD-Text-Display	23
Schaltbild 3: Exemplarische Basisbeschaltung ATmega 8L inklusive Grafikdisplay EA DOGM 128x64.....	24
Schaltbild 4: 10-Pin ISP connection Pinout	26
Schaltbild 5: Adapter zwischen PIN-Header und D-SUB9 Buchse.....	28
Schaltbild 6: Schaltplan des Adafruit FTDI Friend	31
Schaltbild 7: Symbolfestlegung für Spannungsversorgungen.....	41
Schaltbild 8: Beschaltung des Festspannungsreglers	43
Schaltbild 9: Beschaltung des LCD-Displays gemäß Datenblatt.....	47
Schaltbild 10: Schematische Basisbeschaltung des LCD-Displays.....	48
Schaltbild 11: Beschaltung LED's	60
Schaltbild 12: Schaltbild für Definition von Netzklassen	64
Schaltbild 13: Schaltbild myGeigerle - Sheet 1	66
Schaltbild 14: Schaltbild myGeigerle - Sheet 2	67
Schaltbild 15: Schaltbild myGeigerle - Sheet 3	67

5 Softwareverzeichnis

Software 1: Code zur Ansteuerung des LCD-Grafik-Displays	58
Software 2: Source-Code des Projekt myGeigerle	85

6 Historie

<i>Datum</i>	<i>Entscheidung</i>
20.02.2019	Beginn der Projektarbeit und Dokumenterstellung
20.02.2019	Fertigstellung des Projekts und Dokumentations-Restarbeiten
21.02.2019	Fertigstellung der Gesamtdokumentation

Tabelle 1: Historie

7 Sicherheitshinweis

******* WARNING *******

This Geiger Counter kit is for EDUCATIONAL PURPOSES ONLY.

Don't even think about using it to monitor radiation in life-threatening situations, or in any environment where you may expose yourself to dangerous levels of radiation.

Don't rely on the collected data to be an accurate measure of radiation exposure! Be safe!

	Achtung: Aufgrund der im Gerät frei geführten Netzspannung dürfen Aufbau und Inbetriebnahme ausschließlich von Fachkräften durchgeführt werden, die aufgrund ihrer Ausbildung dazu befugt sind. Die einschlägigen Sicherheits- und VDE-Bestimmungen sind unbedingt zu beachten. Der Autor übernimmt keinerlei Haftung über Schaden jeglicher Art, die direkt oder indirekt durch das Gerät verursacht werden.	
---	--	---

******* WARNING *******

Die Schaltung enthält einen Hochspannungsteil.

Es sind Vorsichtsmaßnahmen im Umgang mit Hochspannung ratsam.

Hochspannung IMMER mit einem sehr hochohmigen (1GOhm) Voltmeter messen und justieren.

Ggf. Vorwiderstand dazwischenschalten.

Die Zählröhre SBM-20 ist berührungsempfindlich. Die Röhre muss so gelagert werden, dass Berührung der Röhre vermieden wird.

8 Allgemeines

8.1 Die Entwicklungsumgebung

Die Entwicklungsumgebung meiner Elektronikprojekte:



Abbildung 1: Entwicklungsumgebung für das Projekt myGeigerle

8.2 Allgemeine Beschreibung und Idee zur Realisierung

Schon lange stand das Thema Geigerzähler bzw. Geiger Counter auf meiner ToDo-Liste doch lange ist nichts passiert, da ich mich nicht wirklich an das Thema heranwagen wollte.

Im Rahmen einer neuen und besonderen Wetterstation, welche ich plane und bei der auch die natürliche Radioaktivität erfasst werden soll bekam das Thema aber neue Bedeutung.

Um erst mal Erfahrungen mit den Geigern und der notwendigen Hochspannung zu sammeln begann alles mit der Recherche im Internet bei der ich auf das MightyOhm Geiger Counter Kit <https://mightyohm.com/blog/products/geiger-counter/> aus den USA aufmerksam wurde uns dessen Bausatz in Deutschland durch Elektor vertrieben wurde.

Darüber hinaus gab es in meinem Lagerbestand noch ein Grafikdisplay von EA ohne Bestimmung.

Schon war das Projekt myGeigerle geboren.

Mit einem Kit erste Erfahrungen für später sammeln und eine Auswertelogik auf Basis eines ATmega dazu zu bauen erschien mir als gute Idee.

Hier also die Umsetzung von myGeigerle als eigenes kleines Projekt für alle diejenigen die vielleicht Interesse haben es auch auszuprobieren.

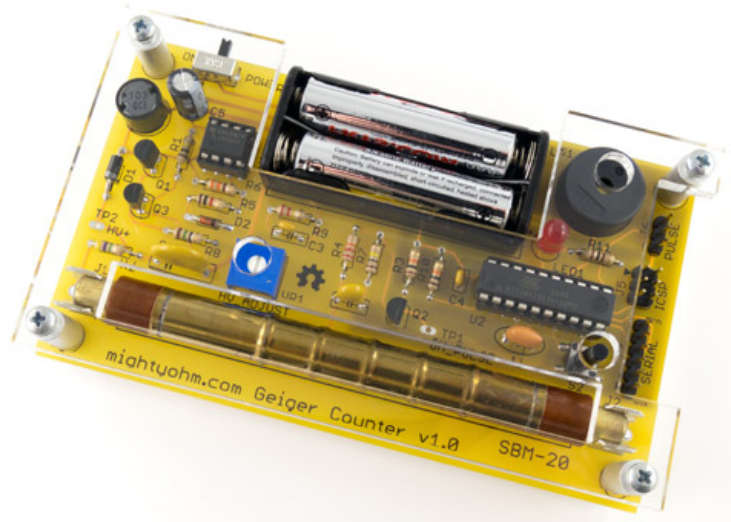


Abbildung 2: Der MightyOhm Geiger Counter ... die Inspiration

Allgemeines und Ideen zu Realisierung:

In den folgenden Beschreibungen und Erklärungen beschränke ich mich im Wesentlichen auf den Teil, welchen ich selbst zum Projekt beigetragen habe.

Was den Geiger Counter von MightyOhm aus USA betrifft so wird hier auf dessen Homepage, welche unter MightyOhm zu finden ist, referenziert.

Die tatsächlich relevanten Teile wie z.B. das Kommunikationsprotokoll werde ich hier im Folgenden zitieren, für den Fall, dass die Website von MightyOhm <https://mightyohm.com/blog/> down sein sollte. Damit kann trotzdem nachvollzogen werden, was ich hier angestellt habe.

An dieser Stelle aber nochmals der Hinweis, was immer zitiert oder erklärt und beschrieben wird das Copyright und die Rechte am MightyOhm und dessen Dokumenten etc. liegt bei Jeff Keyzer in USA.

8.3 Leistungsumfang

im Folgenden wird der Leistungsumfang und die Teilfunktionalität beschrieben welche myGeigerle besitzt und dem Benutzer zur Verfügung stellt:

- Hauptschalter
- LCD-Grafik-Display mit einer Auflösung 128x64 Pixel
 - o Balkendiagramm der gemessenen Radioaktivität bezogen auf CPS (counts per second)
 - o Messwerte und Betriebszustand
 - o Empfangene und ausgewertete Datenpakete vom MightyOhm Counter
- Spannungsversorgung via 5V USB Mini Typ B Stecker
 - o Erzeugung der 3,3V Betriebsspannung mittels Festspannungsregler
- Schnittstellen
 - o RS232 Schnittstelle zum Tracing und zur Datenübertragung der Logging-Daten an PC
 - o ISP Schnittstelle zur direkten Programmierung des Targets
- Hintergrundbeleuchtung und optisches Alive des Mikroprozessors

Das vom MightyOhm Geiger gesendete Protokoll wird vom ATmega8L empfangen und verarbeitet.

Die empfangenen Messdaten werden auf dem Display angezeigt. Die CPS = Counts Per Second werden in einem Balkendiagramm angezeigt und ein weiterer Counter zählt die vom Geiger empfangenen Datenpakete.

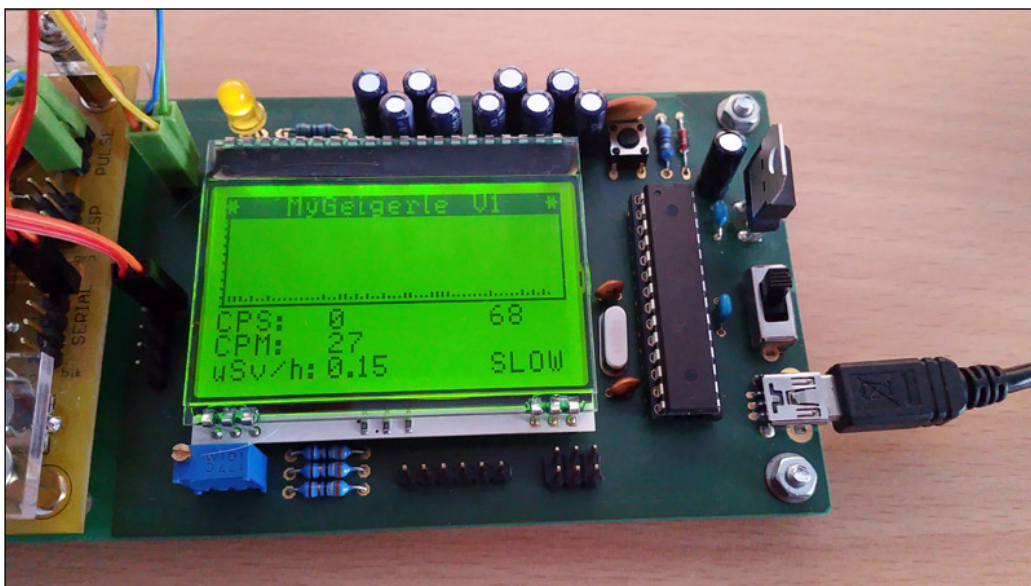


Abbildung 3: Das Display für das Projekt myGeigerle in Betrieb

8.4 Funktionskomponenten

Das Projekt myGeigerle verfügt über die folgenden einzelnen Funktions- / Teilkomponenten:

- | | |
|--|--------|
| • Wichtiges und Basics zum MightyOhm Geiger Counter | MIGHTY |
| • Spannungsversorgung via 5V USB und Erzeugung von 3,3V über Festspannungsregler | SUP |
| • Zentraler Hauptschalter für Spannungsversorgung | Switch |
| • Mikrocontroller ATmega8L (inkl. ISP, UART und Reset) | ATMEGA |
| • UART / FTDI Trace-Schnittstelle | TRACE |
| • LCD-Grafik-Display EA DOGM128E-6 (grün/gelb) von Electronic Assembly | LCD |
| • Hintergrundbeleuchtung und Alive-LED | LED |

9 Elektronische Grundlagen

9.1 Wichtiges und Basics zum MightyOhm Geiger Counter

Der MightyOhm Geiger Counter steht als Bausatz zur Verfügung. Er kann über unterschiedliche Quellen wie Adafruit oder Elektor in Deutschland bezogen werden und kostet ca. 100 Euro.

Der Bausatz besteht neben Platinen und Bauteilen im Kit auch aus Plexiglas-Formteilen für Boden und Deckel sowie aus dem Geiger-Zählrohr SBM-20.

Die Assembly-Anleitung etc. befindet sich unten im Bereich der Dokumente zu diesem Projekt zum Download bereitgestellt.

Herzstück des Geiger Counters ist ein ATtiny Prozessor, der die Auswertung der radioaktiven Zerfallsereignisse bereits on chip verrechnet und mittels UART-Schnittstelle, welche der FTDI Spezifikation entspricht, bereitstellt.

9.1.1 Kommunikationsprotokoll des MightyOhm Geigers

Das Protokoll des MightyOhm Geigers im Details entsprechend Beschreibung:

Logging:

To connect your computer to the serial port (J7), you will need a USB-serial converter that operates at 3.3V TTL levels. The serial header is set up to work with the common FTDI-232R-3V3 serial cable.

Data is sent over the serial port at 9600 baud, 8N1.

The data is reported in comma separated value (CSV) format:

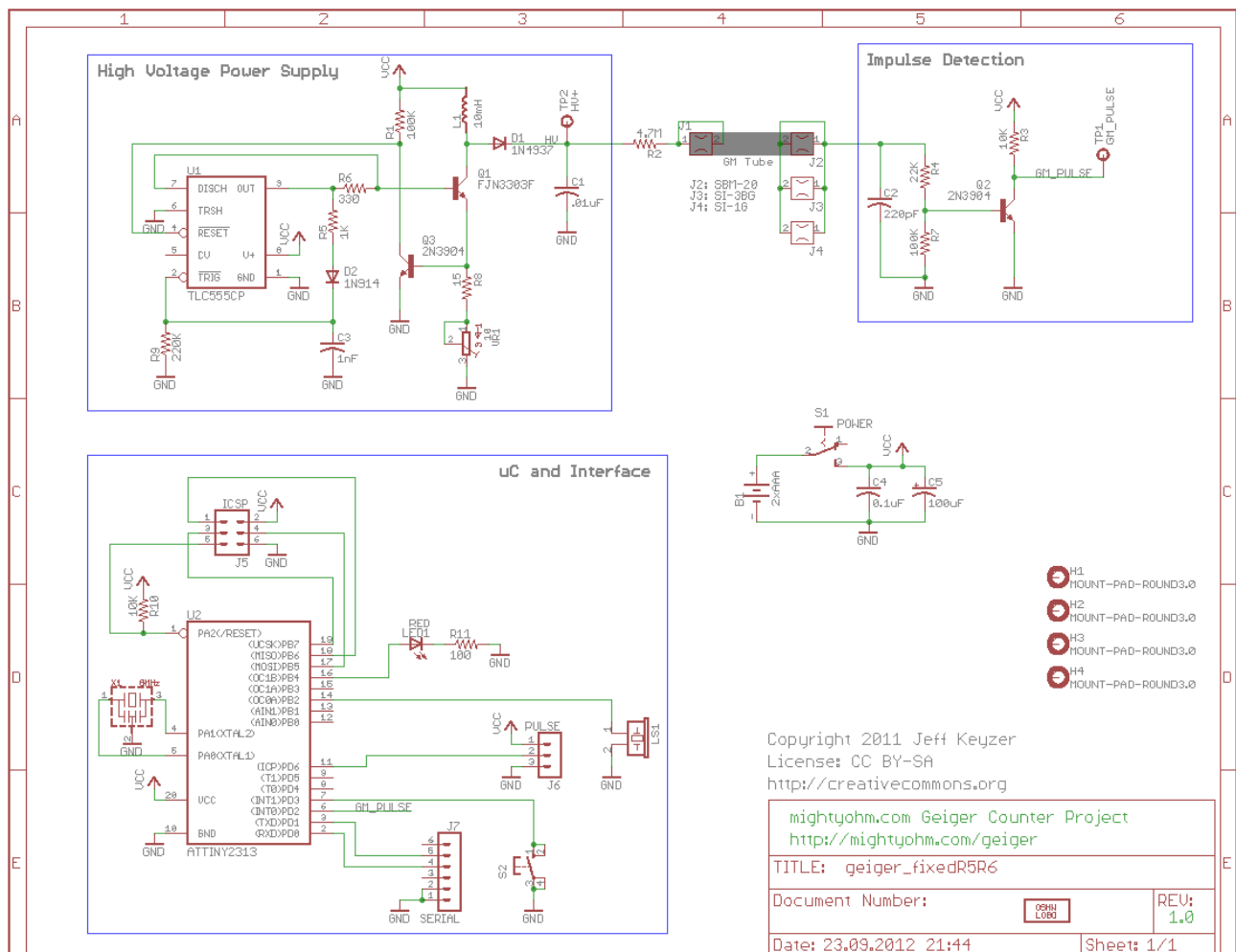
CPS, #####, CPM, #####, uSv/hr, ###.##, SLOW|FAST|INST

There are three modes. Normally, the sample period is LONG_PERIOD (default 60 seconds). This is SLOW averaging mode. If the last five measured counts exceed a preset threshold, the sample period switches to SHORT_PERIOD seconds (default 5 seconds). This is FAST mode, and is more responsive but less accurate. Finally, if CPS > 255, we report CPS*60 and switch to INST mode, since we can't store data in the (8-bit) sample buffer.

The largest CPS value that can be displayed is 65535, but the largest value that can be stored in the sample buffer is 255.

Abbildung 4: Das Kommunikationsprotokoll des MightyOhm Geiger Counters

9.1.2 Der Schaltplan des MightyOhm Geiger Counter v1.0



Schaltbild 1: Der Schaltplan des MightyOhm Geiger Counter v1.0

9.1.3 Das Layout des MightyOhm Geiger Counter v1.0

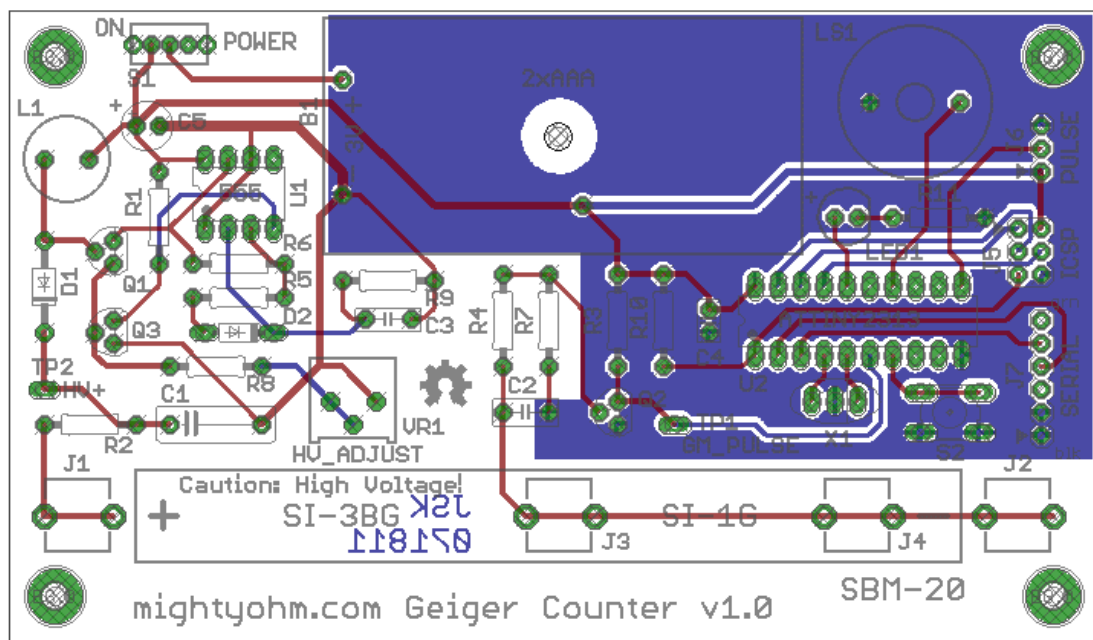


Abbildung 5: Das Layout des MightyOhm Geiger Counter v1.0

9.2 Mikrokontroller ATmega8L

Achtung: Im Projekt myGeigerle wird der ATmega8L mit einer Betriebsspannung von 3,3V betrieben da der MightyOhm Geiger Counter und das verwendete Display auch nur mit 3,3V arbeiten.

Im Projekt wird der Mikrokontroller ATmega8L von ATMEL mit externem Takt von 8MHz eingesetzt. Bei der Erstinbetriebnahme und der Verwendung mit interner Clock des ATmega8L kam es zu Kommunikationsproblemen mit dem MightyOhm Geiger. Die Clock-Drift und die Auswirkungen auf die Baudrate waren zu groß. Daher wurde auf einen genaueren externen Quarz umgestellt.

Erste Inbetriebnahmen und Versuche bzgl. Projektumsetzung wurden mit dem ATmega8L auf eine Breadboard realisiert.

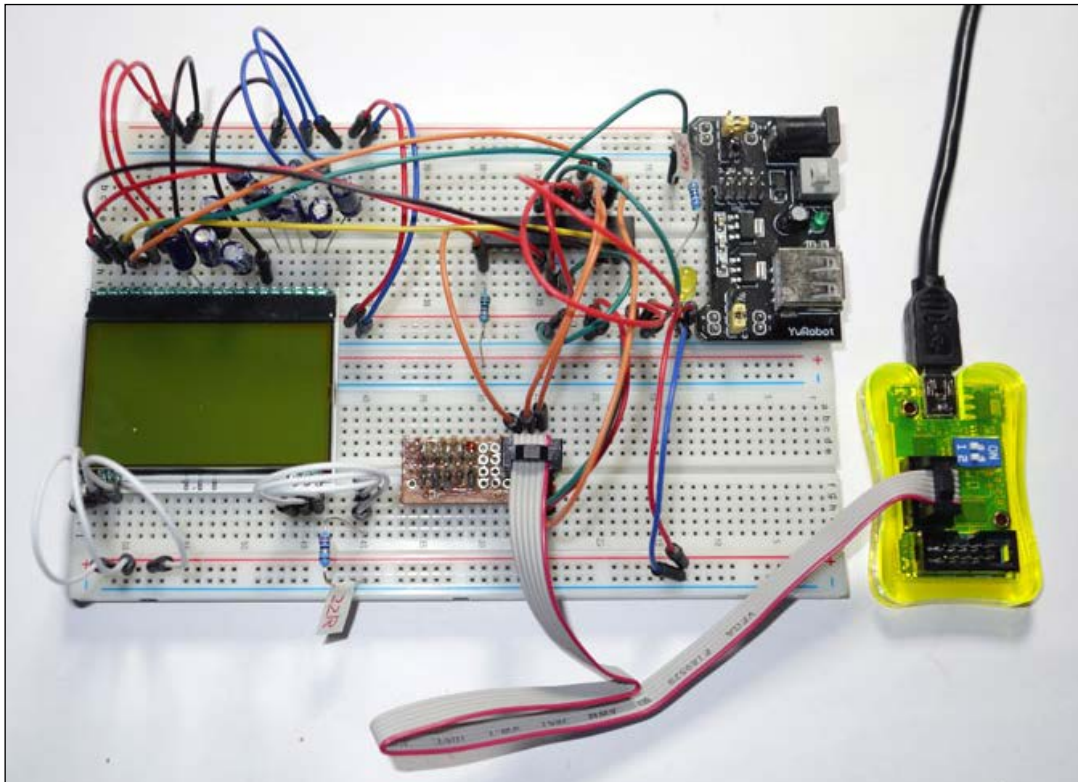


Abbildung 6: Erstinbetriebnahme des Displays mit ATmega8L auf einem Breadboard

PINOut ATmega8L:

PINOut ATmega8LPDIP

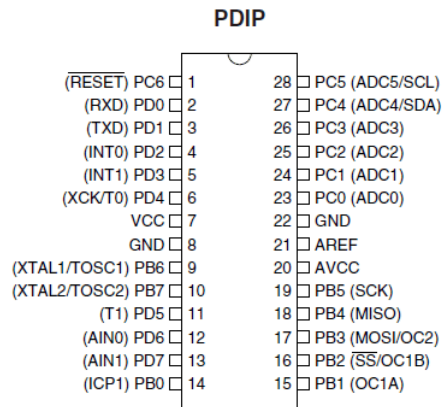


Abbildung 7: PINOut ATmega8L PDIP

PINOut ATmega8LTQFP

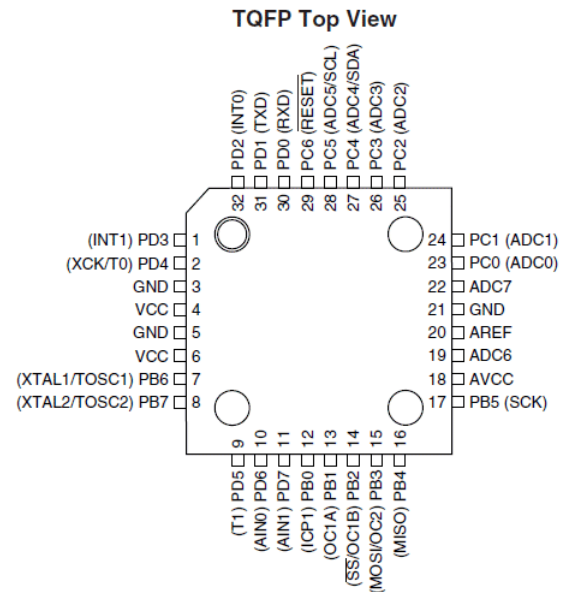


Abbildung 8: PINOut ATmega8L TQFP

Bauteile:

Stückliste: CPU-Unit ATmega8L			
Sonstiges		Halbleiter	
G1	Quarz 8,00 MHz	IC1	ATMEL ATmega8L RISC CPU

Tabelle 2: Stückliste CPU-Unit ATmega8L

9.3 Ressourcenzuordnung ATmega8L im Projekt myGeigerle

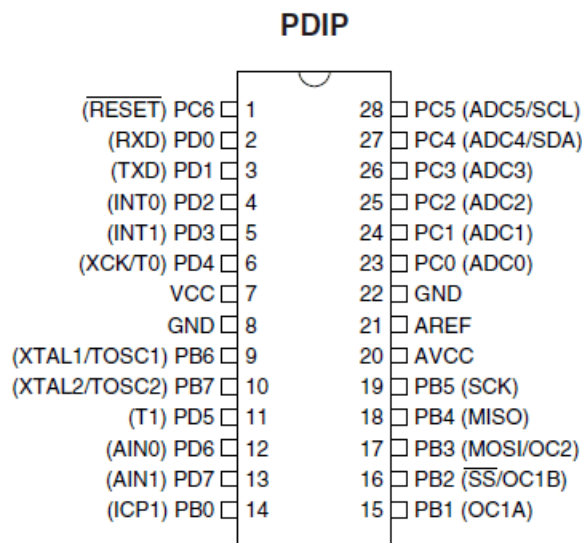


Abbildung 9: PINOut ATmega8L PDIP

<i>PIN</i>	<i>Port</i>	<i>Funktion</i>	<i>Used</i>	<i>Beschreibung</i>	<i>Definition</i>
1	PC6	RESET	J	Externer Reset Eingang	RESET
2	PD0	RXD	J	UART Schnittstelle – Receive Data	RXD
3	PD1	TXD	J	UART Schnittstelle – Transmit Data	TXD
4	PD2	PD2	N	--	--
5	PD3	PD3	N	--	--
6	PD4	PD4	J	Grafik-Display - Chip Select	GLCD_CS1
7	VCC	VCC	J	Spannungsversorgung +3,3V	VCC
8	GND	GND	J	Ground GND	GND
9	PB6	PB6	J	Externer Quarz	XTAL1
10	PB7	PB7	J	Externer Quarz	XTAL2
11	PD5	PD5	J	Grafik-Display – Reset	GLCD_RST
12	PD6	PD6	N	--	--
13	PD7	PD7	J	Grafik-Display – Adressierung A0	GLCD_A0
14	PB0	PB0	N	--	--
15	PB1	PB1	N	--	--
16	PB2	PB2	N	--	--
17	PB3	MOSI	J	ISP Programmierinterface und Grafik-Display SPI: SI	MOSI
18	PB4	MISO	J	ISP Programmierinterface	MISO
19	PB5	SCK	J	ISP Programmierinterface und Grafik-Display SPI: SCL	SCK
20	AVCC	VCC	J	Spannungsversorgung +3,3V	VCC
21	AREF	AREF	J	Spannungsversorgung +3,3V	VCC
22	GND	GND	J	Ground GND	GND
23	PC0	PC0	J	ALIVE-LED	ALIVE
24	PC1	PC1	N	--	--
25	PC2	PC2	N	--	--
26	PC3	PC3	N	--	--
27	PC4	PC4	N	--	--
28	PC5	PC5	N	--	--

Tabelle 3: Ressourcenzuordnung ATmega8L

	Externer Takt und Reset
	Spannungsversorgungen GND und VCC
	Serielle Schnittstelle RS232
	ISP Programmierinterface
	Zu den Analog-Digital-Wandlern gehörend
	Zu Timer2 gehörende Pin's
	GPIO Verwendung

Die unter „Definition“ vergebenen Namen definieren die Namensgebung der Signalleitungen im Schaltplan und ggf. in der Software.

9.4 Interrupt-Vektor-Tabelle ATmega8L

Reset and Interrupt Vectors

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	0x000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

- Notes:
1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 209.
 2. When the IVSEL bit in GICR is set, Interrupt Vectors will be moved to the start of the boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the boot Flash section.

Table 19 shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the boot section or vice versa.

Tabelle 4: Interrupt-Vektor-Tabelle ATmega8L

Reset and Interrupt Vectors Placement

BOOTRST ⁽¹⁾	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

- Note:
1. The Boot Reset Address is shown in Table 82 on page 220. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

Tabelle 5: Interrupt-Vektor-Tabelle ATmega8L

In der aktuellen Umsetzung Version 1 des myGeigerle wird aktuell keine externe Interrupt-Quelle benötigt bzw. verwendet.

Für die Erzeugung des Alive-Signals und damit die blinkende Status LED wird TIMER1 COMPA verwendet.

Die LCD Hintergrundbeleuchtung wird OHNE PWM umgesetzt da der ATmega8L PQM nur auf OC2 zur Verfügung stellt und dieser PIN aber für die SPI zum Display benötigt wird. Aus diesem Grund erfolgt die Dimmung der Hintergrundbeleuchtung mittel Spindeltrimmer.

9.5 Basisbeschaltung eines ATmega8L inkl. Display

Achtung: Die Basisbeschaltung im folgenden Beispiel weicht von der tatsächlichen Implementierung im Projekt ab. Dies ist dem geschuldet, dass die Inbetriebnahme und der Test einzelner Komponenten vorab mit Breadboard stattgefunden hat und das endgültige Ergebnis eine Mischung aus beidem darstellt.

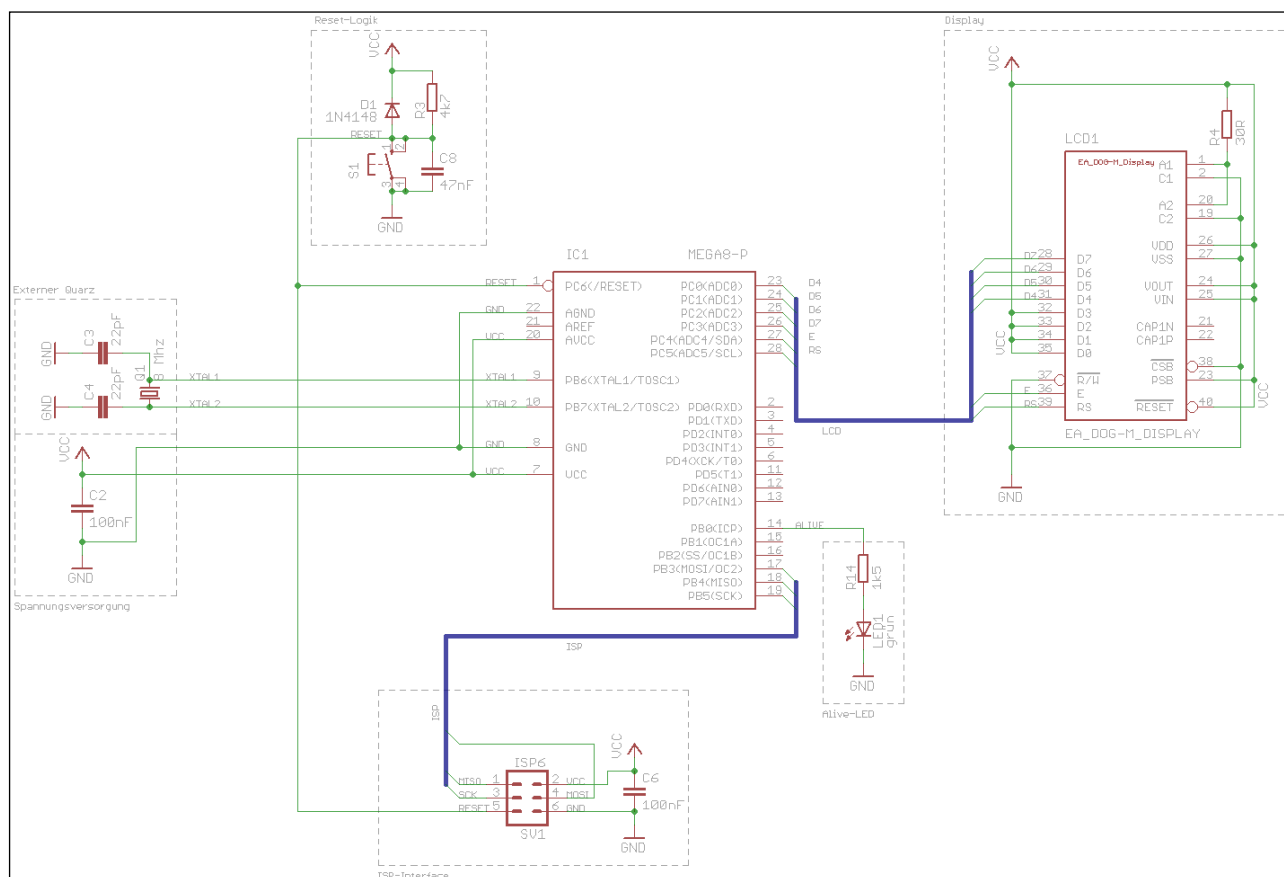
Die im Folgenden dargestellte Schaltung bildet die Basisbeschaltung eines AVR ATmega. Zum Einsatz kommt ein externer Quarz mit 8.000.000 Hz (8 MHz). Weiter ist im Schaltplan der Anschluss eines externen RESET-Tasters vorgesehen.

Die wesentlichen Bestandteile sind vorhanden, welche wären:

- Reset-Logik
- ISP-Interface
- Spannungsversorgung
- Geräuschreduktion für ADC
- Externer Quarz
- Display

Die ALIVE-LED ist an einem anderen PIN angeschlossen und das Display wird nicht über 4-Bit-Mode, sondern über SPI angesteuert. Hierzu folgt aber nochmals ein weiteres Schaltbild für die Erstinbetriebnahme des Displays.

Der ATmega8 im Projekt myGeigerle wird über eine externe 8MHz Taktquelle getaktet.



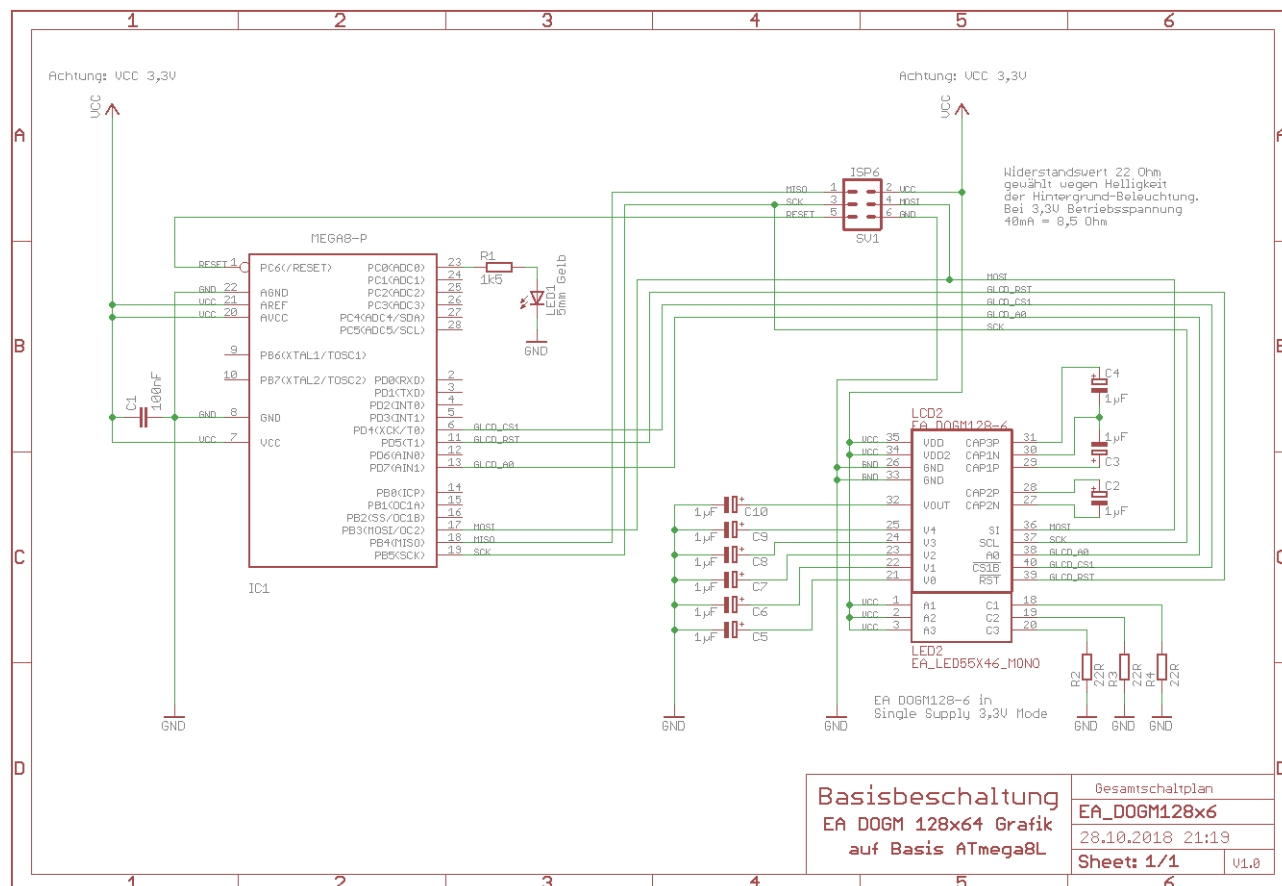
Schaltbild 2: Exemplarische Basisbeschaltung ATmega 8L inklusive LCD-Text-Display

Bauteile:

Stückliste: Basisbeschaltung ATmega8L mit LCD-Text-Display			
Widerstände		Halbleiter	
R3	Metallschichtwiderstand 4k7 Ω	IC1	ATMEL AVR ATmega8L-P
R4	Metallschichtwiderstand 40 Ω	D1	Diode 1N4148
R14	Metallschichtwiderstand 1k5 Ω	LCD1	LCD-Display EA DOGM162
		LED1	Low-Current LED, grün
Kondensatoren		Sonstiges	
C3, C4	Keramikkondensator 22 pF	S1	Kurzhubtaster
C8	Keramikkondensator 47 nF	Q1	Quarz 8.000.000 Hz
C2, C6	MP-Kondensatoren 100nF	SV1	Federleiste MA03-2 (ISP)

Tabelle 6: Stückliste Basisbeschaltung ATmega8L

Im Folgenden hier nochmals ergänzend die Basisbeschaltung eines ATmega8L mit Grafikdisplay EA DOGM 128x64:



Schaltbild 3: Exemplarische Basisbeschaltung ATmega 8L inklusive Grafikdisplay EA DOGM 128x64

Bauteile:

Stückliste: Basisbeschaltung ATmega8L mit LCD-Grafik-Display			
Widerstände		Halbleiter	
R1	Metallschichtwiderstand 1k5 Ω	IC1	ATMEL AVR ATmega8L-P
R2, R3, R4	Metallschichtwiderstand 22 Ω	LCD2	LCD-Grafik-Display EA DOGM128x-6
		LED1	Low-Current LED, gelb
		LED2	LED Hintergrundbeleuchtung EA_LED55x46
Kondensatoren		Sonstiges	
C1	Keramikkondensator 100nF	SV1	Federleiste MA03-2 (ISP)
C2 – C10	Elektrolytkondensator 1µF		

Tabelle 7: Stückliste Basisbeschaltung ATmega8L

Reset-Taster:

Als Reset-Taster wird ein Print-Kurzhubtaster verwendet



Kurzhubtaster 6x6mm, Höhe: 4,3mm, 12V, vertikal
Bestellnummer Reichelt Elektronik: *TASTER 3301*

Tabelle 8: Kurzhubtaster für RESET

9.6 ISP-Schnittstelle / ISP-Programmierung

Zur In-System-Programmierung wird die ATMEL ISP-Schnittstelle umgesetzt. Die Programmierung im Projekt erfolgt direkt über BASCOM mit Hilfe des USB-ISP-Programmer ERFOS AVR-ISP welcher von tremex hergestellt und von Diamex vertrieben wird.

Das Problem ist und war, dass ab Windows 10 BASCOM Probleme mit dem ATMEL AVR ISP mkII hat, die wohl auf Treiber-Inkompatibilitäten beruhen, welche ich bis heute nicht lösen konnte. Daher der Einsatz des anderen Programmers.

Hinweis: Mit Hilfe des ATMEL Studio 7.0 lässt sich der ATMEL AVR ISP mkII noch problemlos verwenden, aber nicht von BASCOM aus.

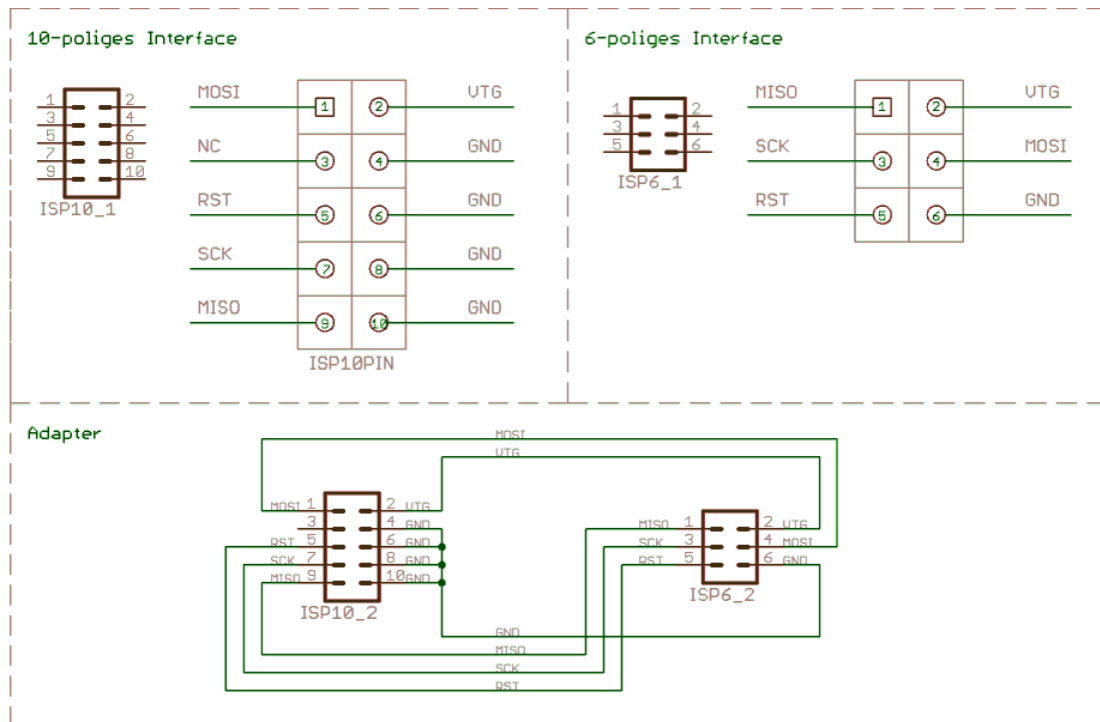
Grundsätzlich ist der Mikrocontroller mit jedem Atmel – ISP – Kompatiblen Programmiergerät programmierbar.

ISP Programmierinterface von ATMEL:

1 2 MISO ● VTG SCK ● MOSI RST ● GND ISP6PIN	1 2 MOSI ● VTG NC ● GND RST ● GND SCK ● GND MISO ● GND ISP10PIN
6-pin ISP connection Pinout	10-pin ISP connection Pinout

Tabelle 9: ISP connection Pinout

Für das Projekt myGeigerle wird das 6-polige Interface in der folgenden Form umgesetzt:



Schaltbild 4: 10-Pin ISP connection Pinout

Der Adapter und das 10-polige Interface werden nicht umgesetzt. Das 6-polige ISP-Interface befindet sich auf allen Boards und kann direkt mit dem USB-ISP-Programmer AVRISP mkII verbunden werden.

Genaue Beschreibung und Datenblätter zum USB-ISP-Programmer befinden sich im Anhang zu diesem Dokument im Kapitel Datenblätter.



Abbildung 10: USB-ISP-Programmer ERFOS AVR-ISP

9.7 Traceschnittstelle

In allen bisherigen Projekten wurde zum Tracen und zur Ausgabe von Daten vom ATmega zum PC eine UART bzw. RS232 Schnittstelle umgesetzt. Zur Vollständigkeit wird diese auch hier in diesem Kapitel nochmals beschrieben.

Aber.....

Beginnend mit dem Projekt myGeigerle wird auf UART-USB-Interface umgestellt. Zum Einsatz kommt die UART-USB--Bridge von FTDI. Um das Rad nicht neu zu erfinden und fine pitch Bauelemente löten zu müssen wird auf bestehende Produkte zurückgegriffen.

Die Wahl fällt dabei auf die beiden Produkte

1. FTDI Friend + Extras - v1.0
2. FTDI Serial TTL-232 USB Cable

von Adafruit welche in diesem Kapitel auch weiter und detailliert beschrieben werden. Damit gibt es eine zukunftssichere Lösung welche für 3,3V und 5V Projekte gleichermaßen eingesetzt werden kann.

9.7.1 RS232-Traceschnittstelle

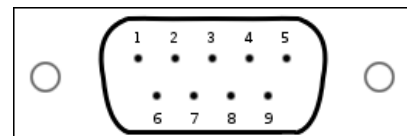
Abkürzung	Name	Beschreibung	Pin-Nr. 25-pol.	Pin-Nr. 9-pol.	In/Out
--	Common Ground	Gemeinsame Abschirmmasse (nicht Datenmasse)	Pin 1	—	—
TxD,TX,TD	Transmit Data	Leitung für ausgehende (gesendete) Daten.	Pin 2	Pin 3	Out
RxD,RX,RD	Receive Data	Leitung für den Empfang von Daten.	Pin 3	Pin 2	In
RTS	Request to Send	„Sendeankündigung“; Eine logische Null an diesem Ausgang signalisiert der Gegenstelle, dass sie Daten Senden kann	Pin 4	Pin 7	Out
CTS	Clear to Send	Eine logische Null an diesem Eingang ist ein Signal der Gegenstelle, dass sie Daten entgegennehmen kann	Pin 5	Pin 8	In
DSR	Dataset Ready	Ein angeschlossenes Gerät signalisiert dem Computer, dass es einsatzbereit (nicht notwendigerweise empfangsbereit) ist, wenn eine logische Null auf dieser Leitung anliegt.	Pin 6	Pin 6	In
GND	Ground	Signalmasse. Die Signalspannungen werden gegen diese Leitung gemessen.	Pin 7	Pin 5	—
DCD,CD	(Data) Carrier Detect	Ein Gerät signalisiert dem Computer, dass es einlaufende Daten auf der Leitung erkennt	Pin 8	Pin 1	In
DTR	Data Terminal Ready	Über diese Leitung signalisiert der PC dem Gerät, dass er betriebsbereit ist. Damit kann ein Gerät eingeschaltet oder zurückgesetzt werden. (Üblicherweise schaltet ein Gerät z.B. Modem diese Leitung auf DSR durch, wenn es einsatzbereit ist)	Pin 20	Pin 4	Out

Abkürzung	Name	Beschreibung	Pin-Nr. 25-pol.	Pin-Nr. 9-pol.	In/Out
RI	Ring Indicator	Das Gerät zeigt dem PC an, dass ein Anruf ankommt ("ring" ist engl. für "klingeln"; besonders bei Modems)	Pin 22	Pin 9	In

Tabelle 10: PIN-Belegung des seriellen RS232-Ports

In/Out wird auch Sicht des PC's aus betrachtet.

RS232 Buchse 9-polig:



RS232 Stecker 9-polig:

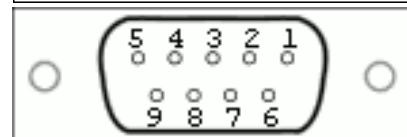
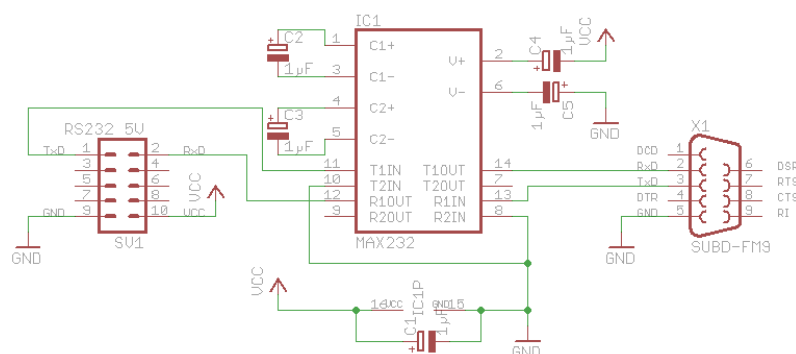


Tabelle 11: PIN-Belegung der 9-poligen RS232 Stecker/Buchse

9.7.2 Adapter RS232 PIN-Header / D-SUB9



Schaltbild 5: Adapter zwischen PIN-Header und D-SUB9 Buchse

Bauteile:

Stückliste: RS232-Adapter zwischen PIN-Header und D-SUB9 Buchse PC			
Sonstiges		Halbleiter	
PIN-HEADER	PFL10 Pfostensteckverbinder	IC1	Maxim MAX232 CPE
D-SUB_9POLIG	SUB-D-Buchse 9-polig.		
Kondensatoren			
C1, C2, C3, C4, C5	Elko 1µF/16V		

Tabelle 12: Stückliste RS232-Adapter zwischen PIN-Header und D-SUB9 Buchse PC

PIN-Belegung Pfostenstecker:

<i>PIN</i>	<i>Funktion</i>
1	TxD
2	RxD
3	n.c.
4	n.c.
5	n.c.
6	n.c.
7	n.c.
8	n.c.
9	GND
10	Spannungsversorgung Target (Vcc)

Tabelle 13: PIN-Belegung RS232-Pfostensteckers

Ressourcenzuordnung zum ATmega8L:

<i>Nummer</i>	<i>Schaltbild</i>	<i>Ressource ATmega8L</i>
1	RXD	PortD.0 [PD0] (RXD)
2	TXD	PortD.1 [PD1] (TXD)

Tabelle 14: Ressourcenzuordnung SW-RS232 für den ATmega8L

9.7.3 Trace-Adapter RS232 Prototyp auf Lochraster

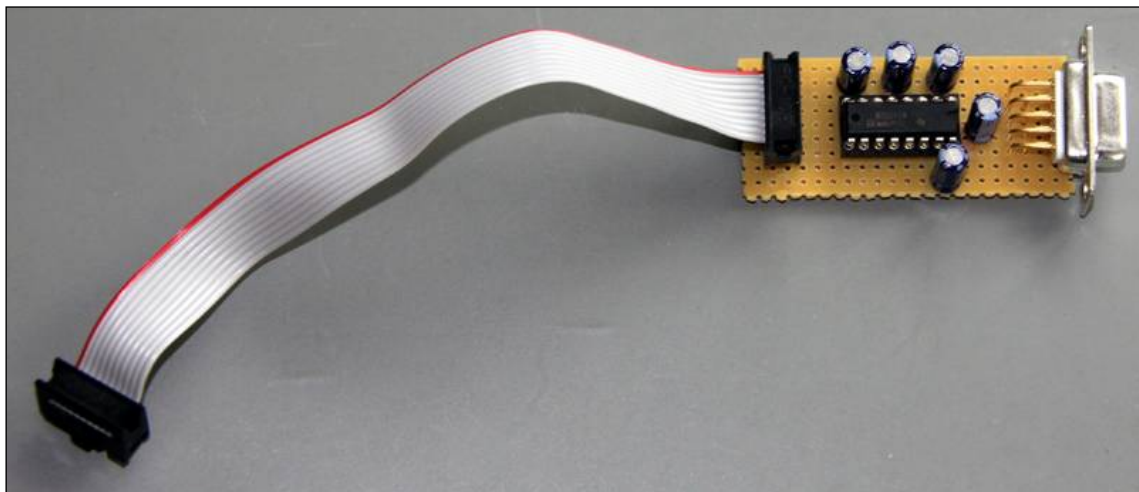


Abbildung 11: RS232-Traceadapter

9.7.4 FTDI Friend + extra – v1.0 UART-USB-Bridge

Für das Tracing und die Anbindung der neuen Projekte wird Anstelle RS232 eine UART-USB-Bridge von Adafruit eingesetzt, FTDI Friend + Extras – v1.0.

Adafruit Produkt ID: 284

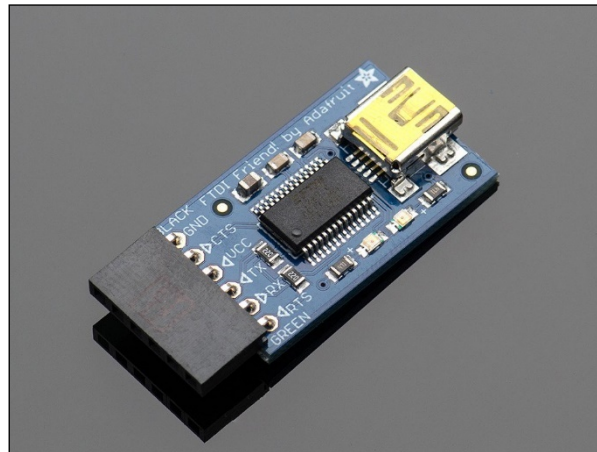
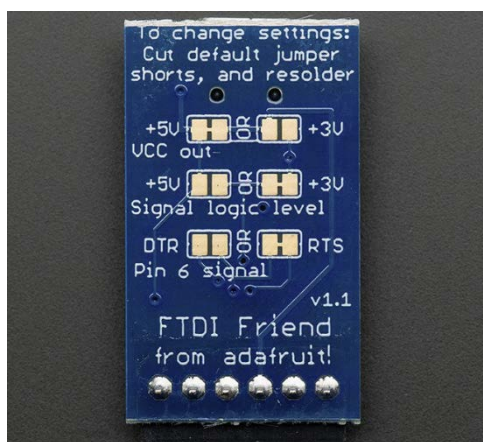


Abbildung 12: FTDI Friend + Extras - v1.0

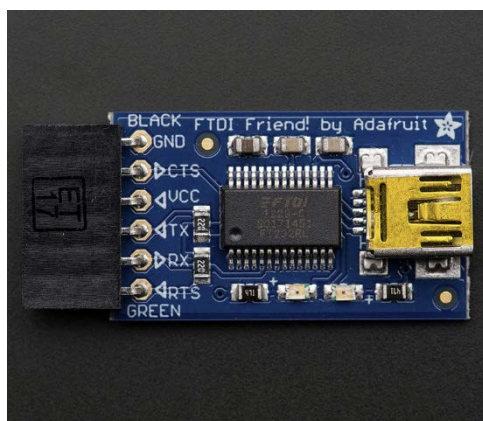


Für die Konfiguration des FTDI Friend existieren auf der Platine-Rückseite Lötflächen bzw. PCB-Brücken die gesetzt und entfernt werden müssen.

Bei VCC kann zwischen 3,3V und 5V gewählt werden (Default ist 5V).

Das Logik-Level für RxD und TxD kann zwischen 3,3V und 5V geändert werden. Default ist 3,3V. Mit dieser Einstellung sind die Pegel aber auch 5V compliant und funktionieren für beide Spannungspegel.

Die Funktion des 6. Pins kann wahlweise auf DTR oder RTS gesetzt werden (Default ist RTS)



Pinbelegung der Stecker:

Pin 1	Schwarz	Ground
Pin 2	Braun	CTS
Pin 3	Rot	VCC (Power), 5V (* 3,3V)
Pin 4	Orange	TxD 3V Level (* 5V Level)
Pin 5	Gelb	RxD 3V Level (* 5V Level)
Pin 6	Grün	RTS (* DTR)

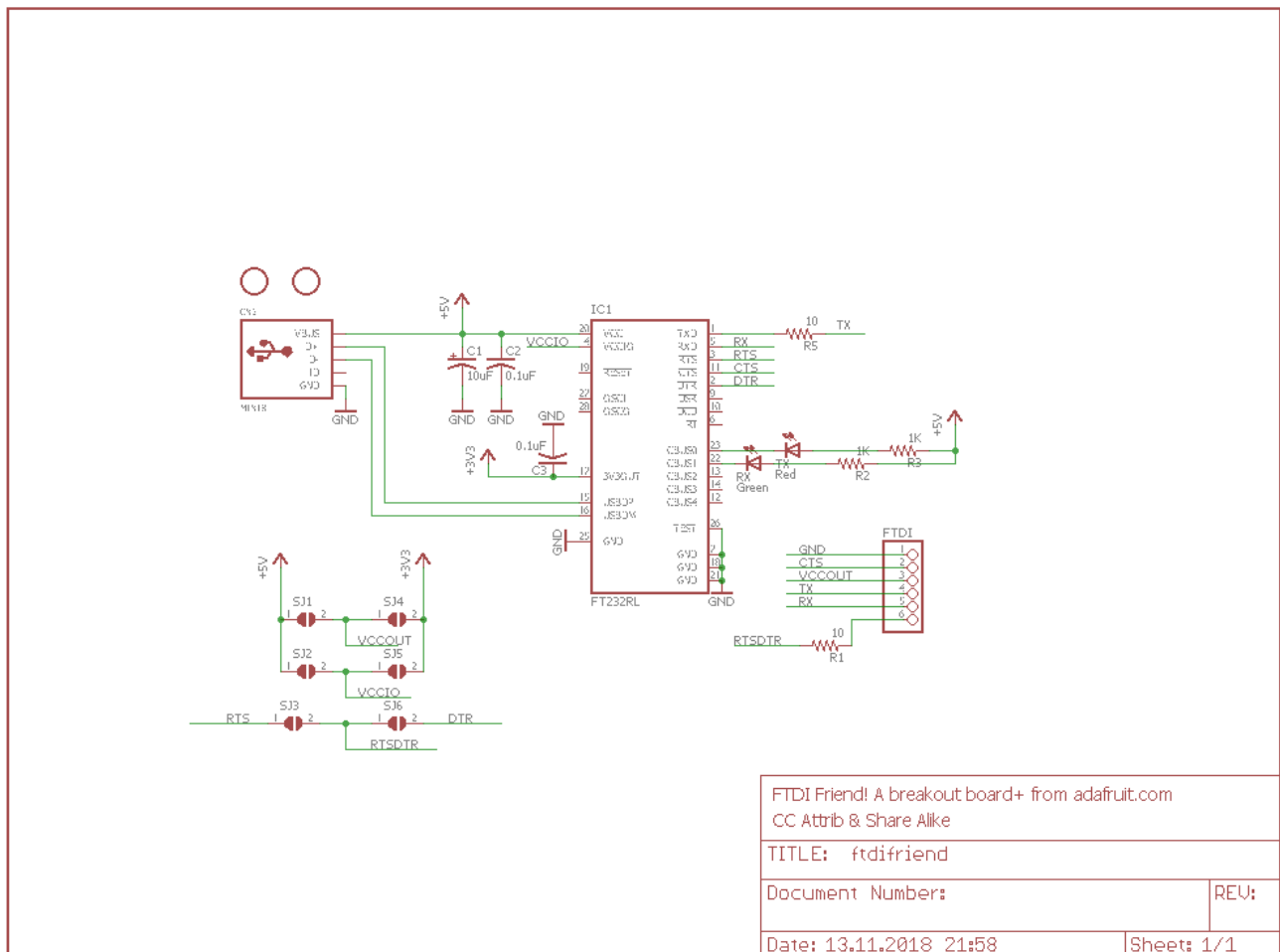
(* = alternative Funktion)

Für weitere Details wird auf das Adafruit Manual verwiesen, welches unter folgendem Link zum Download zur Verfügung steht:

<https://learn.adafruit.com/ftdi-friend/overview>

<https://cdn-learn.adafruit.com/downloads/pdf/ftdi-friend.pdf?timestamp=1550729989>

Schaltplan des FTDI Friend:



Schaltbild 6: Schaltplan des Adafruit FTDI Friend

Layout des FTDI Friend:

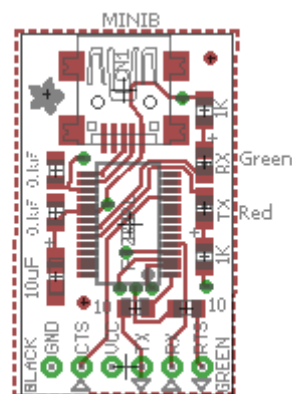


Abbildung 13: Layout des FTDI Friend

9.7.5 FTDI Serial TTL-232 USB Cable

Neben dem FTDI Friend gibt es ein komplett fertiges Kabel FTDI Serial TTL-232 USB Cable von Adafruit welches auch zum Einsatz kommt bzw. kommen kann und die gleiche Funktion abdeckt. Der einzige Unterschied besteht in der festen Konfiguration mit einer Spannungsversorgung von VCC = 5V und einem Signalpegel von 3,3V.

Adafruit Produkt ID: 70



Abbildung 14: FTDI Serial TTL-232 USB Cable

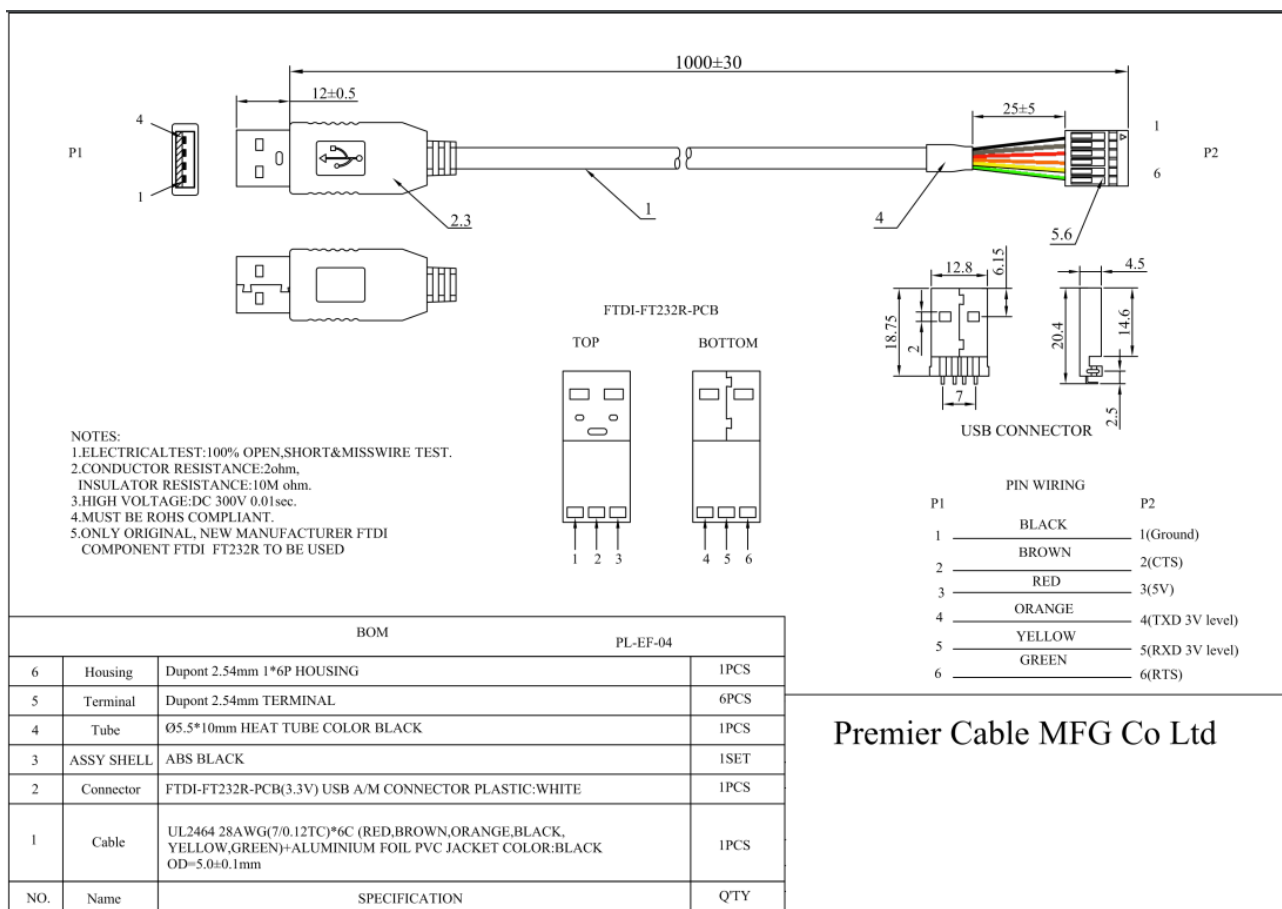


Abbildung 15: FTDI Serial TTL-232 USB Cable Datenblatt



Abbildung 16: FTDI Serial TTL-232 USB Cable Detailansichten

9.8 AVR Fusebits Tutorial

Einleitung - Was sind Fusebits?

Fusebits sind im Grunde genommen nichts anderes als Speicherzellen, die man Programmieren und Löschen kann. Sie dienen jedoch nicht zur Speicherung von Daten, sondern mit ihrer Hilfe kann das Verhalten des AVR beeinflusst werden. Zum Beispiel können bestimmte Funktionen aktiviert und deaktiviert werden.

Was ist zu beachten?

Bevor man zum ersten Mal die Fusebits eines AVR verändert, sollte man wissen das man den Controller damit nicht zerstören kann. Es ist jedoch möglich den Controller so einzustellen das man mit den normalen Werkzeugen nicht mehr darauf zugreifen kann. Grundsätzlich kann ein "verfuster" Controller mit dem richtigen Werkzeug aber wieder repariert werden. Ob dies wirtschaftlich jedoch sinnvoll ist steht auf einem anderen Blatt.

Was anfänglich Probleme verursacht ist die "invertierte Logik" der Fusebits. Ein programmiertes Fusebit ist nicht wie man annehmen könnte auf "1" gesetzt, sondern auf "0". Eine unprogrammierte Fuse ist "1". Manche Programme stellen programmierte Fusebits mit einem gesetzten Häkchen dar (z.B. PonyProg), welches dann als eine 0 (=programmed) interpretiert wird. Möchte man das Fusebit "setzen" oder "programmieren" muss man das Häkchen setzen.

Handelt es sich um ein Enable Fusebit bedeutet ein programmiertes Fusebit (0), dass die Funktion eingeschaltet ist. Ist es ein Disable Fusebit bedeutet ein programmiertes Fusebit (0), dass die Funktion ausgeschaltet ist.

Was braucht man?

Die Fusebits lassen sich über ISP, JTAG oder parallel programmieren. Man kann hierfür die gleiche Hardware und Software verwenden wie zum Programmieren der des Flash Speichers oder des EEPROMs, zum Beispiel PonyProg oder das AVR Studio.

Die Fusebytes

Je nach Controller stehen bis zu drei Fusebytes zur Verfügung: Fuse Low Byte, Fuse High Byte und Extended Fuse Byte. Einige der älteren Controller haben nur ein Fuse Byte und sehr wenige Fusebits.

Die Fusebits

Die folgende Beschreibung listet alle Fusebits auf die es bei den AVR Controllern gibt. Es gibt jedoch keinen Controller in dem alle Fusebits gleichzeitig zu finden sind. Ältere Controller vom Typ AT90S kennen teilweise nur 2 Fusebits.

CKSEL

Die wohl am häufigsten geänderten Fusebits sind CKSEL0 bis CKSEL3 (Select Clock Source). Mit ihrer Hilfe wählt man die Taktquelle aus der der Controller seinen Takt erhält. Hier ist etwas Vorsicht geboten da eine falsche Einstellung den Controller lähmen kann. Eine falsche Einstellung lässt sich jedoch relativ leicht beheben. Die genauen Parameter können zwischen den einzelnen Typen variieren

Default: Interner RC Oszillator mit 1MHz (bzw. 8MHz bei Typen mit Vorteiler)

CKSEL0: 0 (programmiert)
CKSEL1: 1 (unprogrammiert)
CKSEL2: 1 (unprogrammiert)
CKSEL3: 1 (unprogrammiert)

SUT

Mit SUT0 und SUT1 lässt sich die Zeit einstellen wie lange der Reset Impuls nach einem Reset oder Power Up verzögert wird. Je nach Umgebungsbedingung kann die Reset Zeit verlängert oder verkürzt werden. Zusammen mit der Brown Out Detection wird eine externe Resetschaltung (bis auf den üblichen 10kOhm PullUp Widerstand) meist überflüssig.

Default:
SUT0: 0 (programmiert)
SUT1: 1 (unprogrammiert)

CKDIV8

Divide Clock by 8 ist etwas irreführend. Wenn dieses Fusebit gesetzt ist wird ein Vorteiler aktiviert, der den Takt für den Controller durch 8 teilt. Es ist jedoch möglich diesen Vorteiler auf einen anderen Wert einzustellen. Dies ist dann sinnvoll, wenn der Controller aus einer externen Taktquelle gespeist werden soll, die Frequenz aber zu hoch ist. Details dazu im Artikel Taktquelle.

Default:
CKDIV8: 0 (programmiert)

CKOUT

Wird diese Fuse programmiert wird der CPU Takt an dem entsprechenden CLKO Pin ausgegeben.

Default:
CKOUT: 1 (unprogrammiert)

CKOPT

CKOPT kommt zum Einsatz, wenn der AVR von einem externen Quarz getaktet wird. Wird CKOPT programmiert (0) schwingt der Oszillator mit der maximalen Amplitude. Dies kann notwendig werden, wenn der AVR in einer Umgebung mit vielen Störsignalen betrieben werden soll. Ist CKOPT unprogrammiert (1) schwingt der Oszillator mit einer niedrigeren Amplitude. Dadurch verringert sich die Stromaufnahme und die Störabstrahlung.

Default:
CKOPT: 1 (unprogrammiert)

RSTDISBL

Dieses Fuse Bit steuert die Funktion des Reset Pin. Wird es programmiert kann man den Reset Pin als normalen IO Pin verwenden.

Achtung: Wird dieses Bit programmiert kann der Controller nicht mehr über die ISP Schnittstelle erreicht werden

Default:

RSTDISBL: 1 (unprogrammiert)

SPIEN

Mit SPIEN kann die ISP Schnittstelle abgeschaltet werden. Dieses Fusebit lässt sich nur über die parallele Programmierung ändern. Ist die ISP Schnittstelle einmal abgeschaltet kann der Controller nicht mehr über ISP erreicht werden.

Default:

SPIEN: 0 (programmiert)

JTAGEN

JTAGEN aktiviert/deaktiviert die JTAG Schnittstelle.

Default:

JTAGEN: 0 (programmiert)

DWEN

DWEN aktiviert/deaktiviert die debugWire Schnittstelle.

Default:

DWEN: 1 (unprogrammiert)

OCDEN

OCDEN aktiviert/deaktiviert das On-Chip Debug System. Das On-Chip Debug System kann unabhängig von der JTAG Schnittstelle deaktiviert werden. Bei abgeschaltetem OCD kann der Controller über JTAG nur programmiert werden.

Default:

OCDEN: 1 (unprogrammiert)

EESAVE

Wird EESAVE programmiert wird das EEPROM bei einem Chip Erase vor dem Löschen geschützt. Ein Chip Erase löscht normalerweise den kompletten Speicher.

Default:

EESAVE: 1 (unprogrammiert)

BODEN

BODEN aktiviert/deaktiviert die Brown Out Detection. Bei manchen Controllern wird diese Funktion durch die BODLEVEL Fusebits übernommen.

Default:

BODEN: 1 (unprogrammiert)

BODLEVEL

Mit BODLEVEL kann der Spannungswert eingestellt werden, bei dem der Unterspannungsschutz aktiv werden soll. Ältere Controller (z.B. ATmega128) haben nur zwei Schwellwerte. Mit BODLEVEL kann zwischen den Werten gewechselt werden, mit BODEN wird die Funktion komplett deaktiviert. Neuere Controller (z.B. ATmega168) haben 3 BODLEVEL Fusebits mit denen mehrere Schwellwerte eingestellt werden können bzw. die gesamte Funktion deaktiviert wird. Ab Werk ist bei allen Typen die BOD Funktion abgeschaltet.

Default:
BODLEVEL: 1 (unprogrammed)

WDTON

Mit WDTON kann der Watchdog Timer permanent aktiviert werden. Ist dieses Fusebit nicht programmiert (1) kann der Watchdog per Software gesteuert werden.

Default:
WDTON: 1 (unprogrammiert)

BOOTRST

BOOTRST bestimmt an welche Adresse nach einem Reset gesprungen wird. Unprogrammiert (1) springt der Controller nach einem Reset an Adresse \$0000. Wird das Fusebit programmiert springt der Controller nach einem Reset an den Beginn des Bootloaders. Die Adresse hängt vom Controller und von den Einstellungen der BOOTSZ Fusebits ab.

Default:
BOOTRST: 1 (unprogrammiert)

BOOTSZ

Mit BOOTSZ wird die Größe des Speicherbereiches bestimmt, der für den Bootloader reserviert wird. Die Größe ist abhängig vom Controllertyp. Dieser Speicherbereich befindet sich immer am Ende des Flash Adressraumes.

Default:
BOOTSZ: siehe Datenblatt

Compatibility Bits

Viele Controller haben ein Compatibility Bit. Mit diesem Bit lässt sich der Controller in einen Modus versetzen, in dem er sich exakt so verhält wie sein Vorgänger. Beim ATmega128 gibt es z.B. das M103C Bit. Der ATmega128 verhält sich also wie ein ATmega103.

Ob das Compatibility Bit ab Werk programmiert ist oder nicht hängt vom Controller ab.

SELFPRGEN

SELFPRGEN aktiviert/deaktiviert die Self Programming Funktion.

Default:
SELFPRGEN: 1 (unprogrammiert)

HWBEN

HWBEN aktiviert/deaktiviert die Hardware Boot Funktion

Default:
HWBEN: 0 (programmiert)

Sollte ich ein Fusebit vergessen haben oder neue dazukommen bitte ergänzen.

Fusebits mit dem AVR Studio programmieren:

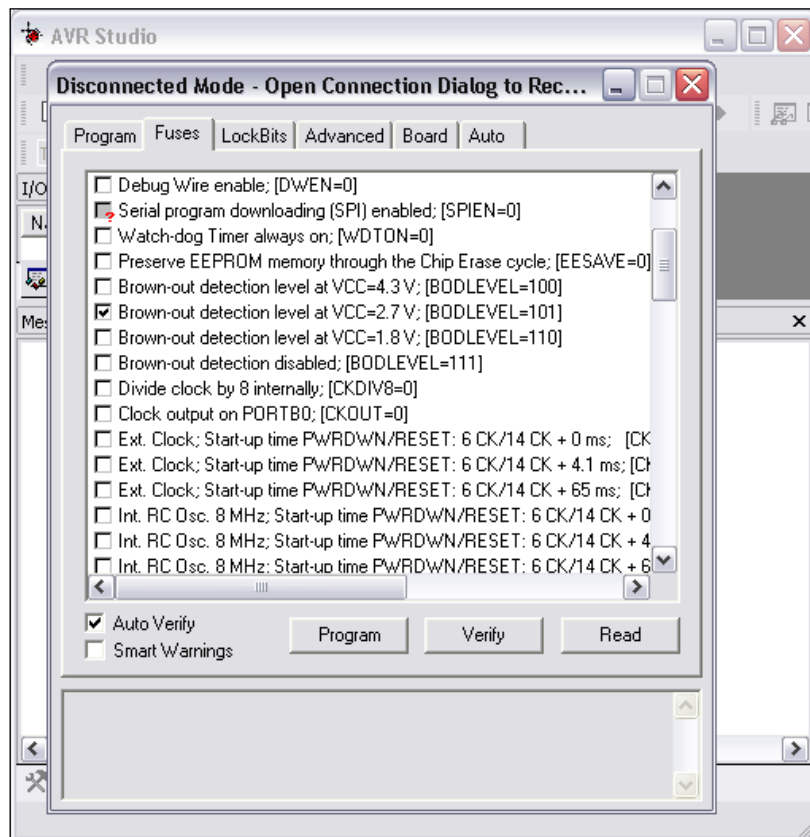


Abbildung 17: Fusebits im AVR Studio

Am einfachsten und intuitivsten lassen sich die Fusebits mit dem AVR Studio programmieren. Für jedes Fusebit gibt es eine kurze Beschreibung und den Default Wert. Gibt es für eine Funktion mehrere Fusebits wird für jede Kombination ein Häkchen mit Beschreibung und Bitkombination angezeigt.

In diesem Beispiel (ATmega168) sieht man z.B. dass die Brown Out Detection auf 2,7V eingestellt wurde.

9.9 AVR Fuse Konfiguration ATmega8L

Der ATmega8L besitzt zwei Fuse-Bytes.

Fuse Bytes des ATmega8L:

The ATmega8 has two fuse bytes. Table 87 and Table 88 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 87. Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL ⁽⁴⁾	7	Select if PC6 is I/O pin or RESET pin	1 (unprogrammed, PC6 is RESET-pin)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN ⁽¹⁾	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT ⁽²⁾	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 82 for details)	0 (programmed) ⁽³⁾
BOOTSZ0	1	Select Boot Size (see Table 82 for details)	0 (programmed) ⁽³⁾
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

- Notes:
1. The SPIEN Fuse is not accessible in Serial Programming mode.
 2. The CKOPT Fuse functionality depends on the setting of the CKSEL bits, see "Clock Sources" on page 26 for details.
 3. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 82 on page 220.
 4. When programming the RSTDISBL Fuse Parallel Programming has to be used to change fuses or perform further programming.

Table 88. Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed) ⁽¹⁾
SUT0	4	Select start-up time	0 (programmed) ⁽¹⁾
CKSEL3	3	Select Clock source	0 (programmed) ⁽²⁾
CKSEL2	2	Select Clock source	0 (programmed) ⁽²⁾
CKSEL1	1	Select Clock source	0 (programmed) ⁽²⁾
CKSEL0	0	Select Clock source	1 (unprogrammed) ⁽²⁾

- Notes:
1. The default value of SUT1..0 results in maximum start-up time. See Table 10 on page 30 for details.
 2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 1MHz. See Table 2 on page 26 for details.

The status of the Fuse Bits is not affected by Chip Erase. Note that the Fuse Bits are locked if lock bit1 (LB1) is programmed. Program the Fuse Bits before programming the Lock Bits.

The fuse values are latched when the device enters Programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

Festlegung der Fuse-Bits für das Projekt myGeigerle:

Fuse High Byte \$D9:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RSTDISBL	WDTON	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST
1	1	0	1	1	0	0	1

Tabelle 15: Fuse High Byte ATmega8L

Fuse Low Byte - \$FF:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
1	1	1	0	1	0	0	0

Tabelle 16: Extended Fuse Byte ATmega8L

Bemerkung:

0 = programmiert (!! negative Logik !!)

1 = unprogrammiert (!! negative Logik !!)

	Default (keine Programmierung)
	Wichtige Programmierung (abweichend von Default)
	Sonstige Systemeinstellung abweichend von Default)

CKSEL3-0 ist im Originalzustand laut Datenblatt auf 0001. Das entspricht dem internen Oszillator mit 1MHz. Uns interessiert aber der Zustand mit einer internen Taktung von 8 MHz. Dazu muss CKSEL3 auf 1 gesetzt werden. Die Start-Up-Time wird durch CKSEL0 und SUT bestimmt. Eine 0 auf CKSEL0 bedeutet zusammen mit 10 auf SUT1-0 eine Startverzögerung von 6 CK + 64 ms. Wie beim ATmega üblich, bedeutet eine Null ein Häkchen und bei einer Eins bleibt das Feld leer.

9.9.1 Die Fuse-Konfiguration von myGeigerle

Die folgenden Screenshots beziehen sich auf die Programmversion 7 des ATMEL Studios. Frühere Versionen weichen von davon ab, funktionieren aber mit Windows 10 nicht mehr.

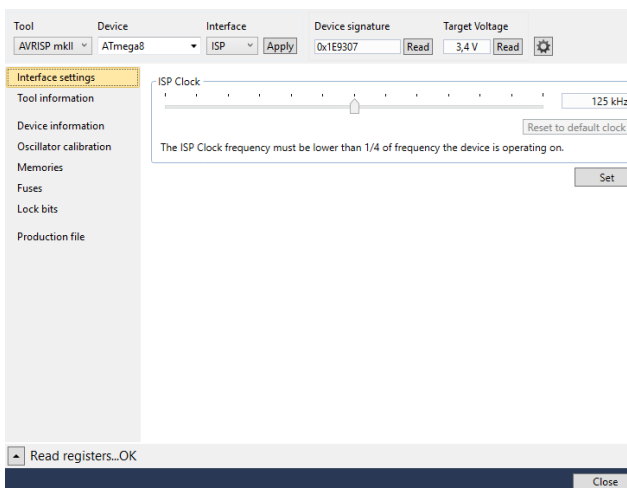


Tabelle 17: AVR-Studio – Interface settings

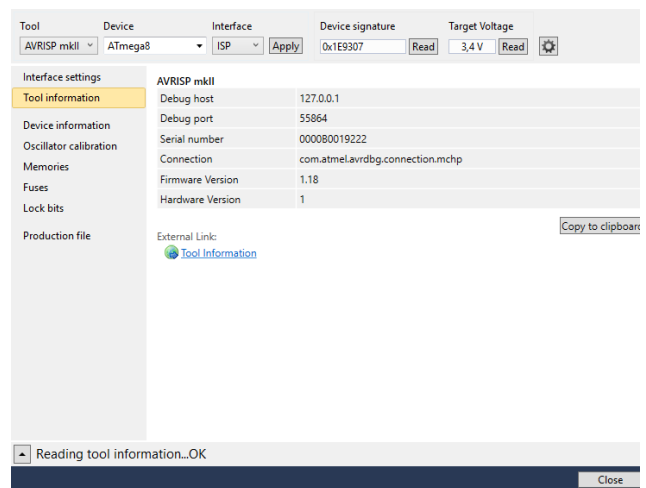


Tabelle 18: AVR-Studio – Tool information

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Detected device

Device names

Device signature

Datasheet information

	ATm ega8 L-8A	ATm ega8 L-8P	ATm ega8 L-8	ATm ega8 -16A	ATm ega8 -16P	ATm ega8 -16	ATm ega8 L-8P	ATm ega8 L-8A	ATm ega8 L-8	ATm ega8 -16A	ATm ega8 -16P	ATm ega8 -16
CPU	U	U	MU	U	U	MU	N	N	MN	N	N	MN

Reading tool information...OK

Close

Tabelle 19: AVR-Studio – Device information

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Calibrate for frequency

Value:

Address:

Memory:

Read

Write

Ensure that the location is unprogrammed before writing. If the location contains data, the write can fail with a validation error. To unprogram the entire chip, use Erase Chip on the Memories page.

Reading tool information...OK

Close

Tabelle 20: AVR-Studio – Oscillator calibration

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Device

Erase Chip

Erase now

Flash (8 KB)

D:\Dokumente\Elektronik\ATMEGA_Projekte\MyGeigerle\MyGeigerle.hex

☒ Erase device before programming

☒ Verify Flash after programming

Advanced

☒ Verify EEPROM after programming

Advanced

OK

Close

Tabelle 21: AVR-Studio - Memories

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Fuse Name

Value

☒ HIGH.RSTDISBL
 ☐ HIGH.WDTON
 ☒ HIGH.SPIEN
 ☒ HIGH.EESAVE
 ☒ HIGH.BOOTSZ
 ☐ HIGH.BOOTRST
 ☐ HIGH.CKOPT
 ☒ LOW.BODLEVEL
 ☐ LOW.BODEN
 ☒ LOW.SUT_KSEL

Boot Flash size=1024 words Boot address=\$0C00

Brown-out detection at VCC=2,7 V

Ext. Crystal/Resonator High Freq.: Start-up time: 16K CK + 64 ms

Fuse Register

Value

HIGH

0xD9

LOW

0xFF

☒ Auto read
 ☒ Verify after programming

Copy to clipboard

Read registers...OK

Close

Tabelle 22: AVR-Studio – Fuses

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Lock Bit

Value

☒ LOCKBIT.LB
 ☒ LOCKBIT.BLB0
 ☒ LOCKBIT.BLB1

No memory lock features enabled

No lock on SPM and LPM in Application Section

No lock on SPM and LPM in Boot Section

Lock Bit Register

Value

LOCKBIT

0xFF

☒ Auto read
 ☒ Verify after programming

Copy to clipboard

Read registers...OK

Close

Tabelle 23: AVR-Studio – Lock bits

Tool	Device	Interface	Device signature	Target Voltage
AVRISP mkII	ATmega8	ISP	0x1E9307	3,4 V

Interface settings

Tool information

Device information

Oscillator calibration

Memories

Fuses

Lock bits

Production file

Program device from ELF production file

☒ Erase memory before programming
 ☒ Verify programmed content

Save to ELF production file

Flash:

EEPROM:

User Signatures:

File Content

File: Not found

OK

Close

Tabelle 24: AVR-Studio – Production file

© Markus Fulde, 2019

Seite 40 von 106

9.10 Grundlagen zur Spannung 5V, Vcc und VDD

Für die Spannungsversorgung des Projekts myGeigerle wird die Spannung +5V und GND bereitgestellt. Diese wird aus einem 9V Block mittels Festspannungsregler erzeugt.

Unterschiedliche Technologien haben unterschiedliche Bezeichnungen für die notwendigen Spannungsversorgungen.

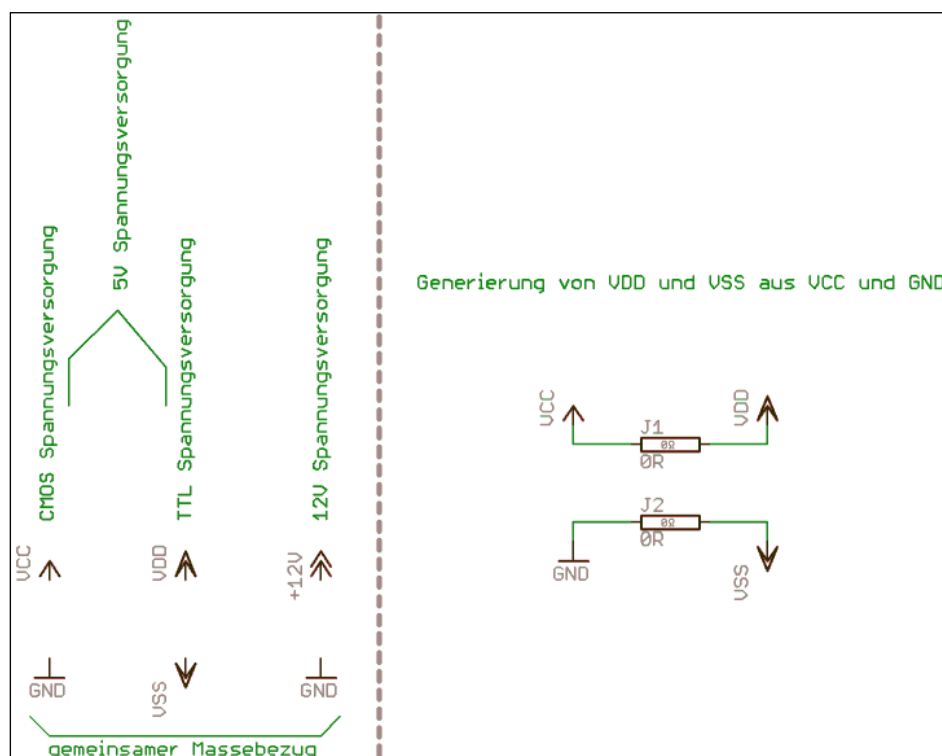
Für CMOS Bausteine gelten die Bezeichnungen VCC und GND.

TTL-Logik verwendet die Bezeichnungen VDD und VSS. VDD und VSS werden im Projekt durch 0Ω-Widerstände (Drahtbrücken) erzeugt, um EAGLE die korrekte Behandlung der Spannungen zu ermöglichen.

Für das Projekt gelten die folgenden Bezeichnungen in den Netzen:

Spannungspotential	Verwendeter Name
GND (Ground) = MASSE = 0V	GND VSS
+5V Spannung	VCC VDD
+12V	+12V

Tabelle 25: Vorgeschriebene Namensgebung für Spannungsversorgungen



Schaltbild 7: Symbolfestlegung für Spannungsversorgungen

10 Elektronische Teilkomponenten

10.1 Hauptschalter

Das Gesamtsystem (myGeigerle + MightyOhm Geiger) kann durch einen kleinen Hauptschalter ein- und ausgeschaltet werden. Folgender Hauptschalter wurde ausgewählt:

Cylewet vertikaler Schiebeschalter / Wechselschalter, 12 mm, mit 3 Pins, PCB-Panel, für Arduino, CLW1016, 10 Stück

12mm

Amazon:

AISIN Nr. B071P5VD49

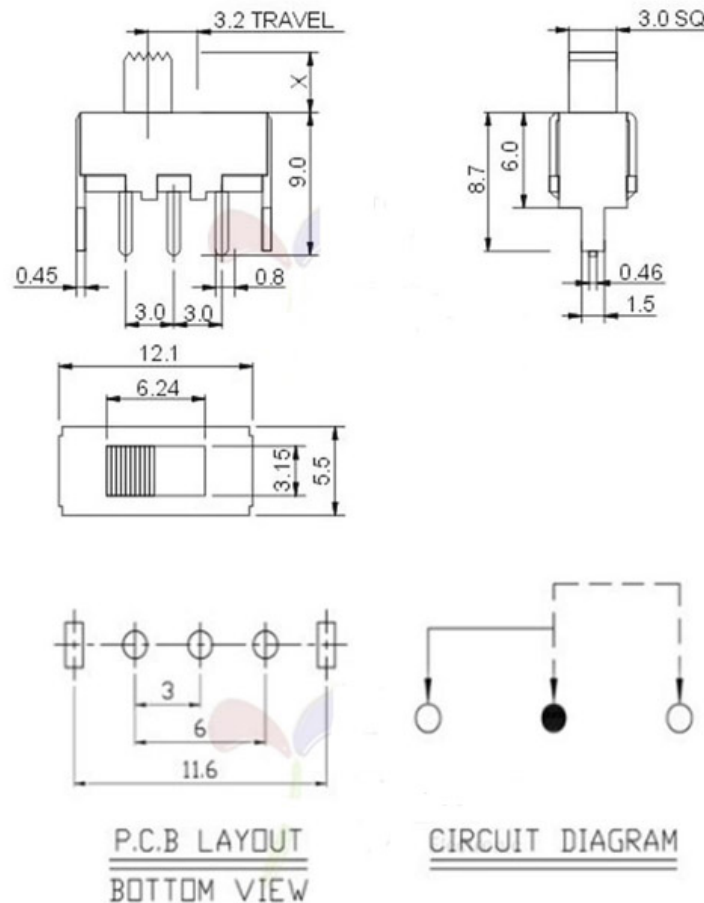
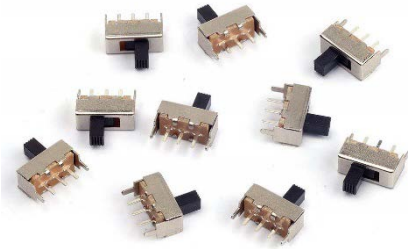


Abbildung 18: Hauptschalter

10.2 Spannungsversorgung

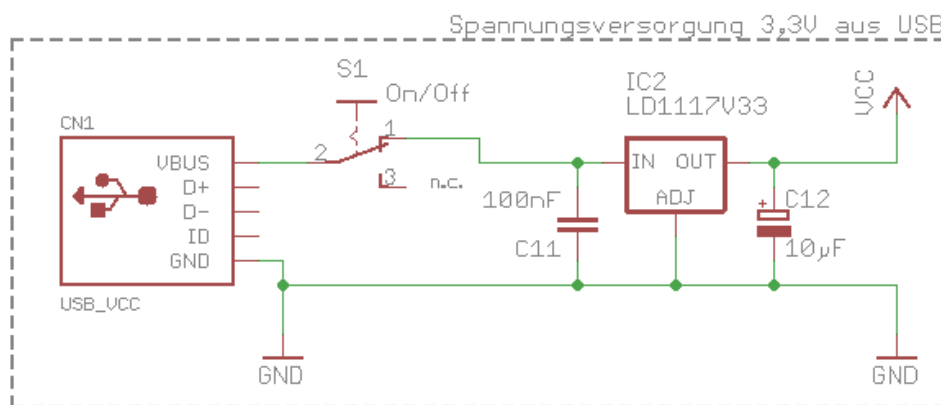
Achtung: Beim Betrieb des MightyOhm Geigers über myGeigerle müssen die Batterien aus dem MightyOhm Geiger entfernt werden!

Die Spannungsversorgung des Geigerzählers myGeigerle erfolgt mittels 5V welche über einen Standard Mini USB 2.0 Typ B Male Stecker zugeführt werden kann. Die Stromaufnahme beträgt ca. 250 - 300 mA so dass sich jedes Universalnetzteil oder jeder PC USB-Anschluss dazu eignet.

Für die Aufnahme zweier Batterien im MightyOhm Gehäuse ist ein Batteriefach vorhanden. Die Batterien müssen jedoch entfernt werden, da die Spannungsversorgung von außen erfolgt und mittels Jumper/Kabelbrücke vom myGeigerle zum MightyOhm Counter erfolgt.

Die Elektronik des Displays und des MightyOhm Counters arbeitet mit 3V bzw. 3,3V. Diese Spannung wird mittels Festspannungsregler aus den 5V USB-Spannung erzeugt.

10.2.1 Beschaltung des Festspannungsreglers:



Schaltbild 8: Beschaltung des Festspannungsreglers

Bauteile:

Stückliste: Beschaltung Spannungsversorgung			
Kondensatoren		Halbleiter	
C11	Folienkondensator 100nF	IC2	3,3V Festspannungsregler LD1117V33
C12	Elektrolytkondensator 10µF	CN1	Einbaubuchse Mini USB Buchse

Tabelle 26: Stückliste Beschaltung Festspannungsregler

Als Einbaubuchse Mini USB kommt das folgende Produkt von Conrad Elektronik zum Einsatz:



Mini USB Typ B liegend 5-polig WR-COM Buchse, Einbau horizontal WR-COM Würth Elektronik

Conrad Electronic:

Bestell-Nr.: 1088421 – 62

Hst.-Teile-Nr.: 651005136521

EAN: 2050002388277

USB 2.0 – 5 polig

Breite 7,8mm

Höhe 6,5 mm

Länge 8,15 mm

10.3 LCD-Grafik-Display

Allgemeines:

Als Anzeigedisplay wird ein LCD-Grafik-Modul der Firma ELECTRONIC ASSEMBLY der DOG_Serie 3,3V EA DOG-M Super Flach / 55x46 mm inkl. Kontroller ST7565R für SPI eingesetzt.

Als Hintergrundbeleuchtung wird LED-Beleuchtung Grün/Gelb eingesetzt.

Zum Einsatz kommt das folgende Display:

LCD-Modul 128x64 Pixel

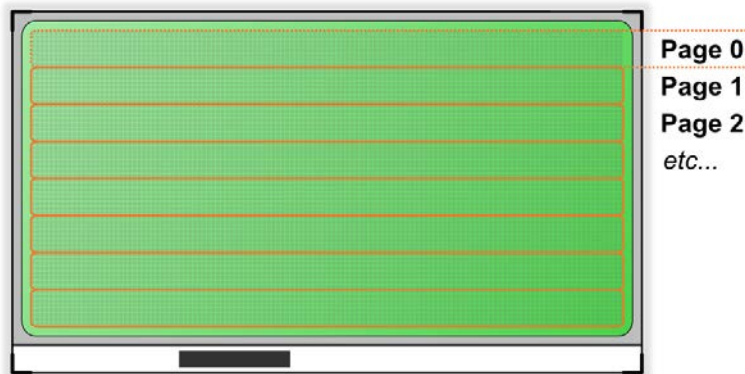
EA DOGM128E-6

LED Hintergrundbeleuchtung Grün/Gelb

EA LED55X46-G

EA DOGM128E-6:

Auflösung 128x64 Pixel



ST7565 - Pages

Abbildung 19: LCD EA DOGM128E-6 128x64 Pixel

Für das LCD-Display wird eine Grün/Gelb LED-Hintergrundbeleuchtung eingesetzt.



Abbildung 20: LED Hintergrundbeleuchtung

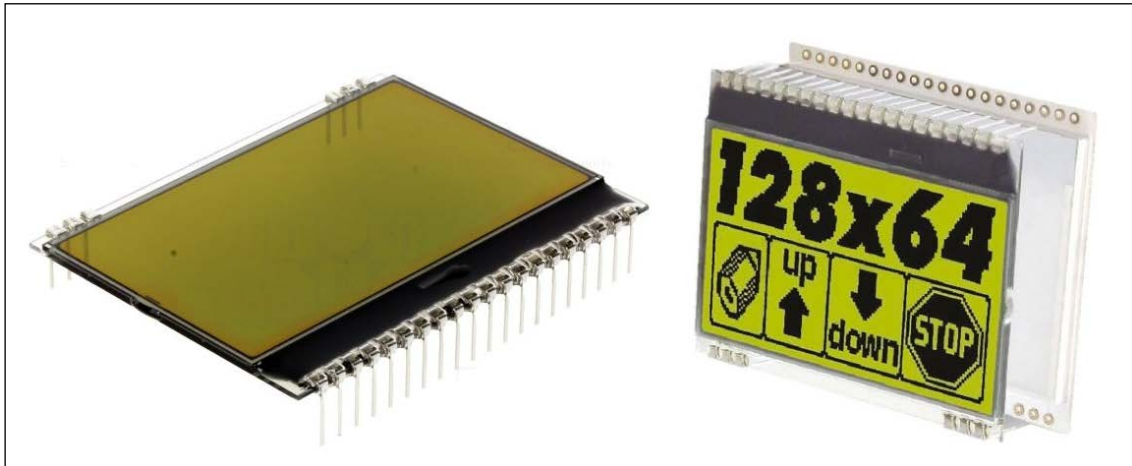


Abbildung 21: LCD-Display EA DOG-M 128x-6

Allgemeine Technische Daten zum Display:

- Kontrastreiche LCD-Supertwist Anzeige
- Optionale LED-Beleuchtungskörper in verschiedenen Farben
- 128x64 Punkte (entspricht 8x21 Zeichen oder 4x16 Zeichen Groß)
- Controller ST 7565R mit SPI (4-Draht) Interface
- Spannungsversorgung Single Supply +3,3V (typ. 270µA)
- Keine zus. Spannungen erforderlich
- Betriebstemperaturbereich -20...+70°C
- LED-Hintergrundbeleuchtung 5..40mA
- Keine Montage erforderlich: einfach nur in PCB einlöten.

Abmessungen:

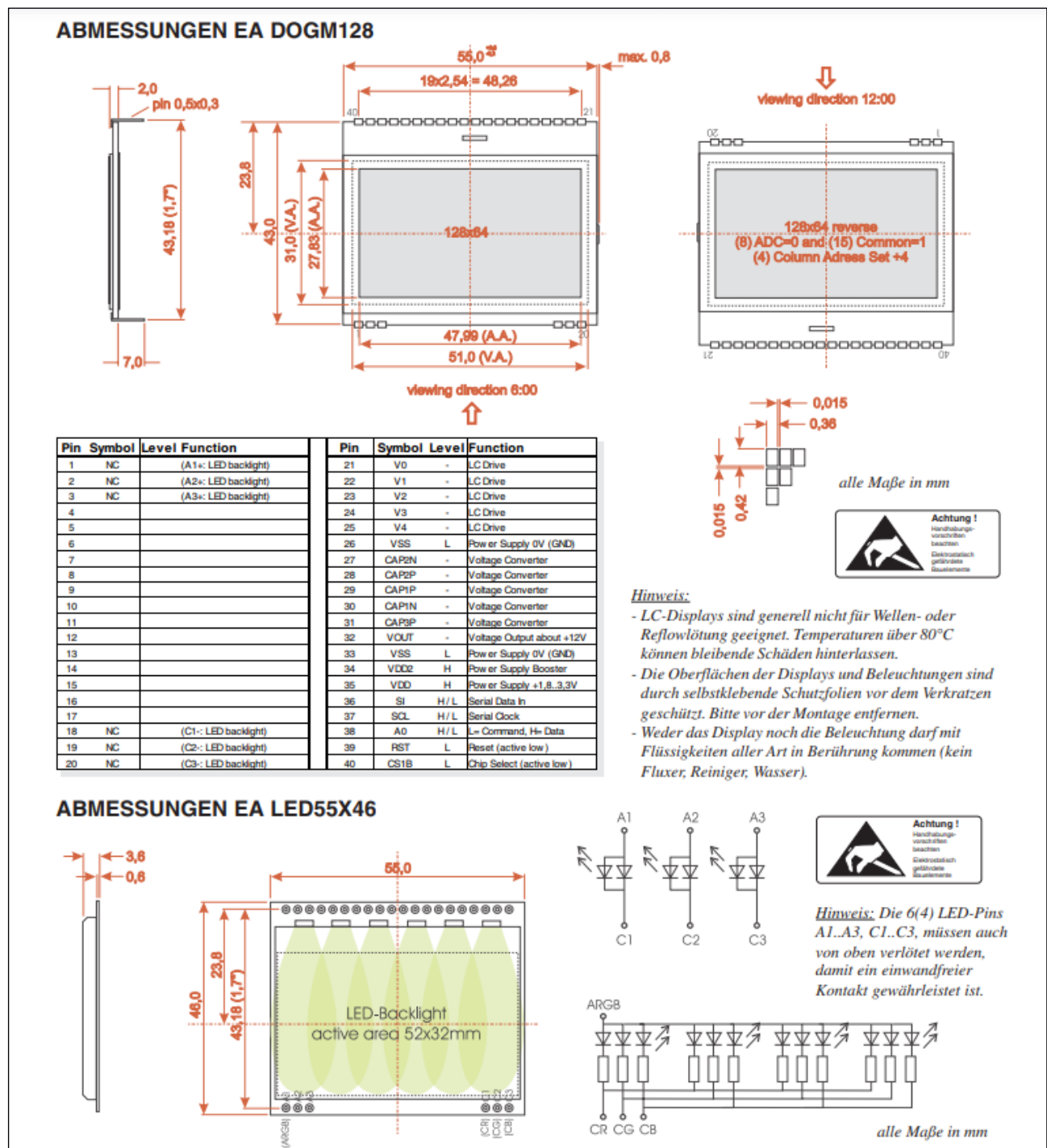


Abbildung 22: Abmessungen und technische Daten DOGM LCD-Display

Kontrasteinstellung:

Für alle Displays der EA DOGM- Serie ist der Kontrast per Befehl einstellbar. In der Regel wird der Kontrast einmalig eingestellt und dann - dank integrierter Temperaturkompensation - über den gesamten Betriebstemperaturbereich (-20..+70°C) konstant gehalten. Dieser einmalige Abgleich ist jedoch in jedem Fall erforderlich.

LED-Hintergrundbeleuchtung für LCD-Display:

LED backlight	Forward voltage (V)			Current max. (mA)	Limiting resistor (ohm, each path)	
	min	typ	max		@3,3 V	@5 V
yellow/green EA LED55x46-G		2,1		40	40	97
white EA LED55x46-W		3,3		25	CAT4238	82
amber EA LED55x46-A		1,9		40	47	103
green EA LED55x46-E		3		40	8,5	57
blue EA LED55x46-B		3,3		30	CAT4238	68
red EA LED55x46-R		1,9		40	47	103
full color EA LED55x46-RGB	red	green	blue	40 each colour	CAT4238	25
	typ. 4V	typ. 4V	typ. 4V			

Für die Beschaltung der Hintergrundbeleuchtung wird ein Vorwiderstand von 30 Ohm benötigt. Eine Serienschaltung der LED's ist nicht möglich, weil hierfür eine Forward-Spannung von 6,4V benötigt wird, das Display aber nur mit 3,3V betrieben wird. Somit kommt nur Parallelschaltung in Frage!

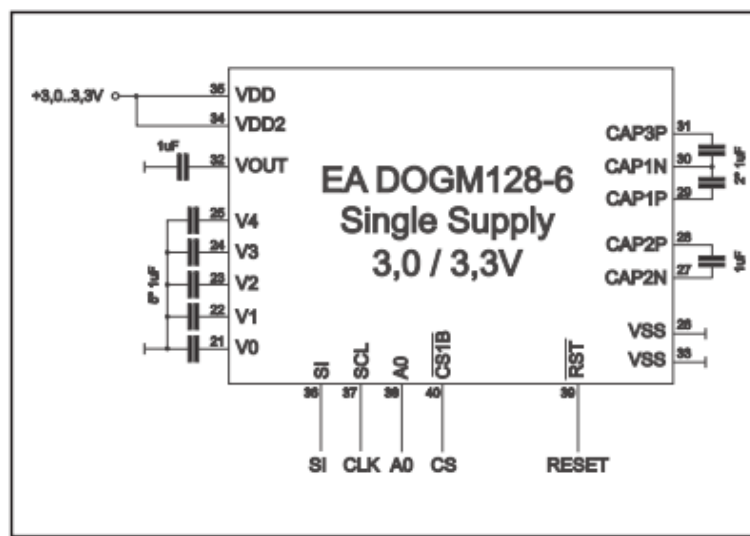
Tabelle 27: LED-Hintergrundbeleuchtung für LCD-Display

Die Hintergrundbeleuchtung wird mittels 100 Ohm Trimmer-Potentiometer einstellbar gemacht. Eine PWM kommt in diesem Projekt nicht in Frage da der Timer2 für den Sekundenimpuls benötigt wird und der notwendige OC2 Ausgang für das SPI Interface.

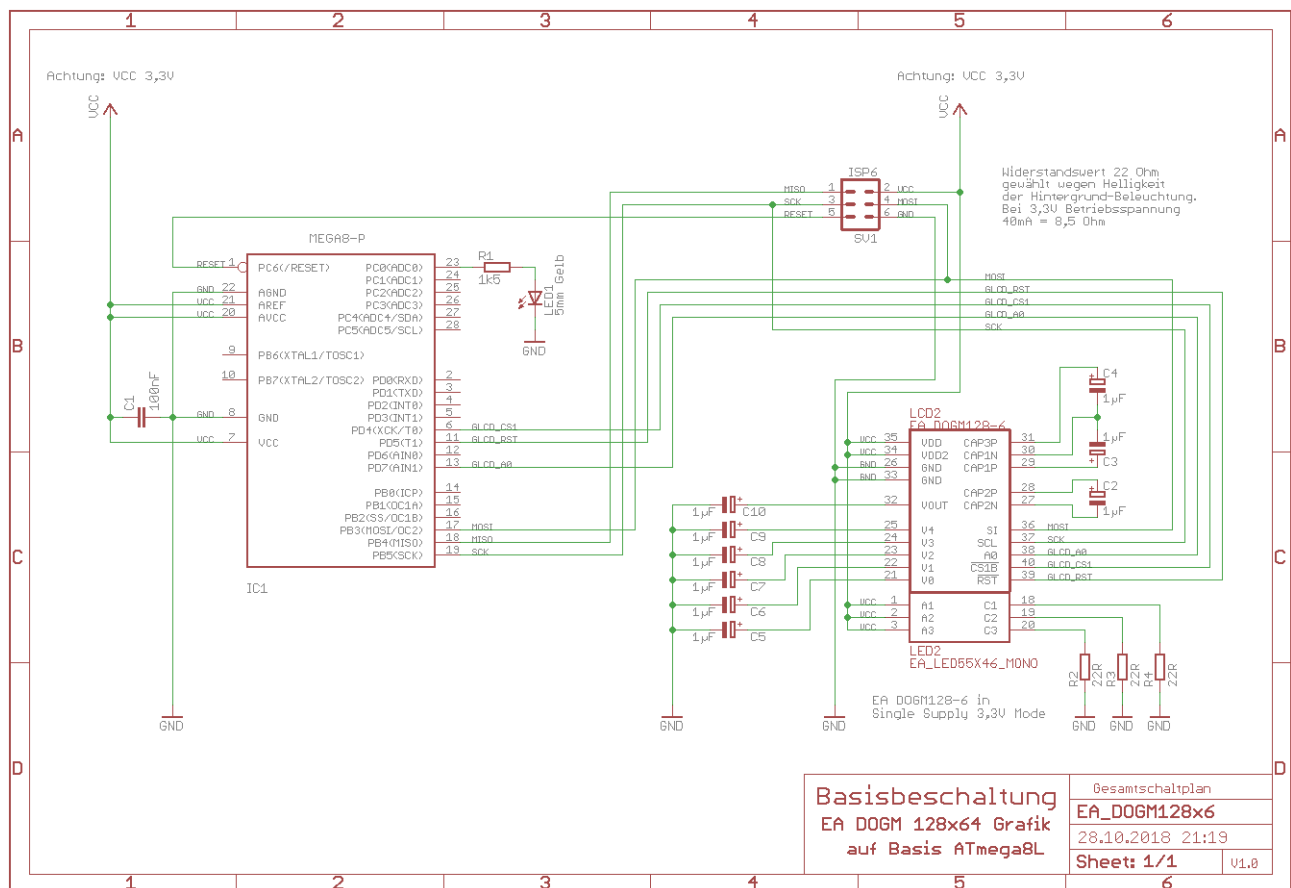
Das LCD-Display muss vor dem verlöten oder stecken auf die Beleuchtungseinheit gelötet werden, damit die LEDs der Beleuchtung über die Kontakte des LCDs Strom bekommen. Es ist ratsam, alle Pins der Beleuchtungseinheit anzulöten, da sich so der Druck beim Einsetzen des Displays in die Fassung besser verteilt. Elektrisch ist dies nicht notwendig. Bitte sehr sparsam mit dem Lötzinn umgehen, da es sonst an den Beinchen herunterläuft und somit das Display nicht in die Fassung passt.

10.3.1 Beschaltung des LCD-Display

Die Beschaltung des LCD-Display an den ATmega erfolgt wie im folgenden Schaltbild aus dem Datenblatt des LCD-Displays gefordert:



Schaltbild 9: Beschaltung des LCD-Displays gemäß Datenblatt



Schaltbild 10: Schematische Basisbeschaltung des LCD-Displays

Bauteile:

Stückliste: Basisbeschaltung ATmega8L mit LCD-Grafik-Display			
Widerstände		Halbleiter	
R1	Metallschichtwiderstand 1k5 Ω	IC1	ATMEL AVR ATmega8L-P
R2, R3, R4	Metallschichtwiderstand 22 Ω	LCD2	LCD-Grafik-Display EA DOGM128x-6
		LED1	Low-Current LED, gelb
		LED2	LED Hintergrundbeleuchtung EA_LED55x46
Kondensatoren		Sonstiges	
C1	Keramikkondensator 100nF	SV1	Federleiste MA03-2 (ISP)
C2 – C10	Elektrolytkondensator 1µF		

Tabelle 28: Stückliste Basisbeschaltung ATmega8L

Ressourcenzuordnung zum ATmega8L:

Nummer	Schaltbild	Ressource ATmega8L	
1	SPI: SI	PortB.3	[MOSI] MOSI
2	SPI: SCK	PortB.5	[SCK] SCK
3	SPI: AO	PortD.7	[PD7] (GPIO)
4	SPI: CS	PortD.4	[PD4] (GPIO)
5	SPI: RST	PortD.5	[PD5] (GPIO)

Tabelle 29: Ressourcenzuordnung ATmega8L für LCD-Display

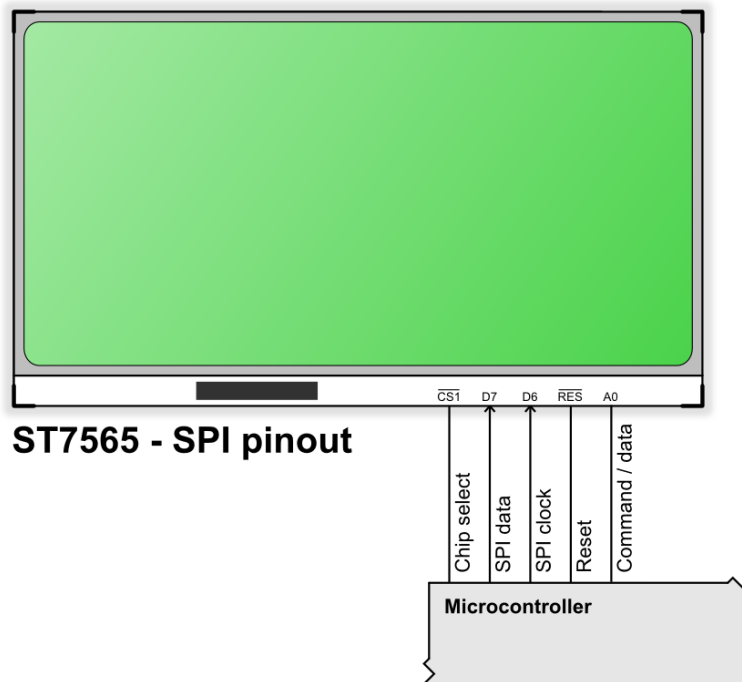


Abbildung 23: ST7565 – SPI Pinout

10.3.2 Befehlstabellen

Befehlstabellen des Displays:

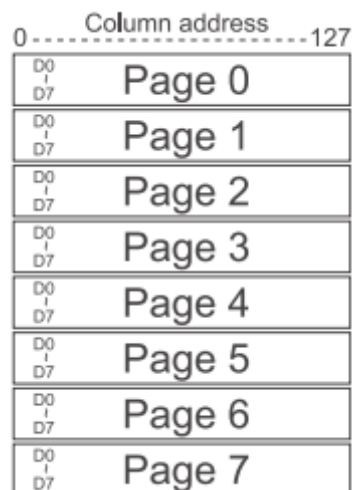


Abbildung 24: Page-Organisation des Displays

Command	Command Code									Function
	A0	D7	D6	D5	D4	D3	D2	D1	D0	
(1) Display ON/OFF	0	1	0	1	0	1	1	1	0	LCD display ON/OFF 0: OFF, 1: ON
(2) Display start line set	0	0	1	Display start address						Sets the display RAM display start line address
(3) Page address set	0	1	0	1	1	Page address				Sets the display RAM page address
(4) Column address set upper bit	0	0	0	0	1	Most significant column address				Sets the most significant 4 bits of the display RAM column address.
Column address set lower bit		0	0	0	0	Least significant column address				Sets the least significant 4 bits of the display RAM column address.
(6) Display data write	1	Write data								Writes to the display RAM
(8) ADC select	0	1	0	1	0	0	0	0	0	Sets the display RAM address SEG output correspondence 0: normal, 1: reverse
(9) Display normal/ reverse	0	1	0	1	0	0	1	1	0	Sets the LCD display normal/ reverse 0: normal, 1: reverse
(10) Display all points ON/OFF	0	1	0	1	0	0	1	0	0	Display all points 0: normal display 1: all points ON
(11) LCD bias set	0	1	0	1	0	0	0	1	0	Sets the LCD drive voltage bias ratio 0: 1/9 bias, 1: 1/7 bias (ST7565R)
(14) Reset	0	1	1	1	0	0	0	1	0	Internal reset
(15) Common output mode select	0	1	1	0	0	0	*	*	*	Select COM output scan direction 0: normal direction 1: reverse direction
(16) Power control set	0	0	0	1	0	1	Operating mode			Select internal power supply operating mode
(17) V ₀ voltage regulator internal resistor ratio set	0	0	0	1	0	0	Resistor ratio			Select internal resistor ratio(Rb/Ra) mode
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	Set the V ₀ output voltage electronic volume register
Electronic volume register set		0	0	Electronic volume value						
(19) Static indicator ON/OFF	0	1	0	1	0	1	1	0	0	0: OFF, 1: ON
Static indicator register set		0	0	0	0	0	0	0	Mode	Set the flashing mode
(20) Booster ratio set	0	1	1	1	1	1	0	0	0	select booster ratio 00: 2x,3x,4x 01: 5x 11: 6x
(21) Power save	0									Display OFF and display all points ON compound command
(22) NOP	0	1	1	1	0	0	0	1	1	Command for non-operation

Tabelle 30: Befehlstabelle des Grafik-Displays EA-DOGM128x-6

Eine detaillierte Beschreibung des hier integrierten Controllers ST7036 finden Sie im Internet unter <https://www.lcd-module.de/pdf/grafik/dogm128.pdf>

Initialisierungsbeispiel für EA DOGM 128x-6 (Single supply +3,0V / +3,3V):

Initialisation example for single supply +3.3V (bottom view)											
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark
(2) Display start line set	0	0	1	0	0	0	0	0	0	\$40	Display start line 0
(8) ADC set	0	1	0	1	0	0	0	0	1	\$A1	ADC reverse *)
(15) Common output mode select	0	1	1	0	0	0	0	0	0	\$C0	Normal COM0~COM63
(9) Display normal/reverse	0	1	0	1	0	0	1	1	0	\$A6	Display normal
(11) LCD bias set	0	1	0	1	0	0	0	1	0	\$A2	Set bias 1/9 (Duty 1/65)
(16) Power control set	0	0	0	1	0	1	1	1	1	\$2F	Booster, Regulator and Follower on
(20) Booster ratio set	0	1	1	1	1	1	0	0	0	\$F8	Set internal Booster to 4x
		0	0	0	0	0	0	0	0	\$00	
(17) V0 voltage regulator set	0	0	0	1	0	0	1	1	1	\$27	Contrast set
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	\$81	
		0	0	0	1	0	1	1	0	\$16	
(19) Static indicator set	0	1	0	1	0	1	1	0	0	\$AC	No indicator
		0	0	0	0	0	0	0	0	\$00	
(1) Display ON/OFF	0	1	0	1	0	1	1	1	1	\$AF	Display on

Tabelle 31: LCD-Display EA DOGM081 Initialisierungsbeispiel (Single power supply)

Initialisierungsbeispiel für EA DOGM 128x-6 (dual power supply):

Initialisation example for dual power supply (bottom view)											
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark
(2) Display start line set	0	0	1	0	0	0	0	0	0	\$40	Display start line 0
(8) ADC set	0	1	0	1	0	0	0	0	1	\$A1	ADC reverse *)
(15) Common output mode select	0	1	1	0	0	0	0	0	0	\$C0	Normal COM0~COM63
(9) Display normal/reverse	0	1	0	1	0	0	1	1	0	\$A6	Display normal
(11) LCD bias set	0	1	0	1	0	0	0	1	0	\$A2	Set bias 1/9 (Duty 1/65)
(16) Power control set	0	0	0	1	0	1	0	1	1	\$2B	Booster off, Regulator, Follower on
(17) V0 voltage regulator set	0	0	0	1	0	0	1	1	1	\$27	Contrast set
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	\$81	
		0	0	0	1	0	1	1	0	\$16	
(19) Static indicator set	0	1	0	1	0	1	1	0	0	\$AC	No indicator
		0	0	0	0	0	0	0	0	\$00	
(1) Display ON/OFF	0	1	0	1	0	1	1	1	1	\$AF	Display on

Tabelle 32: LCD-Display EA DOGM081 Initialisierungsbeispiel (dual power supply)

10.3.3 BASCOM Beispielcode für die Displayansteuerung (Funktionsbibliothek)

```
#####
' LCD-Display.BAS                                     Stand 17.04.2017
' -----                                     (C) Markus Fulde

' Testprogramm zur Inbetriebnahme des Display DOGM128x64 Grafik-Display
'
' Die Inbetriebnahme erfolgt mit einem Steckbrett auf Basis ATmega8L
'
' Die Spezialität bei BASCOM mit diesem Display besteht in der Verwendung einer
' speziellen Library und der nicht vollständigen Unterstützung des Displays
' durch BASCOM. BASCOM selbst bietet nur Zugriffsmöglichkeiten und Textfunk-
' tionen aber keine Grafikfunktionen.
' Diese werde in diesem Beispielprogramm von Hand dazuprogrammiert.
'
' Die für die LCD-Display-Ansteuerung notwendigen Routinen sind
' als solche gekennzeichnet und bereits zur späteren Verwendung
' ausgelagert.
'
' Die Anbindung des Display erfolgt über SPI mit einer Betriebsspannung von
' 3.3V
'
#####

' Die folgenden BASCOM Befehle funktionieren:
' - Glcdcmd
' - Glcddata
' - Setfont
' - Lcdat
' - SHOWPIC
' - Display On | Off

' Die folgenden Befehle funktionieren nicht
' - Cursor On|Off Blink|Noblink
' - Deflcdchar

' Probleme Und Nachteile Dieses Displays:
' - Das Display verfügt nur über eine 3.3V Logik und kann somit nicht an 5V
' betrieben werden bzw. ist eine entsprechende LevelShifter-Logik notwendig
' - Im vom Display zur Verfügung gestellten SPI Mode kann nur schreibend und
' nicht lesend auf das Display zugegriffen werden.
' Um Grafiken abbilden zu können ist daher ein großer Buffer im RAM des
' Controllers notwendig. Die wenigstens ATmega's bieten ausreichend Speicher
' um solch eine Kopie des Grafikspeichers zu halten.
' 128 Spalten und 64 Zeilen ergeben 8192 Bits (1 Bit für jedes Pixel S/W).
' Für den Grafikspeicher sind also 1024 Byte RAM notwendig.

#####

' -----
' Compilerinstruktionen und Compilerdirektiven
' -----

Sregfile = "m8def.dat"                                     ' Definitionsdatei für ATmega128 laden
$crystal = 8000000                                         ' Quarzfrequenz für 16 MHz festlegen

$hwstack = 40
$swstack = 40
$framesize = 40

Sbaud = 9600

' Hier wird die spezielle Grafik-LIB von BASCOM geladen
Slib "glcdeadogm128x64.lbx"

' -----
' Allgemeine Zusatzinformationen zu Programmbeginn
' -----

' Generelle Informationen zum Display

' Das 128x64 Display ist organisiert in 8 Pages, jede Page mit 8 Pixel Höhe
' und einer Tiefe von 132.
' Die einfachste Möglichkeit Grafiken auf das Display zu schreiben ist das
```

```

' Updaten bzw. beschreiben einer gesamten Page
'
' Pages:
'
' +-----+
' | Column Addresses für Page 1 | Page 0 - 0000
' +-----+
' | Column Addresses für Page 2 | Page 1 - 0001
' +-----+
' | .... | Page 2 - 0010
' +-----+
' | .... | Page 3 - 0011
' +-----+
' | .... | Page 4 - 0100
' +-----+
' | .... | Page 5 - 0101
' +-----+
' | .... | Page 6 - 0110
' +-----+
' | .... | Page 7 - 0111
' +-----+
'
' Column Addresses:
'
'      S00 S01 S02 S03 S04 S05 S06      S83 hex - Column address
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ ... +---+
' D0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
' D1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
' D2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
' D3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
'      ...
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
' D7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
'      +---+ +---+ +---+ +---+ +---+ +---+ +---+ +---+
'
' +---- Data (Datenbits innerhalb des geschriebenen Datenbyte) - jedes Bit ist ein Pixel
'
' -----
' Definition von Ressourcen
' -----
'
' ----- LED's -----
' Alive_pin Alias Pinc.0 ' GPIO für Alive-LED (für DDR oder In-
' put)
' Alive Alias Portc.0 ' GPIO für Alive-LED (für Output oder
' Pullup)
'
' ----- LCD-Display -----
' LCD-Display
' Glcd_cs1 Alias Portd.4 ' GPIO PD4
' Glcd_a0 Alias Portd.7 ' GPIO PD7
' Glcd_si Alias Portb.3 ' MOSI
' Glcd_sclk Alias Portb.5 ' SCK
' Glcd_rst Alias Portd.5 ' GPIO PD5
'
' -----
' Definition von Konstanten
' -----
'
' ----- Für Testumgebung bzw. Traceausgaben -----
' Const Main_testmodus = 1 ' Flag für Testmodus Allgemeinsystem
' Const Lcd_testmodus = 1 ' Flag
'
' Const Demo_part_1 = 1 ' Einfache Ausgabe Text
' Const Demo_part_2 = 1 ' Erweiterte Display Funktionen
' Const Demo_part_3 = 1 ' Bildfunktionen
' Const Demo_part_4 = 1 ' Grafikspeicher
'
' ----- Allgemeine Systemkonstanten -----

```

```

' Tatsächliches Allgemeines
' Const Led_aus = 0
' Const Led_ein = 1

Const Led_aus = 1                                ' Achtung !! bei STK500 ist Logik ge-
dreht!!                                          '
Const Led_ein = 0                                ' Achtung !! bei STK500 ist Logik ge-
dreht!!                                          '

Const False = 0
Const True = 1

' Zeitvorgabe für Sekunden-Timer
Const Timervorgabe = 34286                        ' Timer von 1 Sekunden (SekundenTick)

' ----- LCD -----

Const Glcd_cmd_display_off = &B10101110          ' (1.1) LCD Display - off
Const Glcd_cmd_display_on = &B10101111           ' (1.1) LCD Display - on

Const Glcd_cmd_display_start_line = &B01000000    ' (2) Set the display RAM display start
line address                                     '
'                                     | | | |
'                                     variabel: xxxxxx &B01xxxxxx 0x00-0x63

Const Glcd_cmd_page_address_set = &B10110000      ' (3) Sets the display RAM page address
'                                     | | |
'                                     variabel: xxxx &B1011xxxx 0x00 - 0x07 (Page 0 -
7)

Const Glcd_cmd_column_address_set_msca = &B00010000 ' (4.1) Sets the most significant 4 bits
of the display RAM column address               '
'                                     | | |
'                                     variabel: xxxx &B0001xxxx 0x00 - 0x07 (Block 0
- 7)

Const Glcd_cmd_column_address_set_lsca = &B00000000 ' (4.2) Sets the least significant 4
bits of the display RAM column address           '
'                                     | | |
'                                     variabel: xxxx &B0000xxxx 0x00 - 0x0F (Column 0
- 15)

Const Glcd_cmd_disp_normal = &B10100110           ' (9.1) Display - normal
Const Glcd_cmd_disp_inverted = &B10100111         ' (9.2) Display - inverted

Const Glcd_cmd_disp_all_point_normal = &B10100100 ' (10.1) Display all points - normal
display
Const Glcd_cmd_disp_all_point_on = &B10100101     ' (10.2) Display all points - all poins
ON

Const Glcd_cmd_reset = &B11100010                 ' (14) Internal reset

Const Glcd_cmd_set_contrast = &B10000001          ' (18.1) Electronic volume mode set
Const Glcd_cmd_set_contrast_data = &B00000000    ' (18.2) Electronic volume register set
'                                     | | | |
'                                     variabel: xxxxxx &B00xxxxxx 0x00 - 0x3F (0-63)

Const Glcd_cmd_nop = &B11100011                   ' (22) NOP

' -----
' Definition von Variablen und Datentypen
' -----

' ----- Temporäre Hilfsvariablen -----
Dim Temp_byte_1 As Byte                            ' Temporäre Byte Variable 1
Dim Temp_byte_2 As Byte                            ' Temporäre Byte Variable 2
Dim Temp_byte_3 As Byte                            ' Temporäre Byte Variable3

' ----- Variablen für LCD-Display -----
' -----

```

```
' Prototyping
' -----

' ----- Konfiguration und Basiseinstellungen (Projekt und Testumgebung) -----
' ----- CONFIG -----
' ----- Timer -----
' Konfiguration eines Timers für 1 Sekunden Timer-Tick (Scheduler und Alive)
Config Timer1 = Timer , Prescale = 256          ' Timer 1 verwenden
On Timer1 Sekunden_tick                        ' Interrupt Routine
Timer1 = Timervorgabe
Enable Timer1                                ' Interrupt für Sekunden-Tack

' ----- LCD Display -----
' Konfiguration LCD Display
Config Graphlcd = 128 * 64eadoqm , Cs1 = Glcd_cs1 , A0 = Glcd_a0 , Si = Glcd_si , Sclk = Glcd_sclk ,
Rst = Glcd_rst

Initlcd                                     ' LCD Initialisieren (normal beim ersten
LCD Zugriff automatisch)
Cls                                         ' Display löschen

SetFont Font8x8tt                           ' Kleiner FONT laden

' ----- Port's und Pin's -----
' ----- LED-Konfigurationen -----
Config Alive_pin = Output

' ----- Variablen und Werte -----
' ----- LED-Konfigurationen -----
Alive = Led_aus                                ' Alive-LED aus

' ----- LCD-Display -----

' -----
' Und los gehts, hier noch die Restarbeiten
' -----

' ----- Freigabe aller Interrupts -----
Enable Interrupts                            ' Damit auch Empfang von Daten über Buf-
fer

' ----- Gosub's -----

' #####
'
'                         Hauptprogramm ConvCtrl
'
' #####

Print "Programm start"

#if Demo_part_1
  Lcdat 1 , 1 , "11111111"
  Lcdat 2 , 1 , "88888888"
  Lcdat 3 , 1 , "EA-DOGM182x64 Test1" , 1      ' Text invers ausgeben
  Wait 5
  Cls
#endif
```

```
' ----- Hier ist die Programmhauptschleife -----
'
```

Do

' Hauptschleife

```
#if Demo_part_2
```

```
Cls
```

```
Lcdat 1 , 1 , "abcdefgh"
Lcdat 2 , 1 , "ABCDEFGH"
Lcdat 3 , 1 , "EA-DOGM182x64 Test1" , 1
```

' Text invers ausgeben

```
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 1
' Glcdcmd &B00000001
Lcdat 5 , 1 , "Kontrast 1 " , 1
```

'Kontrast einstellen

'Kontrast auf 1 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 2
' Glcdcmd &B00000010
Lcdat 5 , 1 , "Kontrast 2 " , 1
```

'Kontrast setzen

'Kontrast auf 2 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 4
' Glcdcmd &B00000100
Lcdat 5 , 1 , "Kontrast 4 " , 1
```

'Kontrast setzen

'Kontrast auf 4 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 8
' Glcdcmd &B00001000
Lcdat 5 , 1 , "Kontrast 8 " , 1
```

'Kontrast setzen

'Kontrast auf 8 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 16
' Glcdcmd &B00010000
Lcdat 5 , 1 , "Kontrast 16 " , 1
```

'Kontrast setzen

'Kontrast auf 16 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 32
' Glcdcmd &B00100000
Lcdat 5 , 1 , "Kontrast 32 " , 1
```

'Kontrast setzen

'Kontrast auf 32 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 63
' Glcdcmd &B00111111
Lcdat 5 , 1 , "Kontrast max " , 1
```

'Kontrast setzen

'Kontrast auf 63 Setzen

```
Wait 2
Glcdcmd Glcd_cmd_set_contrast
Glcdcmd Glcd_cmd_set_contrast_data + 20
' Glcdcmd 20
Lcdat 5 , 1 , "Kontrast 20 " , 1
```

'Kontrast setzen -> bester Kontrast

'Kontrast auf 20 Setzen

```
Wait 5
#endif
```

```
#if Demo_part_3
```

```
Showpic 0 , 0 , Picture
Wait 5
```

' BGF Grafik anzeigen

```
Display Off
Wait 2
Display On
Wait 2
```

' Display ausschalten mit BASCOM Befehl

' Display einschalten mit BASCOM Befehl

```
Glcdcmd Glcd_cmd_display_off
Wait 2
```

' Display ausschalten mit GLCD Befehl


```

    Glcdcmd Glcd_cmd_display_on                ' Display einschalten mit GLCD Befehl
    Wait 2

    Glcdcmd Glcd_cmd_disp_inverted              ' Display invers betreiben
    Wait 2
    Glcdcmd Glcd_cmd_disp_normal               ' Display Normalmodus
    Wait 5
#endif

#if Demo_part_4

    Cls

    ' ---- Linie auf Display erzeugen ----

    ' Sets the display RAM page adress
    Glcdcmd Glcd_cmd_page_address_set + 0      'Zeile 0

    For Temp_byte_1 = 0 To 7

        ' Sets the most significant 4 bits of the display RAM column address
        Temp_byte_3 = Glcd_cmd_column_address_set_msca + Temp_byte_1
        Glcdcmd Temp_byte_3

        For Temp_byte_2 = 0 To 15
            ' Sets the least significant 4 bits of the display RAM column address
            Temp_byte_3 = Glcd_cmd_column_address_set_lsca + Temp_byte_2
            Glcdcmd Temp_byte_3

            Glcddata &B10100101                ' Pattern erzeugt Linienmuster
        Next Temp_byte_2
    Next Temp_byte_1

    Wait 5

    ' ---- Schachbrettmuster auf Display erzeugen ----

    For Temp_byte_2 = 0 To 7
        Temp_byte_3 = Glcd_cmd_page_address_set + Temp_byte_2
        Glcdcmd Temp_byte_3                    ' Setze RAM Page 1-8, 8 Pixel hoch

        Glcdcmd Glcd_cmd_column_address_set_msca + 0 ' Setze Block 0 von Page x
        Glcdcmd Glcd_cmd_column_address_set_lsca + 0 ' Setze Column 0 von Block 0

        For Temp_byte_1 = 1 To 16

            ' Pattern erzeugt Schachbrettmuster
            Glcddata &HF0
            Glcddata &HF0
            Glcddata &HF0
            Glcddata &HF0
            Glcddata &H0F
            Glcddata &H0F
            Glcddata &H0F
            Glcddata &H0F
        Next Temp_byte_1
    Next Temp_byte_2

    Wait 5

#endif

Loop                                            ' Hauptschleife

'## End Hauptprogramm #####

End

' *****
' Interruptroutinen
' *****

' -----
' Interrupt-Service-Routine (Timer1): Sekunden_tick

```

```
' Routine zur Auswertung des Timer Interrupts
'-----
Sekunden_tick:
    ' ----- Programmcode -----
    Timer1 = Timervorgabe                ' Timer neu laden
    Toggle Alive                        ' Alive-LED toggeln lassen

Return
'-- End Sekunden_tick -----

'*****
' Subroutinen
'*****

' *****
' * LCD-Display *
' *****

'*****
' GOSubroutinen
'*****

' *****
' * LCD-Dsisplay *
' *****

'-----
' Devices schließend und ggf. "Terminate Programm execution"
'-----

' System halt
End                                     'end program

'-----
' Definition von globalen Konstantenfeldern
'-----

' ----- Grafiken für das GLCD -----

#include the picture data
Picture:
$bgf "BASCOM.bgf"

' ----- FONT's für das GLCD -----

' Include Font nach dem END und damit nach dem Programmcode damit nicht FONT-
' Data als Code interpretiert wird (sollte ganz am Ende stehen)
$include "font8x8TT.font"

'-----

'##### END #####

'##### Historie #####
' 28.10.2018 : Version 1.0
'           Erstellung der 1. Version der generellen Inbetriebnahme DOGM128
'#####
```

Software 1: Code zur Ansteuerung des LCD-Grafik-Displays

10.3.4 Prototyp LCD-Display-Ansteuerung

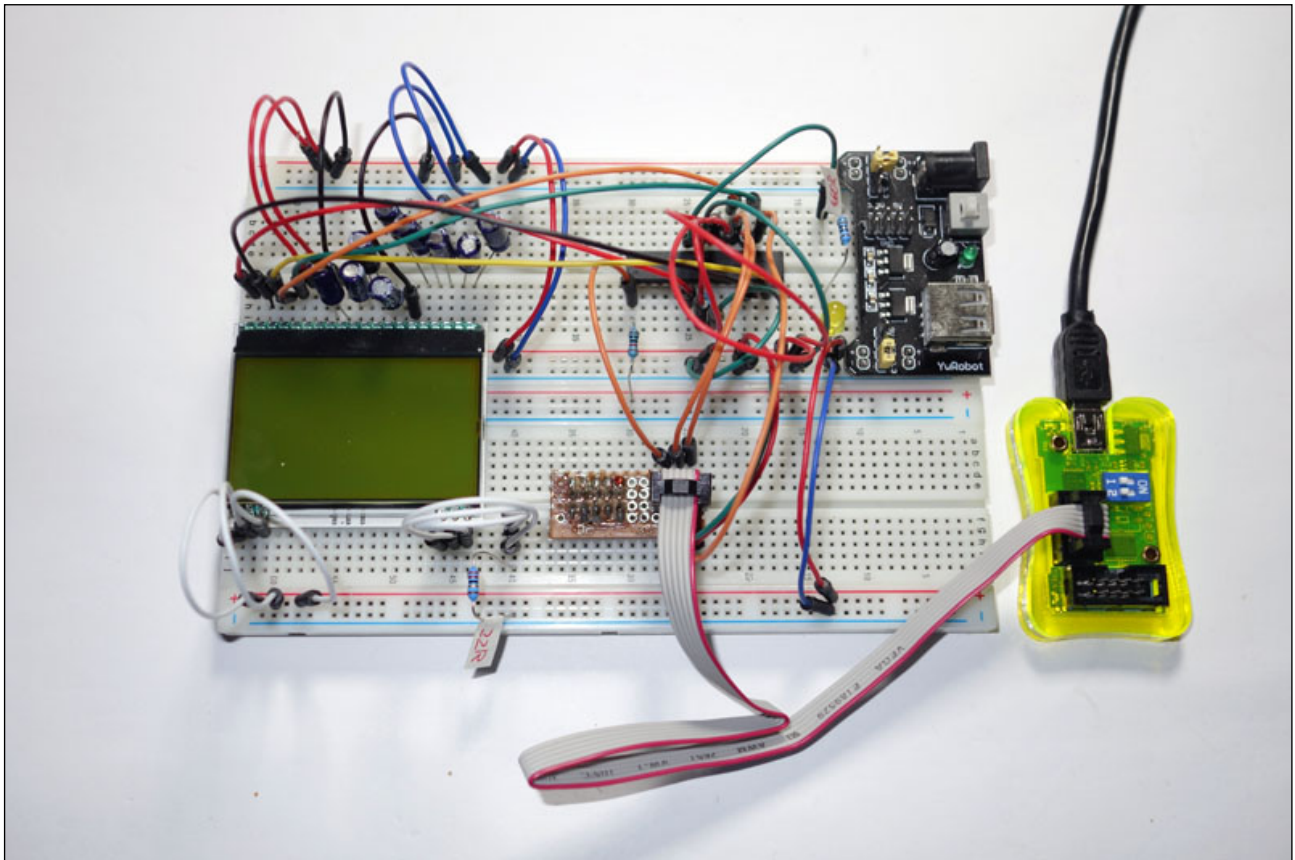


Abbildung 25: Prototyp LCD-Display-Ansteuerung mit Breadboard

10.4 LED-Zuordnung Schaltplan

Im Projekt myGeigerle wird insgesamt nur 1 LED verwendet. Eine 5mm Low-Current-LED welche ohne Transistorstufe mittels Metallschichtvorwiderstand direkt an einem GPIO-Pin des ATmega8L-Controllerst betrieben wird.

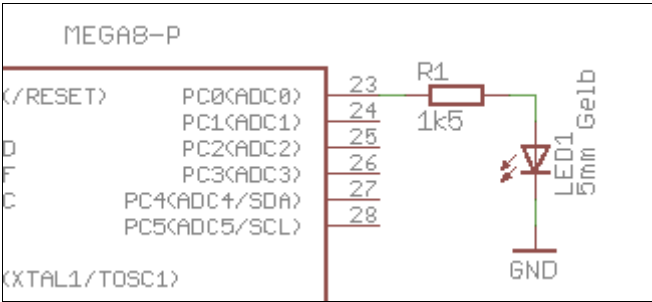
Die folgende Tabelle zeigt die LED des Projekt beschrieben und der HW-Ressource im Schaltplan inkl. Funktion zugeordnet:

LED	Funktion	Abkürzung	Farbe
1	Alive LED – 3 mm	LED1	GELB

Tabelle 33: LED-Zuordnung Schaltplan

Die gelbe 5mm Alive LED zeigt an, dass der Controller bzw. das System normal arbeitet und der Hauptschalter eingeschaltet ist. Dabei blinkt die LED im Sekundenrhythmus.

Beschaltung:



Schaltbild 11: Beschaltung LED's

Bauteile:

Stückliste: LED-Ansteuerung			
Widerstände		Halbleiter	
R1	Metallschichtwiderstand 1k5 Ω	LED1	Standard LED, 5mm, yellow

Tabelle 34: Bauteile für LED-Ansteuerung

Ressourcenzuordnung zum ATmega8L:

Nummer	Schaltbild		Ressource ATmega8	
1	ALIVE	ALIVE	PortC.0	[PC0] (GPIO Output)

Tabelle 35: Ressourcenzuordnung für LED-Ansteuerung

11 Mechanik

Die Platine des myGeigerle wird von den Abmaßen so groß ausgeführt, dass der MightyOhm Geiger Counter ohne Plexiglas-Bodenplatte auf dem myGeigerle verschraubt werden kann.

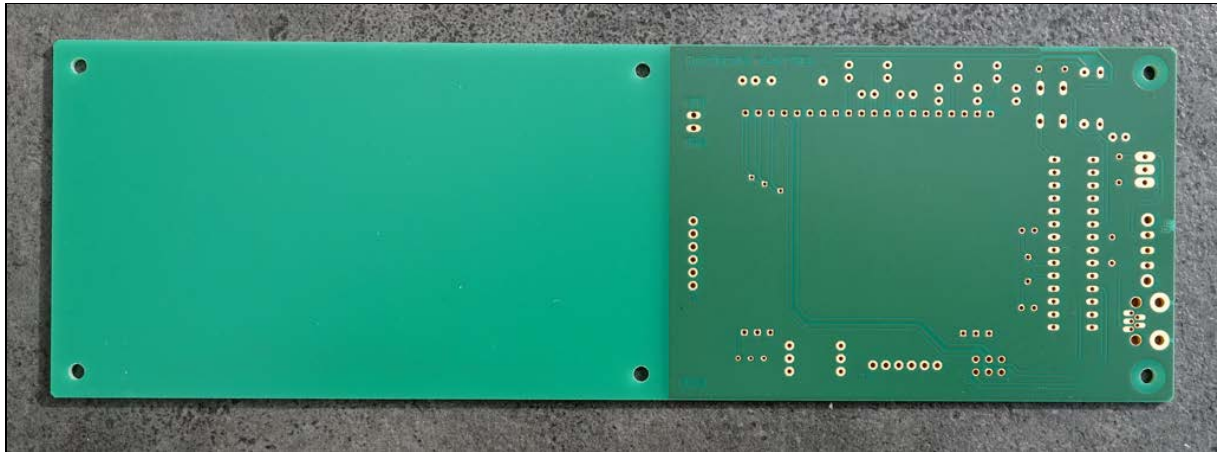


Abbildung 26: Platine des myGeigerle mit Träger für MightyOhm Geiger

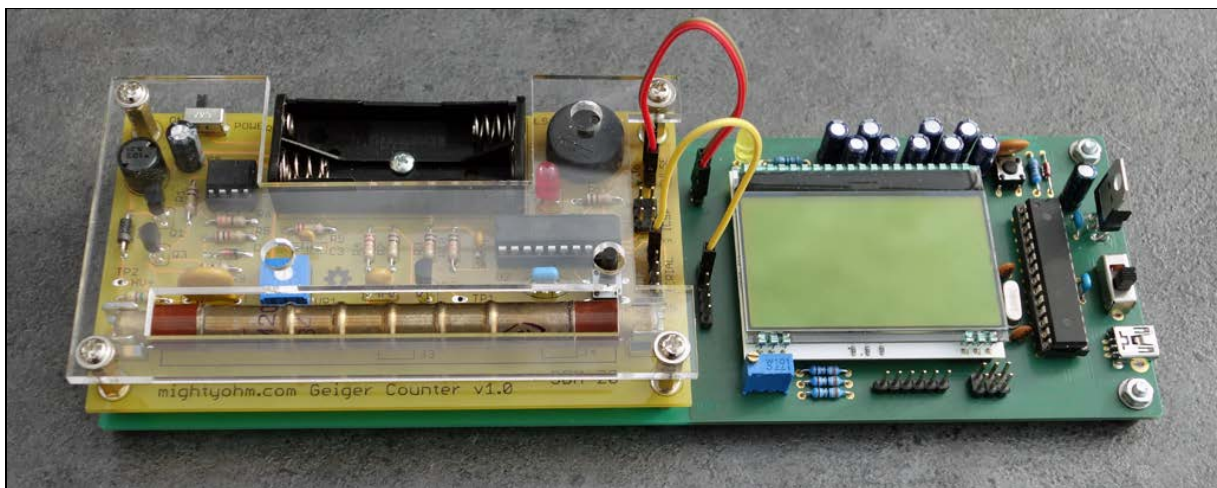


Abbildung 27: MightyOhm Geiger auf myGeigerle montiert

Für den nötigen Abstand der myGeigerle PCB zum Untergrund sorgen Anschlagpuffer



Anschlagpuffer D Ø6-200mm M3-M20 Gummipuffer Maschinenfuß Gummi-Metall-Puffer

Gewinde M3 x 6
Durchmesser 8mm
Höhe 6 mm

Die Anschlagpuffer wurden über eBay bezogen.

12 Bauteile und Bauteilbeschaffung

Für das Prototyping und die Inbetriebnahme der im obigen Kapitel „Grundlagen“ beschriebenen Einzelthemen werden die Einzelteile in Verbindung mit einem Breadboard aufgebaut und in Betrieb genommen.

Die folgenden Stücklisten geben einen Überblick über die benötigten Bauelemente und ergeben gleichzeitig die Bestelllisten von Conrad Electronic, Reichelt Electronic, Amazon und ggf. eBay.

Bauelemente Reichelt Elektronik:

Stk.	Beschreibung	Bestellnummer	Einzelpreis	Gesamtpreis
Kondensatoren				
2	Vielschicht-Keramikkondensator 100nF 50 VDC	Z5U-2,5 100N	0,042	0,084
2	Keramikkondensator 22pF	KERKO 22P	0,052	0,104
1	Keramikkondensator 47nF	KERKO 48N	0,094	0,188
1	Elektrolytkondensator 10µF	JAM NKR100M1VE11	0,062	0,062
9	Elektrolytkondensator 1µF	JAM SSR101M1CE07	0,031	0,279
Widerstände				
1	Metallschichtwiderstand 600 Ohm (590 Ohm)	METALL 590	0,086	0,086
1	Metallschichtwiderstand 100 Ohm	METALL 100	0,086	0,086
1	Metallschichtwiderstand 4k7 Ohm	METALL 4,70K	0,086	0,086
3	Metallschichtwiderstand 40 Ohm (40,2 Ohm)	METALL 40,2	0,086	0,258
Halbleiter				
1	Diode 1N4148	1N 4148	0,01	0,01
1	Spannungsregler 3,3V TO-220	LD1117V33	0,33	0,33
1	EA DOGM 128x-6 LCD Modul Farbe grün	EA DOGM128E-6	18,22	18,22
1	EA LED55x46 Led-Beleuchtung Farbe: grün	EA LED55X46-G	4,18	4,18
1	LED 5mm, Standard, Gelb	LED 5MM ST GE	0,126	0,126
1	ATmega AVR-RISC-Controller, S-DIL-28	ATMEGA 8L8 DIP	1,52	1,52
Sonstiges				
1	MA03-2 für ISP06	SL 2X50G 2,54	0,89	0,89
1	Quarz 8.000 MHz	8,0000-HC49U-S	0,26	0,26
1	IC-Sockel, 28-polig, schmal	GS 28P-S	0,43	0,43
2	Stiftleiste, gerade, RM 2,54 – 6-polig	SL 1X50G 2,54	0,39	0,39
1	MA02-1 Stiftleiste, gerade, RM 2,54 – 2-polig	--	--	--
1	Kurzhubtaster 6x6mm, Höhe: 4,3mm, 12V, vertikal	TASTER 3301	0,137	0,137

Tabelle 36: Bauelemente Reichelt Elektronik

Bauelemente Conrad Electronic:

Stk.	Beschreibung	Bestellnummer	Einzelpreis	Gesamtpreis
Sonstiges				
1x	Spindeltrimmer 3296W-101 - 100 Ohm	447481 - 62	0,62	0,62
1x	Mini USB Typ B liegend 5-polig WR-COM Buchse, Einbau horizontal WR-COM Würth Elektronik	1088421 – 62	2,32	2,32

Tabelle 37: Bauelemente Conrad Electronic

Bauelemente Amazon:

<i>Stk.</i>	<i>Beschreibung</i>	<i>Bestellnummer</i>	<i>Einzelpreis</i>	<i>Gesamtpreis</i>
Sonstiges				
1x	Cylewet vertikaler Schiebeschalter / Wechselschalter, 12 mm	CLW1016	0,84	0,84

Tabelle 38: Bauelemente Amazon

<

Bauelemente eBay:

<i>Stk.</i>	<i>Beschreibung</i>	<i>Bestellnummer</i>	<i>Einzelpreis</i>	<i>Gesamtpreis</i>
Sonstiges				
6	Anschlagpuffer M3x6, 8mm x 6mm	--	1,58	9,48

Tabelle 39: Bauelemente Conrad Electronic

Die vorausgerechneten Materialkosten belaufen sich auf ca. 43,866 Euro.

13 Hardware

13.1 Festlegung von Netzklassen im Projekt

Für die Umsetzung der PCB in EAGLE werden die folgenden Netzklassen definiert:

Nr	Name	Width	Drill	Clearance
☒ 0	default	10mil	0mil	15mil
☐ 1	Digital	10mil	0mil	15mil
☐ 2	Analog	10mil	0mil	15mil
☐ 3	PowerSupply	20mil	0mil	15mil
☐ 4		0mil	0mil	0mil
☐ 5		0mil	0mil	0mil
☐ 6		0mil	0mil	0mil
☐ 7		0mil	0mil	0mil
☐ 8		0mil	0mil	0mil
☐ 9		0mil	0mil	0mil
☐ 10		0mil	0mil	0mil
☐ 11		0mil	0mil	0mil
☐ 12		0mil	0mil	0mil
☐ 13		0mil	0mil	0mil
☐ 14		0mil	0mil	0mil
☐ 15		0mil	0mil	0mil

Abbildung 28: Definition der Netzklassen

Diese Vorgaben führen bei einem einfachen Schaltbild zur folgenden Umsetzung auf dem Board:



Schaltbild 12: Schaltbild für Definition von Netzklassen

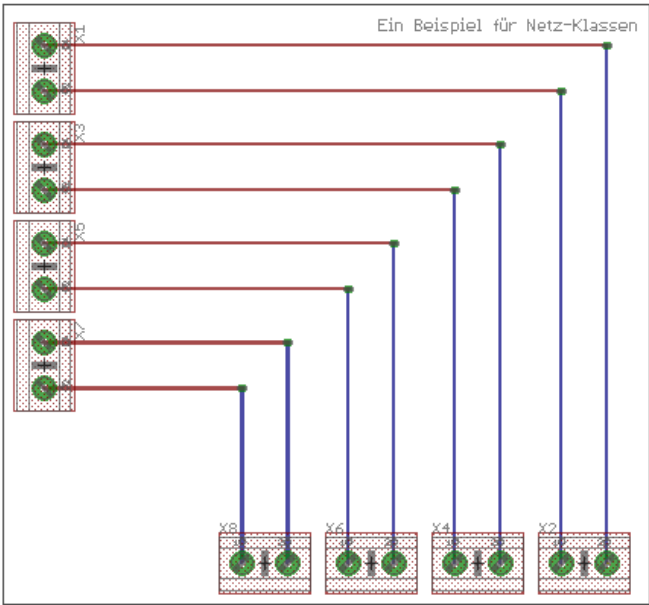
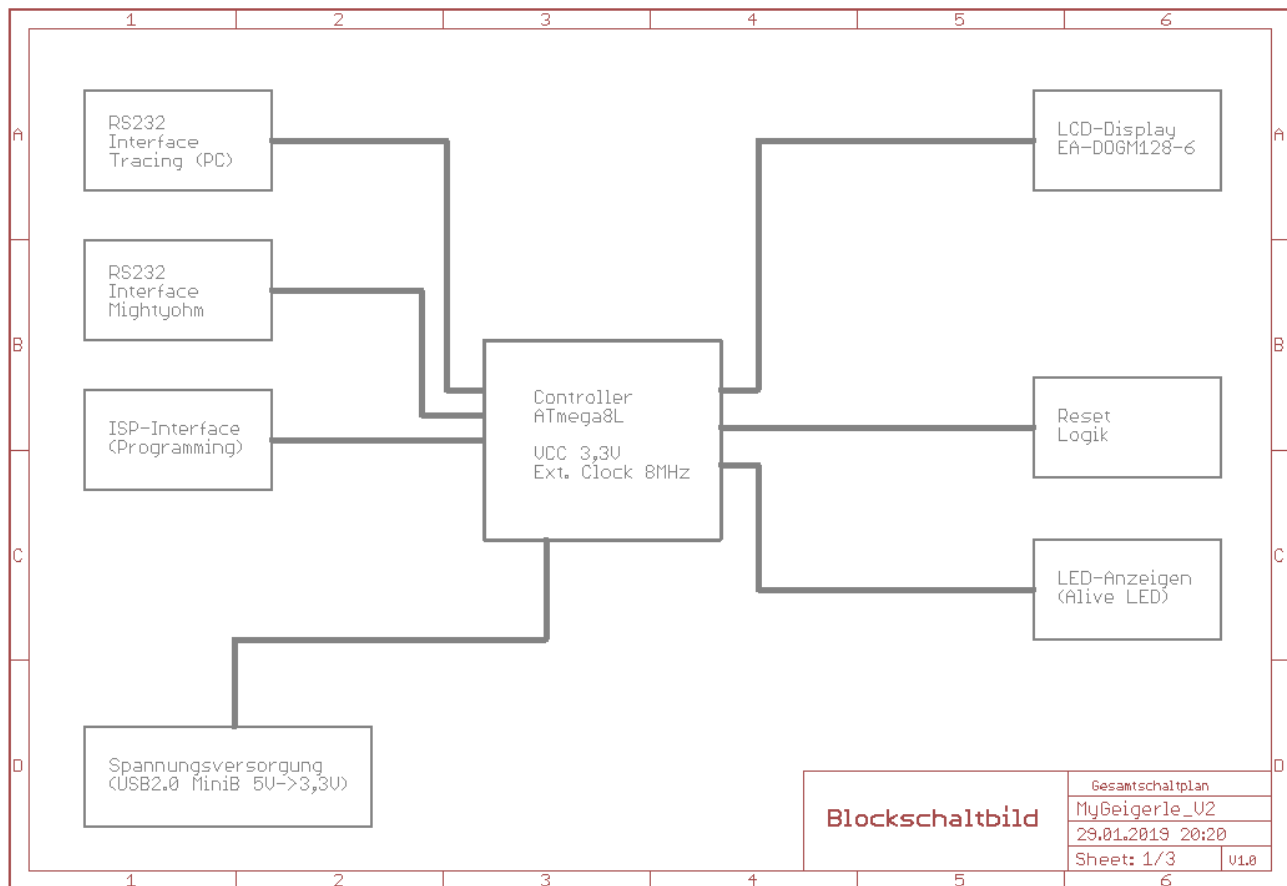


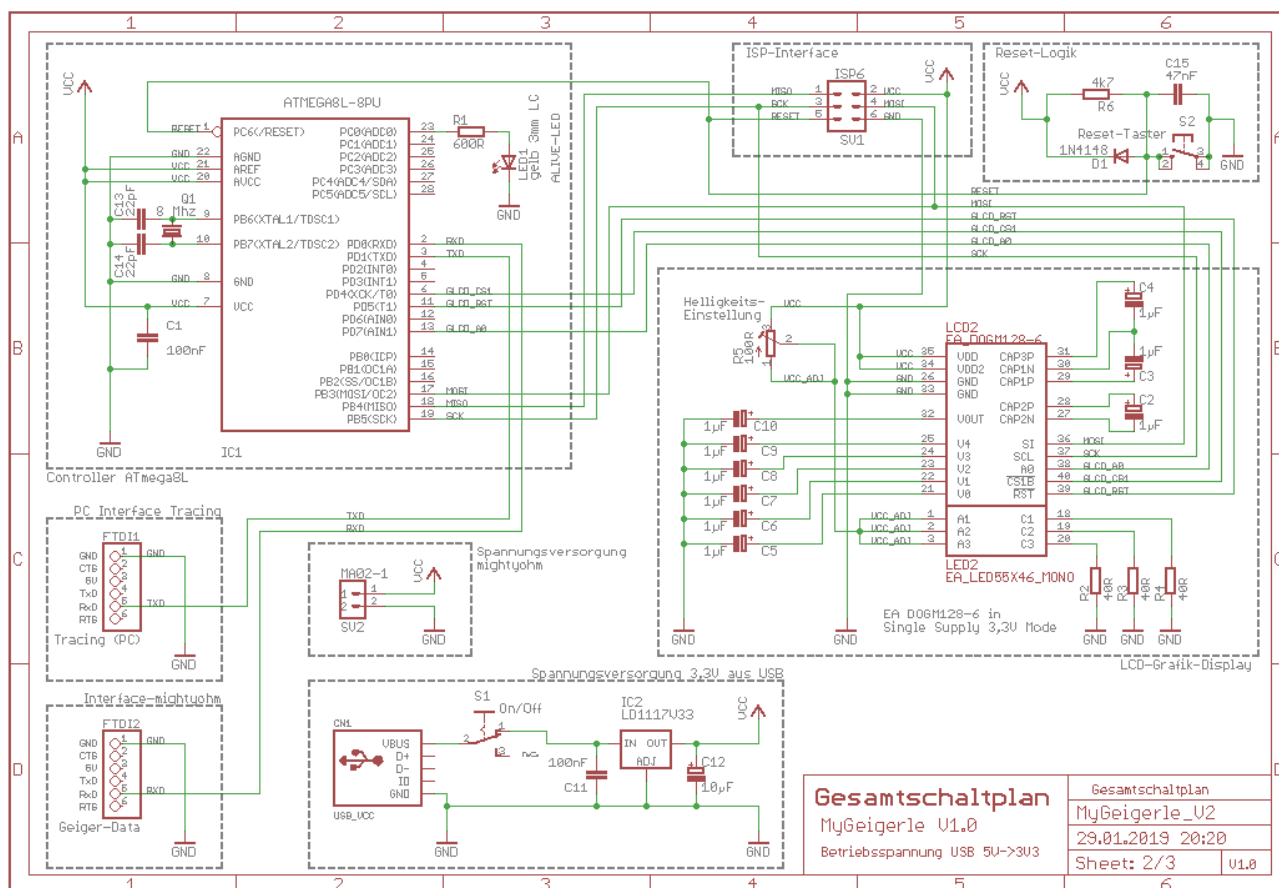
Abbildung 29: Demoboard Netzklassen

13.2 Die PCB zum Projekt myGeigerle

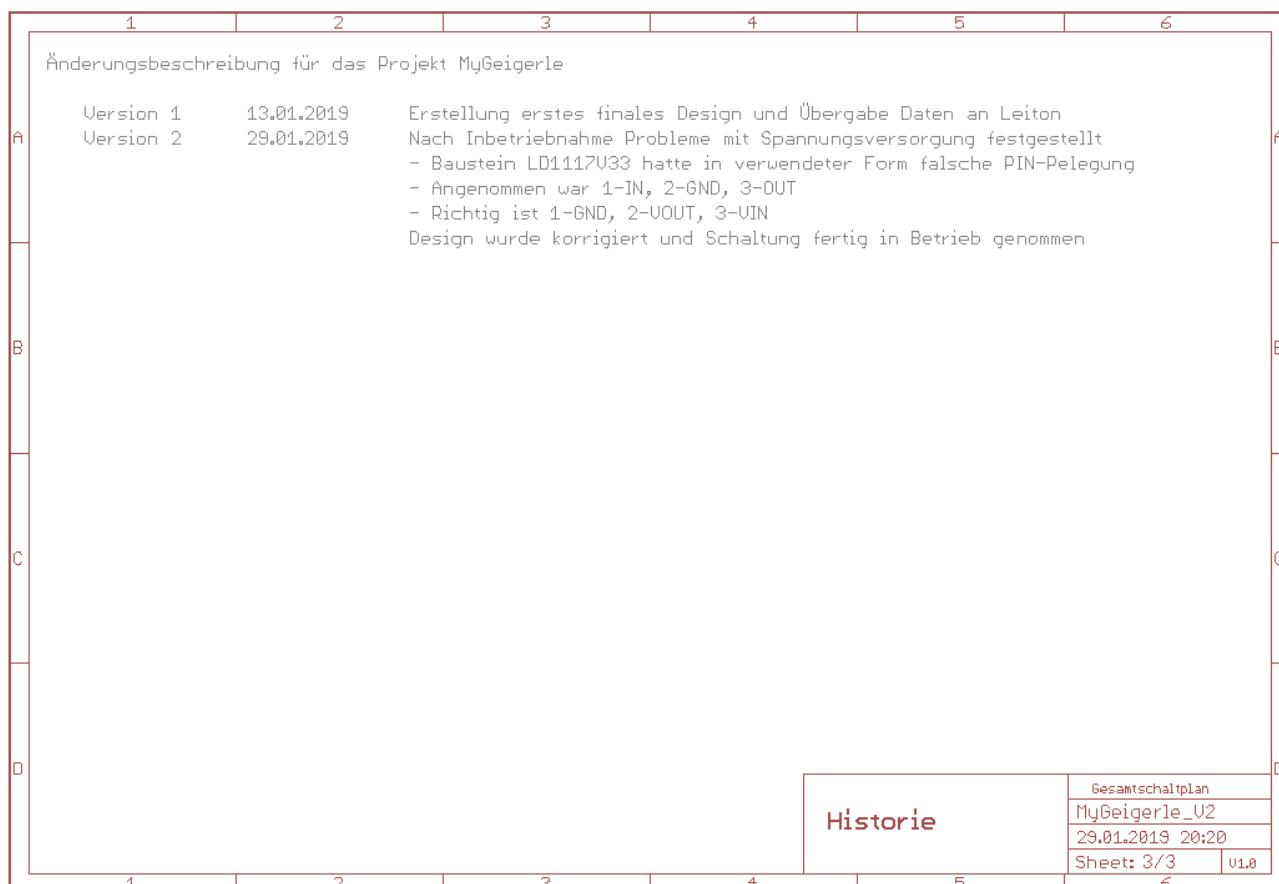
13.2.1 Schematic



Schaltbild 13: Schaltbild myGeigerle - Sheet 1



Schaltbild 14: Schaltung myGeigerle - Sheet 2



Schaltbild 15: Schaltung myGeigerle - Sheet 3

13.2.2 Layout, Layer und Bestückung

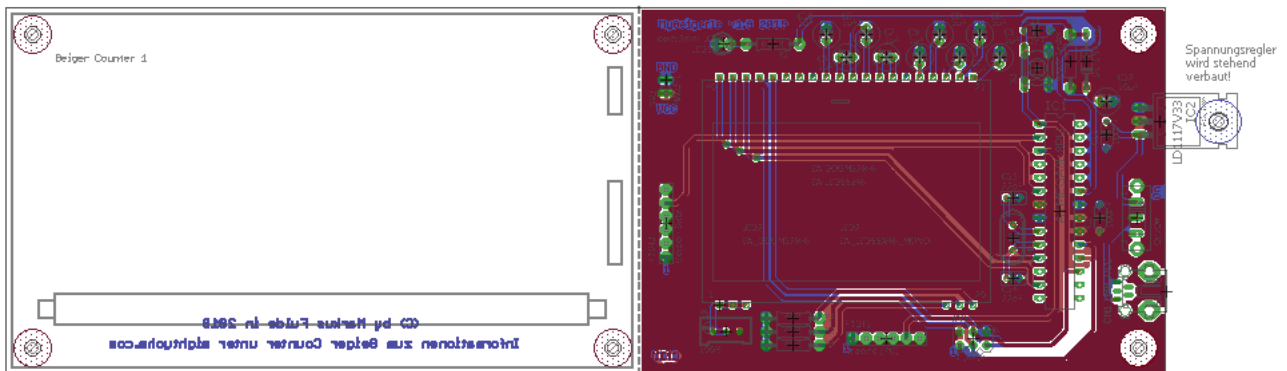


Abbildung 30: PCB myGeigerle – Layout gesamt

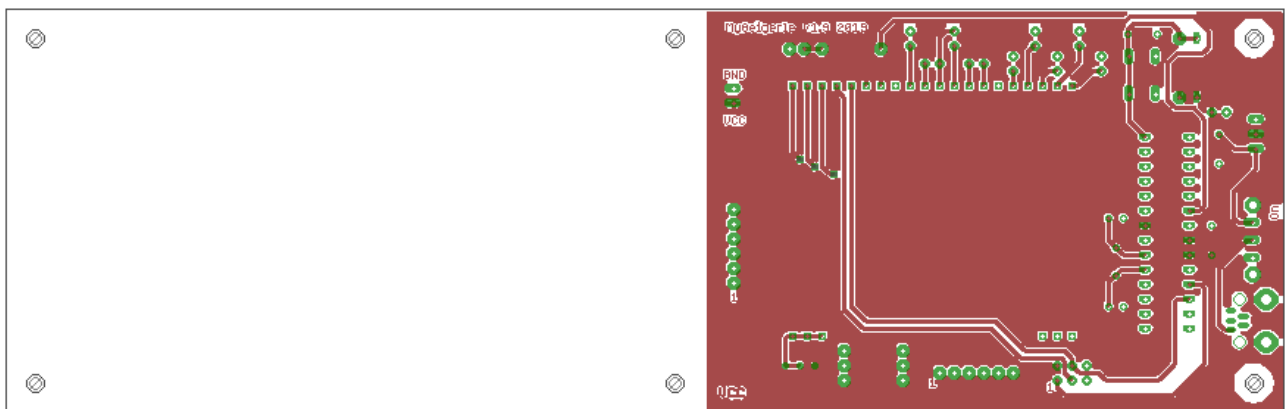


Abbildung 31: PCB myGeigerle – Top Layer

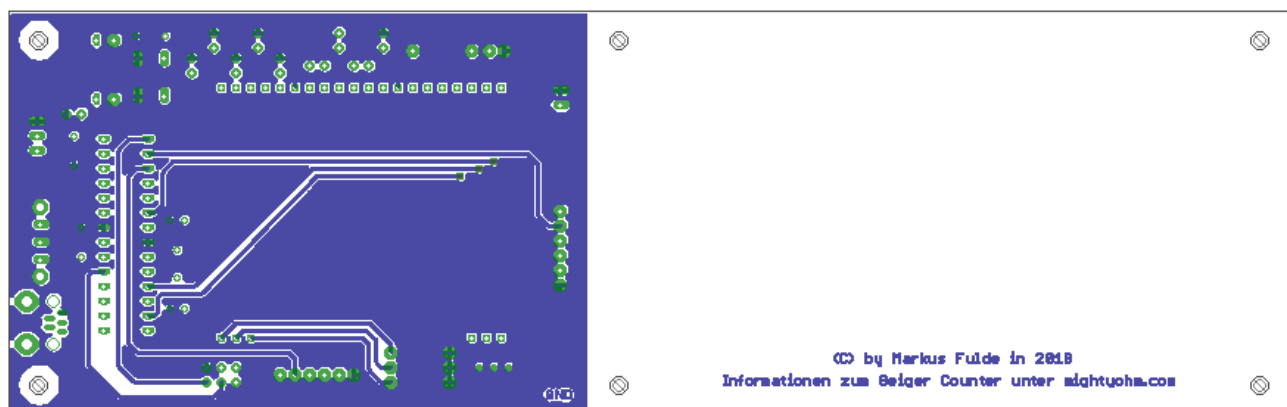


Abbildung 32: PCB myGeigerle – Bottom Layer

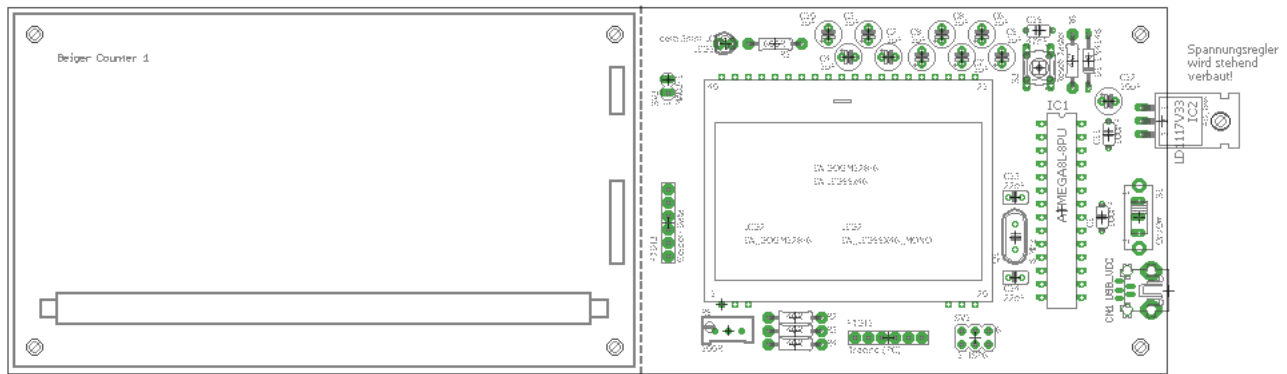


Abbildung 33: PCB myGeigerle – Bestückung Top Layer

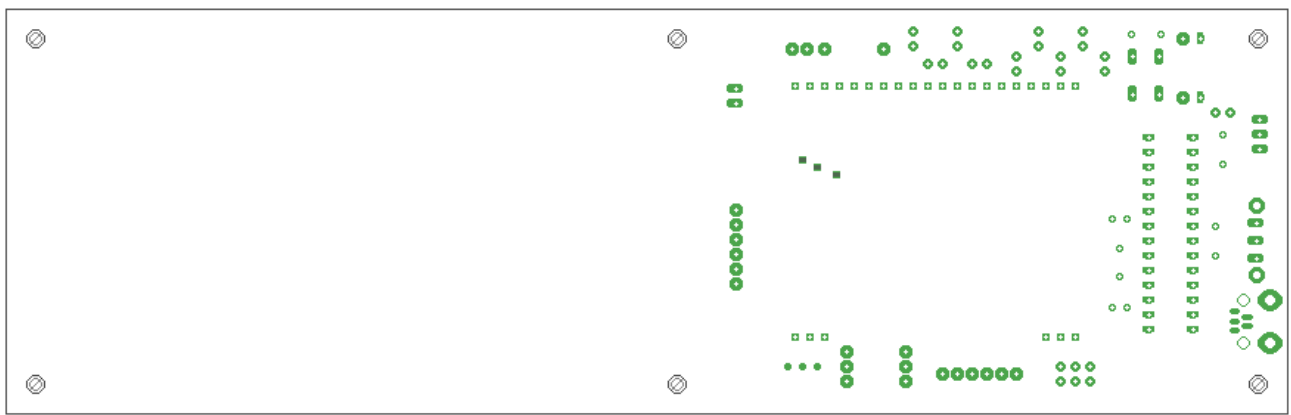


Abbildung 34: PCB myGeigerle – Pads und Vias

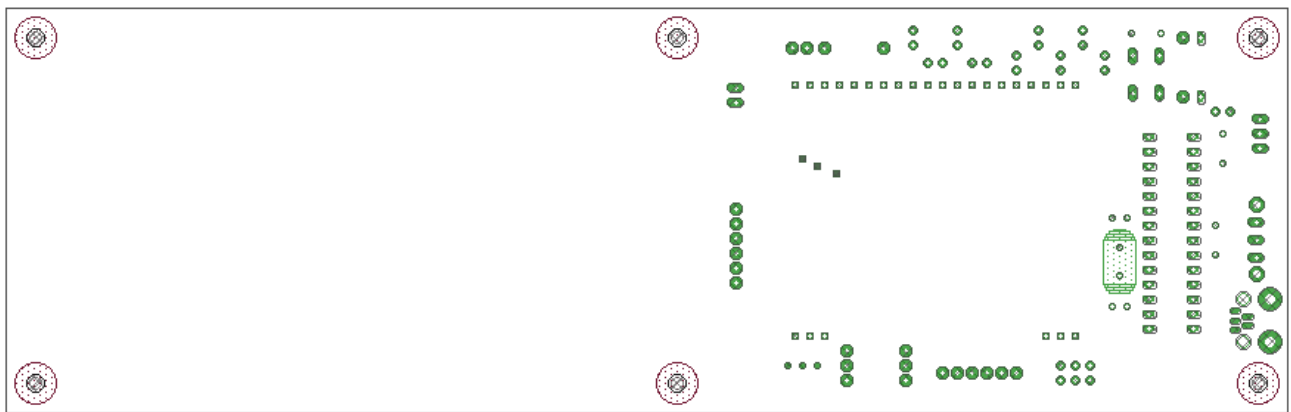


Abbildung 35: PCB myGeigerle – Restricted Areas

13.2.3 Bestückungen in Groß

Zur besseren Lesbarkeit und Erkennung hier nochmals die Bestückungsdrucke in Groß

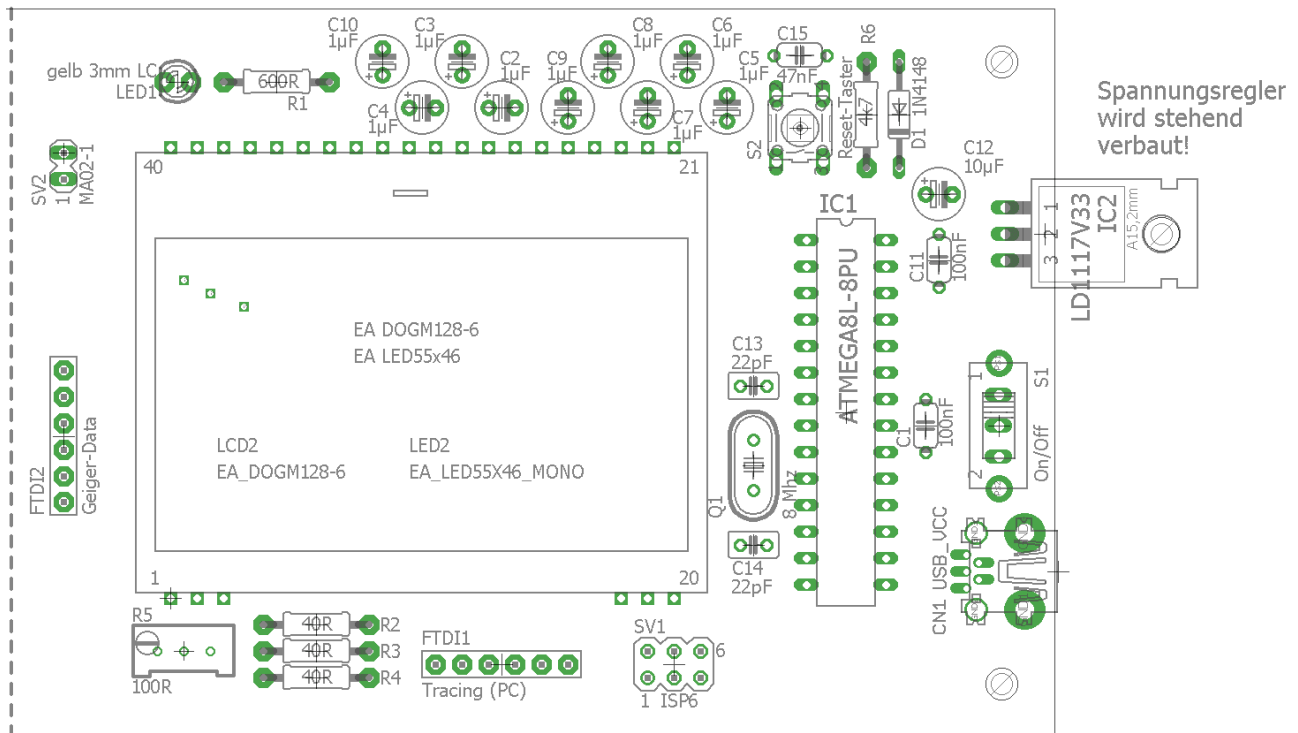


Abbildung 36: PCB myGeigerle – Bestückung

13.2.4 Eagle-BOM

Pos.	Bauteile	Menge	Wert	Device
1.	D1	1	1N4148	DIODE
2.	C13, C14	2	22pF	CAPACITOR, European Symbol
3.	C1, C11	2	100nF	CAPACITOR, European Symbol
4.	C15	1	47nF	CAPACITOR, European Symbol
5.	S1	1	On/Off	SLIDING SWITCH
6.	Q1	1	8 MHz	CRYSTAL
7.	LCD2	1	EA_DOGM128-6	ELECTRONIC ASSEMBLY GmbH
8.	LED2	1	EA_LED55X46_MONO	ELECTRONIC ASSEMBLY GmbH
9.	FTDI2	1	Geiger-Data	PIN HEADER - FTDS Friend - Adafruit
10.	FTDI1	1	Tracing (PC)	PIN HEADER - FTDS Friend - Adafruit
11.	IC2	1	LD1117V33	Low drop fixed and adjustable positive voltage regulators 1 A
12.	LED1	1	gelb 3mm LC	LED
13.	SV2	1	MA02-1	PIN HEADER
14.	SV1	1	ISP6	PIN HEADER
15.	IC1	1	ATMEGA8L-8PU	MICROCONTROLLER
16.	S2	1	Reset-Taster	Kurzhubtaster 6x 6mm
17.	C12	1	10µF	POLARIZED CAPACITOR, European Symbol
18.	C2, C3, C4, C5, C6, C7, C8, C9, C10	9	1µF	POLARIZED CAPACITOR, European Symbol
19.	R2, R3, R4	3	40R	RESISTOR, European Symbol
20.	R6	1	4k7	RESISTOR, European Symbol
21.	R1	1	600R	RESISTOR, European Symbol
22.	R5	1	100R	Trimm resistor
23.	CN1	1	USB_VCC	USB Connectors

Tabelle 40: Eagle BOM für das Projekt myGeigerle

Folgende Bauteile werden noch außerhalb von Eagle benötigt:

Pos.	Bauteile	Menge	Wert	Device
1.	--	6	Anschlagpuffer	--

Tabelle 41: Weitere Bauteile für das Projekt myGeigerle

13.2.5 Das Board

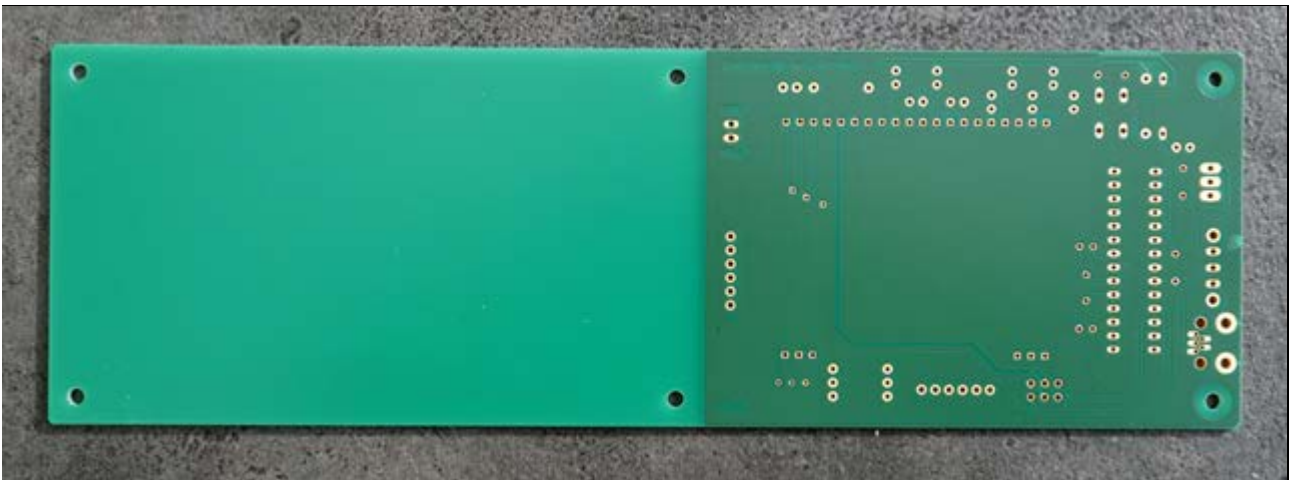


Abbildung 37: PCB myGeigerle TOP

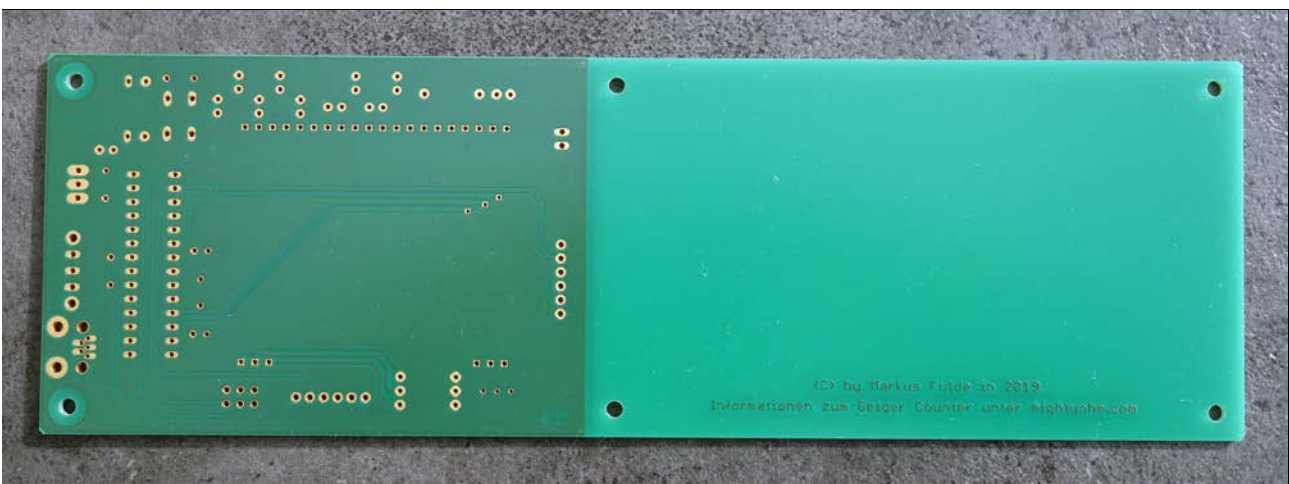


Abbildung 38: PCB myGeigerle BOTTOM

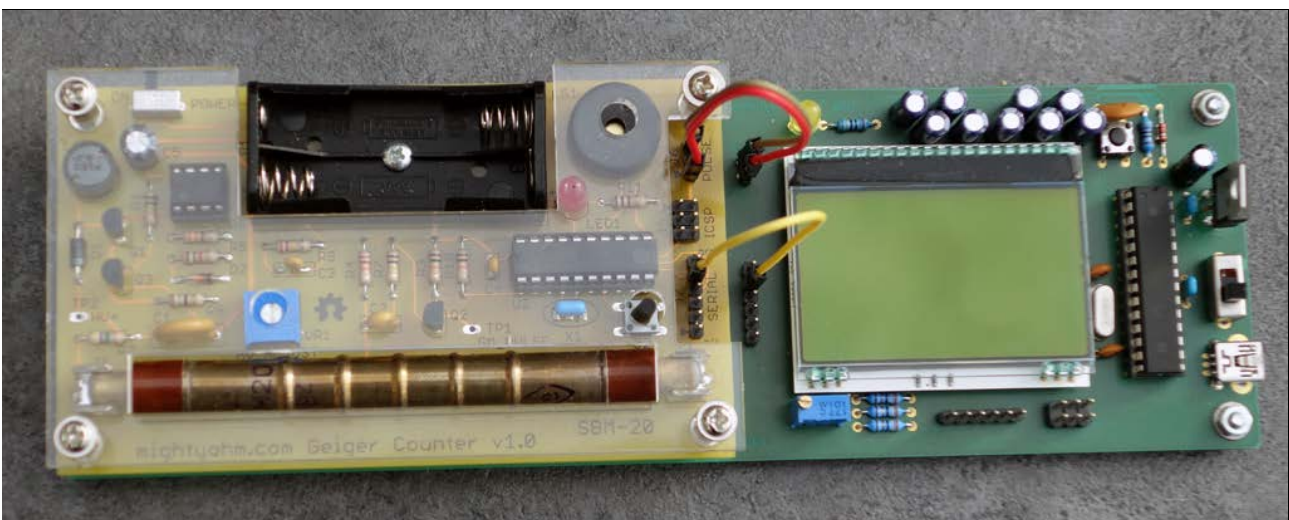


Abbildung 39: PCB TOP fertig bestückt

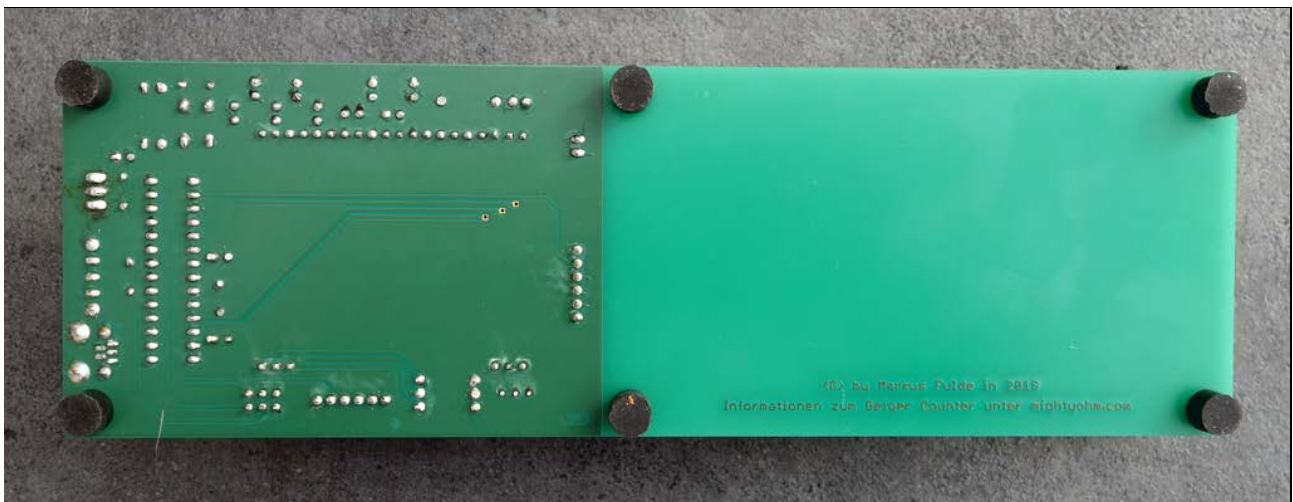


Abbildung 40: PCB BOTTOM fertig bestückt

13.3 Die fertige Hardware

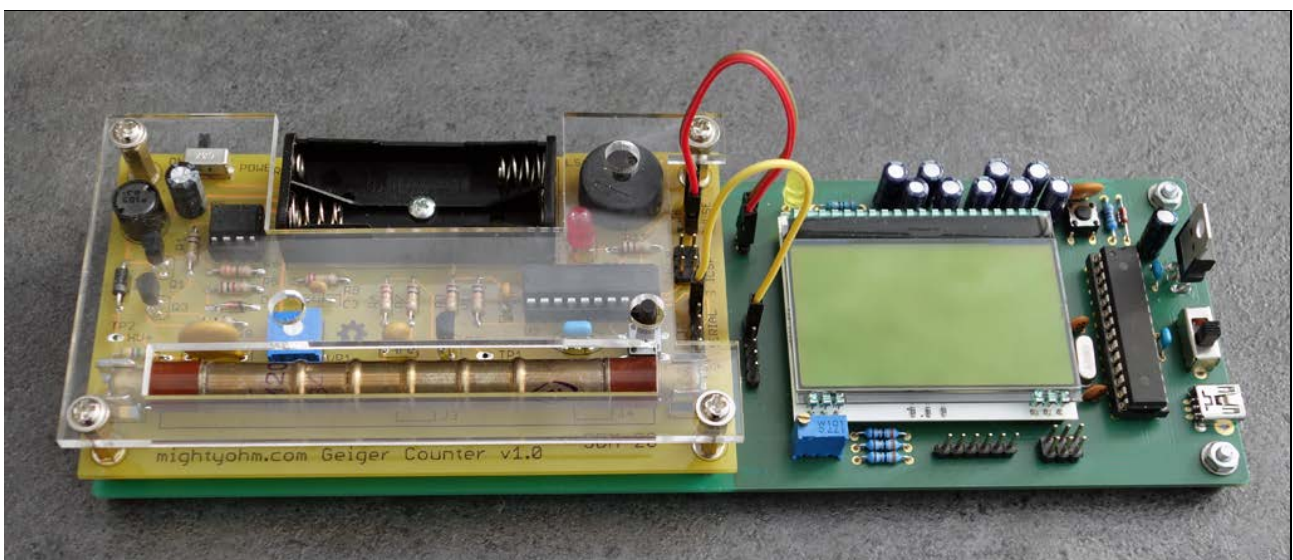


Abbildung 41: Die fertige Platine

14 Software

14.1 Systemfestlegungen und Definitionen

14.1.1 Timerfestlegungen

Die Timer im Gesamtsystem haben entsprechend der Priorität die folgende Reihenfolge:

1. Timer0 8-Bit Timer
2. Timer1 16-Bit Timer
3. Timer2 8-Bit Timer

Timer0:

Nicht verwendet!

Timer1:

Der Timer1 ist im ATmega8L der Timer mit der mittleren Priorität. Mit seiner Hilfe wird ein SW-Timer aufgebaut. Der Timer1 versorgt das Gesamtsystem mit einem 1-Sekunden-Timertick und sorgt für das Toggeln der Betriebs-LED.

Timer2:

Nicht verwendet!

14.2 Verwendete SW

Zur Erstellung dieses Projekts kam folgende Software zum Einsatz:

- Workstation DELL XPS8920: Betriebssystem Windows 10 Pro 64 Bit
- Notebook DELL Inspiron 17R SW: Betriebssystem Windows 10 Pro 64 Bit

- BASCOM-AVR Basic Compiler BASCOM 2.0.8.1
 MCS Electronics
- EAGLE 7.7.0 Standard NON-PROFIT
- ATMEL AVR Studio 7.0

Zur Erstellung dieses Projekts kam folgende HW-Umgebung und SDK's zum Einsatz:

- Breadboards mit Steckverbindern
- ATMEL ISP-Programmer AVRISP mkII
- Tremex USB-ISP-Programmer
- Fertiges Platinen Layout mit Hilfe von Eagle und Herstellung durch Leiton Berlin
- Labornetzteil Rohde&Schwarz NGE102B

- Oszilloskop Tektronix MSO2024B

14.3 Der Source-Code zum Projekt myGeigerle

```
#####
' MyGeigerle.BAS                                     Stand 07.01.2019
' -----                                           (C) Markus Fulde
'
' Programm zur Visualisierung der gemessenen Radioaktivität des Geiger-Müller-
' zähler von mightyohm.
' Quelle des Zählers: http://mightyohm.com/geiger
'
' Die Inbetriebnahme erfolgt mit einem Steckbrett auf Basis ATmega8L
'
' Die Visualisierung erfolgt mittels EA-DOGM128x6 Grafikdisplay mit Beleuch-
' tung.
' Die Anbindung des Display erfolgt über SPI mit einer Betriebsspannung von
' 3.3V
'
' Die Betriebsspannung des Projekt beträgt 3,3 Volt. Der Geigerzähler wird
' nicht mittels Batterie bestromt sondern fremdgespeist.
'
#####
' Details:
' - Prozessor ATmega8L mit externem Clock 8 MGz.
' - Datenrate Geigerzähler 9600 bauds, 8N1.
' - Protokoll: CPS, #####, CPM, #####, uSv/hr, ###.##, SLOW|FAST|INST
'
' Aus der Beschreibung des Geigerzählers:
' There are three modes. Normally, the sample period is LONG_PERIOD (default
' 60 seconds). This is SLOW averaging mode. If the last five measured counts
' exceed a preset threshold, the sample period switches to SHORT_PERIOD seconds
' (default 5 seconds). This is FAST mode, and is more responsive but less
' accurate. Finally, if CPS > 255, we report CPS*60 and switch to INST mode,
' since we can't store data in the (8-bit) sample buffer.
'
' Anmerkung:
' [1] Aus Geschwindigkeitsgründen werden keine Funktionen und Gosubs erwendet
' weil deren Anspringen und Rückspringen Zeit benötigt. Da aber der Sekündliche
' Empfang von Daten vom Geiger vorgegeben ist und eine Bearbeitung und Dar-
' stellung von Daten auf dem Display in dieser Zeit gewährleistet sein muss
' wir dort wo es möglich ist und wo keine Redundanz entsteht sequentiell ohne
' Funktionsaufrufe programmiert.
' [2] Der Clock des ATmega wird mittels externem 8 MHz Quarz erzeugt da bei
' längerem Betrieb der interne Clock bzgl. Genauigkeit nicht ausreichte die
' RS232 Schnittstelle sauber zu bedienen.
'
#####
' -----
' Compilerinstruktionen und Compilerdirektiven
' -----
Sregfile = "m8def.dat"                                ' Definitionsdatei für ATmega128 laden
Scrystal = 8000000                                     ' Quarzfrequenz für 16 MHz festlegen

Shwstack = 32
$swstack = 32
$framesize = 128

Sbaud = 9600                                           ' Baud Rate

' Hier wird die spezielle Grafik-LIB von BASCOM geladen
$lib "glcdeadogm128x6.lbx"

' -----
' Allgemeine Zusatzinformationen zu Programmbeginn
' -----

' Generelle Informationen zum Display

' Das 128x64 Display ist organisiert in 8 Pages, jede Page mit 8 Pixel Höhe
' und einer Tiefe von 132.
' Die einfachste Möglichkeit Grafiken auf das Display zu schreiben ist das
' updaten bzw. beschreiben einer gesamten Page
```

```

Pages:
+-----+
| Column Addresses für Page 1 | Page 0 - 0000
+-----+
| Column Addresses für Page 2 | Page 1 - 0001
+-----+
| ....                        | Page 2 - 0010
+-----+
| ....                        | Page 3 - 0011
+-----+
| ....                        | Page 4 - 0100
+-----+
| ....                        | Page 5 - 0101
+-----+
| ....                        | Page 6 - 0110
+-----+
| ....                        | Page 7 - 0111
+-----+


Column Addresses:

      S00 S01 S02 S03 S04 S05 S06      S83 hex - Column address
    +---+---+---+---+---+---+   ... +---+
D0 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+   +---+   +---- Column most significant 4 bit
D1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+   +---+   +---- Column least significant 4 bit
D2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+   +---+
D3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+   +---+
...
+---+---+---+---+---+---+   +---+
D7 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+   +---+
|
+---- Data (Datenbits innerhalb des geschriebenen Datenbyte) - jedes Bit ist ein Pixel


-----
Definition von Ressourcen
-----


' ----- LED's -----
Alive_pin Alias Pinc.0                               ' GPIO für Alive-LED (für DDR oder In-
put)                                                    put)
Alive Alias Portc.0                                   ' GPIO für Alive-LED (für Output oder
Pullup)                                                 Pullup)


' ----- LCD-Display -----
LCD-Display
Glcd_cs1 Alias Portd.4                                ' GPIO PD4
Glcd_a0 Alias Portd.7                                  ' GPIO PD7
Glcd_si Alias Portb.3                                 ' MOSI
Glcd_sclk Alias Portb.5                              ' SCK
Glcd_rst Alias Portd.5                               ' GPIO PD5


-----
Definition von Konstanten
-----


' ----- Für Testumgebung bzw. Traceausgaben -----
Const Main_testmodus = 0                             ' Flag für Testmodus Allgemeinsystem
Const Trace_mode = 1                                 ' Flag wird verwendet um Traces auf se-
rieller Schnittstelle auszugeben


' ----- Allgemeine Systemkonstanten -----
Tatsächliches Allgemeines
Const Led_au = 0
Const Led_ein = 1

```

```

Const Led_aus = 1                                ' Achtung !! bei STK500 ist Logik ge-
dreht!!                                           dreht!!
Const Led_ein = 0                                ' Achtung !! bei STK500 ist Logik ge-
dreht!!                                           dreht!!

Const False = 0                                  ' Boolsche Variable FALSE nachgebildet
Const True = 1                                   ' Boolsche Variable TRUE nachgebildet

' Zeitvorgabe für Sekunden-Timer
Const Timervorgabe = 34286                        ' Timer von 1 Sekunden (SekundenTick)

' ----- LCD -----

Const Glcd_cmd_display_off = &B10101110          ' (1.1) LCD Display - off
Const Glcd_cmd_display_on = &B10101111           ' (1.1) LCD Display - on

Const Glcd_cmd_display_start_line = &B01000000    ' (2) Set the display RAM display start
line address                                     '
'                                     variabel: xxxxxx
'                                     &B01xxxxxx 0x00-0x63

Const Glcd_cmd_page_address_set = &B10110000      ' (3) Sets the display RAM page address
'                                     &B1011xxxx 0x00 - 0x07 (Page 0 -
7)                                                '
'                                     variabel: xxxx

Const Glcd_cmd_column_address_set_msca = &B00010000 ' (4.1) Sets the most significant 4 bits
of the display RAM column address               '
'                                     &B0001xxxx 0x00 - 0x07 (Block 0
- 7)                                             '
'                                     variabel: xxxx

Const Glcd_cmd_column_address_set_lsca = &B00000000 ' (4.2) Sets the least significant 4
bits of the display RAM column address           '
'                                     &B0000xxxx 0x00 - 0x0F (Column 0
- 15)                                           '
'                                     variabel: xxxx

Const Glcd_cmd_disp_normal = &B10100110          ' (9.1) Display - normal
Const Glcd_cmd_disp_inverted = &B10100111        ' (9.2) Display - inverted

Const Glcd_cmd_disp_all_point_normal = &B10100100 ' (10.1) Display all points - normal
display                                          '
Const Glcd_cmd_disp_all_point_on = &B10100101    ' (10.2) Display all points - all poins
ON                                              '

Const Glcd_cmd_reset = &B11100010                ' (14) Internal reset

Const Glcd_cmd_set_contrast = &B10000001          ' (18.1) Electronic volume mode set
Const Glcd_cmd_set_contrast_data = &B00000000    ' (18.2) Electronic volume register set
'                                     &B00xxxxxx 0x00 - 0x3F (0-63)
'                                     variabel: xxxxxx

Const Glcd_cmd_nop = &B11100011                  ' (22) NOP

' ----- Serielle Kommunikation ----
Const Char_cr = &H0D                              ' ASCII Code für CR
Const Char_lf = &H0A                              ' ASCII Code für LF

Const Serial_in_buffer_size = 64                  ' Buffer soll 48 Byte haben

' ----- Variablen für Verarbeitung von Messdaten -----
Const String_data_delimiter = ","

' ----- Definition von Variablen und Datentypen -----
' ----- Temporäre Hilfsvariablen -----
Dim Temp_byte_1 As Byte                          ' Temporäre Byte-Variable 1
Dim Temp_byte_2 As Byte                          ' Temporäre Byte-Variable 2

```

```

Dim Temp_word_1 As Word                                ' Temporäre Word-Variable 1

' ----- Variablen für LCD-Display -----
' Ein Teil des Display wird als Grafikspeicher abgebildet
Dim Display_memory(512) As Byte                        ' 4 Zeilen a 128 Zeichen = 512 Bytes
Dim Display_memory_line1(128) As Byte At Display_memory(1) Overlay ' Overlay für Zeile 1
Dim Display_memory_line2(128) As Byte At Display_memory(129) Overlay ' Overlay für Zeile 2
Dim Display_memory_line3(128) As Byte At Display_memory(257) Overlay ' Overlay für Zeile 3
Dim Display_memory_line4(128) As Byte At Display_memory(385) Overlay ' Overlay für Zeile 4

' ----- Variablen für Datenstring -----
Dim Serial_receive_string As String * 48              ' Empfangsstring vom Geiger

' ----- Variablen für Verarbeitung von Messdaten -----
Dim String_data_elements(7) As String * 10           ' Datenstring von Geiger enthält 7 Elemente
Dim String_element_count As Byte                     ' Zähler für enthaltene Elemente
Dim Cps_value As Word                                ' Arbeitsvariable für counts per second
Dim Data_counter As Word                             ' Zähler für empfangene Pakete
Dim Cps_array(60) As Byte                             ' Byte Array für 60 CPS Messwerte für Grafik
Dim Bitmuster_dword As Dword                         ' Bitmuster 32 BIT
Dim Bitmuster_byte_array(4) As Byte At Bitmuster_dword Overlay ' Bitmuster in 4 einzelne Bytes unterteilt

' ----- Prototyping -----
' Prototyping
' ----- kein Prototyping in diesem Projekt -----

' ----- Konfiguration und Basiseinstellungen (Projekt und Testumgebung) -----
' ----- CONFIG -----
' ----- Timer -----
' Konfiguration eines Timers für 1 Sekunden Timer-Tick (Scheduler und Alive)
Config Timer1 = Timer , Prescale = 256                ' Timer 1 verwenden
On Timer1 Sekunden_tick                               ' Interrupt Routine
Timer1 = Timervorgabe
Enable Timer1                                          ' Interrupt für Sekunden-Tack

' ----- LCD Display -----
' Konfiguration LCD Display
Config Graphlcd = 128 * 64eadogm , Cs1 = Glcd_cs1 , A0 = Glcd_a0 , Si = Glcd_si , Sclk = Glcd_sclk ,
Rst = Glcd_rst

Initlcd                                              ' LCD Initialisieren (normal beim ersten
LCD Zugriff automatisch)
Cls                                                  ' Display löschen
SetFont Font8x8tt                                   ' Kleiner FONT laden

' ----- Port's und Pin's -----
' ----- LED-Konfigurationen -----
Config Alive_pin = Output                            ' Toggelt und zeigt dass CPU lebt

' ----- Serieller Datenempfang über Interrupt mit Buffer -----

```

```

Config Input = Crlf , Echo = Crlf ' Endsymbole für String definieren
Config Serialin = Buffered , Size = Serial_in_buffer_size ' Empfang von Daten via Interrupt und
Empfangsbuffer
' -> auf diese Weise wird sichergestellt dass kein Byte verloren geht

' ----- Variablen und Werte -----
' ----- LED-Konfigurationen -----
Alive = Led_aus ' Alive-LED aus

' ----- Verarbeitung von Messwerten -----
' Data Empfangscounter der auf dem Display angezeigt wird (Sekundenrhythmus) auf NULL setzen
Data_counter = 0

' Messwerteouffer von 60 Messwerten für Grafik initialisieren
' Anmerkung: Messwerte werden von rechts nach links in das Display eingeschoben
For Temp_byte_1 = 1 To 60 Step 1
    Cps_array(temp_byte_1) = 0
Next Temp_byte_1

' ----- LCD-Display -----
' Grafikspeicher initialisieren
For Temp_word_1 = 1 To 512 Step 1
    Display_memory(temp_word_1) = &H00 ' Bildschirmspeicher auf NULL setzen
Next Temp_word_1

' Linien für Vorlage erzeugen (Bitwerte entsprechen einzelnen Pixeln auf dem Display)
' Linien mit Skalenunterteilung y-Achse
Display_memory(1) = &B01001000
Display_memory(129) = &B10010010
Display_memory(257) = &B00100100
Display_memory(385) = &B01001001

Display_memory(2) = &B11111111
Display_memory(130) = &B11111111
Display_memory(258) = &B11111111
Display_memory(386) = &B01111111

' Strich x-Achse
For Temp_word_1 = 387 To 512 Step 1
    Display_memory(temp_word_1) = &B01000000
Next Temp_word_1

' Rahmenlinie rechter Rand
Display_memory(128) = &B11111111
Display_memory(256) = &B11111111
Display_memory(384) = &B11111111
Display_memory(512) = &B01111111

' ----- Und los gehts, hier noch die Restarbeiten -----
' ----- Freigabe aller Interrupts -----
Enable Interrupts ' Damit auch Empfang von Daten über Buf-
fer

' ----- Gosub's -----

' #####
'
' Hauptprogramm ConvCtrl
' #####

```



```

Print "Programm start"                                ' Wird unabhängig von Trace Flags ausge-
geben                                                  geben

' Bildschirm löschen
Cls

' Kopfzeile in erste Linie schreiben
Lcdat 1 , 1 , "* MyGeigerle V1 *" , 1                ' Text invers ausgeben
Lcdat 6 , 1 , "CPS:"
Lcdat 7 , 1 , "CPM:"
Lcdat 8 , 1 , "uSv/h:"

' Grafikrahmen zeichnen
Gosub Write_display_memory

' Auf erste Nachricht Warten und dann auslesen / löschen und weiter -> Sync für vollständige Nach-
richt
Input Serial_receive_string Noecho
Clear Serialin

' -----
' ----- Hier ist die Programmhauptschleife -----
' -----

Do                                                    ' Hauptschleife

' Mit INPUT einen gesamten String der vom Geiger empfangen wurde auslesen
' Achtung: Input wartet wegen Konfiguration auf CR+LF

Input Serial_receive_string Noecho
Clear Serialin                                     ' Buffer löschen

                                                    ' Message Counte erhöhen

Incr Data_counter

' Zunächst alle Leerzeichen in dem Empfangsstring entfernen
Delchars Serial_receive_string , " "

' Nun Zeichenkette in Elemente zerschneiden
String_element_count = Split(serial_receive_string , String_data_elements(1) , String_data_delim-
iter)

' String_data_elements(1) - fester String "CPS"
' String_data_elements(2) - variable CPS (byte)
' String_data_elements(3) - fester String "CPM"
' String_data_elements(4) - variable CPM (word)
' String_data_elements(5) - fester String "uSv/h"
' String_data_elements(6) - variable uSv/h (float xx.xx)
' String_data_elements(7) - Mode SLOW/FAST/INST

' Ggf. Zeichen wieder auf serieller Schnittstelle ausgeben
#if Trace_mode
Print Serial_receive_string
Print String_data_elements(1) ; "      : " ; String_data_elements(2)
Print String_data_elements(3) ; "      : " ; String_data_elements(4)
Print String_data_elements(5) ; " : " ; String_data_elements(6)
Print "Mode      : " ; String_data_elements(7)
Print
#endif

' Count Per Second (= Ereignisse pro Sekunde im Zählrohr) von String in Byte wandeln
Cps_value = Val(string_data_elements(2))

' Daten in Messwertarray für Grafik speichern, dazu Array links schieben und speichern
For Temp_byte_1 = 1 To 59 Step 1

    Cps_array(temp_byte_1) = Cps_array(temp_byte_1 + 1)

Next Temp_byte_1
Cps_array(60) = Cps_value

' Grafikspeicher aktualisieren (mir ist auf die Schnelle keine bessere Lösung eingefallen)

```

```

For Temp_byte_1 = 1 To 60 Step 1
    ' Schleife über alle Messwerte
    Select Case Cps_array(temp_byte_1)
        Case 0 : Bitmuster_dword = &H50000000 : Goto Hiergehtesraus ' 1: 0101 0000 0000 0000
0000 0000 0000 0000
        Case Is <= 9 : Bitmuster_dword = &H58000000 : Goto Hiergehtesraus ' 2: 0101 1000 0000
0000 0000 0000 0000
        Case Is <= 19 : Bitmuster_dword = &H5C000000 : Goto Hiergehtesraus ' 3: 0101 1100
0000 0000 0000 0000 0000 0000
        Case Is <= 28 : Bitmuster_dword = &H5E000000 : Goto Hiergehtesraus ' 4: 0101 1110
0000 0000 0000 0000 0000 0000
        Case Is <= 38 : Bitmuster_dword = &H5F000000 : Goto Hiergehtesraus ' 5: 0101 1111
0000 0000 0000 0000 0000 0000
        Case Is <= 47 : Bitmuster_dword = &H5F800000 : Goto Hiergehtesraus ' 6: 0101 1111
1000 0000 0000 0000 0000 0000
        Case Is <= 57 : Bitmuster_dword = &H5FC00000 : Goto Hiergehtesraus ' 7: 0101 1111
1100 0000 0000 0000 0000 0000
        Case Is <= 66 : Bitmuster_dword = &H5FE00000 : Goto Hiergehtesraus ' 8: 0101 1111
1110 0000 0000 0000 0000 0000
        Case Is <= 76 : Bitmuster_dword = &H5FF00000 : Goto Hiergehtesraus ' 9: 0101 1111
1111 0000 0000 0000 0000 0000
        Case Is <= 85 : Bitmuster_dword = &H5FF80000 : Goto Hiergehtesraus ' 10: 0101 1111
1111 1000 0000 0000 0000 0000
        Case Is <= 94 : Bitmuster_dword = &H5FFC0000 : Goto Hiergehtesraus ' 11: 0101 1111
1111 1100 0000 0000 0000 0000
        Case Is <= 104 : Bitmuster_dword = &H5FFE0000 : Goto Hiergehtesraus ' 12: 0101 1111
1111 1110 0000 0000 0000 0000
        Case Is <= 113 : Bitmuster_dword = &H5FFF0000 : Goto Hiergehtesraus ' 13: 0101 1111
1111 1111 0000 0000 0000 0000
        Case Is <= 123 : Bitmuster_dword = &H5FFF8000 : Goto Hiergehtesraus ' 14: 0101 1111
1111 1111 1000 0000 0000 0000
        Case Is <= 132 : Bitmuster_dword = &H5FFFC000 : Goto Hiergehtesraus ' 15: 0101 1111
1111 1111 1100 0000 0000 0000
        Case Is <= 142 : Bitmuster_dword = &H5FFFE000 : Goto Hiergehtesraus ' 16: 0101 1111
1111 1111 1110 0000 0000 0000
        Case Is <= 151 : Bitmuster_dword = &H5FFFF000 : Goto Hiergehtesraus ' 17: 0101 1111
1111 1111 1111 0000 0000 0000
        Case Is <= 161 : Bitmuster_dword = &H5FFFF800 : Goto Hiergehtesraus ' 18: 0101 1111
1111 1111 1111 1000 0000 0000
        Case Is <= 170 : Bitmuster_dword = &H5FFFFC00 : Goto Hiergehtesraus ' 19: 0101 1111
1111 1111 1111 1100 0000 0000
        Case Is <= 179 : Bitmuster_dword = &H5FFFFE00 : Goto Hiergehtesraus ' 20: 0101 1111
1111 1111 1111 1110 0000 0000
        Case Is <= 189 : Bitmuster_dword = &H5FFFFFF00 : Goto Hiergehtesraus ' 21: 0101 1111
1111 1111 1111 1111 0000 0000
        Case Is <= 198 : Bitmuster_dword = &H5FFFFFF80 : Goto Hiergehtesraus ' 22: 0101 1111
1111 1111 1111 1111 1000 0000
        Case Is <= 208 : Bitmuster_dword = &H5FFFFFFC0 : Goto Hiergehtesraus ' 23: 0101 1111
1111 1111 1111 1111 1100 0000
        Case Is <= 217 : Bitmuster_dword = &H5FFFFFFE0 : Goto Hiergehtesraus ' 24: 0101 1111
1111 1111 1111 1111 1110 0000
        Case Is <= 227 : Bitmuster_dword = &H5FFFFFFF0 : Goto Hiergehtesraus ' 25: 0101 1111
1111 1111 1111 1111 1111 0000
        Case Is <= 236 : Bitmuster_dword = &H5FFFFFFF8 : Goto Hiergehtesraus ' 26: 0101 1111
1111 1111 1111 1111 1111 1000
        Case Is <= 246 : Bitmuster_dword = &H5FFFFFFFC : Goto Hiergehtesraus ' 27: 0101 1111
1111 1111 1111 1111 1111 1100
        Case Is <= 255 : Bitmuster_dword = &H5FFFFFFFE : Goto Hiergehtesraus ' 28: 0101 1111
1111 1111 1111 1111 1111 1110

    End Select

    Hiergehtesraus: ' Das ist nicht schön aber es muss sein
    :-)

    ' Mit Overlay Grafikspeicher befüllen
    Temp_byte_2 = Temp_byte_1 * 2 ' Dies ist der Index in die Datenstruk-
    turen
    Temp_byte_2 = Temp_byte_2 + 2
    Display_memory_line1(temp_byte_2) = Bitmuster_byte_array(1)
    Display_memory_line2(temp_byte_2) = Bitmuster_byte_array(2)
    Display_memory_line3(temp_byte_2) = Bitmuster_byte_array(3)
    Display_memory_line4(temp_byte_2) = Bitmuster_byte_array(4)

    Next Temp_byte_1

```

```

' Und jetzt da der Grafikspeicher aktualisiert wurde kann er auch an das Display übertragen wer-
den
Gosub Write_display_memory

' Prüfen ob Bytegrenze überschritten und Umrechnung erfolgen muss (wird vom Geiger gemeldet)
If String_data_elements(7) = "INST" Then
    Cps_value = Cps_value * 60
End If

' Alte Daten löschen: Leerzeichen anstelle CLS und Restinhalt beizubehalten
Lcdat 6 , 41 , " "
Lcdat 6 , 100 , " "
Lcdat 7 , 41 , " "
Lcdat 8 , 41 , " "
Lcdat 8 , 100 , " "

' Neue Werte auf Display schreiben
Lcdat 6 , 41 , Cps_value
Lcdat 6 , 100 , Data_counter
Lcdat 7 , 41 , String_data_elements(4)
Lcdat 8 , 41 , String_data_elements(6)
Lcdat 8 , 100 , String_data_elements(7)

Loop                                     ' Hauptschleife

'## End Hauptprogramm #####

End

' *****
' Interruptroutinen
' *****

' -----
' Interrupt-Service-Routine (Timer1): Sekunden_tick
' Routine zur Auswertung des Timer Interrupts
' -----

Sekunden_tick:

    ' ----- Programmcode -----

    Timer1 = Timervorgabe                ' Timer neu laden
    Toggle Alive                        ' Alive-LED toggeln lassen

Return
'-- End Sekunden_tick -----

' *****
' Subroutinen
' *****

' *****
' * Vorlage *
' *****

' ----- Keine Subroutinen in diesem Projekt -----

' *****
' GOSubroutinen
' *****

' *****
' * LCD-Display *
' *****

' -----
' GOSUB-Routine: Write_display_memory
' Routine überträgt den Memory-Speicher im RAM in den Speicher des Displays
' -----

' Anmerkung:
' Da Displayspeicher 132 Speicheradressen pro Zeile besitzt, das Display aber
' nur 128 anzeigt, ich aber im Display-Speicher sauber rechnen können will

```

```
' habe ich aktuell keine andere Lösung gefunden als die 4 Zeilen aus dem
' Grafikspeicher sauber separat zu übertragen
```

```
Write_display_memory:
```

```

' ----- Zeile 1 -----

' Sets the display RAM page address
Glcdcmd Glcd_cmd_page_address_set + 1           'Zeile 1

' Setze Block 0 von Page x
Glcdcmd Glcd_cmd_column_address_set_msca + 0

' Setze Column 0 von Block 0
Glcdcmd Glcd_cmd_column_address_set_lsca + 0

' Schleife über gesamten Grafikspeicher
For Temp_word_1 = 1 To 128 Step 1

    Glcddata Display_memory(temp_word_1)

Next Temp_word_1

' ----- Zeile 2 -----

' Sets the display RAM page address
Glcdcmd Glcd_cmd_page_address_set + 2           'Zeile 2

' Setze Block 0 von Page x
Glcdcmd Glcd_cmd_column_address_set_msca + 0

' Setze Column 0 von Block 0
Glcdcmd Glcd_cmd_column_address_set_lsca + 0

' Schleife über gesamten Grafikspeicher
For Temp_word_1 = 129 To 256 Step 1

    Glcddata Display_memory(temp_word_1)

Next Temp_word_1

' ----- Zeile 3 -----

' Sets the display RAM page address
Glcdcmd Glcd_cmd_page_address_set + 3           'Zeile 3

' Setze Block 0 von Page x
Glcdcmd Glcd_cmd_column_address_set_msca + 0

' Setze Column 0 von Block 0
Glcdcmd Glcd_cmd_column_address_set_lsca + 0

' Schleife über gesamten Grafikspeicher
For Temp_word_1 = 257 To 384 Step 1

    Glcddata Display_memory(temp_word_1)

Next Temp_word_1

' ----- Zeile 4 -----

' Sets the display RAM page address
Glcdcmd Glcd_cmd_page_address_set + 4           'Zeile 4

' Setze Block 0 von Page x
Glcdcmd Glcd_cmd_column_address_set_msca + 0

' Setze Column 0 von Block 0
Glcdcmd Glcd_cmd_column_address_set_lsca + 0

```

```

' Schleife über gesamten Grafikspeicher
For Temp_word_1 = 385 To 512 Step 1

    Glcddata Display_memory(temp_word_1)

Next Temp_word_1

Return
'-- End Write_display_memory -----

' -----
' Devices schließend und ggf. "Terminate Programm execution"
' -----

' System halt
End                                     'end program

' -----
' Definition von globalen Konstantenfeldern
' -----

' ----- Grafiken für das GLCD -----
' include the picture data
' -> es gibt aktuell hier keinen Grafiken

' ----- FONT's für das GLCD -----

' Include Font nach dem END und damit nach dem Programmcode damit nicht FONT-
' Data als Code interpretiert wird (sollte ganz am Ende stehen)
$include "font8x8TT.font"

' -----

' ##### END #####

' ##### Historie #####
' 07.01.2019 : Version 1.0
' Erstellung der 1. Version
' #####

```

Software 2: Source-Code des Projekt myGeigerle

15 Interessantes und wichtige Links

15.1 Bücher und Literatur

- myAVR Lehrbuch Mikrocontroller-Programmierung
Laser & Co. Solutions GmbH
- LCD Lehrheft
Laser & Co. Solutions GmbH
- Projekt „myTWI“
Laser & Co. Solutions GmbH
- Mikrocomputertechnik mit Controllern der Atmel AVR-RISC-Familie
G. Schmitt, Oldenburgverlag, ISBN 3-486-58016-7
- Leiterplattendesign mit EAGLE
André Ketler, Marc Neujahr, mit Verlag, ISBN 978-3-8266-1340-1
- Messen, Steuern und Regeln mit AVR_Controllern
Wolfgang Trampert, Franzis Verlag, ISBN 3-7723-4298-1
- Programmierung der AVR RISC Mikrocontroller mit BASCOM-AVR
Claus Kühnel, Books on Demand, ISBN 3907857046

15.2 Internet

15.2.1 Firmen und Foren

SW

MCS Electronics

URL: <http://www.mcselec.com/>
- BASCOM-AVR BASIC Compiler und Forum

HW

Bauelemente:

Conrad Electronic

URL: <http://www.conrad.de>
- Bauelemente und Zubehör (zum Teil aber sehr teuer)

Pollin Electronic

URL: <http://www.pollin.de/shop/shop.php>
- Bauelemente
- ATmega Evaluationsboard

Reichelt Elektronik

URL: <http://www.reichelt.de/>
- Günstige Bauelemente
- STK500 von ATMEL

Farnell Deutschland

URL: <http://de.farnell.com/jsp/home/homepage.jsp>
- Bauelemente

ELV

- URL: <http://shop.elv.de/output/controller.aspx>
 - Günstige Bauelemente
 - LCD-Displays

RS Components GmbH

- URL: <http://www.rsonline.de>
 - Große Auswahl an Bauelementen zu guten Preisen

Hersteller und Spezialitäten:

ATMEL Corporation

- URL: <http://www.atmel.com/>
 - ATmega
 - STK500

Sensirion

- URL: <http://www.sensirion.com/>
 - Halbleiterhersteller für Temperatursensoren und Feuchtigkeitssensoren

TAOS

- URL: <http://www.taosinc.com>
 - Halbleiterhersteller für Lichtsensoren

Riesen + Kern GmbH

- URL: <http://www.driesen-kern.de>
 - Deutscher Distributor für Sensirion Halbleiter

rb-Messtechnik Reinhardt

- URL: <http://www.rb-messtechnik.de>
 - Windgeber

DatasheetCatalog.COM

- URL: <http://www.datasheetcatalog.com>
 - Datenblätter zu fast allen bekannten elektr. Bauelementen

Worls Of Electronic – Elektronikprojekte

- URL: <http://www.woe.onlinehome.de/projekte.htm>
 - AVR JTAG Emulator

Plantinenservice und Hersteller:

GS Electronic

Sven Schult
 Spillbahnstraße 19a
 53844 Troisdorf

Tel. 02241-3010465
 Fax 02241-3010469

eMail gselectronic@gsel.de

- URL: <http://www.gsel.com/>
 - Plantinenservice
 - Einzelstücke

- Kleinserien

Leiton GmbH

Gottlieb-Dunkel-Str. 47-48
12099 Berlin

Tel.: +49-(0)30-701 73 49-0
Fax: +49-(0)30-701 73 49-19

E-Mail: kontakt@leiton.de

URL: <http://www.leiton.de>

- Platinenservice

- Einzelstücke

- Kleinserien

- Sehr günstige Preise

- Leiton stellt Download für Eagle DesignRules zur Verfügung

15.2.2 ATmega SW und HW-Lösungen

Laser & Co. Solutions GmbH

URL: <http://www.myavr.de>

- Bausätze zum ATmega8

- SW-Lösungen zum Selbststudium

- Dokumente

15.2.3 Foren

RoboterNetz.de

URL: <http://www.roboternetz.de>

- Großer Portal für Robotik, Elektronik und Mikrocontroller

MCS Electronics

URL: <http://www.mcselec.com/>

- Forum rund um BASCOM-AVR

AVR feaks

URL: <http://www.avrfreaks.net/>

- Forum AVRFREAKS.NET

Pony-Prog Tutorial

URL: http://www.mikrocontroller.net/articles/Pony-Prog_Tutorial

- Pony-Prog Tutorial

QSLnet

URL: <http://www.qsl.net>

URL: <http://www.qsl.net/pa3ckr/index.html>

URL: <http://www.qsl.net/pa3ckr/bascom%20and%20avr/arrays%20and%20data/index.html>

- Forum für Elektronik und SW-Lösungen (entstanden aus Radio Amateur Community)

- Zusammenfassung BASCOM und AVR Lösungen (Arrays usw.)

AVR_Praxis

URL: <http://www.avr-praxis.de/index.php>

AVR-PRAXiS ist ein Forum, das ausschließlich für einen Gedankenaustausch und als Diskussionsplattform für Interessierte bereitstellt, welche sich privat, durch das Studium oder beruflich mit der AVR-Mikrocontrollerfamilie beschäftigen wollen oder müssen.

Microcontroller.net

URL: <http://www.mikrocontroller.net>

Großes Portal mit Forum und Chat

BASCOM-Forum

URL: <http://www.bascom-forum.de>

Forum für Projekte, Hardware und Diskussionen

Infos rund um den ATmega:

URL: <http://www.dieelektronikerseite.de/uC%20Ecke/Module/Ports%20-%20Wenn%20der%20AVR%20steuert.htm>

URL: <http://www.kreatives-chaos.com/artikel/avr-grundsaltungen>

Informationen zum TWI / I²C-Bus:

URL: <http://www.roboternetz.de/wissen/index.php/I2C>

URL: <http://www.roboternetz.de/wissen/index.php/TWI>

URL: http://www.roboternetz.de/wissen/index.php/TWI_Praxis

URL: http://www.roboternetz.de/wissen/index.php/TWI_Praxis_Multimaster

URL: http://www.roboternetz.de/wissen/index.php/Bascom_I2C_Master

URL: http://www.roboternetz.de/wissen/index.php/Bascom_und_USI-Kommunikation

URL: http://www.roboternetz.de/wissen/index.php/Bascom_Soft-I2c_Library

URL: http://www.roboternetz.de/wissen/index.php/Bascom_Inside-Code

16 Entwicklungsbegleitende Notizen und Informationen

16.1 Projektcheckliste für AVR Systemdesigns

Diese Checkliste beinhaltet einige grundlegende Regeln beim Design mit AVR Mikrocontrollern.

http://www.mikrocontroller.net/articles/AVR_Checkliste

Dies sind zusammengefasst in Kürze:

16.1.1 Abblockkondensator(en) ordnungsgemäß installiert?

Abblockkondensatoren ("Bunker-Kondensatoren") dienen dazu, sehr kurze Versorgungsspannungseinbrüche, die durch Schaltvorgänge verursacht werden können, zu kompensieren. Diesen Zweck erfüllen sie optimal, wenn folgende Regeln eingehalten werden:

- Ein Abblockkondensator sollte möglichst dicht am IC sitzen.
- Jedes IC in einer Schaltung sollte einen Abblockkondensator besitzen.
- Bei ICs mit mehreren Anschlüssen für VCC und GND sollte jedes VCC-GND-Paar mit einem eigenen Abblockkondensator beschaltet werden (z. B. AVR in SMD-Bauform wie dem ATmega16A also mit vier Kondensatoren).
- Es sollten keramische Kondensatoren mit einer Kapazität von 100 nF verwendet werden. Größere Kondensatoren, etwa 10 µF-Elkos, sind für diese Aufgabe *nicht* geeignet, weil sie "zu langsam" sind!

16.1.2 Spannungsversorgung richtig angeschlossen?

Der AVCC-Pin ist der Versorgungsanschluss für den AD-Wandler und den zugehörigen Port. Er ist nicht an allen AVR vorhanden; wenn er aber vorhanden ist, so muss er auf jeden Fall angeschlossen sein, auch wenn der AD-Wandler nicht benutzt wird. Wird der AD-Wandler verwendet, sollte zur Verbesserung der Genauigkeit der AVCC-Pin über einen Lowpass-Filter angeschlossen werden (siehe Datenblatt). Oft funktioniert die Programmierung des Controllers auch, wenn Vcc oder GND nicht richtig angeschlossen ist. Zur Sicherheit kann man mit einem Messgerät direkt an den Anschlüssen des AVR kontrollieren (VCC-GND, AVCC-GND) prüfen, ob die Verbindungen korrekt sind. Es empfiehlt sich, vor dem Einsetzen bzw. Einlöten des Controllers die Versorgungsanschlüsse nochmals zu prüfen, um sicherzustellen, dass man den IC nicht durch eine zu hohe Spannung aufgrund eines Fehlers in der Versorgung zerstört.

16.1.3 Reset-Pin korrekt beschaltet?

Der Reset-Anschluss am AVR ist 'active-low', d. h. wenn man den Pin mit GND (Masse) verbindet, wird der Controller resettet. Zwar haben AVR einen internen Pullup-Widerstand, der den Reset-Pin gegen VCC "zieht", dieser ist jedoch relativ hochohmig (ca. 50 kOhm, vgl. Datenblatt) und reicht unter Umständen nicht aus, um den Reset-Pin sicher "hochzuhalten". Als Mindestbeschaltung empfiehlt sich dringend, einen externen Pullup-Widerstand vorzusehen (typisch 10 kOhm), der den Reset-Pin mit VCC verbindet. Er sollte nicht kleiner als 4,7 kOhm sein, da der Programmieradapter sonst eventuell den Reset-Pin während des Programmiervorgangs nicht sicher auf "low" ziehen kann. Zusätzlich sollte man auch noch einen Kondensator 47 nF oder 100 nF zwischen Reset-Pin und GND anordnen. Dieses RC-Glied sorgt dafür, dass der Controller beim Einschalten der Versorgungsspannung für eine definierte Zeitspanne im Reset gehalten wird. Im laufenden Betrieb sorgt der Kondensator dafür, dass der Reseteingang unempfindlich gegenüber Spikes und Glitches wird. Er sollte deshalb unmittelbar in Pin-Nähe beim Prozessor untergebracht werden. Dieser Kondensator darf jedoch nicht verwendet werden, wenn DebugWire möglich sein soll.

Atmel empfiehlt zusätzlich noch zum Schutz vor Überspannungen eine externe Diode nach VCC ("Clamp-Diode"), da für den Reset-Pin keine interne vorhanden ist. Diese Diode bereitet jedoch bei manchen Programmieradaptern Schwierigkeiten.

16.1.4 Alle Ground-Anschlüsse beschaltet?

Bei AVR's mit mehreren Ground-Anschlüssen müssen alle Anschlüsse beschaltet werden.

Siehe

<http://www.mikrocontroller.net/forum/read-1-107259.html>

16.2 Datenblätter

USB ISP-Programmer für Atmel AVR, Rev.2

USB- ISP-Programmer

Installation

Bitte laden Sie sich das Treiberpaket „AVR-ISP-Driver“ für den ISP-Programmer bei Ihrem Händler im Downloadbereich herunter. Auch in der Produktbeschreibung ist ein Download anwählbar. www.diamex.de im Download-Bereich.

„Stange ISP Prog“ für Win XP
 „DX_ISP.inf“ Vista oder WIN2000
 „avr-isp.inf“ für Win7 - Windows8 Hinweis am Ende der Beschreibung.

Die *.inf Datei auf Festplatte oder einem Suchpfad ablegen. Programmer USB-seitig anschließen und Treiberassistent (automatisch) ausführen.
 Bei Installation fragt Windows nun nach einem Treiber. Es wird der Windows-eigene Treiber verwendet, das wird dem System durch die heruntergeladene *.inf-Datei mitgeteilt. Dazu Häkchen wie Bild setzen und „Weiter“ klicken:

Im nächsten Dialog angeben, wo sich die .inf-Datei befindet.

ISP-SPI-Geschwindigkeiten:

Durch die USB-Anbindung ergeben sich leicht abweichende ISP-Frequenzen gegenüber einem STK500:

STK500	ISP-Prog
921,6kHz	1MHz
230,4kHz	250kHz
57,6kHz	62,5kHz
28,8kHz	28,2kHz
4kHz	4kHz
603Hz	779Hz

LEDs

- USB-LED: Blinken bei USB-Aktivität
- Target-LED: Aus im Leerlauf, Blinken bei Targetzugriffen

Funktionen

- FLASH, EEPROM, Fusebits, Lockbits schreiben, lesen, Chip erase, OSCCAL-Register lesen

Protokoll

- STK500v2 (AVR-Studio Version 4/5/6 anwendbar)

Softwareunterstützung

- -AVR Studio (COM1..COM9)
- -AVRDUDE
- -Bascom mit der Einstellung Options->Programmer = STK500:
 C:\Programme\Atmel\AVR Tools\STK500\stk500.exe (STK500 native driver) und installiertem AVR Studio 4/5/6

Schalterstellung für normale Programmierung in einer Anwenderschaltung = beide Schalter auf OFF

Falls Spannungsversorgung des Target nötig ist:

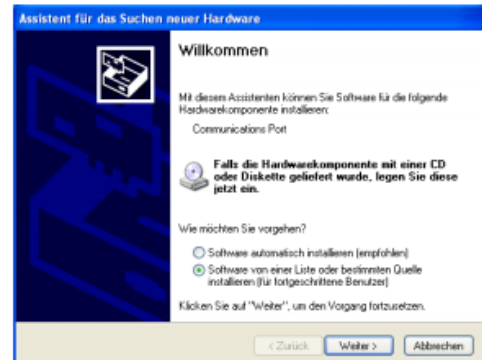
- Für einen einzelnen AVR-Controller nur ohne eigene Schaltung (z.B. im Diamex-Programmieradapter/Schwenkhebler) sind 5V bereitzustellen: 1=on 2=on
- Allgemein: 3,3V maximal 120mA, 5V 150-500mA abhängig vom PC. Schalter 1 off = Target-Spannung aus / on = ein; Schalter 2 off = 3,3V / on = 5,0V.
- Der Programmer benötigt zum Programmierbetrieb keine Stromversorgung vom Target;

Lieferumfang

- ISP-Programmer im bunten Plexiglas-Gehäuse
- USB 2.0 Anschluss
- SPI-Anschlusskabel 6 polig (optional 6 auf 10 Adapter Atmel-Standard – Schaltung rechte Abbildung)
 ca. 15cm Länge
- Beschreibung

Unterstützung folgender Controller:

- **Classic**
 AT90S1200, AT90S2313, AT90S2333, AT90S2343, AT90S4414, AT90S4433, AT90S4434, AT90S8515, AT90S8535
- **Tiny**
 ATtiny12, ATtiny13, ATtiny15, ATtiny2313, ATtiny24, ATtiny25, ATtiny26, ATtiny44, ATtiny45, ATtiny84, ATtiny85, ATtiny86, ATtiny261, ATtiny461, ATtiny861
- **Mega**
 Atmega88, Atmega103, Atmega128, Atmega1280, Atmega1281, Atmega16, Atmega161, Atmega162, Atmega163, Atmega164, Atmega168, Atmega169, Atmega2560, Atmega2561, Atmega32, Atmega324, Atmega325, Atmega3250, Atmega329x, Atmega48, Atmega64, Atmega644, Atmega645, Atmega6450, Atmega649x, Atmega8, Atmega8515, Atmega8535,
- **Weitere**
 AT90CAN128, AT90PWM2, AT90PWM2B, AT90PWM3, AT90PWM3B, AT90USB162, AT90USB646, AT90USB647, AT90USB1286, AT90USB1287
- alle weiteren ATMEGA Prozessoren, die die ISP-Programmierschnittstelle unterstützen. Auch Arduino-Boards.



Treiberinstallation unter Windows 8:

Auch auf Windows 8 funktionieren die Windows 7 Treiber. Es erscheint die Meldung:
 "Die INF-Datei des Drittanbieters enthält keine Digitalsignaturinformationen."
 Einstellungs-Charme > "Ein/Aus" > Umschaltaste gedrückt halten und mit der Maus auf "Neu starten" klicken oder
 Einstellungs-Charme > "PC-Einstellungen ändern" > "Allgemein" > "Erweiterter Start" dann
 "Problembehandlung" > "Erweiterte Optionen" > "Starteinstellungen"
 Der PC startet neu und zeigt ein Menü an, dort die Taste 7 drücken.
 ("Erzwingen der Treibersignatur deaktivieren")
 Nun den Treiber installieren.

ERFOS AVR-ISP

USB-Programmieradapter für AVR-Controller

V07/16

DIP-SCHALTER

SCHALTER 1 + 2: Stromversorgung extern und Spannungspegel

 	1 = ON/OFF, 2 = OFF (externe Spannung aus) An PIN 2 des 6- und 10-poligen Programmieranschlusses liegt keine Spannung an. Die Versorgung des angeschlossenen Microcontrollers muss über eine externe Stromversorgung geschehen. Bitte unbedingt die externe Spannung an PIN2 anlegen! Die Höhe der externen Spannung bestimmt den Pegel auf den Programmierleitungen!
	1 = OFF (3,3V), 2 = ON (externe Spannung ein) Höhe der Spannung auf den Datenleitungen und extern = 3,3 Volt Eine externe Schaltung bzw. ein angeschlossener Controller kann vom AVR-ISP mit Strom versorgt werden.
	1 = ON (5V), 2 = ON (externe Spannung ein) Höhe der Spannung auf den Datenleitungen und extern = 5 Volt Eine externe Schaltung bzw. ein angeschlossener Controller kann vom AVR-ISP mit Strom versorgt werden.

Hinweise:

- Die Höhe der externen Spannung bei Einstellung 5 Volt ist abhängig von der Spannung an der USB-Buchse.
- Die maximale Belastung am Programmieranschluss ist 500mA bei 5 Volt bzw. 100mA bei 3,3 Volt. Wird ein größerer Strom benötigt, bitte die angeschlossene Schaltung mit externem Strom versorgen und DIP-Schalter 2 in Stellung OFF.
- Wenn sich DIP-Schalter 2 in Stellung ON befindet, bitte keine externe Spannung an den Programmieranschluss anlegen.
- Wenn sich DIP-Schalter 2 in Stellung OFF befindet, muss die externe Spannung an den Programmieranschluss (PIN2) angelegt werden um die Höhe der Signalpegel auf den Programmierleitungen festzulegen.
- Wenn PIN2 des 6- oder 10-poligen Programmieranschlusses unbeschaltet ist, muss Schalter 2 auf ON stehen.

LEUCHTDIODEN

LED Grün	Leuchtet konstant, wenn eine USB-Verbindung zum PC besteht.
LED Rot	Flackert beim Programmieren von AVR-Controllern.

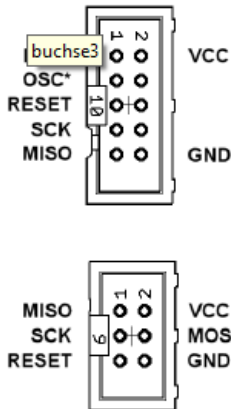
ERFOS AVR-ISP

USB-Programmieradapter für AVR-Controller

V07/16

Anschluss eines AVR-Controllers

Verbinden Sie die Leitungen des Programmieranschlusses auf dem ERFOS-AVR-ISP direkt mit den Pins am Controller:



10-pol. Stiftleiste	6-pol. Stiftleiste	AVR-Controller
PIN1 (MOSI)	PIN4 (MOSI)	MOSI oder PDI
PIN5 (RESET)	PIN5 (RESET)	RESET
PIN7 (SCK)	PIN3 (SCK)	SCK
PIN9 (MISO)	PIN1 (MISO)	MISO oder PDO
Optional: PIN3 (OSC)		XTAL1 (XTALIN)

Die Pin-Nummern der AVR-Controllers variieren abhängig vom Typ und der Gehäuseform. Schauen Sie bitte ins Datenblatt um die zu Ihrem Controller passenden Pin-Nummern zu ermitteln.

HINWEISE, TIPPS!

- Einige AVR-Controller (z.B. AT90CAN32/64/128, ATmega64/128) haben spezielle Programmierpins PDI, PDO. Bitte benutzen Sie diese Leitungen anstatt MISO/MOSI (siehe Datenblatt).
- ERFOS-AVR-ISP unterstützt die adaptive SPI-Bitratenanpassung. Eine zu hoch eingestellte SPI-Bitrate führt in der Regel dazu, dass der angeschlossene AVR-Controller nicht erkannt wird. Die Regel besagt, dass die SPI-Bitrate viermal so hoch sein muss, wie die Taktrate des Controllers. Sollte mit der eingestellten Bitrate keine Verbindung hergestellt werden können, schaltet AVR-ISP automatisch auf niedrigere Bitraten um, bis der Controller antwortet.
- Wenn der angeschlossene AVR-Controller trotzdem nicht antworten will, kontrollieren Sie bitte zunächst die Leitungsverbindungen. MISO an MISO und MOSI an MOSI, das wird häufig vertauscht. Hat der angeschlossene Microcontroller eine Stromversorgung? Eventuell fehlt dem AVR-Controller der Systemtakt, weil die Fuses auf externen Takt eingestellt sind. In diesem Fall kann ein Quarz zur Takterzeugung an die XTAL-Pins des Controllers angeschlossen werden oder das Taktsignal von PIN3 (OSC) der 10-poligen Stiftleiste auf XTALIN oder XTAL1 (siehe Datenblatt des Controllers) gelegt werden. Die Taktfrequenz kann mit ATMEL-Studio über die Funktion „Board-Setting – Clock Generator“ eingestellt werden.
- ERFOS-AVR-ISP unterstützt nicht ATXmega-Controller mit PDI-Schnittstelle und ATTiny-Controller mit TPI-Schnittstelle.

ERFOS AVR-ISP

USB-Programmieradapter für AVR-Controller

V07/16

HINWEISE

© Erwin Reuß; Folker Stange. Nutzung und Weitergabe dieser Informationen auch Auszugsweise nur mit Erlaubnis der Copyright-Inhaber. Alle Markennamen, Warenzeichen und eingetragenen Warenzeichen sind Eigentum Ihrer rechtmäßigen Eigentümer und dienen hier nur der Beschreibung.

HAFTUNGSHINWEIS

Der Hersteller übernimmt keine Haftung für Schäden die durch Anwendung des ERFOS-AVR-ISP entstehen könnten.

LINKS

LED-Genial Online-Shop

<http://www.led-genial.de>

Diamex Online-Shop

<http://www.diamex.de>

AVR/ATMEL-Studio Download

http://www.mikrocontroller.net/articles/Atmel_Studio

Vertrieb



DIAMEX Produktion und Handel GmbH

Innovationspark Wuhlheide
Köpenicker Straße 325, Haus 41
12555 Berlin

Telefon: 030-65762631

E-Mail: info@diamex.de

Homepage: <http://www.diamex.de>

Herstellung

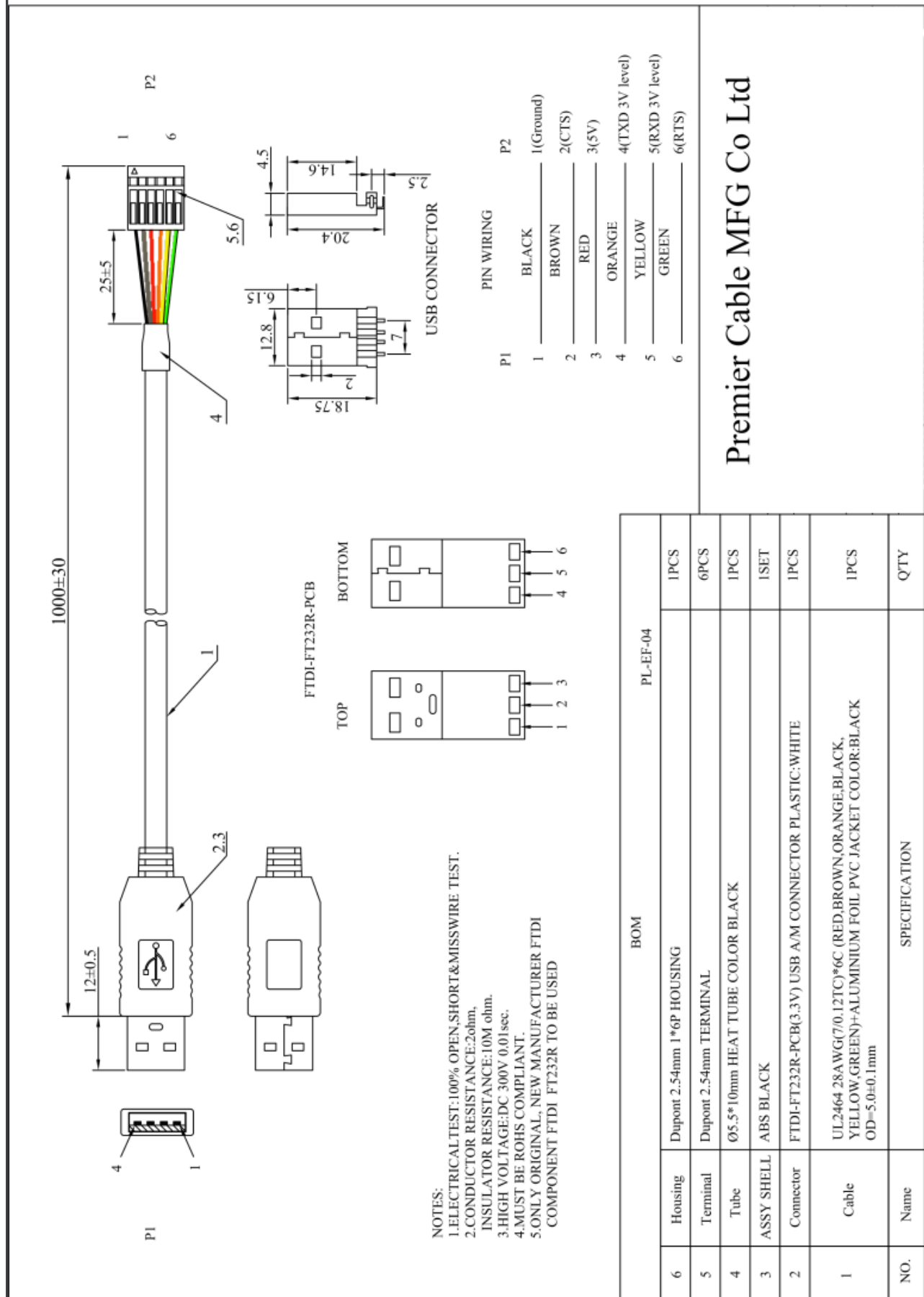


www.tremex.de

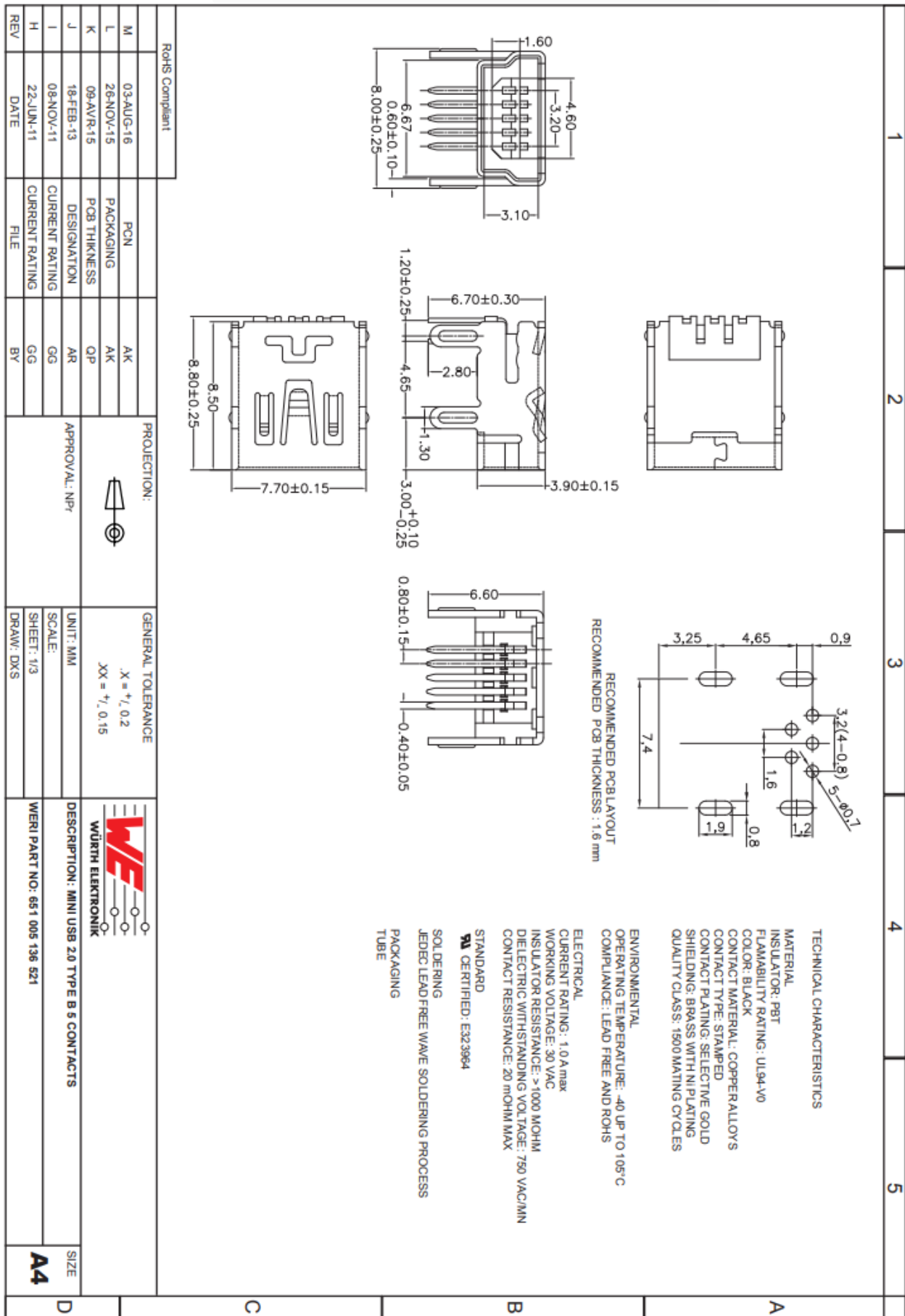
Köpenicker Str. 325 12555 Berlin
Tel. 030-65762631

Hersteller: Tremex GmbH
DIAMEX * OBD-DIAG * TREMEX
WEE-Reg.Nr. DE 51673403

FTDI Serial TTL-232 USB Cable



Mini USB Typ B liegend 5-polig WR-COM Buchse, Einbau horizontal WR-COM Würth Elektronik:



LCD Display

Electronic Assembly EA DOGM128x-6

Zum Beenden des Vollbildmodus **F11** drücken

DOGM GRAFIK SERIE

128x64 PIXEL

Stand 10.2009

ab 1 Stück lieferbar !
auch mit LED: 5,6mm flach



EA DOGM128W-6 +
EA LED55x46-A



EA DOGM128B-6 +
EA LED55x46-W



EA DOGM128W-6 +
EA LED55x46-W

TECHNISCHE DATEN

- * KONTRASTREICHE LCD-SUPERTWIST ANZEIGE (STN UND FSTN) MIT 15µm DOTGAP
- * OPTIONALE LED-BELEUCHTUNG IN VERSCHIEDENEN FARBEN
- * 128x64 PUNKTE (ENTSPRICHT 8x21 ZEICHEN ODER 4x16 ZEICHEN GROSS)
- * KONTROLLER ST 7565R MIT SPI (4-DRAHT) INTERFACE
- * SPANNUNGSVERSORGUNG SINGLE SUPPLY 3,0..3,3V (typ. 270µA)
- * KEINE ZUS. SPANNUNGEN ERFORDERLICH
- * BETRIEBSTEMPERATURBEREICH -20..+70°C (LAGER -30..+80°C)
- * LED-HINTERGRUNDBELEUCHTUNG 5..40mA
- * KEINE MONTAGE ERFORDERLICH: EINFACH NUR IN PCB EINLÖTEN

BESTELLBEZEICHNUNG

GRAFIK 128x64, 55x46mm

x: *W* = weisser Hintergrund (FSTN pos. transfective)

E = Gelb/grüner Hintergrund (STN pos. transmissive)

B = blauer Hintergrund (STN neg. transmissive)

S = schwarzer Hintergrund (FSTN neg. transmissive)

L = Gelb/grüner Hintergrund (STN pos. reflective, nicht beleuchtbar)

LED-BELEUCHTUNG WEISS

LED-BELEUCHTUNG GELB/GRÜN

LED-BELEUCHTUNG BLAU

LED-BELEUCHTUNG ROT

LED-BELEUCHTUNG AMBER

LED-BELEUCHTUNG GRÜN

LED-BELEUCHTUNG RGB VOLLFARBIG

EA DOGM128x-6

EA LED55X46-W

EA LED55X46-G

EA LED55X46-B

EA LED55X46-R

EA LED55X46-A

EA LED55X46-E

EA LED55X46-RGB

ZUBEHÖR

USB-TESTBOARD FÜR PC (WINDOWS 2000, XP, VISTA)

TOUCH PANEL, 4-DRAHT ANALOG ZUM AUFKLEBEN

NULLKRAFTSTECKER FÜR TOUCH, BOTTOM CONTACT

BUCHSENLEISTE 4,8mm hoch (2 Stück erforderlich)

DISKETTE MIT 10 ZEICHENSÄTZEN z.B. 6x8, 8x8, 8x16, kyrillisch

EA 9780-1USB

EA TOUCH128-1

EA WF100-04S

EA FL-20P

EA USBSTICK-FONT

**ELECTRONIC
ASSEMBLY**

making things easy

Zeppelinstraße 19 · D-82205 Gilching · Tel. +49-(0)8105-778090 · Fax: +49-(0)8105-778099 · www.lcd-module.de · info@lcd-module.de

EA DOGM128-6

Seite 2

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

EA DOGM128-6

Mit dem EA DOGM128 erweitert ELECTRONIC ASSEMBLY die EA DOGM Serie um ein Grafikdisplay mit 128x64 Punkten. Auch dieses ist mit Pins ausgestattet, welche eine schnelle und einfache Montage ermöglicht.

Optional sind 7 verschiedene LED-Hintergrundbeleuchtungen lieferbar. Damit sind durch Kombination mit 5 verschiedenen Displaytechniken bis etwa 20 verschiedene Designs denkbar.

Konzipiert für kompakte Handgeräte bietet diese moderne LCD-Serie mit und ohne Beleuchtung eine Reihe echter Vorteile:

- * extrem kompakt mit 55x46 mm bei einem Sichtfenster von 51x31 mm !
- * superflach mit 2,0mm unbeleuchtet bzw. 5,8mm inkl. LED-Beleuchtung
- * serielles SPI Interface (4-Draht)
- * weiter Versorgungsspannungsbereich von 1,8V (dual power) bis zu 3,3V (single supply)
- * nur typ. 270µA Stromverbrauch in vollem Betrieb (LED-Beleuchtung weiss ab 5mA)
- * simple Montage durch einfaches Einlöten
- * verschiedenste Designvarianten ab 1 Stück lieferbar
- * Farbwechsel in der Beleuchtung möglich

LED-BELEUCHTUNGEN

Zur individuellen Hintergrundbeleuchtung sind 7 verschiedene Varianten erhältlich: weiss, gelb/grün, grün, blau, rot, amber und eine Vollfarb-Version.

Bei den monochromen Beleuchtungen stehen jeweils 3 separate LED-Pfade zur Verfügung, welche zur optimalen Anpassung an die Systemspannung parallel oder in Serie geschaltet werden können. Dadurch sind die meisten Beleuchtungen alternativ mit 3,3V oder auch mit 5V zu betreiben!

Zum Betrieb der Hintergrundbeleuchtung empfehlen wir eine Stromquelle (z.B. CAT4238TD) oder einen externen Vorwiderstand zur Strombegrenzung. Dieser errechnet sich aus $R=U/I$; die Werte entnehmen Sie aus der Tabelle nebenan. Für eine optimale Lebensdauer empfehlen wir den Einsatz einer Stromquelle.

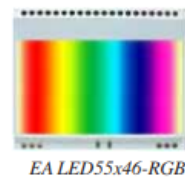
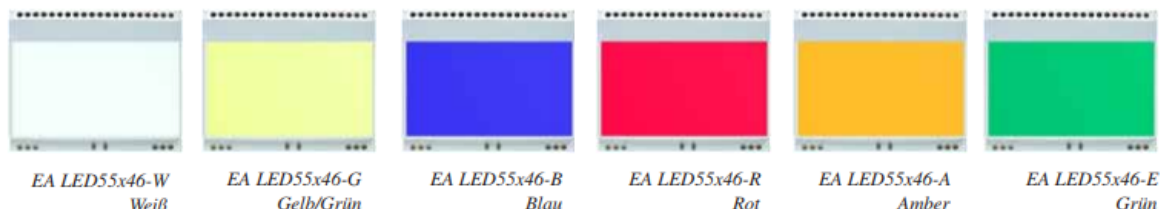
Die Lebensdauer der gelb/grünen, roten und amber-farbenen Beleuchtung beträgt 100.000 Stunden, die der weißen und blauen Beleuchtung deutlich darunter. Wir empfehlen diese wann immer möglich zu dimmen oder abzuschalten.

Achtung: Betreiben Sie die Beleuchtung nie direkt an 5V/3,3V; das kann zur sofortigen Zerstörung der LED's führen! Beachten Sie ein Derating bei Temperaturen >25°C.

LED backlight	Forward voltage (V)			Current max. (mA)	Limiting resistor (ohm, each path)	
	min	typ	max		@3,3 V	@5 V
yellow/green EA LED55x46-G		2,1		40	40	97
white EA LED55x46-W		3,3		25	CAT4238	82
amber EA LED55x46-A		1,9		40	47	103
green EA LED55x46-E		3		40	8,5	57
blue EA LED55x46-B		3,3		30	CAT4238	68
red EA LED55x46-R		1,9		40	47	103
full color EA LED55x46-RGB	red typ. 4V	green typ. 4V	blue typ. 4V	40 each colour	CAT4238	25

6 VERSCHIEDENE BELEUCHTUNGEN

Zur Anpassung an unterschiedlichste Designs stehen 6 verschiedene Beleuchtungsfarben zur Auswahl. Die effektivste und gleichzeitig hellste Beleuchtung ist die weiße EA LED55x46-W.



**ELECTRONIC
ASSEMBLY**
making things easy

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

EA DOGM128-6

Seite 3

5 VERSCHIEDENEN DISPLAY-TECHNOLOGIEN

Als Standard sind 5 verschiedene Technologien in STN und FSTN lieferbar:

Displaytyp	Technologie	optionale Beleuchtung	Lesbarkeit	Displayfarbe unbeleuchtet	Displayfarbe mit Beleuchtung	empfohlene Beleuchtung
	FSTN pos. transflektiv	mit und ohne Beleuchtungskörper zu verwenden	auch bei abgeschalteter Bel. lesbar	schwarz auf weiß	schwarz auf Beleuchtungsfarbe	weiß, blau, grün, RGB
	STN pos. gelb/grün transmissiv	Beleuchtungskörper erforderlich	auch bei abgeschalteter Bel. lesbar	dunkelgrün auf gelb/grün	schwarz auf gelb/grün	gelb/grün, grün
	STN neg. blau transmissiv	nur beleuchtet zu verwenden	---	---	Beleuchtungsfarbe auf blauem Hintergrund	weiß, gelb/grün, grün
	FSTN neg. transmissiv	nur beleuchtet zu verwenden	---	---	Beleuchtungsfarbe auf schwarzem Hintergrund	weiß, grün, RGB
	STN pos. gelb/grün reflektiv	keine Beleuchtung möglich	ohne Beleuchtung bestens lesbar	dunkelgrün auf gelb/grün	---	---

KONTRASTEINSTELLUNG

Für alle Displays der EA DOGM- Serie ist der Kontrast per Befehl einstellbar. In der Regel wird der Kontrast einmalig eingestellt und dann - dank integrierter Temperaturkompensation - über den gesamten Betriebstemperaturbereich (-20...+70°C) konstant gehalten. Dieser einmalige Abgleich ist jedoch in jedem Fall erforderlich.

MONTAGE

Zuerst werden das Display und der jeweilige Beleuchtungskörper aufeinandergesteckt. Dann wird die gesamte Einheit einfach in eine Platine gesteckt und dort verlötet. Bitte beachten Sie, dass die 6 Pins für die Beleuchtung auch von oben verlötet werden müssen.

Achtung: Auf dem Display befinden sich 1-2 Schutzfolien (oben und/oder unten) und auf der Beleuchtung jeweils eine Schutzfolie. Diese müssen entfernt werden.

Achtung: Weder das Display noch die Beleuchtung darf mit Flüssigkeiten aller Art in Berührung kommen (kein Fluxer, Reiniger, Wasser).

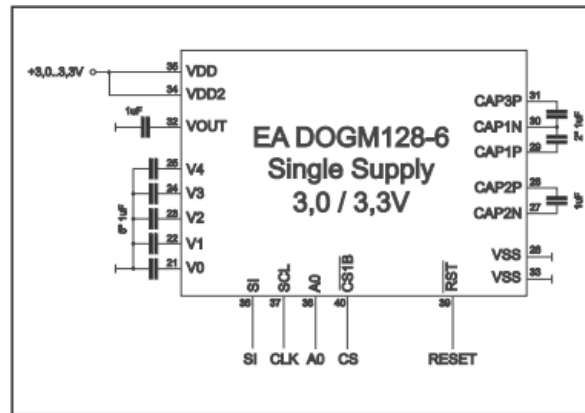
**ELECTRONIC
ASSEMBLY**
making things easy

EA DOGM128-6

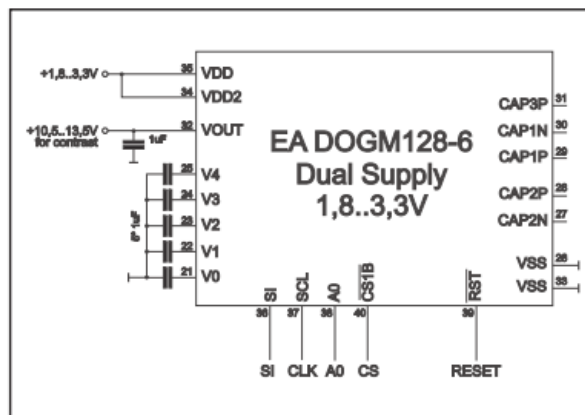
Seite 4

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

APPLIKATIONSBEISPIELE



Zum Betrieb an +3,0V oder +3,3V (single supply)
sind 9 externe Kondensatoren erforderlich.



Stehen +10,5..+13,5V (unstabilisiert) für den
Kontrast zur Verfügung, kann das Display ab 1,8V
Logikspannung betrieben werden.

USB-TESTBOARD EA 9780-1USB

Zum einfachen Start ist ein USB-Testboard für den Anschluss an einen PC erhältlich. Im Lieferumfang ist ein USB-Kabel, sowie eine Windows-Software enthalten. Hierüber können Texte und Bilder (BMP) direkt am angeschlossenen Display dargestellt werden. Weitere Informationen zum Testboard finden Sie im Datenblatt zu EA 9780-1USB.

SIMULATION UNTER WINDOWS

Ein Simulatorfenster zeigt zusätzlich den Displayinhalt. Das bedeutet, dass mit dieser Software bereits alle Displays und Farben ohne Hardware simuliert werden können. Die Software steht auf unserer Website kostenfrei zum Download bereit:

<http://www.lcd-module.de/deu/disk/startdog.zip>



**ELECTRONIC
ASSEMBLY**
making things easy

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

EA DOGM128-6

Seite 5

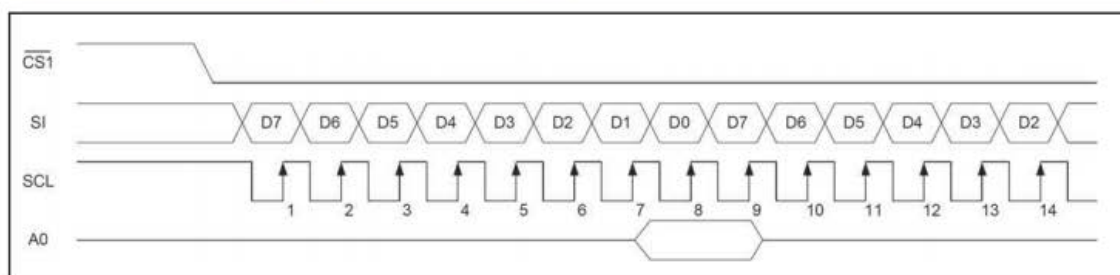
BEFEHLSTABELLE

Command	Command Code									Function
	A0	D7	D6	D5	D4	D3	D2	D1	D0	
(1) Display ON/OFF	0	1	0	1	0	1	1	1	0	LCD display ON/OFF 0: OFF, 1: ON
(2) Display start line set	0	0	1	Display start address				1		Sets the display RAM display start line address
(3) Page address set	0	1	0	1	1	Page address				Sets the display RAM page address
(4) Column address set upper bit	0	0	0	0	1	Most significant column address				Sets the most significant 4 bits of the display RAM column address.
Column address set lower bit		0	0	0	0	Least significant column address				Sets the least significant 4 bits of the display RAM column address.
(6) Display data write	1	Write data								Writes to the display RAM
(8) ADC select	0	1	0	1	0	0	0	0	0	Sets the display RAM address SEG output correspondence 0: normal, 1: reverse
(9) Display normal/reverse	0	1	0	1	0	0	1	1	0	Sets the LCD display normal/ reverse 0: normal, 1: reverse
(10) Display all points ON/OFF	0	1	0	1	0	0	1	0	0	Display all points 0: normal display 1: all points ON
(11) LCD bias set	0	1	0	1	0	0	0	1	0	Sets the LCD drive voltage bias ratio 0: 1/9 bias, 1: 1/7 bias (ST7565R)
(14) Reset	0	1	1	1	0	0	0	1	0	Internal reset
(15) Common output mode select	0	1	1	0	0	0	*	*	*	Select COM output scan direction 0: normal direction 1: reverse direction
(16) Power control set	0	0	0	1	0	1	Operating mode		Select internal power supply operating mode	
(17) V ₀ voltage regulator internal resistor ratio set	0	0	0	1	0	0	Resistor ratio		Select internal resistor ratio(Rb/Ra) mode	
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	Set the V ₀ output voltage electronic volume register
Electronic volume register set		0	0	Electronic volume value						
(19) Static indicator ON/OFF	0	1	0	1	0	1	1	0	0	0: OFF, 1: ON
Static indicator register set		0	0	0	0	0	0	0	0	Mode
(20) Booster ratio set	0	1	1	1	1	1	0	0	0	select booster ratio 00: 2x,3x,4x 01: 5x 11: 6x
(21) Power save	0	step-up value								Display OFF and display all points ON compound command
(22) NOP	0	1	1	1	0	0	0	1	1	Command for non-operation

Column address	
D0 D7	Page 0
D0 D7	Page 1
D0 D7	Page 2
D0 D7	Page 3
D0 D7	Page 4
D0 D7	Page 5
D0 D7	Page 6
D0 D7	Page 7

DATENÜBERTRAGUNG

Die Datenübertragung ist uni-direktional ausgelegt. Das bedeutet, dass Daten nur geschrieben, nicht aber wieder gelesen werden können. Eine bei anderen Displays notwendige Busy-Abfrage ist bei diesem Display nicht notwendig. Die Taktrate der SCL Leitung kann je nach Versorgungsspannung bis zu 20 MHz (3,3V) betragen. Nähere Hinweise zum Timing finden Sie auf den Seiten 64 und 65 im Datenblatt des Controllers ST7565R, auf unserer Homepage unter <http://www.lcd-module.de/eng/pdf/zubehoer/st7565r.pdf>



**ELECTRONIC
ASSEMBLY**
making things easy

EA DOGM128-6

Seite 6

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

INITIALISIERUNGSBEISPIEL

Interne Spannungserzeugung (single supply +3,0V / +3,3V)

Initialisation example for single supply +3.3V (bottom view)											
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark
(2) Display start line set	0	0	1	0	0	0	0	0	0	\$40	Display start line 0
(8) ADC set	0	1	0	1	0	0	0	0	1	\$A1	ADC reverse *)
(15) Common output mode select	0	1	1	0	0	0	0	0	0	\$C0	Normal COM0~COM63
(9) Display normal/reverse	0	1	0	1	0	0	1	1	0	\$A6	Display normal
(11) LCD bias set	0	1	0	1	0	0	0	1	0	\$A2	Set bias 1/9 (Duty 1/65)
(16) Power control set	0	0	0	1	0	1	1	1	1	\$2F	Booster, Regulator and Follower on
(20) Booster ratio set	0	1	1	1	1	1	0	0	0	\$F8	Set internal Booster to 4x
		0	0	0	0	0	0	0	0	\$00	
(17) V0 voltage regulator set	0	0	0	1	0	0	1	1	1	\$27	Contrast set
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	\$81	
		0	0	0	1	0	1	1	0	\$16	No indicator
(19) Static indicator set	0	1	0	1	0	1	1	0	0	\$AC	
		0	0	0	0	0	0	0	0	\$00	Display on
(1) Display ON/OFF	0	1	0	1	0	1	1	1	1	\$AF	

*) Bitte beachten Sie, dass für die 6:00 Darstellung ADC auf „reverse“ gesetzt werden muss (gespiegeltes Layout) !



INITIALISIERUNGSBEISPIEL

Externe Versorgungsspannung für Kontrast (dual power supply)

Einbaulage 6:00 (Bottom View)

Initialisation example for dual power supply (bottom view)											
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark
(2) Display start line set	0	0	1	0	0	0	0	0	0	\$40	Display start line 0
(8) ADC set	0	1	0	1	0	0	0	0	1	\$A1	ADC reverse *)
(15) Common output mode select	0	1	1	0	0	0	0	0	0	\$C0	Normal COM0~COM63
(9) Display normal/reverse	0	1	0	1	0	0	1	1	0	\$A6	Display normal
(11) LCD bias set	0	1	0	1	0	0	0	1	0	\$A2	Set bias 1/9 (Duty 1/65)
(16) Power control set	0	0	0	1	0	1	0	1	1	\$2B	Booster off, Regulator, Follower on
(17) V0 voltage regulator set	0	0	0	1	0	0	1	1	1	\$27	Contrast set
(18) Electronic volume mode set	0	1	0	0	0	0	0	0	1	\$81	
		0	0	0	1	0	1	1	0	\$16	No indicator
(19) Static indicator set	0	1	0	1	0	1	1	0	0	\$AC	
		0	0	0	0	0	0	0	0	\$00	Display on
(1) Display ON/OFF	0	1	0	1	0	1	1	1	1	\$AF	

*) Bitte beachten Sie, dass für die 6:00 Darstellung ADC auf „reverse“ gesetzt werden muss (gespiegeltes Layout) !

**ELECTRONIC
ASSEMBLY**
making things easy

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

EA DOGM128-6

Seite 7

12:00 BLICKWINKEL, TOP VIEW EINBAULAGE

Wird das Display überwiegend von oben abgelesen (z.B. in der Front eines Labornetz-
teils), kann der Vorzugsblickwinkel auf 12:00 Uhr eingestellt werden. Dazu wird das
Display um 180° gedreht eingebaut und der Inhalt per Software
korrigiert. Dazu ist eine geringfügig andere Initialisierung erforder-
lich.

Die Column Adresse ganz links verschiebt sich dann auf die Adresse
4.



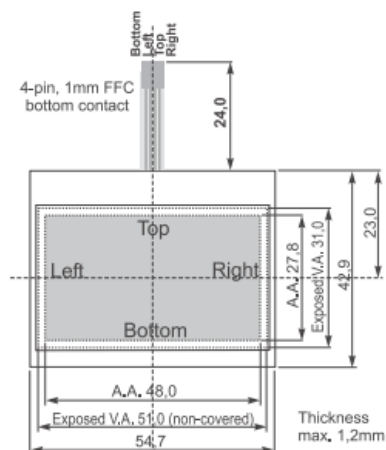
Einbaulage 12:00 (Top View)

Column address	
00	Page 0
01	Page 1
02	Page 2
03	Page 3
04	Page 4
05	Page 5
06	Page 6
07	Page 7

Initialisation example top view												
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark	
(8) ADC set	0	1	0	1	0	0	0	0	0	\$A0	ADC normal	
(15) Common output mode select	0	1	1	0	0	1	0	0	0	\$C8	Reverse COM63-COM0	

TOUCH PANEL EA TOUCH128-1 (OPTIONAL)

Als Zubehör ist ein analoges Touch Panel lieferbar. Es verfügt über eine selbstklebende Rückseite und wird auf das
Display aufgeklebt. Der Anschluss erfolgt über ein 4-poliges Flexkabel für einen ZIF-Stecker im Raster 1,0mm (Zube-
hörer: ZIF-Stecker EA WF100-04S). Der Biegeradius soll mind. 5mm betragen. Für beste Lesbarkeit empfehlen wir
das Display mit einer Beleuchtung auszustatten.

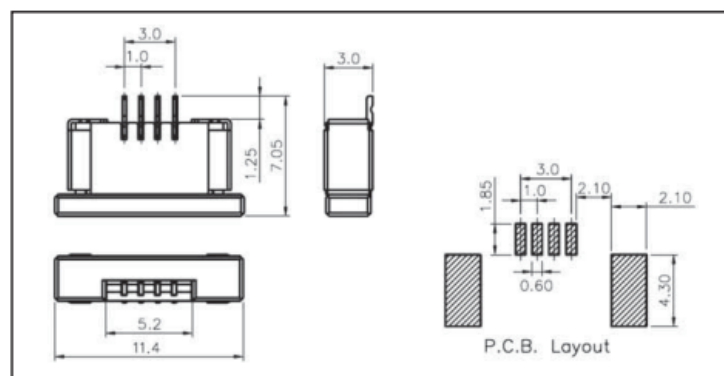


Zur Anbindung an ein Prozessorsystem empfehlen wir entweder die Ver-
wendung eines ext. Touchpanel-Kontrollers oder einen Controller mit
Analogeingängen. Das Touchpanel
ist dann mit einem Potentiometer
vergleichbar: Legt man an die Pins
Top-Bottom eine Spannung von
z.B. 3,3V, kann man am Pin *Left*
oder *Right* eine Spannung welche
linear zur Y-Richtung der Berührung
ist messen. Den Berührungspunkt in
X-Richtung erhält man über das
Anlegen der Spannung an
Left-Right und Messen an *Top* oder
B o t t o m.
Pinbelegung siehe Masszeichnung.

Specification			
Specification	min	max	Unit
Top-Bottom	120	300	Ω
Left-Right	580	900	Ω
Voltage	3	12	V
Current	5	25	mA
Linearity		1,5	%
Force	45	65	g
Contact Bounce	5	10	ms
Op. Temperatur	-20	+60	°C
Stor. Temperatur	-20	+70	°C
Transmission	75	85	%
Life Time	10000		Cycles

NULLKRAFTSTECKER EA WF100-04S

Als Zubehör ist für das Touchpanel ein
Nullkraftstecker (4-polig) im Raster
1,0mm lieferbar. Die Kontaktierung er-
folgt an der Unterseite des Kabels.



alle Maße in mm

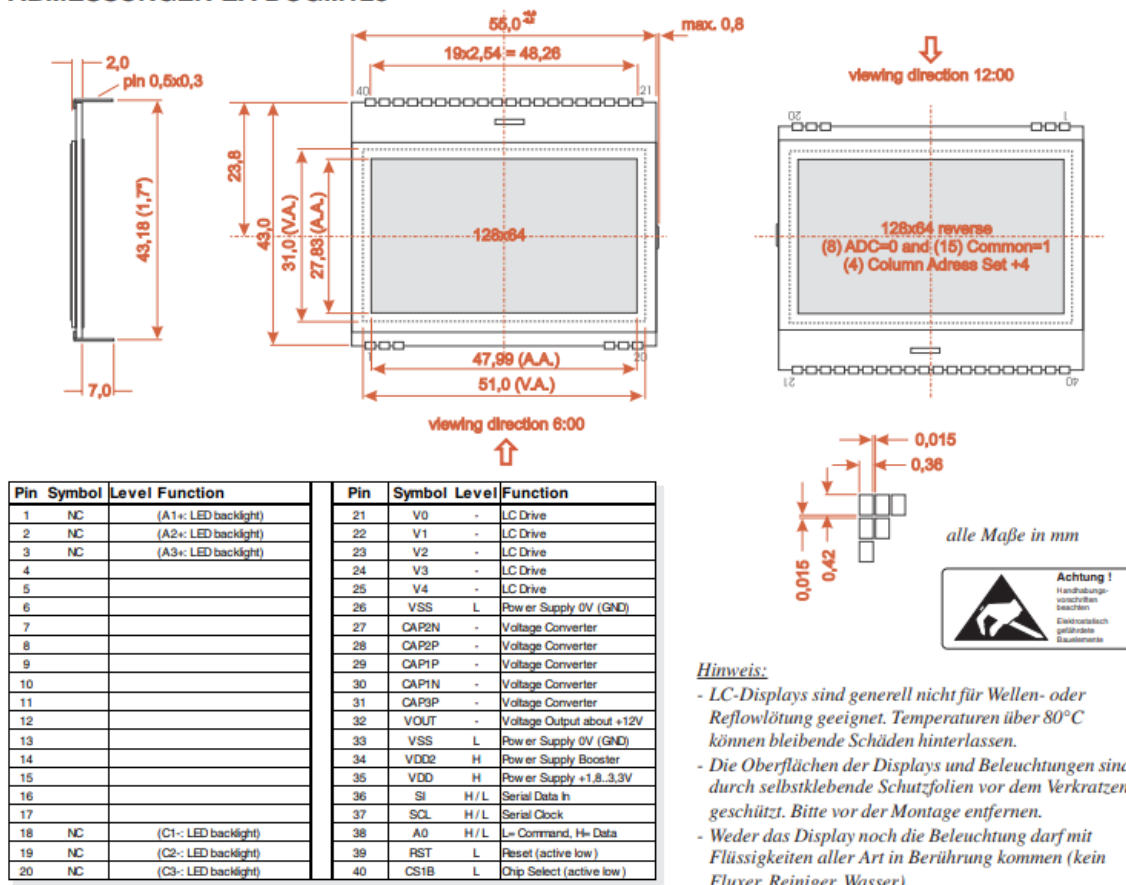
**ELECTRONIC
ASSEMBLY**
making things easy

EA DOGM128-6

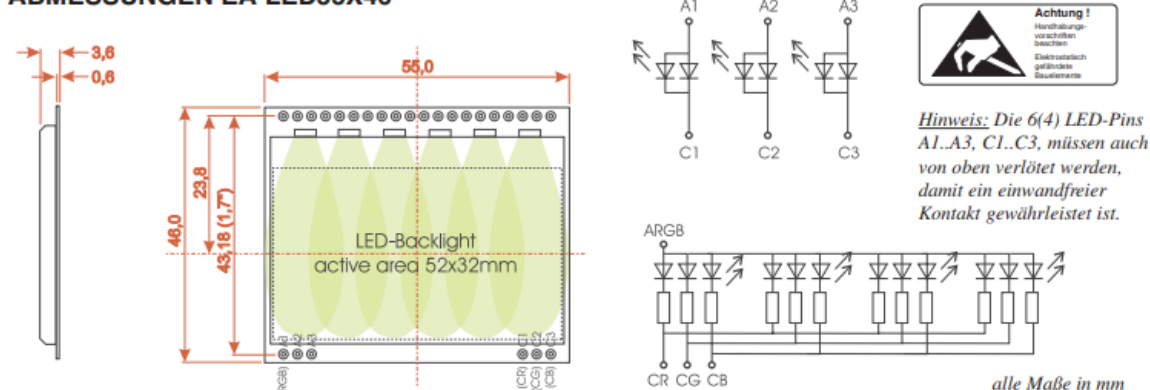
Seite 8

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für
Druckfehler und Applikationsbeispiele.

ABMESSUNGEN EA DOGM128



ABMESSUNGEN EA LED55X46



ELECTRONIC ASSEMBLY

making things easy

Zeppelinstraße 19 · D-82205 Gilching · Tel. +49-(0)8105-778090 · Fax: +49-(0)8105-778099 · www.lcd-module.de · info@lcd-module.de
