Senso

Version 1.0

Geocaching Spiel
Senso

Lonsee im April 2017

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

2 Abbildungsverzeichnis

3 Tabellenverzeichnis

4 Schaltbilderverzeichnis

5 Softwareverzeichnis

6 Historie

Datum	***Entscheidung***
21.12.2016	Beginn der Projektarbeit und Dokumenterstellung
28.03.2017	Fertigstellung des Projekts und Dokumentations-Restarbeiten
29.03.2017	Fertigstellung der Gesamtdokumentation

Tabelle 1: Historie

7 Allgemeines

7.1 Die Entwicklungsumgebung

Die Entwicklungsumgebung meiner Elektronikprojekte:

Abbildung 1: Entwicklungsumgebung Senso

7.2 Allgemeine Beschreibung und Idee zur Realisierung

Die Wikipedia schreibt zum Senso folgendes:

Senso (im englischen Sprachraum Simon) ist ein von Ralph Baer – dem Entwickler der 1972 erschienenen ersten TV-Spielekonsole Magnavox Odyssey – und Howard J. Morrison entwickeltes und in mehreren Ländern patentiertes elektronisches Spiel, welches 1978 bei Milton Bradley veröffentlicht wurde und sich besonders in den 1980er Jahren großer Beliebtheit erfreute. 1979 befand es sich auf der Auswahlliste zum Spiel des Jahres. Neuerdings wird das Spiel auch in Deutschland als Simon verkauft.

Idee
Senso kann allein oder mit mehreren Personen gespielt werden. Das Spiel besteht aus vier großen Feldern in den Farben Rot, Blau, Gelb und Grün. Diese leuchten abwechselnd auf und geben dabei für jede Farbe einen kurzen, individuellen Signalton von sich. Der Spieler muss sich diese Reihenfolge merken und nach Abschluss der Vorgabe durch das Spiel wiederholen. Mit jeder Runde kommt eine weitere Farb-Ton-Kombination hinzu. Mit steigendem Schwierigkeitsgrad leuchten die Felder in schnellerer Reihenfolge auf.

In diesem Geocaching-Projekt Senso wird das Spielkonzept dazu benutzt, eine kleine Elektronik zu entwickeln welche im Gelände versteckt wird, die nach jedem erfolgreichem Spiel die finalen Geokoordinaten der finalen Dose anzeigt.

Es muss eine Reihenfolge von insgesamt 10 Vorgaben richtig absolviert werden, damit das Spiel sein Geheimnis und damit die finalen Koordinaten Preis gibt.

Im Wesentlichen geht es bei diesem Projekt aber auch umn den Ersatz des originalen Geocaching-Spiels das schon seit vielen Jahren Vorort liegt und durch Vandalismus zerstört wurde. Wasser und Elektronik vertragen sich einfach zu schlecht.

7.3 Leistungsumfang

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

- Hauptschalter
- LCD-Display
  - Bedienung des Spiels
  - Statusmeldungen
- 4 LED's in den Farben Rot, Grün, Gelb und Blau mit passenden zugehörigen Tastern
  - Steuerung des Spiels
  - Auswahl und Bestätigung der Vorgabereihenfolge
- Schnittstellen
  - RS232 Schnittstelle zum Tracing und zur Datenübertragung der Logging-Daten an PC
  - ISP Schnittstelle zur direkten Programmierung des Target
- Akustische Signalisierungen über Lautsprecher

7.4 Funktionskomponenten

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

- Spannungsversorgung via 9V Batterie und Erzeugung von 5V über Festspannungsregler SUP
- Mikrocontroller ATmega8L (inkl. ISP, RS232 und Reset) µC
- LCD-Display 1x8 EA DOG-M 081 von Electronic Assembly LCD
  - Die Helligkeit des Display soll über PWM einstellbar sein
  - Bei Innaktivität soll das Display abgeschaltet werden können
- Hauptschalter für Spannungsversorgung SW
- 4 Tasten zur Eingabe der Vorgabeserien (Spieletasten) in den Farben KEY
  Blau (Key1), Rot (Key2), Grün (Key3) und Gelb (Key4)
- LED-Anzeigen für Vorgabeserien und Betriebsanzeige LED
  - Low-Current LED Grün zur Anzeige des Betriebs (Allive LED)
  - LED Rot, Grün, Gelb, Blau als Standard LED für Spielefarben
- Gehäuse mit Batteriefach und Displayfenster BOX
- Akustisches Feedback über Lautsprecher SOUND

8 Elektronische Grundlagen

8.1 Mikrokontroller ATmega8L

Im Projekt wird der Mikrokontroller ATmega8L von ATMEL mit internem Takt von 8MHz eingesetzt. Es wird ein minimalistischer Ansatz gewählt und da Genauigkeit keine Rolle spielt ist der internne Takt ausreichend.

Erste Inbetriebnahmen und Versuche bzgl. Projektumsetzung werden mit dem ATmega8L auf dem STK500 von ATMEL realisiert.

PINOut ATmega8L:

PINOut ATmega8LPDIP

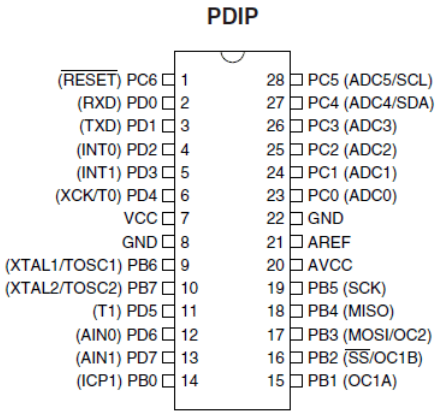


Abbildung 2: PINOut ATmega8L PDIP

PINOut ATmega8LTQFP

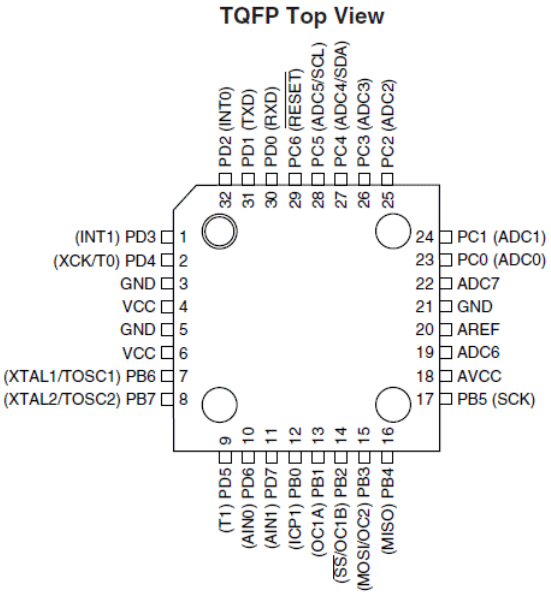


Abbildung 3: PINOut ATmega8L TQFP

Bauteile:

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

Tabelle 2: Stückliste CPU-Unit ATmega8L

8.2 Ressourcenzuordnung ATmega8L im Projekt Senso

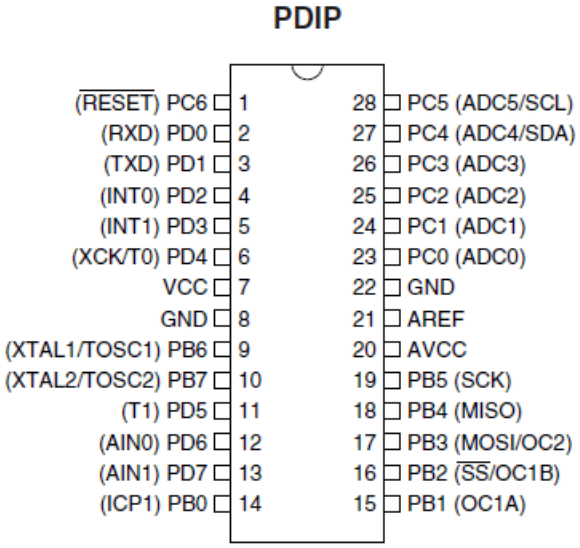


Abbildung 4: PINOut ATmega8L PDIP

PIN	*Port*	*Funktion*	*Used*	*Beschreibung*	*Definition*
1	PC6	RESET	J	Externer Reset Eingang	RESET
2	PD0	RXD	J	RS232 Schnittstelle – Receive Data	RXD
3	PD1	TXD	J	RS232 Schnittstelle – Transmit Data	TXD
4	PD2	PD2	J	Taster 2 Rot (Input)	KEY_2_ROT
5	PD3	PD3	J	Taster 1 Blau (Start mit Interrupt) (Input)	KEY_1_BLAU
6	PD4	PD4	J	Taster 3 Grün (Input)	KEY_3_GRUEN
7	VCC	VCC	J	Spannungsversorgung +5V	VCC
8	GND	GND	J	Ground GND	GND
9	PB6	PB6	J	LED 1 Blau (Output)	LED_1_BLAU
10	PB7	PB7	J	LED 2 Rot (Output)	LED_2_ ROT
11	PD5	PD5	J	LED 3 Grün (Output)	LED_3_GRUEN
12	PD6	PD6	J	LED 4 Gelb (Output)	LED_4_4_GELB
13	PD7	PD7	J	Taster 4 Gelb (Input)	KEY_4_GELB
14	PB0	PB0	J	Alive-LED	ALIVE
15	PB1	PB1	J	Ton / DTMF (Output)	SPEAKER
16	PB2	PB2	J	LCD Datenbus E	LCD_E
17	PB3	MOSI OC2	J	ISP Programmierinterface und PWM für Displayhelligkeit	MOSI
18	PB4	MISO	J	ISP Programmierinterface	MISO
19	PB5	SCK	J	ISP Programmierinterface	SCK
20	AVCC	VCC	J	Spannungsversorgung +5V	VCC
21	AREF	AREF	J	Spannungsversorgung +5V	VCC
22	GND	GND	J	Ground GND	GND
23	PC0	ADC0	J	ADC0 für Batteriespannung	UBATT
24	PC1	PC1	J	LCD Datenbus RS	LCD_RS
25	PC2	PC2	J	LCD Datenbus Bit 0	LCD_D0
26	PC3	PC3	J	LCD Datenbus Bit 1	LCD_D1
27	PC4	PC4	J	LCD Datenbus Bit 2	LCD_D2
28	PC5	PC5	J	LCD Datenbus Bit 3	LCD_D3

Tabelle 3: Ressourcenzuordnung ATmega8L

	Externe Betaktung 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.

8.3 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 Senso-Spiels wird aktuell keine externe Interrupt-Quelle benötigt bzw. verwenden.

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

Die LCD Hintergrundbeleuchtung wird mittels PWM des Timer 2 erzeugt, welche aber keine programmierte Interrupt-Service-Routine anspringt.

8.4 Basisbeschaltung eines ATmega8L inkl. Display

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.

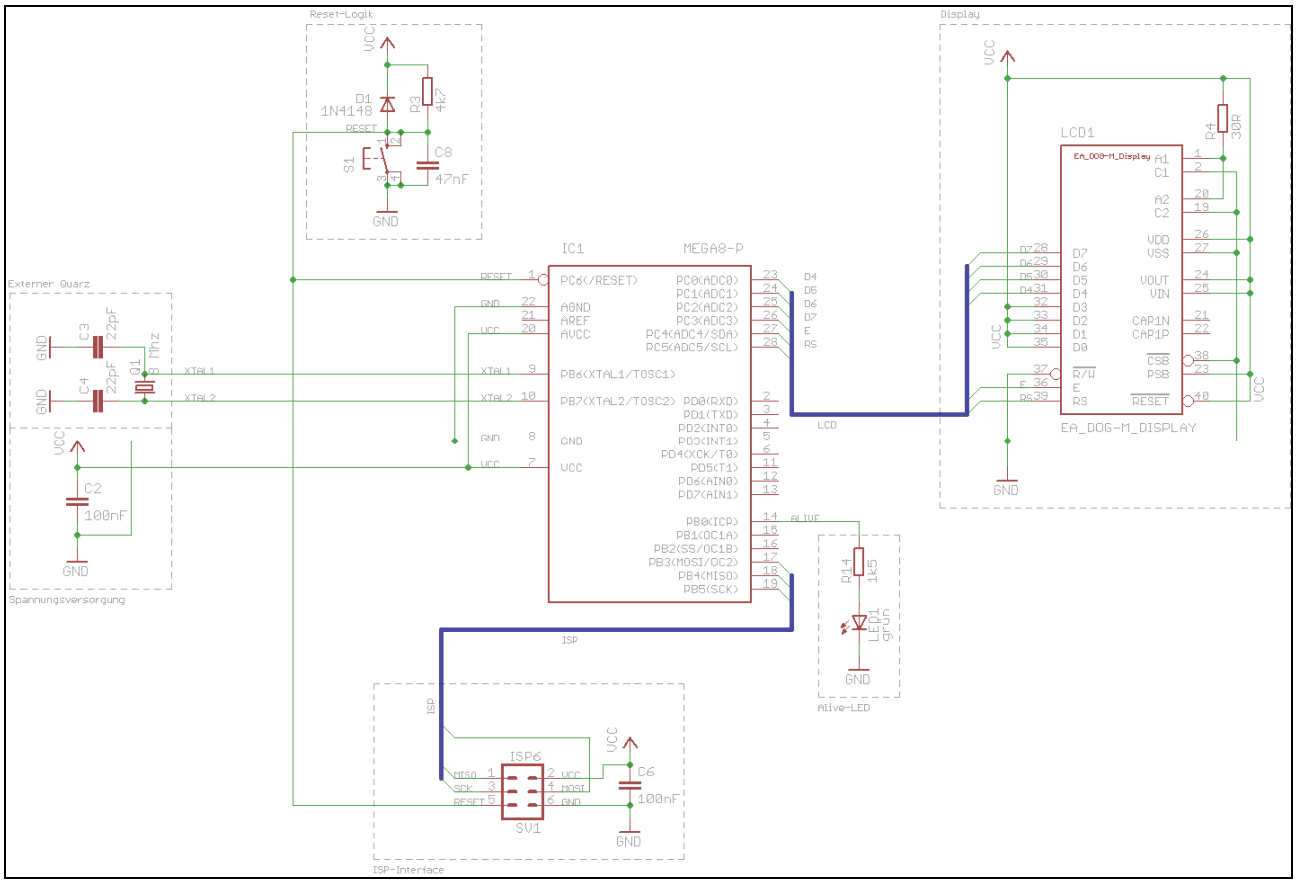
Die wesentlichen Bestandteile sind vorhanden, welche wären:

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

Die Basisbeschaltung im folgenden Beispiel weicht von der tatsächlichen Implementierung im Projekt durch eine Transistor-Stufe ab mittels derer über PWM die Hintergrundbeleuchtung gesteuert und ggf. auch abgeschaltet werden kann.

Im Projekt Senso wird ebenso auf die Reset-Logik verzichtet um Bauteile und Platzverbrauch zu sparen.

Der ATmega8 im Projekt Senso wird über die interne 8MHz Taktquelle getaktet so dass hier kein externer Quarz benötigt wird.



Schaltbild 1: Basisbeschaltung ATmega 8L inklusive Display

Bauteile:

Stückliste: Basisbeschaltung ATmega8L			
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

Reset-Taster:

Als Reset-Taster wird ein Print-Kurzhubtaster verwendet

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

Tabelle 7: Kurzhubtaster für RESET

8.5 ISP-Schnittstelle / ISP-Programmierung

Zur In-System-Programmierung wird die ATMEL ISP-Schnittstelle umgesetzt. Die Programmierung im Projekt erfolgt direkt über BASCOM mit Hilfe von STK500 oder der USB-Programmieradapter AVRISPMKII von ATMEL.

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

ISP Programmierinterface von ATMEL:

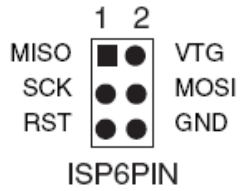
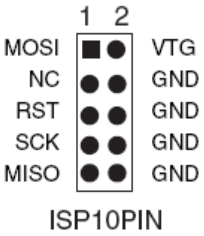
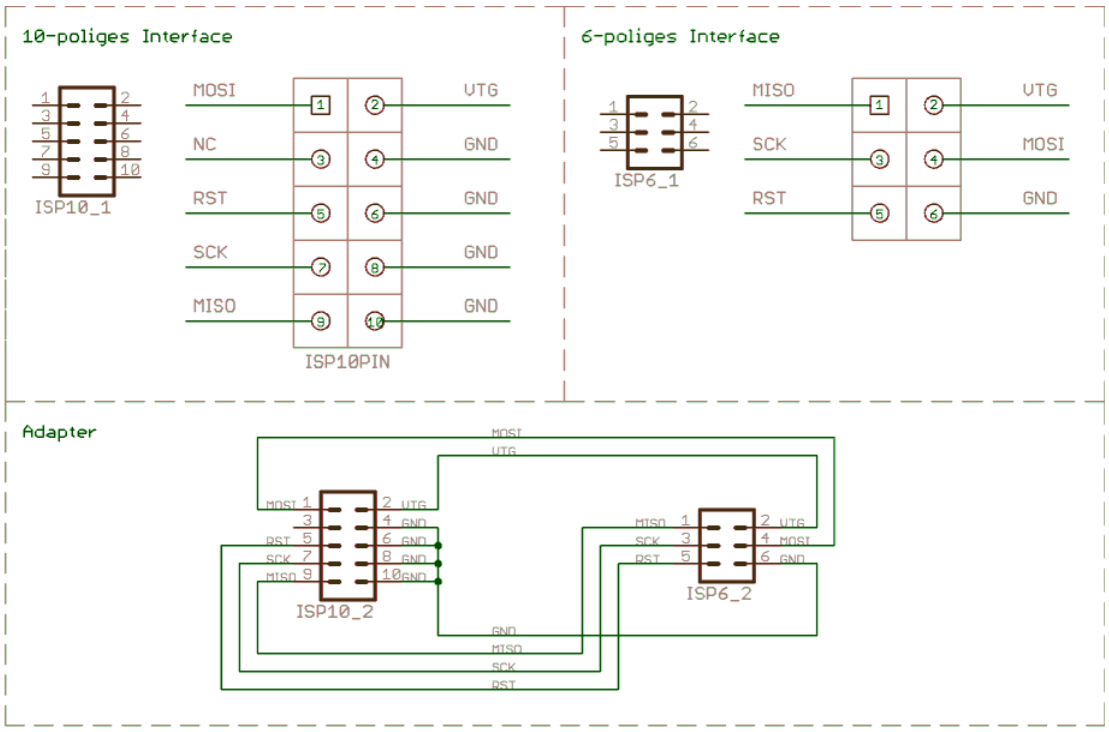
	
6-pin ISP connection Pinout	10-pin ISP connection Pinout

Tabelle 8: ISP connection Pinout

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



Schaltbild 2: 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 AVRISP mkII verbunden werden.

8.6 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	„Sendeanforderung"; 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
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 9: PIN-Belegung des seriellen RS232-Ports

In/Out wird auch Sicht vom PC aus gesehen.

RS232 Buchse 9-polig:

RS232 Stecker 9-polig:

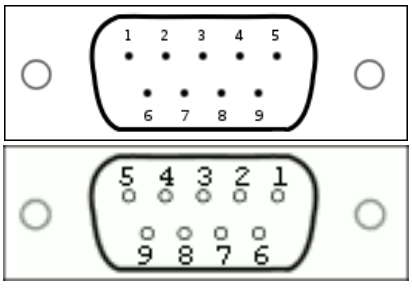
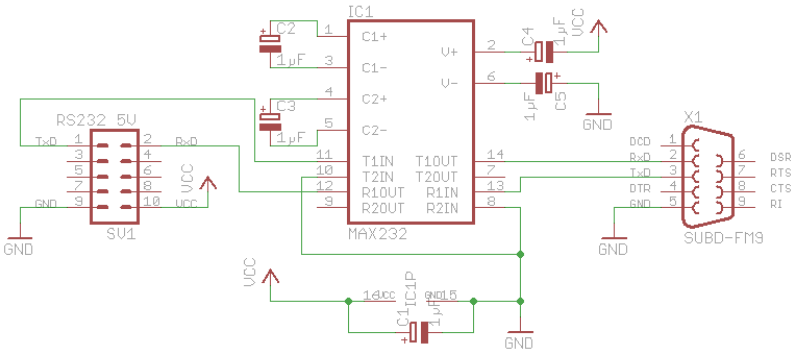


Tabelle 10: PIN-Belegung der 9-poligen RS232 Stecker/Buchse

8.6.1 Adapter RS232 PIN-Header / D-SUB9



Schaltbild 3: 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 11: Stückliste RS232-Adapter zwischen PIN-Header und D-SUB9 Buchse PC

PIN-Belegung Pfostenstecker:

PIN	***Funktion***
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 12: PIN-Belegung RS232-Pfostensteckers

Ressourcenzuordnung zum ATmega8L:

Nummer	***Schaltbild***	***Ressource ATmega8L***
1	RXD	PortD.0 [PD0] (RXD)
2	TXD	PortD.1 [PD1] (TXD)

Tabelle 13: Ressourcenzuordnung SW-RS232 für den ATmega8L

8.6.2 Trace-Adapter RS232 Prototyp auf Lochraster

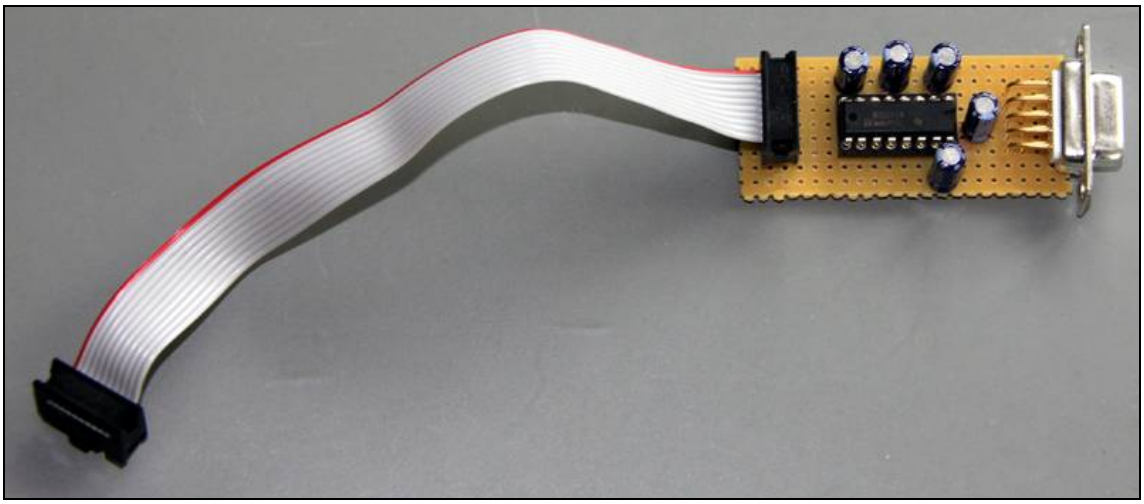


Abbildung 5: RS232-Traceadapter

8.7 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 "verfuseter" 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), das die Funktion eingeschaltet ist. Ist es ein Disable Fusebit bedeutet ein programmiertes Fusebit (0), das die Funkiton 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 Umgebungsbedinung 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 (zb ATmega128) haben nur zwei Schwellwerte. Mit BODLEVEL kann zwischen den Werten

gewechselt werden, mit BODEN wird die Funktion komplett deaktiviert. Neuere Controller (zb 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 zB 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:

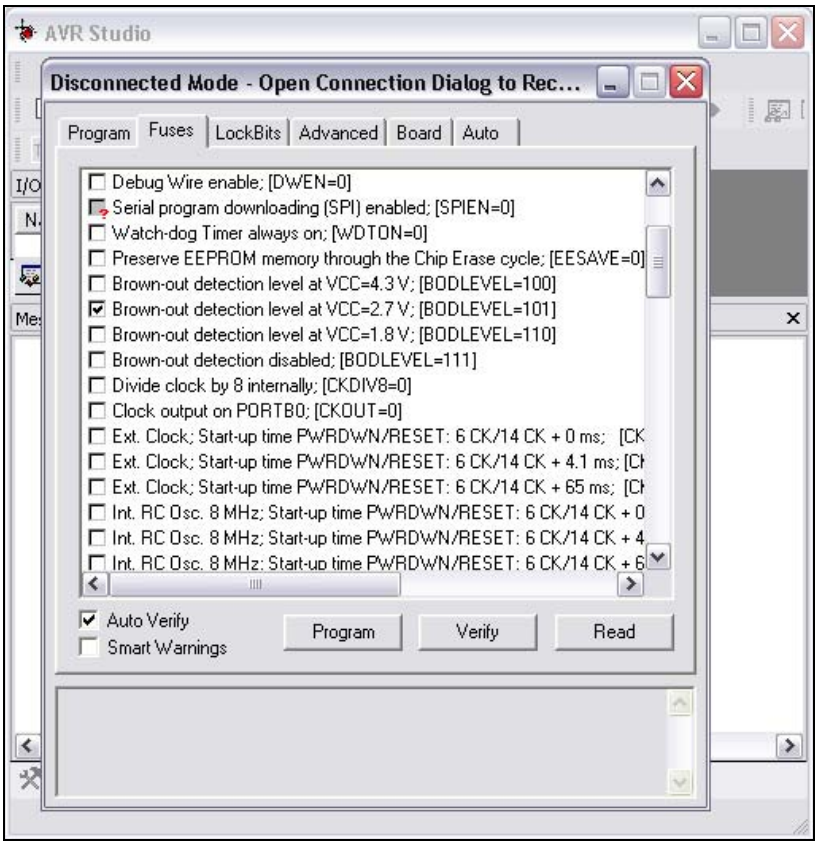


Abbildung 6: Fusebits im AVR Studio

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 zb das die Brown Out Detection auf 2,7V eingestellt wurde.

8.8 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. SeeTable 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 Senso:

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 14: 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 15: 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.

8.8.1 Die Fuse-Konfiguration von Senso

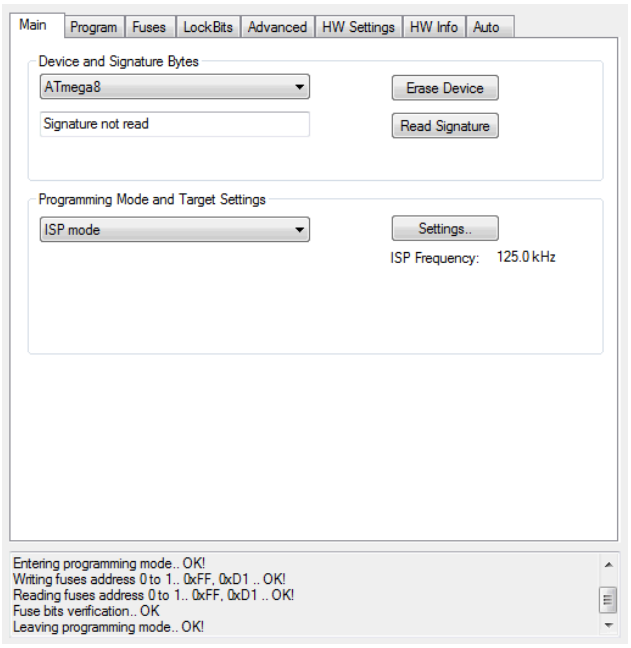


Tabelle 16: AVR-Studio - Main

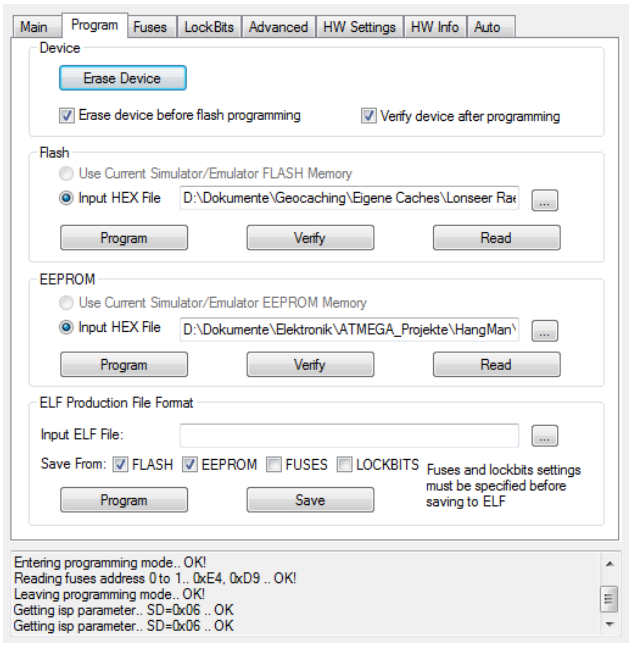


Tabelle 17: AVR-Studio - Program

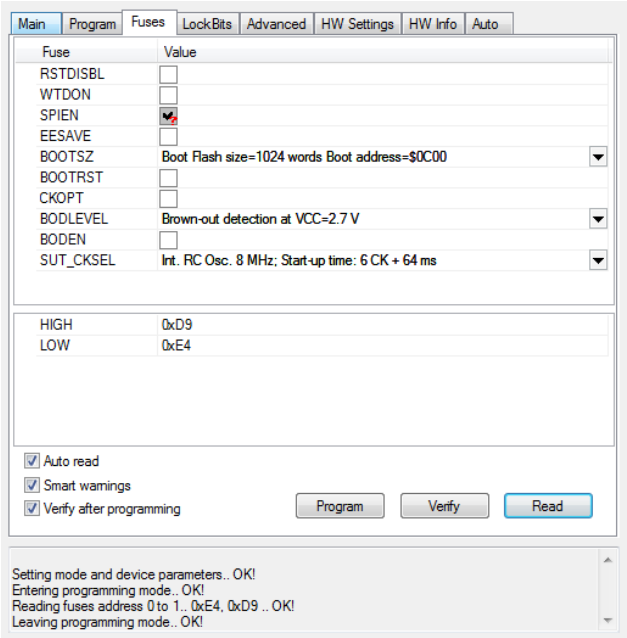


Tabelle 18: AVR-Studio - Fuses

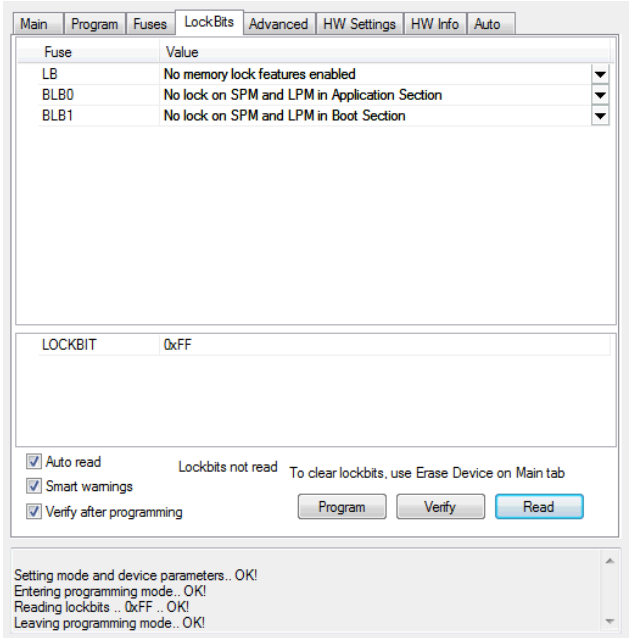


Tabelle 19: AVR-Studio - LockBits

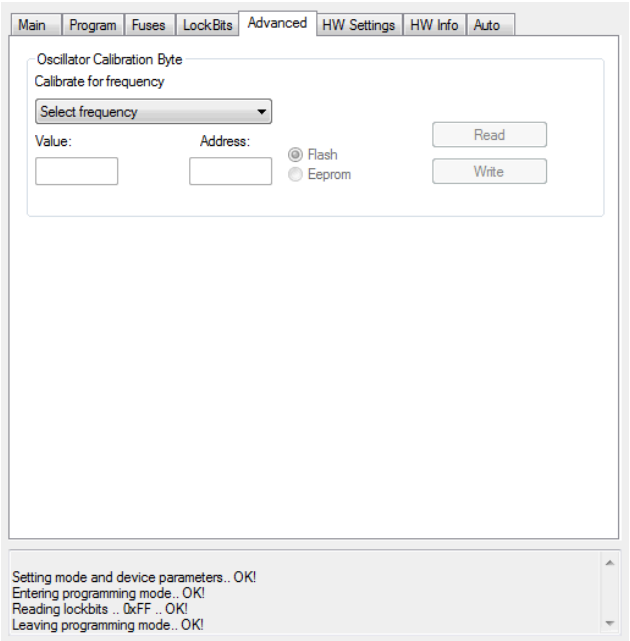


Tabelle 20: AVR-Studio - Advanced

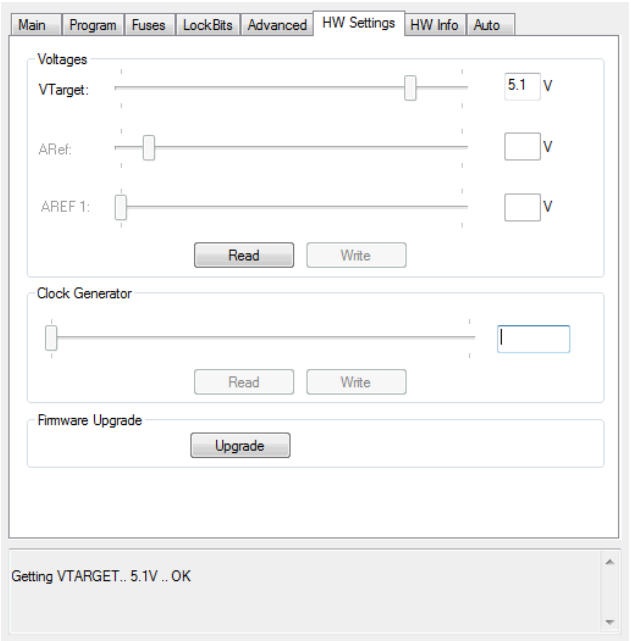


Tabelle 21: AVR-Studio – MainHW Settings

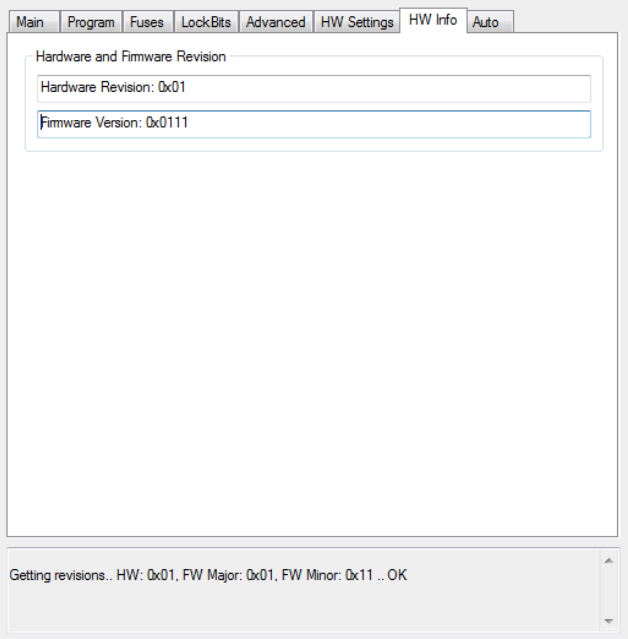


Tabelle 22: AVR-Studio – HW Info

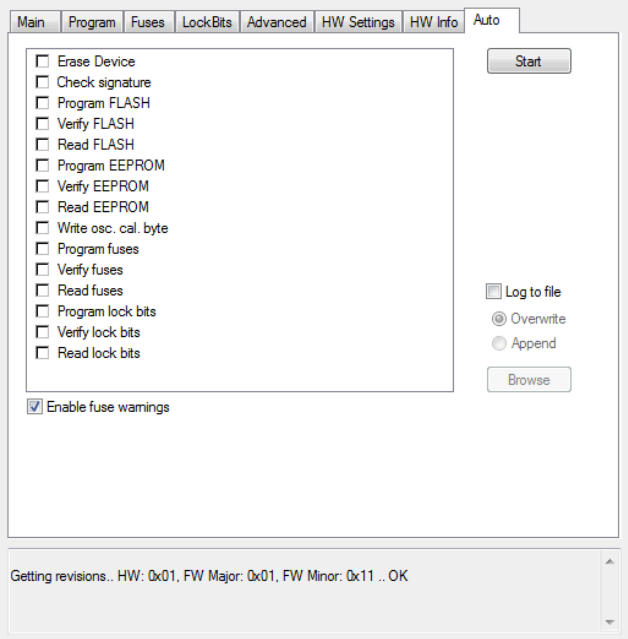


Tabelle 23: AVR-Studio - Auto

8.9 Grundlagen zur Spannung 5V, Vcc und VDD

Für die Spannungsversorgung des Projekts Senso 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.

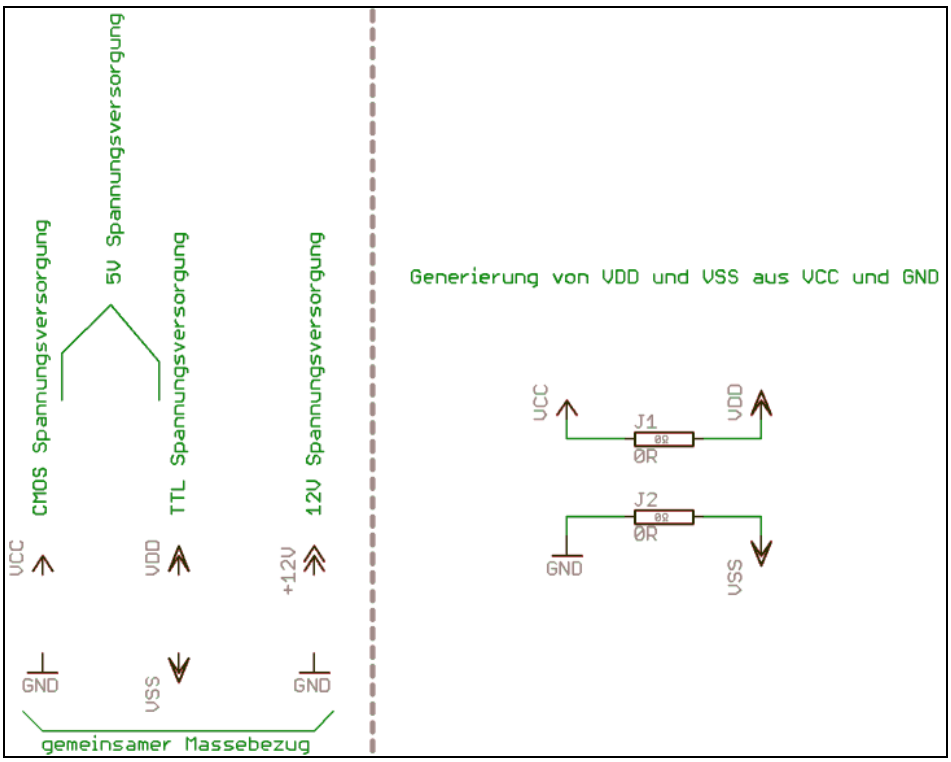
Für CMOS Bausteine gelten die Bezeichnungen VCC und GND.

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 24: Vorgeschriebene Namensgebung für Spannungsversorgungen



Schaltbild 4: Symbolfestlegung für Spannungsversorgungen

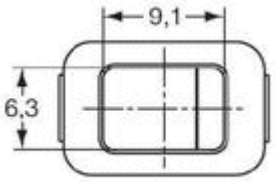
9 Elektronische Teilkomponenten

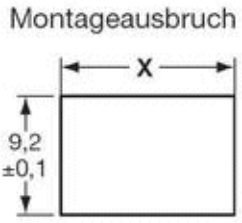
9.1 Hauptschalter

Das Spiel kann durch einen kleinen Hauptschalter ein- und ausgeschaltet werden. Folgender Hauptschalter wurde ausgewählt:

Einpoliger Subminiatur-Wippenschalter 250 V/AC 3 A

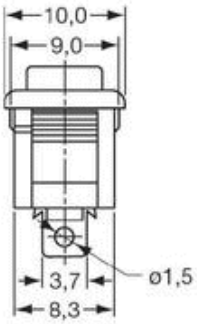
Conrad:
Bestell-Nr.: 700039 - 62
EAN: 2050000213762





X	Länge Ausschnitt	Frontplattenstärke
A	13,6 $^{+0,00}_{-0,10}$	0,75~1,25
B	13,7 $^{+0,00}_{-0,10}$	1,25~2,00
C	13,8 $^{+0,00}_{-0,10}$	2,00~2,50*

*: 2,50mm = maximale Plattenstärke



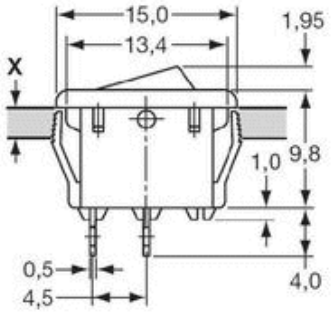


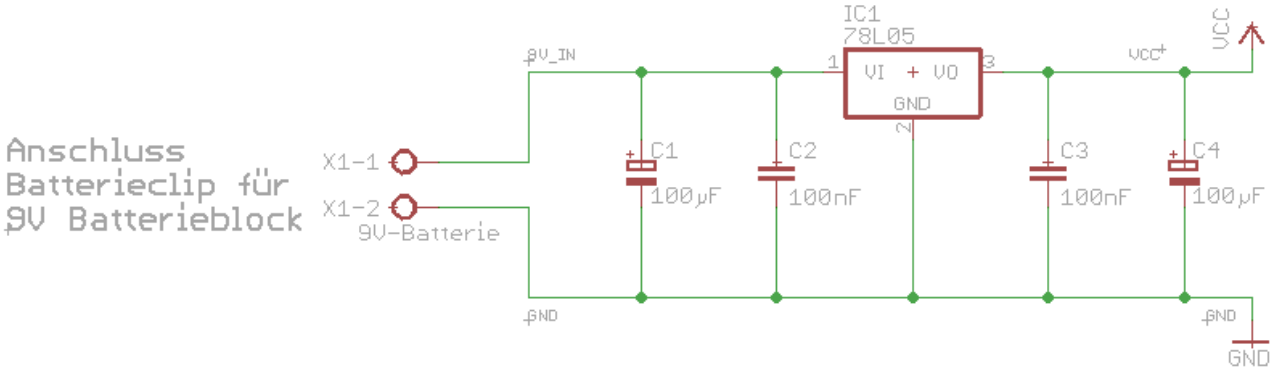
Abbildung 7: Hauptschalter

9.2 Spannungsversorgung

Die Spannungsversorgung des Spiels erfolgt mittels 9V Blockbatterie. Für die Aufnahme der Batterie im Gehäuse ist ein Batteriefach vorhanden.

Die Elektronik des Spiels arbeitet mit +5V. Diese Spannung wird mittels Festspannungsregler erzeugt.

9.2.1 Beschaltung des Festspannungsreglers:



Schaltbild 5: Beschaltung des Festspannungsreglers

Bauteile:

Stückliste: Beschaltung Spannungsversorgung			
Konensatoren		***Halbleiter***	
C2, C3	Folienkondensator 100nF	IC2	5V Festspannungsregler 78L05
C1, C4	Elektrolytkondensato 100µF		

Tabelle 25: Stückliste Beschaltung Festspannungsregler

9.3 LCD-Display

Allgemeines:

Als Anzeigedisplay wird ein LCD-Modul der Firma ELECTRONIC ASSEMBLY der DOG_Serie 3,3V EA DOG-M Super Flach / 55x27 mm inkl. Kontroller ST7036 für 4-/8-BIT SPI (4-Draht) eingesetzt.

Als Hintergrundbeleuchtung wird LED-Beleuchtung Weiß eingesetzt.

Zum Einsatz kommt das folgende Display:

LCD-Modul 1x8 – 12 mm	EA DOGM081E-A
LED Hintergrundbeleuchtung Grün	EA LED55X31-G

EA DOGM081E-A: Buchstabenhöhe 12 mm

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Abbildung 8: LCD 2x16

Für das LCD-Display wird eine weiße LED-Hintergrundbeleuchtung eingesetzt.

Abbildung 9: LCD-Display EA DOG-M

Allgemeine Technische Daten zum Display:

- Kontrastreiche LCD-Supertwist Anzeige
- Optionale LED-Beleuchtungskörper in verschiedenen Farben
- 1x8, 2x16 und 3x16 Zeichen mit 12,0 mm / 5,6 mm und 3,6 mm Schrift
- Kontroller ST 7036 für 4-BIT, 8-BIT und SPI (4-DRAHT) Interface
- Spannungsversorgung +3,3V oder +5V single supply (typ. 250µA)
- Keine zus. Spannungen erforderlich
- Betriebstemperaturbereich -20..+70°C
- LED-Hintergrundbeleuchtung 3..80mA@3,3V oder 2..40mA@5V
- Keine Montage erforderlich: einfach nur in PCB einlöten.

Abmessungen:

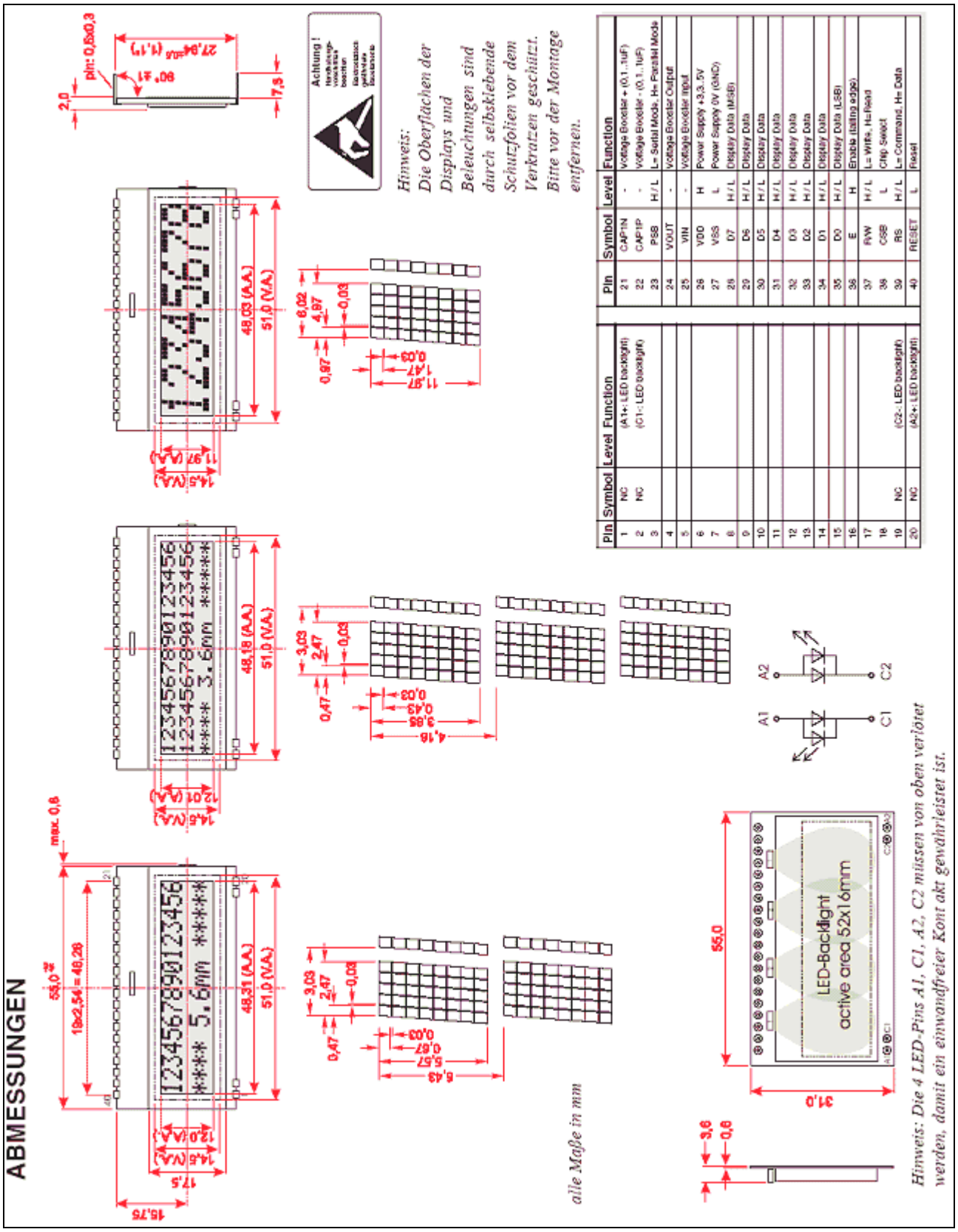


Abbildung 10: Abmessungen und technische Daten DOGM LCD-Display

Kontrasteinstellung:

Für alle Displays der EA DOG- Serie ist der Kontrast per Befehl einstellbar. Dies erfolgt über die Bits C0..C5 in den Befehlen "Contrast Set" und "Power/Icon Control/Contrast Set". In der Regel wird der Kontrast einmalig eingestellt und dann - dank integrierter Temperaturkompensation - über den gesamten Betriebstemperaturbereich (-20..+70°C) konstant gehalten.

Insgesamt benötigen die Displays selbst im 3,3V Betrieb keine zusätzliche negative Spannung!

LED-Hintergrundbeleuchtung für LCD-Display:

white EA LED55x31-W	Forward voltage	Current max.	Limiting resistor	
			@ 3,3 V	@ 5 V
Connected in parallel	3,2 V	60 mA	1,6 ohm	30 ohm
Connected in series	6,4 V	30 mA	-	-

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 5V betrieben wird. Somit kommt nur Parallelschaltung in Frage!

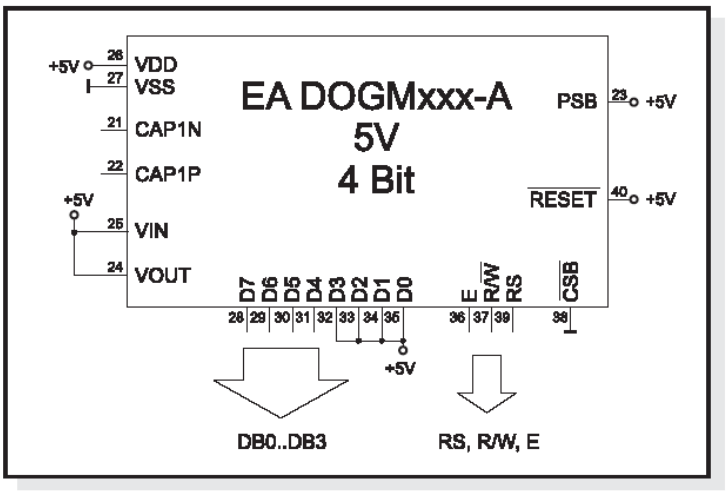
Tabelle 26: LED-Hintergrundbeleuchtung für LCD-Display

Für die Hintergrundbeleuchtung wird eine PWM über den ATmega realisiert mit Hilfe derer die Beleuchtungshelligkeit eingestellt werden kann. Siehe hierzu Beschaltung des LCD-Displays und separate Transistor-Stufe.

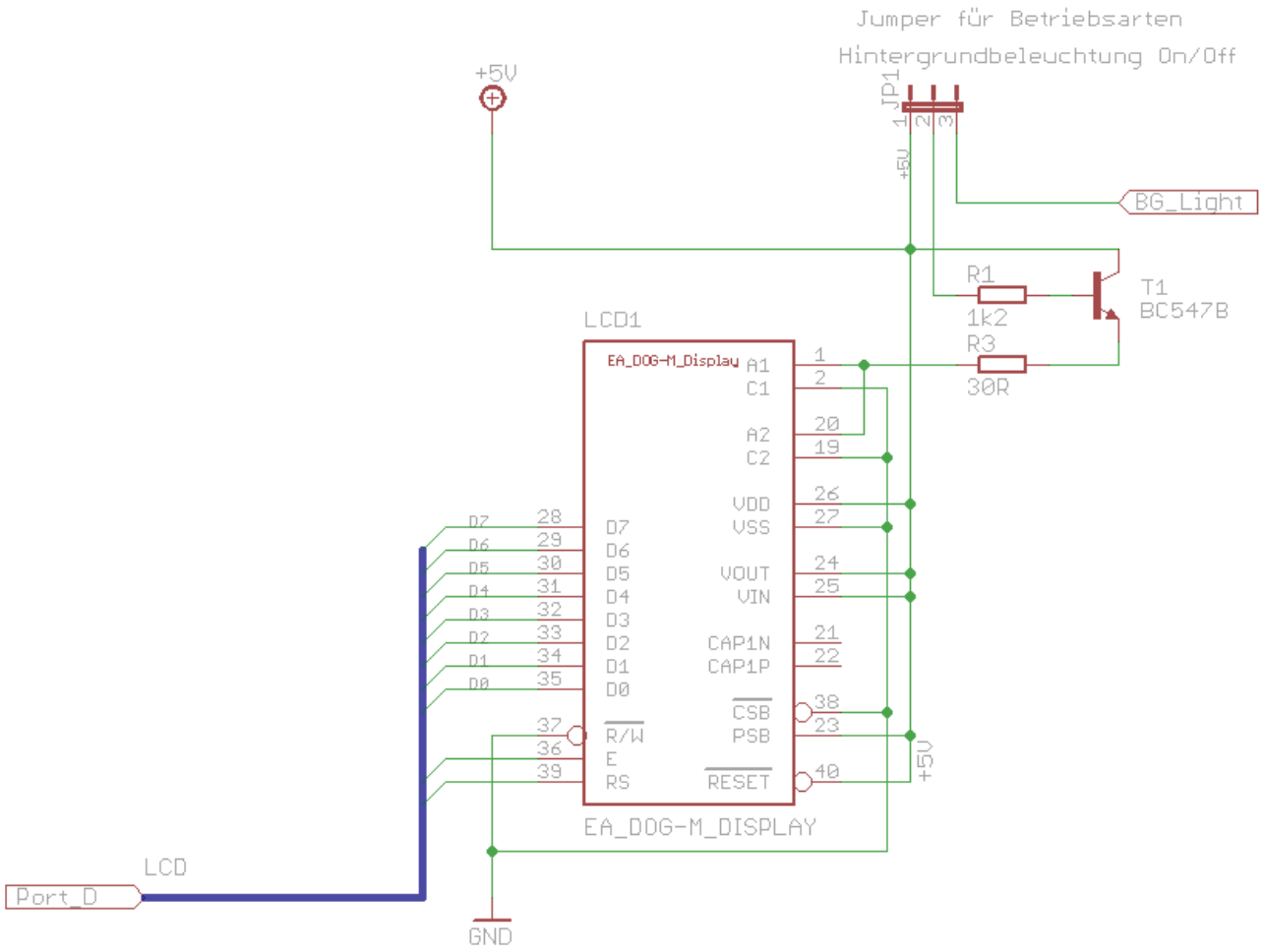
Das LCD-Display muss vor dem verlöten oder stecken auf die Beleuchtungseinheit gelötet werden, damit die LEDs der Beleuchtung über die Kontakte des LCDs Strom bekommen. Es ist ratsam, alle Pins der Beleuchtungseinheit anzulöten, da sich so der Druck beim Einsetzten des Displays in die Fassung besser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

9.3.1 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 6: Beschaltung LCD-Display gemäß Datenblatt



Schaltbild 7: Schematische Basisbeschaltung LCD-Display

Bemerkung:

Auf der Zielplatine des Projekts wird auf den Jumper verzichtet da die Helligkeitseinstelung nur über PWM realisiert wird. Hierzu dient die PWM des Timers 2 welcher sich bei OC2 den Pin mit MOSI teilt!
Auf der Zielplatine wird das LCD-Display im 4-Bit-Modus angesprochen und nicht im 8-Bit-Modus!

Bauteile:

Stückliste: LCD Beschaltung			
Widerstände		***Halbleiter***	
R1	1k2 Ω	LCD1	LCD Display EA DOGM162B-A
R3	30 Ω	T1	Transistor BC547B
Sonstiges			
JP1	Jumper 3-polig, 2,54mm Raster		

Tabelle 27: Stückliste LCD Beschaltung

Bemerkung zur LCD-Beschaltung:

- Während der Entwicklung kann über den Jumper JP1 eingestellt werden, ob die Beleuchtung mit konstanter Spannung +5V und gleichbleibender Helligkeit betrieben wird oder ob eine Ankopplung an den ATmega zur PWM erfolgt.

Ressourcenzuordnung zum ATmega8L:

Nummer	***Schaltbild***		***Ressource ATmega8L***		
1	MOSI_PWM	[PWM]	PortB.3	[OC2]	(Timer2)
2	Out_Port LCD:D4	[LCD_D0]	PortC.2	[PC2]	(GPIO)
3	Out_Port LCD:D5	[LCD_D1]	PortC.3	[PC3]	(GPIO)
4	Out_Port LCD:D6	[LCD_D2]	PortC.4	[PC4]	(GPIO)
5	Out_Port LCD:D7	[LCD_D3]	PortC.5	[PC5]	(GPIO)
6	Out_Port LCD:E	[LCD_E]	PortD.6	[PD6]	(GPIO)
7	Out_Port LCD:RS	[LCD_RS]	PortD.7	[PD7]	(GPIO)

Tabelle 28: Ressourcenzuordnung ATmega8L für LCD-Display

9.3.2 Zeichensatz und Befehlstabellen

Der unten abgebildete Zeichensatz ist integriert. Zusätzlich können 8 eigene Zeichen frei definiert werden.

Tabelle 29: Zeichensatz des LCD-Displays

Befehlstabellen des Display:

Instruction	Instruction Code										Description	Instruction Execution Time		
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		OSC= 380kHz	OSC= 540kHz	OSC= 700kHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.08 ms	0.76 ms	0.59 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.08 ms	0.76 ms	0.59 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	26.3 µs	18.5 µs	14.3 µs
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	26.3 µs	18.5 µs	14.3 µs
Function Set	0	0	0	0	1	DL	N	DH	IS2	IS1	DL: interface data is 8/4 bits N: number of line is 2/1 DH: double height font IS[2:1]: instruction table select	26.3 µs	18.5 µs	14.3 µs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	26.3 µs	18.5 µs	14.3 µs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0	0	0
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs

Tabelle 30: LCD-Display EA DOGM Instruction Code

Instruction table 0(IS[2:1]=[0,0])														
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	X	X	S/C and R/L: Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	26.3 µs	18.5 µs	14.3 µs
Set CGRAM	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	26.3 µs	18.5 µs	14.3 µs

Tabelle 31: LCD-Display EA DOGM Instruction table 0

Instruction table 1(IS[2:1]=[0,1])														
Bias Set	0	0	0	0	0	1	BS	1	0	FX	BS=1:1/4 bias BS=0:1/5 bias FX: fixed on high in 3-line application and fixed on low in other applications.	26.3 µs	18.5 µs	14.3 µs
Set ICON Address	0	0	0	1	0	0	AC3	AC2	AC1	AC0	Set ICON address in address counter.	26.3 µs	18.5 µs	14.3 µs
Power/ICON Control/ Contrast Set	0	0	0	1	0	1	Ion	Bon	C5	C4	Ion: ICON display on/off Bon: set booster circuit on/off C5,C4: Contrast set for internal follower mode.	26.3 µs	18.5 µs	14.3 µs
Follower Control	0	0	0	1	1	0	Fon	Rab 2	Rab 1	Rab 0	Fon: set follower circuit on/off Rab2~0: select follower amplified ratio.	26.3 µs	18.5 µs	14.3 µs
Contrast Set	0	0	0	1	1	1	C3	C2	C1	C0	Contrast set for internal follower mode.	26.3 µs	18.5 µs	14.3 µs

Tabelle 32: LCD-Display EA DOGM Instruction table 1

Instruction table 2(IS[2:1]=[1,0])														
Double Height Position Select	0	0	0	0	0	1	UD	X	x	x	UD: Double height position select	26.3 µs	18.5 µs	14.3 µs
Reserved	0	0	0	1	X	X	X	X	X	X	Do not use (reserved for test)	26.3 µs	18.5 µs	14.3 µs

Tabelle 33: LCD-Display EA DOGM Instruction table 2

Eine detaillierte Beschreibung des hier integrierten Kontrollers ST7036 finden Sie im Internet unter
http://www.lcd-module.de/eng/pdf/zubehoer/st7036.pdf

Initialisierungsbeispiel für EA DOGM081 bei 8-Bit und 5V:

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 5V EA DOGM162												
Befehl	**RS**	**R/W**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Hex**	**Bemerkung**
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	1	1	0	0	$1C	BS: 1/4, 2-zeiliges LCD
Power Control	0	0	0	1	0	1	0	0	1	0	$52	Booster aus, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	0	0	1	$69	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	0	1	0	0	$74	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Tabelle 34: LCD-Display EA DOGM081 Initialisierungsbeispiel

Initialisierungsbeispiel für EA DOGM081 bei 8-Bit und 5V:

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 5V EA DOGM163												
Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	1	1	0	1	$1D	BS: 1/4, 3-zeiliges LCD
Power Control	0	0	0	1	0	1	0	0	0	0	$50	Booster aus, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	1	0	0	$6C	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	1	1	0	0	$7C	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Tabelle 35: LCD-Display EA DOGM081 Initialisierungsbeispiel

9.3.3 PWM-Einstellungen für Hintergrundbeleuchtung

Die Hintergrundbeleuchtung des LCD Display wird mittels vom ATmega erzeugten PWM eingestellt.

Der Helligkeitswert des Display kann aber zur Spielzeit nicht geändert werden sondern er ist empirisch ermittelt und softwareseitig festgelegt.

Der Quarz für den Prozessor arbeitet mit 8 MHz. Für die Prescaler-Einstellung wird der Wert 128 festgelegt. Damit ergibt sich für die PWM-Frequenz folgender Wert:

Quarz = 8 MHz ; Prescaler = 128 ; Timer = 8 Bit ergibt: (8000000Hz / 128) / (256*2) = **122,07 Hz**

9.3.4 BASCOM Beispielcode für die Displayansteuerung (Funktionsbibliothek)

```
'#################################################################################
' LCD-Display.BAS                                                Stand 03.05.2014
' ---------------                                                (C) Markus Fulde
'
' Testprogramm zur Inbetriebnahme des Display DOGM163
' Die Inbetriebnahme erfolgt mit dem Eva-Board STK500 + STK501
' Die für die LCD-Display-Ansteuerung notwendigen Routinen sind
' als solche gekennzeichnet und bereits zur späteren Verwendung
' ausgelagert.
'#################################################################################

'--------------------------------------------------------------------------------
' Compilerinstruktionen und Compilerdirektiven
'--------------------------------------------------------------------------------
$regfile = "m8def.dat"                                    ' Definitionsdatei für ATmega8 laden
$crystal = 8000000                                        ' Quarzfrequenz für 16 MHz festlegen

$baud = 19200                                             ' Baudrate für RS232 Traceausgabe defi-
nieren


'--------------------------------------------------------------------------------
' Allgemeine Zusatzinformatonen zu Programmbeginn
'--------------------------------------------------------------------------------


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

' ----- LED's -----
Alive_pin Alias PinB.0                                    ' GPIO für Alive-LED (für DDR oder In-
put)
```

```
Alive Alias PortB.0                                    ' GPIO für Alive-LED (für Output oder
Pullup)

Pwrled_pin Alias PINB.1                                ' GPIO für Power-LED (für DDR oder In-
put)
Pwrled Alias PORTB.1                                   ' GPIO für Power-LED (für Output oder
Pullup)

' ----- LCD-Display -----
' LCD-Display
Db4_pin Alias PortC.2                                  ' GPIO für LCD Pin4
Db5_pin Alias PortC.3                                  ' GPIO für LCD Pin5
Db6_pin Alias PortC.4                                  ' GPIO für LCD Pin6
Db7_pin Alias PortC.5                                  ' GPIO für LCD Pin7
E_pin Alias PortD.6                                    ' GPIO für LCD E
Rs_pin Alias PortD.7                                   ' GPIO für LCD RS

' ----- Test: Tasten zu Testzwecken -----
Key1_pin Alias PIND.2                                  ' GPIO für Key 1 (für DDR oder Input)
Key1 Alias PORTD.2                                     ' GPIO für Key 1 (für Output oder
Pullup)

Key2_pin Alias PIND.3                                  ' GPIO für Key 2 (für DDR oder Input)
Key2 Alias PORTD.3                                     ' GPIO für Key 2 (für Output oder
Pullup)


'-------------------------------------------------------------------------------
' Definition von Konstaten
'-------------------------------------------------------------------------------

' ----- Für Testumgebung bzw. Traceausgaben -----
Const Main_testmodus = 1                               ' Flag für Testmodus Allgemeinsystem
Const Lcd_testmodus = 1                                ' Flag


' ----- Allgemeine Systemkonstanten -----

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

Const Led_aus = 1 ' Achtung !! bei STK500 ist Logik gedreht!!
Const Led_ein = 0 ' Achtung !! bei STK500 ist Logik gedreht!!

Const False = 0
Const True = 1

Const Pullup_aus = 0
Const Pullup_ein = 1

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


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

' Anmerkung: Wert von 35 ist der beste Wert welcher durch Versuche ermittelt wurde!!
Const Lcd_kontrast_default = 35                        ' LCD-Default-Kontrast
' Anmerkung: Wert von 70 ist der beste Wert welcher durch Versuche ermittelt wurde!!
Const Lcd_helligkeit_default = 70                      ' LCD-Default-Helligkeit
Const Lcd_helligkeit_aus = 0                           ' Wert für Display aus


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

' ----- Arbeitsvariablen für Spieleablauf -----
Dim Game_hangman_status As Byte                        ' Arbeitsvariable für Hangman-Status

' ----- Variablen für LCD-Display -----
Dim Lcd_kontrastwert As Byte                           ' Arbeitsvariable für Kontrastwert
Dim Lcd_helligkeit As Byte                             ' Arbeitsvariable für Displayhelligkeit
```

```
'-----------------------------------------------------------------------
' Prototyping
'-----------------------------------------------------------------------

' ----- LCD und Print -----
Declare Sub Lcd_print_hangingman(byval Value As Byte)       ' Funktion zum schrittweisen Aufbau des
Hanging Man


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

' Konfiguration Timer 2 für Hardware-PWM an OC2 (D.7)
Config Timer2 = Pwm , Prescale = 128 , Compare Pwm = Clear Up
Enable TIMER2

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

' Konfiguration LCD Display
Config Lcdpin = Pin , Db4 = Db4_pin , Db5 = Db5_pin , Db6 = Db6_pin , Db7 = Db7_pin , E = E_pin , Rs
= Rs_pin
Config Lcd = 16 * 2 , Chipset = Dogm162v5                   ' DOG-M Treiber laden
Config Lcdbus = 4                                           ' LCD arbeitet über 4-Bit
Initlcd                                                     ' LCD initialisieren
Waitms 100                                                  ' 100ms nach Init warten
Cursor Off Noblink                                          ' Blinkenden Cursor abschalten

' Definition benutzerdefinierter Zeichen
' LCD-Zeichen linker Sockel des Galgens
Deflcdchar 0 , 16 , 16 , 16 , 16 , 16 , 16 , 30 , 30
' LCD-Zeichen linke obere Ecke des Galgens
Deflcdchar 1 , 15 , 9 , 10 , 12 , 8 , 8 , 8 , 8
' LCD-Zeichen rechter Galgen ohne Männchen
Deflcdchar 2 , 28 , 4 , 32 , 32 , 32 , 32 , 32 , 32
' LCD-Zeichen rechter Galgen mit Kopf
Deflcdchar 3 , 28 , 4 , 4 , 14 , 17 , 17 , 14 , 4
' LCD-Zeichen Bauch
Deflcdchar 4 , 4 , 4 , 4 , 4 , 32 , 32 , 32 , 32
' LCD-Zeichen Bauch mit linkem Beine
Deflcdchar 5 , 4 , 4 , 4 , 4 , 8 , 16 , 16 , 32
' LCD-Zeichen Bauch mit beiden Beinen
Deflcdchar 6 , 4 , 4 , 4 , 4 , 10 , 17 , 17 , 32
' LCD-Zeichen kompletter Männchen-Körper
Deflcdchar 7 , 14 , 21 , 21 , 4 , 10 , 17 , 17 , 32

Cls                                                         ' Clear Screen


' ------------------------------ Port's und Pin's ----------------------------

' ----- LED-Konfigurationen -----
Config Alive_pin = Output
Config Pwrled_pin = Output

' ----- Test: Tasten-Konfiguration ------
Config Key1_pin = Input
Config Key2_pin = Input

' ------------------------------ Variablen und Werte -------------------------

' ----- LED-Konfigurationen -----
Alive = Led_aus                                             ' Alive-LED aus
Pwrled = Led_aus                                            ' LED für Spannungsüberwachung

' ----- LCD-Display -----
Lcd_kontrastwert = Lcd_kontrast_default                     ' Kontrastwert
Lcd_helligkeit = Lcd_helligkeit_default                     ' Displayhelligkleit

' ----- Spielesteuerung -----
```

```
Game_hangman_status = 0


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

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

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

Gosub Lcd_kontrast_set                                  ' LCD Kontrast einstellen
Gosub Lcd_helligkeit_set                                ' Displayhelligkeit einstellen


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

' Kleiner LCD-Test zu Beginn
#if Lcd_testmodus
   ' Mit kleiner Schleife gesamtes Display mit * füllen
   For Temp_byte_1=1 to 2 Step 1
      For Temp_byte_2=1 to 16 Step 1
         Locate Temp_byte_1, Temp_byte_2
         Lcd "*"
         waitms 50
      Next Temp_byte_2
   Next Temp_byte_1

   ' Mit kleiner Schleife gesamtes Display leeren
   For Temp_byte_1=2 to 1 Step -1
      For Temp_byte_2=16 to 1 Step -1
         Locate Temp_byte_1, Temp_byte_2
         Lcd " "
         waitms 50
      Next Temp_byte_2
   Next Temp_byte_1

   ' Noch auf beide Zeilen die Positionszahlen ausgeben und gut
   Locate 1 , 1 : Lcd "1234567890123456"
   Locate 2 , 1 : Lcd "1234567890123456"

#endif


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

' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Main_testmodus
   Print "** OK, let's GO **"
#endif

Do                                                      ' Hauptschleife

   ' Test für Kontrasteinstellung
   ' Debounce Key1_pin , 0 , Lcd_inccontrast , Sub            ' Erhöhe den Kontrast
   ' Debounce Key2_pin , 0 , Lcd_deccontrast , Sub            ' Verringere den Kontrast

   ' Test für Helligkeitseinstellung
   ' Debounce Key1_pin , 0 , Lcd_inchelligkeit , Sub          ' Erhöhe den Helligkeit
   ' Debounce Key2_pin , 0 , Lcd_dechelligkeit , Sub          ' Verringere den Helligkeit

   ' Test für Ausschalten / Einschalten Hintergrundbeleuchtung
   ' Debounce Key1_pin , 0 , Lcd_beleuchtung_ein , Sub        ' Beleuchtung ein
   ' Debounce Key2_pin , 0 , Lcd_beleuchtung_aus , Sub        ' Beleuchtung aus

   ' Test für Galgenmännchen
   Debounce Key1_pin , 0 , Lcd_print_hangingman_all , Sub   ' Galgenmännchen zeichnen

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 *
' ***************

'---------------------------------------------------------------------------------
' LCD - Subroutine: Lcd_print_hangingman
'
' Subroutine schreibt je nach uebergebenem Status den Zustand des Hanging Man
' auf das Display
'    Status 0: Galgen ohne Maennchen
'    Status 1: Galgen mit Kopf
'    Status 2: Galgen mit Kopf und Rumpf
'    Status 3: Galgen mit Kopf, Rumpf und linken Bein
'    Status 4: Galgen mit Kopf, Rumpf, linken und rechten Bein
'    Status 5: Galgen mit kompletten Körper
'
' Parameter:    Value = Step n des Hanging-Man
' Rückgabewert: keine
'
' Globale Variable:
' --
'---------------------------------------------------------------------------------
Sub Lcd_print_hangingman(byval Value As Byte)

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

   ' Galgen zeichnen
   Locate 1 , 1 : Lcd Chr(1)
   Locate 2 , 1 : Lcd Chr(0)

   Select Case Value
      Case 0 :
         Locate 1 , 2 : Lcd Chr(2)
      Case 1 :
         Locate 1 , 2 : Lcd Chr(3)
      Case 2 :
         Locate 1 , 2 : Lcd Chr(3)
         Locate 2 , 2 : Lcd Chr(4)
      Case 3 :
         Locate 1 , 2 : Lcd Chr(3)
         Locate 2 , 2 : Lcd Chr(5)
      Case 4 :
         Locate 1 , 2 : Lcd Chr(3)
         Locate 2 , 2 : Lcd Chr(6)
      Case 5 :
         Locate 1 , 2 : Lcd Chr(3)
         Locate 2 , 2 : Lcd Chr(7)
   End Select

End Sub
'-- End Lcd_Print_hanginman ----------------------------------------------------


'*********************************************************************************
```

```
' GOSubroutinen
'*****************************************************************************

' ***************
' * LCD-Dsiplay *
' ***************

'-----------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_contrast_set
' Routine berechnen neue Kontrastwerte und steuert direkt den
' Kontroller des Display an.
' Aufgrund von 6 Bit sind nur Kontrastwerte zwischen 0 und 63 möglich!
'-----------------------------------------------------------------------------
Lcd_kontrast_set:                                       ' Kontrasteinstellung Display

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

   ' Verarbeitung des Kontrastwertes für High-Byte und Low-Byte
   Temp_byte_1 = Lcd_kontrastwert And &B00001111
   Temp_byte_1 = Temp_byte_1 + &B01110000

   Temp_byte_2 = Lcd_kontrastwert
   Shift Temp_byte_2 , Right , 4
   Temp_byte_2 = Temp_byte_2 And &B00000011
   Temp_byte_2 = Temp_byte_2 + &B01010100

   ' Instruction Table 1 einstellen [0,1]
   _temp1 = &B00101001
   !rCall _Lcd_control

   ' Tempvar_1 = &B0111xxxx für Kontrast Set Instruction Table 1 - Low Byte
   _temp1 = Temp_byte_1
   !rCall _Lcd_control

   ' Temovar_2 = &B010101xx für Kontrast Set Instruction Table 1 - High Byte
   _temp1 = Temp_byte_2
   !rCall _Lcd_control

   ' Zurückschalten auf Instruction Table 0 [0,0]
   _temp1 = &B00101000
   !rCall _Lcd_control

Return
'-- End Lcd_kontrast_set -------------------------------------------------------


'-----------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_IncContrast
' Routine erhöht bei Einsprung den Kontrastwert um 1
'-----------------------------------------------------------------------------
Lcd_inccontrast:

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

   ' Aufgrund von 6 Bit sind nur Kontrastwerte zwischen 0 und 63 möglich!
   If Lcd_kontrastwert < 63 Then
      Incr Lcd_kontrastwert
   end if
   Gosub Lcd_kontrast_set
   Print "Kontrast: " ; Lcd_kontrastwert

   Return
'-- End Lcd_IncContrast ----------------------------------------------------


'-----------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_DecContrast
' Routine verringert bei Einsprung den Kontrastwert um 1
'-----------------------------------------------------------------------------
Lcd_deccontrast:

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

   If Lcd_kontrastwert > 1 Then
      Decr Lcd_kontrastwert
   End If
   Gosub Lcd_kontrast_set
   Print "Kontrast: " ; Lcd_kontrastwert

Return
```

```
'-- End Lcd_DecContrast ------------------------------------------------------


'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_helligkeit_set
'
' Routine setzt den Helligkeitswert aus der globalen Variable Lcd_helligkeit in
' das Controllregister
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Lcd_helligkeit = Helligkeitswert für PWM
'--------------------------------------------------------------------------------
Lcd_helligkeit_set:

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

   ' Variable in Timer-Count-Register laden
   Ocr2 = Lcd_helligkeit

Return
'-- End Ir_set_helligkeit -----------------------------------------------------


'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_IncHelligkeit
' Routine erhöht bei Einsprung die Helligkeit um 1
'--------------------------------------------------------------------------------
Lcd_inchelligkeit:

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

   If Lcd_helligkeit < 250 Then
      Lcd_helligkeit = Lcd_helligkeit + 10
   end if
   Gosub Lcd_helligkeit_set
   #if Lcd_testmodus
      Print "Helligkeit: " ; Lcd_helligkeit
   #endif

Return
'-- End Lcd_IncHelligkeit -------------------------------------------------------


'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_DecHelligkeit
' Routine verringert bei Einsprung die Helligkeit um 1
'--------------------------------------------------------------------------------
Lcd_dechelligkeit:

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

   If Lcd_helligkeit > 10 Then
      Lcd_helligkeit = Lcd_helligkeit - 10
   End If
   Gosub Lcd_helligkeit_set
   #if Lcd_testmodus
      Print "Helligkeit: " ; Lcd_helligkeit
   #endif

Return
'-- End Lcd_Dechelligkeit --------------------------------------------------------


'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_beleuchtung_aus
'
' Routine schaltet die LCD-Hintergrundbeleuchtung aus
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'--------------------------------------------------------------------------------
Lcd_beleuchtung_aus:

   ' ----- Programmcode -----
```

```
   Lcd_helligkeit = Lcd_helligkeit_aus
   Gosub Lcd_helligkeit_set

   #if Lcd_testmodus
      Print "Beleuchtung aus"
   #endif

Return
'-- End Lcd_beleuchtung_aus ---------------------------------------------------


'-------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_beleuchtung_ein
'
' Routine schaltet die LCD-Hintergrundbeleuchtung ein
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'-------------------------------------------------------------------------------
Lcd_beleuchtung_ein:

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

   Lcd_helligkeit = Lcd_helligkeit_default
   Gosub Lcd_helligkeit_set

   #if Lcd_testmodus
      Print "Beleuchtung ein"
   #endif

Return
'-- End Lcd_beleuchtung_ein ----------------------------------------------------


'-------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_Print_hangingman
'
' Routine zeichnet Galgenmaennchen auf dem Display
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'-------------------------------------------------------------------------------
Lcd_print_hangingman_all:

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

   Cls                                                  ' Bildschirm löschen

   Call Lcd_print_hangingman(game_hangman_status )      ' Maennchen gemaess Status zeichnen
   Incr Game_hangman_status                             ' Statuscounter erhoehen

   ' Pruefen ob Statuscounter noch valide und ggf. zuruecksetzen
   If Game_hangman_status > 5 Then
      Game_hangman_status = 0
   End If

Return
'-- End Lcd_beleuchtung_ein ----------------------------------------------------



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

' System halt
End                                                  'end program

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

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


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


'################################## Historie #################################
' 03.05.2014 : Version x.x
'              Beginn der Implementierungen für Hanging Man
' 03.05.2014 : Versions 1.0
'              Fertigstellung Inbetriebnahme LCD-Display und wichtigster
'              Funktionen
'#############################################################################
```

Software 1: Code zur Ansteuerung des LCD-Displays

Anmerkung:

Der oben gezeige / abgebildete Beispielcode ist allgemein zu verstehen und bezieht sich auf die Verwendung von 2-zeiligen bzw. auch 3-zeiligen Display der DOG-M Serie. Im Projekt Senso kommt nur ein einzeiliges Display mit 8 Zeichen zumk Einsatz. Dies ändert aber an der prinzipiellen Ansteuerung des Dsiplay nichts. Lediglich die außerhalb des Bereichs liegenden zeichen werden nicht angezeigt. Daher kann der obige Code denoch als Referenzimplementierung verstanden werden.

9.3.5 Prototyp LCD-Display-Ansteuerung

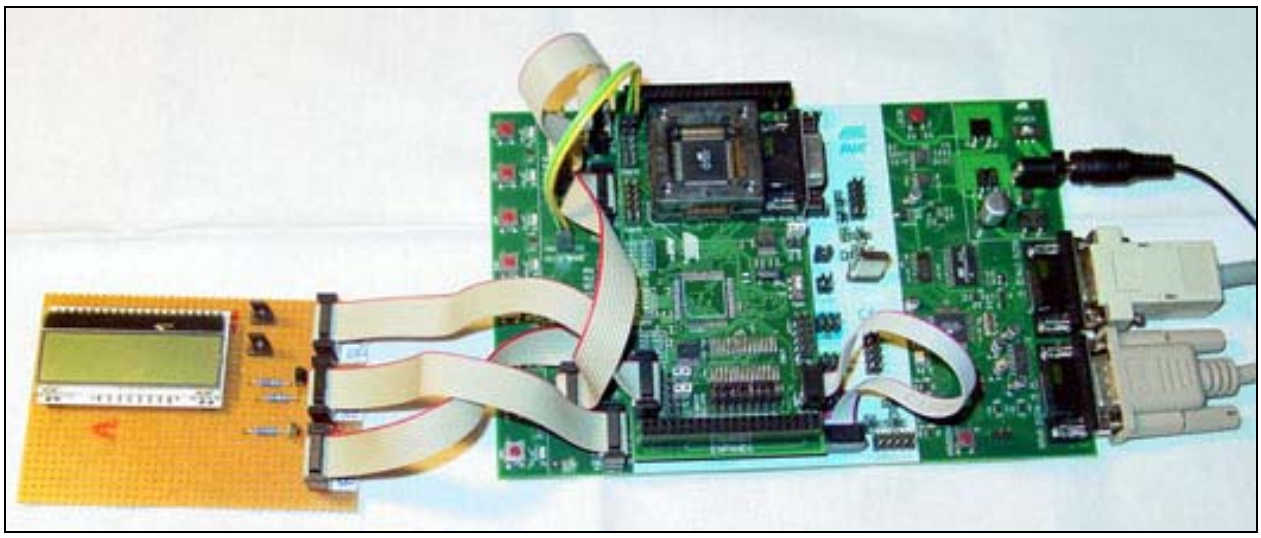


Abbildung 11: Prototyp LCD-Display-Ansteuerung mit STK

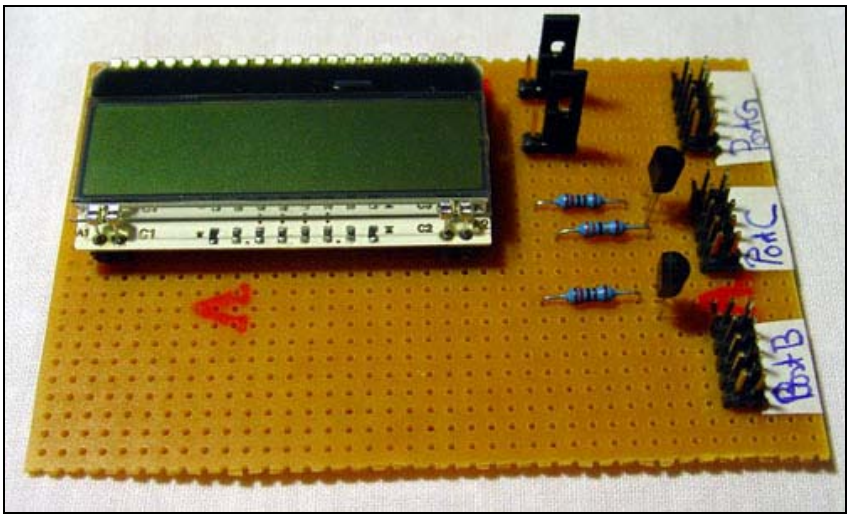


Abbildung 12: Prototyp LCD-Display-Ansteuerung PCB

9.4 Tasteneingabe

Allgemeines:

Um das Spiel Senso spielen zu können werden 4 Miniatur-Drucktaster in den Farben Blau, Rot, Grün und Gelb verwendet.

Zum Einsatz kommt der folgende Typ:

Miniatur-Drucktaster, Ein 0,5A-250VAC, blau

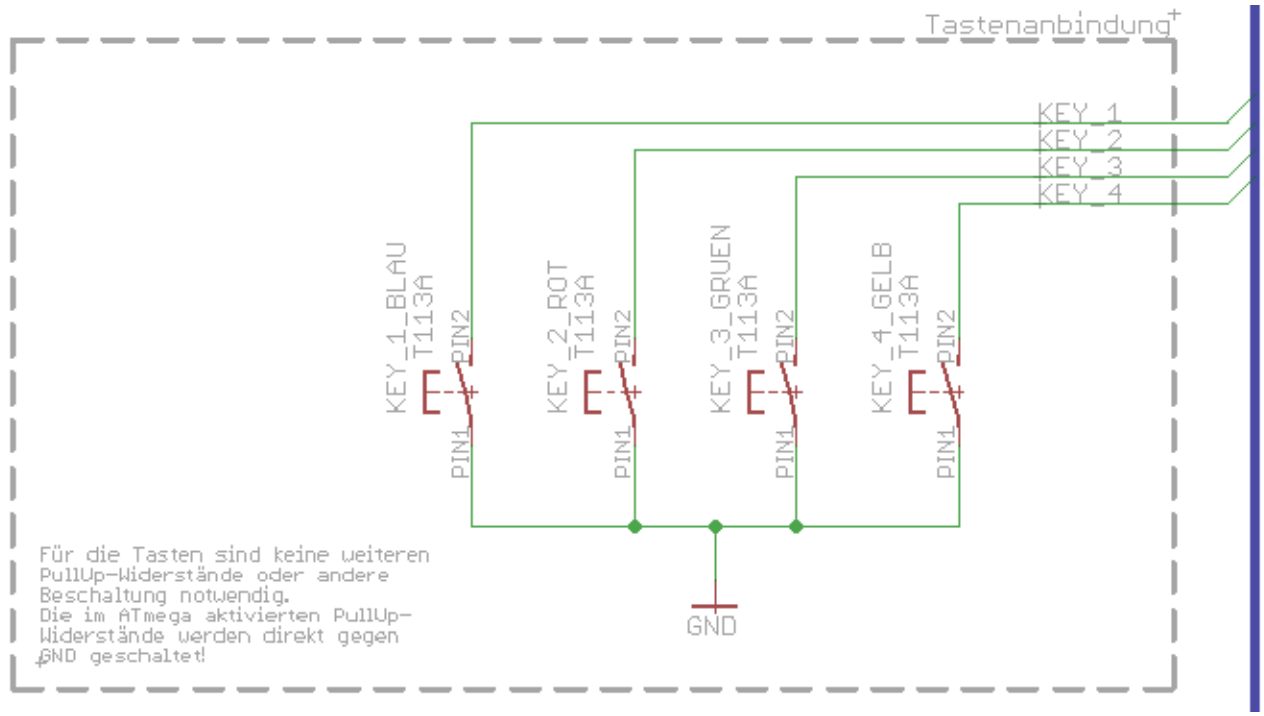
Typ:	Drucktaster
Ausführung:	Arbeitskontakt / Schließer
Aufbau:	verstärkte Ausführung
Anschluss:	Lötanschluss
Belastungsgrenze:	250 V / 0,5 A
Farbe:	blau, rot, gruen, gelb
Befestigung:	M7x0,75
Bohrdurchmesser:	7,0 mm
Schaltzyklen:	50.000
Länge:	21,1 mm
Ø:	9,5 mm
Temperatur, max.:	-25 … +85 °C
Verpackungsgewicht:	0,004 kg
Reichelt Bestellnummer:	*T 113A BL, T 113A RT, T 113A GN, T 113A GE*

Abbildung 13: Miniatur-Drucktaster

Für das Einlesen der Taster werden die ATmega internen PullUp-Widerstände aktiviert. So entsteht kein Aufwand für eine weitere pheriphere Beschaltung.

9.4.1 Beschaltung der Taster

Die Beschaltung des Taster erfolgt wie im folgenden Schaltbild dargestellt:



Schaltbild 8: Beschaltung Taster im Projekt Senso

Bauteile:

Stückliste: Beschaltung Taster		
Sonstiges		
KEY_1_BLAU	Miniatur-Drucktaster Blau	
KEY_2_ROT	Miniatur-Drucktaster Rot	
KEY_3_GRUEN	Miniatur-Drucktaster Grün	
KEY_4_GELB	Miniatur-Drucktaster Gelb	

Tabelle 36: Stückliste Beschaltung Taster

Ressourcenzuordnung zum ATmega8L:

Nummer	***Schaltbild***	***Ressource ATmega8L***
1	KEY_1_BLAU	PortD.3 [PD3] (GPIO)
2	KEY_2_ROT	PortD.2 [PD2] (GPIO)
3	KEY_3_GRUEN	PortD.5 [PD5] (GPIO)
4	KEY_4_GELB	PortD.7 [PD7] (GPIO)

Tabelle 37: Ressourcenzuordnung ATmega8L für Drehimpulsgeber

9.5 Sound

Die Soundausgabe im Projekt Senso erfolgt mittels Kleinlautsprecher.

Zum Einsatz kommt der Kleinlautsprecher km EKULIT LSM-S36K.
Reichelt Bestellnummer: *LSM-S36K*
bzw. bei einem 2. Muster des Projekts der sich dann auch im Feld befinden wir der Lautsprecher Visaton K 50 Fl
Reichelt Bestellnummer: *VIS K50FL-16*

Der Wechsel auf einen anderen Lautsprecher wurde während der Inbetriebnahme im gehäuse notwendig. Es zeigte sich, dass bei geschlossenem Gehäuse die Außenlautstärke unzureichend erscheint. Daher wurde auf den K 50 FL gewechselt. Hierzu wurde die Gehäuse-Rückseite mehrfach gebohrt und der Lautsprecher mit Kleber abschließend verklebt. Die Membrane des Lautsprechers sind aus Kunststoff und dichten daher ab.

Der Lautsprecher LSM-S36K ist für die Printmontage geeignet.

Typ:	Kleinlautsprecher
Impedanz:	16 Ohm
Resonanzfrequenz:	450 Hz
Frequenzbereich:fo	5000 Hz
Leistung:	0,25 W
Musikbelastbarkeit:	0,5 W
Geräuschpegel:	83 dB
Temperaturbereich:	-40 … +95 °C
Anschluss:	Printanschuss
Breite:	48 mm
Höhe:	40 mm
Tiefe:	4,8 mm
Ausführung:	Kunststoff
Verpackungsgewicht:	0,009 kg

Bestellnummer: *LSM-S36K*

5 cm (2") Kleinlautsprecher mit Kunststoffmembran.
Signalausgabe in Geräten und Anlagen aller Art, in denen wenig Platz zur Verfügung steht.
Gut geeignet für Anwendungen im Außenbereich und bei kritischen Umgebungseinflüssen.

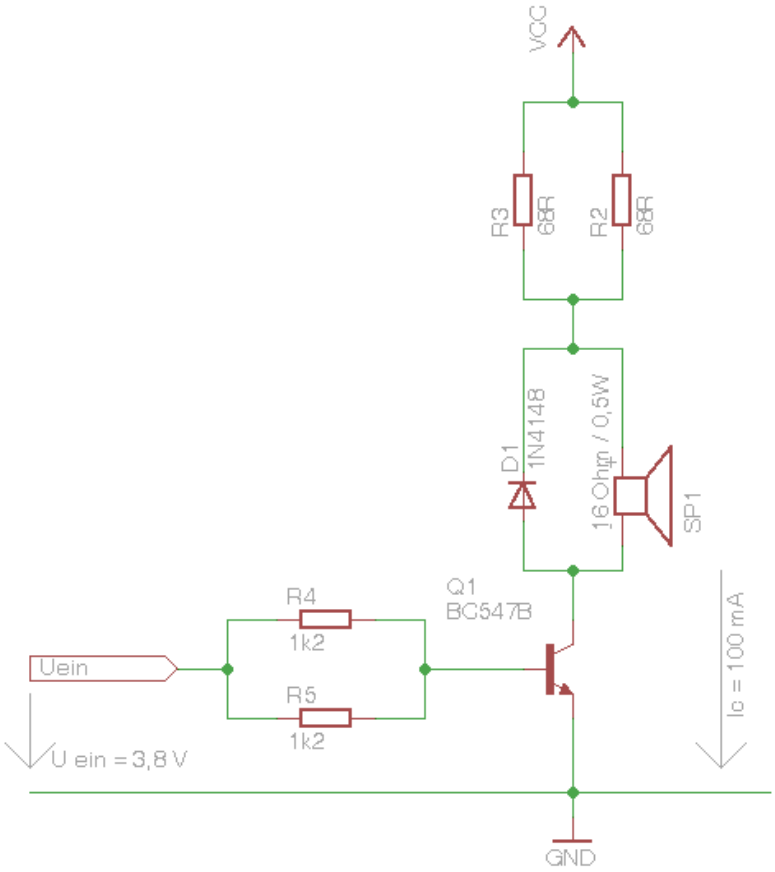
Nenn-/Musikbelastbarkeit:	1 W / 2 W
Impedanz:	16 Ohm
Übertragungsbereich	(– 10 dB): 150–20000 Hz
Mittlerer Schalldruckpegel:	80 dB (1 W/1 m)
Resonanzfrequenz:	340 Hz
Schwingspulendurchmesser:	13 mm Ø
Wickelhöhe:	2 mm
Schutzklasse:	IP 65
Temperaturbereich:	-40 ... 70 °C
Schallwandöffnung:	46 mm Ø
Gewicht netto:	0,014 kg

Bestellnummer: *VIS K50FL-16*

Tabelle 38: Lautsprecher im Projekt Senso

Der Lautsprecher wird über eine Transistorstufe vom Controller aus angesteuert.

Beschaltung:



Schaltbild 9: Lautsprecher

Bauteile:

Stückliste: Soundgeber			
Sonstiges		***Halbleiter***	
SP1	Lautsprecher LSM-S36K	D1	Diode 1N4148
Widerstand		Q1	Transistor BC547B
R2	Metallschichtwiderstand 68Ω		
R3	Metallschichtwiderstand 68Ω		
R4	Metallschichtwiderstand 1,2 kΩ		
R5	Metallschichtwiderstand 1,2 kΩ		

Tabelle 39: Bauteile für Soundgeber

Ressourcenzuordnung zum ATmega8L:

Nummer	***Schaltbild***	***Ressource ATmega8***
1	SOUND SOUND	PortB.1 [PB1] (GPIO Output)

Tabelle 40: Ressourcenzuordnung für 5V Soundgeber

9.5.1 Tonerzeugung mit dem Sound-Befehl von BASCOM-AVR

Für die Ausgabe einfacher Töne stellt der Bascom AVR Compiler den Befehl „Sound“ zu Verfügung. Dieser Befehl ist allerdings nicht sehr genau. Sollen Töne mit genauer Frequenz ausgegeben werden so müssen andere Methoden angewendet werden, auf die ich hier aber nicht eingehen werde. Die folgenden Informationen zum Sound-Befehl finden sich in der Bascom Hilfe

Action

Sends pulses to a port pin.

Syntax

SOUND pin, duration, pulses

Remarks

Pin	Any I/O pin such as PORTB.0 etc.
Duration	The number of pulses to send. Byte, integer/word or constant.
Pulses	The time the pin is pulled low and high. This is the value for a loop counter.

Hilfe 1: Bascom Hilfe zum Sound-Befehl

Weiter Informationen sind nicht verfügbar, also musste ich weiter recherchieren. Die Frage die sich dabei stellte, wie kann eine definierte Frequenz für eine Note ausgegeben werden ? Diese Frage lies sich nach einigem suchen beantworten, sie kann mit den folgenden Formeln berechnet werden. (Schön wäre es wenn die Bascom Hilfe dazu einen Hinweis geben würde.)

$$\text{pulses} = \int \left(\frac{f_{\text{Quarz}}}{(12 \cdot f_{\text{Note}})} \right) \quad \begin{matrix} f_{\text{Quarz}} - \text{Taktfrequenz des Mikrokontroller} \\ f_{\text{Note}} - \text{Frequenz der auszugebenden Note} \end{matrix}$$

$$\text{duration} = \int \left(\frac{T \cdot f_{\text{Quarz}}}{12 \cdot \text{pulses}} \right) \quad \begin{matrix} T - \text{Ton Dauer in Sekunden} \\ 12 - \text{ist die Anzahl Takte für eine Periode} \end{matrix}$$

Mit diesen Informationen allein können wir jedoch noch nichts anfangen es fehlt noch ein entscheidender Teil nämlich die Frequenz der zu verwendenden Note. Dazu werfen wir eine Blick auf die Informationen zur „chromatische Tonleiter“ (i). Bei einer gleichtemperierten Stimmung wird jede Oktave in zwölf identische Halbton-Schritte aufgeteilt, die Schrittweite zwischen den Halbnoten beträgt $\sqrt[12]{2} \approx 1,0594630943593$. Mit diesem Wissen lässt sich für jede Grundfrequenz die komplette Oktave und auch die folgenden Oktaven berechnen, siehe Tabelle.

Tonname	reine Stimmung (ii)		gleichstufige Stimmung (iii)	
	Verhältnis	Frequenz in Hz	Intervall	Frequenz in Hz
c'	1/1	264	$\sqrt[12]{2^0}$	261,63
cis'	25/24	275	$\sqrt[12]{2^1}$	277,18
d'	9/8	297	$\sqrt[12]{2^2}$	293,67
es'	6/5	317	$\sqrt[12]{2^3}$	311,13
e'	5/4	330	$\sqrt[12]{2^4}$	329,63
f'	4/3	352	$\sqrt[12]{2^5}$	349,23
fis'	25/18	367	$\sqrt[12]{2^6}$	369,99
g'	3/2	396	$\sqrt[12]{2^7}$	392,00
as'	8/5	422	$\sqrt[12]{2^8}$	415,31
a'	5/3	440	$\sqrt[12]{2^9}$	440,00
b'	9/5	475	$\sqrt[12]{2^{10}}$	466,16
h'	15/8	495	$\sqrt[12]{2^{11}}$	493,88
c''	2/1	528	$\sqrt[12]{2^{12}}$	523,25

Tabelle 41: Verhältnisse und Intervalle für eine reine- und gleichstufige Stimmung

Grundfrequenz der Oktave
Kammerton A

Die Grundfrequenz der Oktave beträgt immer das doppelte der vorhergehenden Oktave z.B.
$c' = 264; c'' = c' \cdot 2$

Da wir nun wissen wie die Frequenzen der Oktaven berechnet werden können wir die o.g. Formeln für den Bascom Sound-Befehl auflösen, dazu ein Beispiel in dem wir die Werte für den Kammerton a' berechnen. Der Kammerton a' ist mit 440 Hz sicherlich einer der bekanntesten Töne, aber das nur am Rande.

Los geht's

In dem Beispiel werde ich die Frequenz für den gesuchten Ton nach dem beschriebenen Verfahren herleiten.

Es soll, wie bereits erwähnt, der Kammerton a' als 1/4 Note erklingen. Der Mikrocontroller wird mit einer Frequenz von 4 MHz getaktet, der Ausgabe Pin soll PORTD.6 sein.

Die Frequenz von c' für die gleichstufige Stimmung ist gegeben zu 261,63 Hz daraus berechnen wir nach der Formel aus Tabelle 1 die Frequenz für den Kammerton a'

$$a' = \sqrt[12]{2^9} \cdot c'$$

zuerst einmal lösen wir die Wurzel auf, und erhalten

$\sqrt[12]{2^9} = \sqrt[12]{512} = 512^{\frac{1}{12}} = 512^{0,0\overline{833}} = 1,681792831$ für das Intervall

nun noch eine simple Multiplikation und wir erhalten die Frequenz für den Kammerton a':

$$a' = \text{Intervall} \cdot \text{Grundfrequenz} = 1,681792831 \cdot 261,63 = 440,0074584Hz$$

Damit ist auch die Plausibilität der Berechnung bewiesen.

Im nächsten Schritt berechnen wir die Anzahl der High/Low Übergänge (pulses) des Signals, hier nehme ich, wie bereits erwähnt, eine Taktfrequenz des Mikrocontrollers von 4 MHz an, natürlich ist auch jede andere Taktfrequenz möglich.

Die Formel verlangt die Angabe der Frequenzen in Hz. Das Ergebnis ist eine Ganzzahl für den pulses Wert des Bascom Befehls (siehe Bascom Hilfe zum Sound-Befehl).

Bevor wir weiter rechnen, möchte ich noch etwas zu den Notenwerten (iv) sagen. Sicherlich ist bekannt das Noten unterschiedliche Längen haben können, damit ist gemeint wie lange die Note ertönt, diese Zeit wird in Sekunden angegeben. In Tabelle 2 habe ich Notenwerte und Zeiten in Sekunden aufgeführt, die für T verwendet werden können.

Note	1/1	1/2	1/4	1/8	1/16	/32	1,64
Dauer Sek.	2	1	0,5	0,25	0,125	0,0625	0,03125

Tabelle 42: Standardnotenwerte und ihre Zeiten in Sekunde

Wir lösen also die folgende Formel mit den bekannten Werten.

$$duration = \int\left(\frac{T \cdot f_{\text{Quarz}}}{12 \cdot \text{pulses}}\right) = \int\left(\frac{0,5 \cdot 4000000}{12 \cdot 757}\right) = 220,1673272 = 220$$

Damit ist die Berechnung beendet und es bleibt nur noch den Sound-Befehl mit diesen Werten auszuführen, für das Beispiel würde der Befehl wie folgt lauten:

Sound PORTD.6 , 757 , 440

Ich hoffe, das ich mit dieser kurzen Anleitung etwas Licht in die Dunkelheit um den Bascom Sound-Befehl bringen konnte. In diesem Sinne wünsche ich viel Spaß beim experimentieren mit dem Sound-Befehl.

9.5.2 Berechnung der Ausgangsstufe

In der oben abgebildeten Schaltung ist der Anschluss eines einfachen Latsprechers an einem AVR ATMEL ATmega8 Mikrocontroller dargestellt.
Im Folgenden ist beschrieben, welche Berechnungen für die Dimensionierung dieser Schaltung durchgeführt wurden.

Gegebene Werte Lautsprecher (am Beispiel Lautsprecher LSM-S36K)**:**

Impedanz		16±15%
Leistung (Nominal)	W	0.25
Leistung (Maximum)	W	0.5
Resonanzfrequenz	Hz	450±20%
Schalldruckpegel	dB	83±3
Frequenzbereich	Hz	fo~5.000
Betriebstemperatur	°C	-40~+90
Lagertemperatur	°C	-40~+95
Gewicht	g	-

Elektrotechnik Karl-Heinz Mauz GmbH

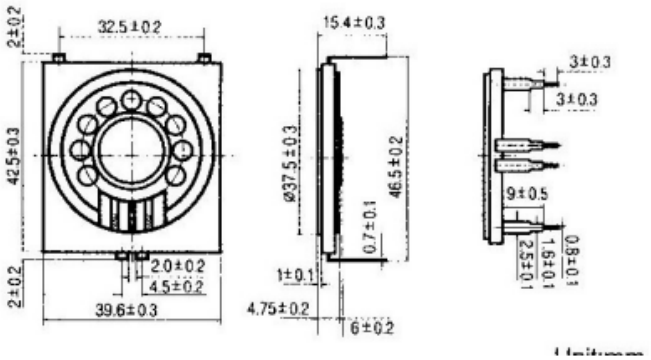
LSM-S36K (Artikel-Nr. 140045)

EKULIT

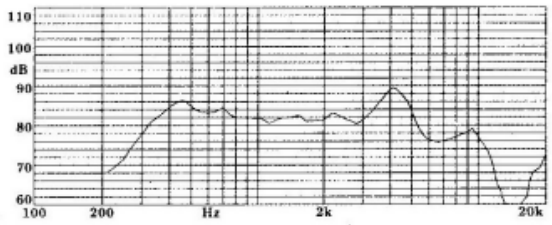
SPECIFICATIONS:

TYPE	UNIT	LSM-S36K
Impedance	Ω	16±15%
Power (Nominal)	W	0.25
Power (Maximum)	W	0.5
Resonant Frequency	Hz	450±20%
Sound Pressure Level	dB	83±3
Frequency Range	Hz	Fo~5.000
Operating Temperature	°C	-40~+90
Storage Temperature	°C	-40~+95

DIMENSIONS :
(Unit: mm)

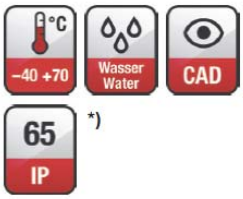


Frequency response:



Adresse: **Felix-Wankel-Str. 35 • 73760 Ostfildern/Nellingen**
Tel: **+49-711/3414023** Fax: **+49-711/3414024**
E-mail: **info@ekulit.de** Web site: **www.EKULIT.de**

K 50 FL Art. No. 2948 – 8 Ω, Art. No. 2949 – 16 Ω, Art. No. 2950 – 50 Ω

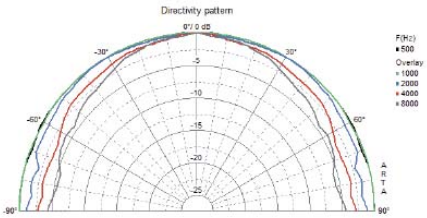


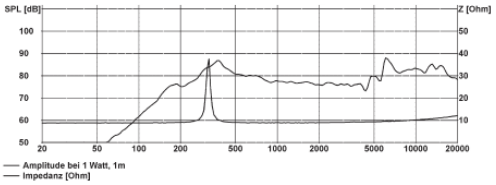
5 cm (2“) Kleinlautsprecher mit Kunststoffmembran. Sehr flache Bauweise. Gute Sprachverständlichkeit.

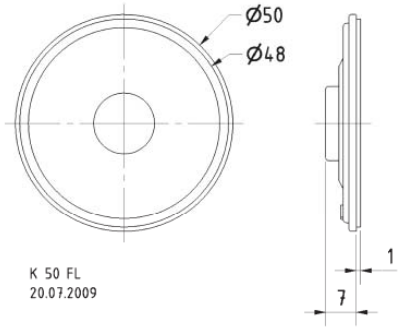
Anwendungsmöglichkeiten: Signalausgabe in Geräten und Anlagen aller Art, in denen wenig Platz zur Verfügung steht. Gut geeignet für Anwendungen im Außenbereich und bei kritischen Umgebungseinflüssen.

5 cm (2”) miniature speaker *with plastic diaphragm. Very compact design. Good speech reproduction.*

Typical applications: *Signal output in machines and other equipment where space is at a premium. Well suited to outdoor applications and where ambient conditions are generally unfavourable.*







		K 50	K 50 FL
Nenn-/Musikbelastbarkeit	*Rated/maximum power*	2 W / 3 W	1 W/2 W
Impedanz	*Impedance*	8 Ω/50 Ω	8 Ω/16 Ω/50 Ω
Übertragungsbereich (–10 dB)	*Frequency response (–10 dB)*	250–10000 Hz	150–20000 Hz
Mittlerer Schalldruckpegel	*Mean sound pressure level*	83 dB (1 W/1 m)	80 dB (1 W/1 m)
Grenzauslenkung x_{mech}	*Excursion limit* x_{mech}	± 0,5 mm	–
Resonanzfrequenz	*Resonance frequency*	400 Hz	340 Hz
Obere Polplattenhöhe	*Height of front pole-plate*	2 mm	–
Schwingspulendurchmesser	*Voice coil diameter*	14 mm Ø	13 mm
Wickelhöhe	*Height of winding*	3 mm	2 mm
Schallwandöffnung	*Cutout diameter*	45 mm Ø	46 mm
Gewicht netto	*Net weight*	0,05 kg	0,014 kg

*) Für Frontseite bei Einbau in ein abgedichtetes Gehäuse
For front side when built into a sealed enclosure

Gegebene Werte der Schaltung:

+UB
5.0 Volt

BC547 – Datenblatt
Ic = 100 mA, siehe unter (Collector Current Continous)
Ptot – 500 mW

Berechnung der Widerstände R2 und R3

Der Kollektorwiderstand besteht aus dem Widerstand R2||R3 plus dem Widerstand des Lautsprechers. Da der Widerstand des Lautsprechers feststeht können nur die Werte von R2||R3 berechnet werden.

Als Transistor soll der BC547B verwendet werden. Dem Datenblatt kann entnommen werden dass der maximale Strom für Ic mit 100mA angegeben ist. Somit ergibt sich für die Berechnung des Widerstand R2||R3 die folgende Formel aus dem Ohm'schen Gesetzt

$$R = \frac{U}{I}$$

$$R2||R3 = \frac{+UB}{Ic} = \frac{5V}{100mA} = 50\,\Omega$$

Wie oben beschrieben besteht der Kollektorwiderstand aus R2||R3 und dem Widerstand des Lautsprechers. Somit muss vom errechneten Widerstandswert für R2||R3 der Impedanzwert des Lautsprechers abgezogen werden.

Damit erhält man für einen 8 Ohm Lautsprecher

$$R2||R3 = R2||R3 - R_{Lautsprecher} = 50\,\Omega - 8\,\Omega = 42\Omega$$

bzw. für einen 16 Ohm Lautsprecher

$$R2||R3 = R2||R3 - R_{Lautsprecher} = 50\,\Omega - 16\,\Omega = 34\Omega$$

Der Widerstandwert von 34 Ohm befindet sich nicht in jeder Widerstandreihe. Da ich nur 68 Ohm Widerstände zur Hand hatte musste ich eine Parallelschaltung für diesen verwenden.
Der Gesamtwert der Parallelschaltung von zwei Widerständen berechnet sich nach der folgenden Formel

$$R_{ges} = \frac{R1 * R2}{R1 + R2}$$

Der Widerstandswert R_{ges} einer Parallelschaltung von zwei Widerständen ist immer kleiner als der kleinste Einzelwiderstand.

Für die verwendeten Widerstände ergibt sich folgende Rechnung

$$R_{ges} = \frac{R1 * R2}{R1 + R2} = \frac{68\Omega * 68\Omega}{68\Omega + 68\Omega} = \frac{1624}{136} = 34\,\Omega$$

Damit liegen wir mit 2 parallel geschalteten 68 Ohm Widerständen genau auf dem gewünschten Wert.

Der Kollektorstrom Ic

Der tatsächliche Kollektorwiderstand Ic mit dem zuvor ermittelten Widerstand von R2||R3 berechnet sich nach

$$Ic = \frac{U}{R_{ges}} = \frac{5V}{34\Omega + 16\Omega} = 0{,}1A = 100\,mA$$

Verlustleistung P

Die Leistung errechnet sich mit

$$P = U * I$$

Im Datenblatt des BC547 ist eine maximale Verlustleistung Ptot = 500mW angegeben. Setzt man die oben berechneten Werte in die Formel ein so erhält man

$$P = 5V * 100mA = 500mW$$

Die berechnete Leistung liegt im Bereich des erlaubten.

Basisstrom und Basisvorwiderstand

Im Datenblatt zum BC547 findet sich in der Kennlinie für V_{CEsat} ein Wert für B von 20 in Sättigung. Dieser Wert wird für die Berechnung des Basisstroms benötigt. Es ergibt sich damit folgende Berechnungsformel

$$I_B = \frac{I_C}{B} = \frac{100\,mA}{20} = 0{,}5\,mA$$

Hinweis: Findet sich keine Angabe zur Verstärkung (B) in Sättigung ist es kein Fehler generell mit einem Wert von 20 bis 30 zu rechen.

Als nächstes muss die Höhe der Eingangsspannung ermittelt werden. Im Datenblatt ist der Spannungsabfall der BE-Strecke in Sättigung U_{BEsat} von V_{BEsat} = 900 mV angegeben.
Die Ausgangsspannung des Microcontrollers liegt typisch bei ca. 5 Volt. Von dieser Spannung werden sicherheitshalber noch ca. 5% abgezogen. Somit ergibt sich für die Eingangsspannung der Treiberstufe der folgende Spannungswert

$$U_{Eingang} = 5V - (5\% + U_{BEsat}) = 5V - (\,0{,}25V + 900mV\,) = 3{,}85V$$

Wir möchten den Ausgang des Microcontrollers mit 2 mA belasten. Somit ergibt sich für den Basisvorwiderstand die folgenden Berechnung

$$R_{vor} = \frac{U_{BE}}{(2mA - I_B)} = \frac{900\,mV}{(2mA - 0{,}5mA)} = 600\,Ohm$$

Ich habe für den Basisvorwiderstand eine Parallelschaltung aus zwei 1k2 Widerständen verwendet.

Die mir dieser Treiberstufe erzeugte Lautstärke der Tonausgabe ist für einen Signalgeber ausreichend. Wird mehr Leistung benötigt so kann ein Audioverstärker IC wie z.B. der Baustein TD7052 mit einer Leistung von 1 Watt verwendet werden. Allerdings muss hier dann auf einen anderen Lautsprecher-Typ gewechselt werden.

9.5.3 BASCOM Beispielcode zur Soundausgabe

```
'################################################################################
'
' DrMaFu_Senso_SoundCheck.BAS                                    Stand 04.01.2017
' ---------------------------                                    (C) Markus Fulde
'
' Testprogramm für Senso - Soundausgabe
'
'################################################################################


'--------------------------------------------------------------------------------
' Compilerinstruktionen und Compilerdirektiven
'--------------------------------------------------------------------------------
$regfile = "m8def.dat"                                    ' Definitionsdatei für ATmega128 laden
$crystal = 8000000                                        ' Quarzfrequenz für 16 MHz festlegen

$hwstack = 128                                            ' HW-Stack auf 128 Bytes erweitern
$swstack = 64                                             ' SW-Stack auf 64 Bytes erweitern
$framesize = 80                                           ' Framesize auf 80 Byte festlegen

$baud = 19200                                             ' Baudrate für RS232 Traceausgabe defi-
nieren


'--------------------------------------------------------------------------------
' Allgemeine Zusatzinformatonen zu Programmbeginn
'--------------------------------------------------------------------------------

' -- Fuses aus ATmega8 ausgelesen ---------------------------------------------
'   HIGH 0xD9
'   LOW  0xE4 = Int. RC Osc. 8 MHz; Start-up time: 6 CK + 64 ms


' -- Allgemeine Informationen zur Sounderzeugung -----------------------------
'     | |C||D| | |F||G||A| | |C||D| | |F||G||A| | |C||D| | |F||G||A| |
'     | |1||1| | |1||1||1| | |2||2| | |2||2||2| | |3||3| | |3||3||3| | |
'     | |#||#| | |#||#||#| | |#||#| | |#||#||#| | |#||#| | |#||#||#| | |
'     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
'     |C1|D1|E1|F1|G1|A1|H1|C2|D2|E2|F2|G2|A2|H2|C3|D3|E3|F3|G3|A3|H3| |
'     |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__| |
'
' (1/1)   - 2 sec
' (1/2)   - 1 sec
' (1/4)   - 0,5 sec
' (1/8)   - 0,25 sec
' (1/16)  - 0,125 sec
' (1/32)  - 0,0625 sec
'
' *****   FORMAT TO FUNCTIONS SOUND  *****
'
' Sound Speaker , Pulses , Periods
' Speaker - port for generations of sound
' Periods - sound frequency (1-65535)
' Pulses  - duration of sound (1-65535)
'
' Periods = Abc[F_crystal / (k * F_nota)]
' Pulses  = Abc[(T_period * F_crystal) / (k * Periods)]
'
' F_crystal - clockrate of controller, Hz
' F_nota    - frequency a notes, Hz
' k = 12    - amount of tacts, for which is formed one period of sound
' T_period  - duration of sounding a notes, sec
' Abc       - function of truncation whole number
'
'================================================================================
' Table of values Pulses,Periods for 3 octaves at frequency of quartz 8000000 Hz
'================================================================================
'Note|Freqvency|Periods|Pulses 1/1|Pulses 1/2|Pulses 1/4|Pulses 1/8|Pulses 1/16|
'================================================================================
' C1 |  261,63 |  2548 |    523   |    262   |    131   |    65    |     33    |
'Cis1|  277,18 |  2405 |    554   |    277   |    139   |    69    |     35    |
' D1 |  293,66 |  2270 |    587   |    294   |    147   |    73    |     37    |
'Dis1|  311,13 |  2143 |    622   |    311   |    156   |    78    |     39    |
' E1 |  329,63 |  2022 |    659   |    330   |    165   |    82    |     41    |
' F1 |  349,23 |  1909 |    698   |    349   |    175   |    87    |     44    |
```

```
'Fis1|  369,99 |  1802 |    740   |    370   |    185   |    92    |     46    |
' G1 |  392,00 |  1701 |    784   |    392   |    196   |    98    |     49    |
'Gis1|  415,30 |  1605 |    831   |    415   |    208   |   104    |     52    |
' A1 |  440,00 |  1515 |    880   |    440   |    220   |   110    |     55    |
'Ais1|  466,16 |  1430 |    932   |    466   |    233   |   117    |     58    |
' B1 |  493,88 |  1350 |    988   |    494   |    247   |   124    |     62    |
'=============================================================================
' C2 |  523,25 |  1274 |   1047   |    523   |    262   |   131    |     65    |
'Cis2|  554,36 |  1203 |   1109   |    554   |    277   |   139    |     69    |
' D2 |  587,32 |  1135 |   1175   |    587   |    294   |   147    |     73    |
'Dis2|  622,26 |  1071 |   1245   |    622   |    311   |   156    |     78    |
' E2 |  659,26 |  1011 |   1319   |    659   |    330   |   165    |     82    |
' F2 |  698,46 |   954 |   1397   |    698   |    349   |   175    |     87    |
'Fis2|  739,98 |   901 |   1480   |    740   |    370   |   185    |     92    |
' G2 |  784,00 |   850 |   1568   |    784   |    392   |   196    |     98    |
'Gis2|  830,60 |   803 |   1661   |    831   |    415   |   208    |    104    |
' A2 |  880,00 |   758 |   1720   |    880   |    440   |   220    |    110    |
'Ais2|  932,32 |   715 |   1865   |    932   |    466   |   233    |    117    |
' B2 |  987,75 |   675 |   1976   |    988   |    494   |   247    |    124    |
'=============================================================================
' C3 | 1046,50 |   637 |   2093   |   1047   |    523   |   262    |    131    |
'Cis3| 1108,70 |   601 |   2218   |   1109   |    554   |   277    |    139    |
' D3 | 1174,60 |   566 |   2350   |   1175   |    587   |   294    |    147    |
'Dis3| 1244,50 |   536 |   2490   |   1245   |    622   |   311    |    156    |
' E3 | 1318,50 |   483 |   2638   |   1319   |    659   |   330    |    165    |
' F3 | 1396,90 |   477 |   2794   |   1397   |    698   |   349    |    175    |
'Fis3| 1480,00 |   450 |   2960   |   1480   |    740   |   370    |    185    |
' G3 | 1568,00 |   425 |   3136   |   1568   |    784   |   392    |    196    |
'Gis3| 1661,20 |   401 |   3322   |   1661   |    831   |   415    |    208    |
' A3 | 1720,00 |   388 |   3440   |   1720   |    880   |   440    |    220    |
'Ais3| 1864,60 |   358 |   3730   |   1865   |    932   |   466    |    233    |
' B3 | 1975,50 |   337 |   3952   |   1976   |    988   |   494    |    247    |
'=============================================================================



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

' ----- LED's -----
Alive_led_pin Alias Pinb.0                                  ' GPIO für Alive-LED (für DDR oder In-
put)
Alive_led Alias Portb.0                                     ' GPIO für Alive-LED für Output oder
Pullup)


' ----- Sound -----
Speaker_pin Alias Pinb.1                                    ' GPIO für Sound-Ausgabe (für DDR oder
Input)
Speaker Alias Portb.1                                       ' GPIO für Sound-Ausgabe (für Output
oder Pullup)


' ----- Key / Tasten -----

' BLAU
Key_taster1_pin Alias Pind.3                                ' KEY1: GPIO für Taste 1 BLAU (für DDR
oder Input)
Key_taster1 Alias Portd.3                                   ' KEY1: GPIO für Taste 1 BLAU (für Out-
put oder Pullup)

' ROT
Key_taster2_pin Alias Pind.2                                ' KEY2: GPIO für Taste 2 ROT (für DDR
oder Input)
Key_taster2 Alias Portd.2                                   ' KEY2: GPIO für Taste 2 ROT (für Output
oder Pullup)

' GRÜN
Key_taster3_pin Alias Pind.4                                ' KEY3: GPIO für Taste 3 GRÜN (für DDR
oder Input)
Key_taster3 Alias Portd.4                                   ' KEY3: GPIO für Taste 3 GRÜN (für Out-
put oder Pullup)

' GELB
Key_taster4_pin Alias Pind.7                                ' KEY4: GPIO für Taste 4 GELB (für DDR
oder Input)
Key_taster4 Alias Portd.7                                   ' KEY4: GPIO für Taste 4 GELB (für Out-
put oder Pullup)
```

```
'-----------------------------------------------------------------------------
' Definition von Konstaten
'-----------------------------------------------------------------------------

' ----- Für Testumgebung bzw. Traceausgaben -----
Const Main_testmodus = 1                                    ' Flag für Testmodus Allgemeinsystem


' ----- Allgemeine Systemkonstanten -----

' Tatsächliches Allgemeines
' Const Led_aus = 0                                           ' LED Konstake für LED aus im echten
Spiel
' Const Led_ein = 1                                           ' LED Konstake für LED ein im echten
Spiel

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

Const Pullup_aus = 0
Const Pullup_ein = 1

Const Key_pressed_yes = 0
Const Key_pressed_no = 1


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


' ----- SOUND -----
Const Sound_ton1 = 637
Const Sound_dauer1 = 262

Const Sound_ton2 = 483
Const Sound_dauer2 = 330

Const Sound_ton3 = 425
Const Sound_dauer3 = 392

Const Sound_ton4 = 337
Const Sound_dauer4 = 494


'-----------------------------------------------------------------------------
' 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
Dim Temp_word_2 As Word                                     ' Temporäre Word Variable 2


' ----- System -----
Dim Sectick_counter As Word                                 ' Globaler Sekundenzähler



'-----------------------------------------------------------------------------
' 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 freigeben



' ------------------------------ Port's und Pin's --------------------------

' ----- LED-Konfigurationen -----
Config Alive_led_pin = Output                             ' GPIO für Alive-LED ist Output


' ----- SOUND-Konfigurationen -----
Config Speaker_pin = Output                               ' GPIO für Soundausgabe ist Output


' ----- KEY-Tasten-Konfiguration -----
Config Key_taster1_pin = Input                            ' GPIO für Taste 1 auf Inupt schalten
und PullUp aktivieren
Key_taster1 = Pullup_ein

Config Key_taster2_pin = Input                            ' GPIO für Taste 2 auf Inupt schalten
und PullUp aktivieren
Key_taster2 = Pullup_ein

Config Key_taster3_pin = Input                            ' GPIO für Taste 3 auf Inupt schalten
und PullUp aktivieren
Key_taster3 = Pullup_ein

Config Key_taster4_pin = Input                            ' GPIO für Taste 4 auf Inupt schalten
und PullUp aktivieren
Key_taster4 = Pullup_ein



' ------------------------------ Variablen und Werte -------------------------


' ----- System -----
Sectick_counter = 0                                       ' Globaler Sekundenzähler initialisieren


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

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


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



' ###############################################################################
'
'                              Hauptprogramm SoundCheck
'
' ###############################################################################

' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Main_testmodus
   Print ""
   Print "** OK, let's GO **"
#endif


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

Do                                                        ' Hauptschleife

   ' -------------------- Testaufgaben waehrend Entwicklung -------------------
```

```
   ' Tasten 1 - 4 abfragen und abhängig von Tasten Ton auslösen

   If Key_taster1_pin = Key_pressed_yes Then
      Sound Speaker , Sound_ton1 , Sound_dauer1
   End If

   If Key_taster2_pin = Key_pressed_yes Then
      Sound Speaker , Sound_ton2 , Sound_dauer2
   End If

   If Key_taster3_pin = Key_pressed_yes Then
      Sound Speaker , Sound_ton3 , Sound_dauer3
   End If

   If Key_taster4_pin = Key_pressed_yes Then
      Sound Speaker , Sound_ton4 , Sound_dauer4
   End If

Loop                                                    ' Hauptschleife

'## End Hauptprogramm ###########################################################

End


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

'-------------------------------------------------------------------------------
' Interrupt-Service-Routine (Timer1): Sekunden_tick
'
' Routine zur Auswertung des Timer Interrupts
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Sectick_counter = Variable für den Sekundencounter
'-------------------------------------------------------------------------------
Sekunden_tick:

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

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

   ' Timervariable erhöhen
   Incr Sectick_counter

Return
'-- End Sekunden_tick  -------------------------------------------------------



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


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



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

' System halt
End                                                     'end program


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


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

```
'#################################### END ###################################
```

Software 2: Code zur Ansteuerung des Lautsprechers

9.6 LED-Zuordnung Schaltplan

Im Projekt Senso werden insgesamt 5 LED's verwendet. Eine 3mm Low-Current-LED und 4 Standard-LED's welche ohne Transistortufe mittels Metallschichtvorwiderstand direkt an den GPIO-Pin's des ATmega8L-Controllerst betrieben werden.

In der folgenden Tabelle befinden sich alle LED's des Projekt beschrieben und den HW-Ressourcen im Schaltplan inkl. Funktion zugeordnet:

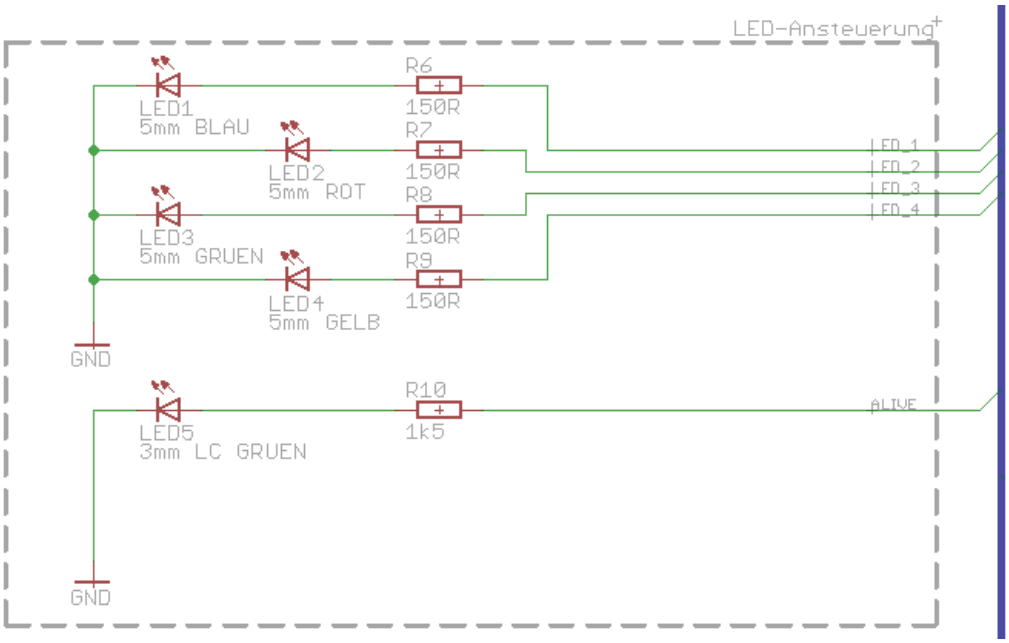
LED	*Funktion*	*Abkürzung*	*Farbe*
1	Spiel-LED BLAU – 5 mm	LED1	BLAU
2	Spiel-LED ROT – 5 mm	LED2	ROT
3	Spiel-LED GRÜN – 5 mm	LED3	GRÜN
4	Spiel-LED GELB – 5 mm	LED4	GELB
5	Alive LED – 3 mm	LED5	GRÜN

Tabelle 43: LED-Zuordnung Schaltplan

Die grüne 3mm Alive LED zeigt an, dass der Controller bzw. das Spiel normal arbeitet und der Hauptschalter eingeschaltet ist. Dabei blinkt die LED im Sekundenrhythmus. Die LED befindet sich aber nur intern im Gehäuse und ist nicht nach außen geführt.

Die 5mm-LED's in den Farben Blaut, Rot, Grün und Gelb dienen für den Spielablauf.

Beschaltung:



Schaltbild 10: Beschaltung LED's

Bauteile:

Stückliste: LED-Ansteuerung			
Widerstände		***Halbleiter***	
R6, R7, R8. R9	Metallschichtwiderstand 150 Ω	LED5	Low Current LED, 3mm, green
R10	Metallschichtwiderstand 1k5 Ω	LED1	Standard LED, 5mm, blue
		LED2	Standard LED, 5mm, red
		LED3	Standard LED, 5mm, green
		LED4	Standard LED, 5mm, yellow

Tabelle 44: Bauteile für LED-Ansteuerung

Ressourcenzuordnung zum ATmega8L:

Nummer	***Schaltbild***		***Ressource ATmega8***		
1	ALIVE	ALIVE	PortB.0	[PB0]	(GPIO Output)
2	LED_1	LED_1	PortB.6	[PB6]	(GPIO Output)
3	LED_2	LED_2	PortB.7	[PB7]	(GPIO Output)
4	LED_3	LED_3	PortD.5	[PD5]	(GPIO Output)
5	LED_4	LED_4	PortD.6	[PD6]	(GPIO Output)

Tabelle 45: Ressourcenzuordnung für LED-Asnsteuerung

Die Gehäusedurchführung / Gehäusefassung für die 5mm LED's wird als LED-Fassung aus Neopren ausgeführt.

Wasserdichte Neopren Fassungen für 5mm LEDs

Passend für LED 5 mm

Bestellnummer LEDPROFISHOP: 09-023

10 Mechanik

Die Schwierigkeit bestand darin, ein formschönes Gehäuse zu finden welches nicht zu groß ist sondern das auch nicht gut in der Hand gehalten werden kann und in dem sowohl das ausgewählt Display als auch der 9V-Batterieblock Platz findet.

Nach langer Suche wird nun das folgende Gehäuse eingesetzt:

BOPLA BOS 750

Digitales Handgehäuse

Die Tastenfläche ist vertieft zur Aufnahme einer Folientastatur. Batteriefach für 9V-Block, mit Panoramascheibe (muss von außen geklebt werden) Tastenfeld: 70 x 86mm

Hersteller : BOPLA
Artikelnummer des Herstellers : 34750000
Verpackungsgewicht : 0.109 kg
RoHS : konform

Bestllnummer Reichelt: *BOPLA BOS 750*

Typ	Handgehäuse
Farbe	schwarz
Ausführung	mit Panoramascheibe
Länge	157 mm
Breite	84 mm
Höhe	30 mm

Tabelle 46: Gehäuse

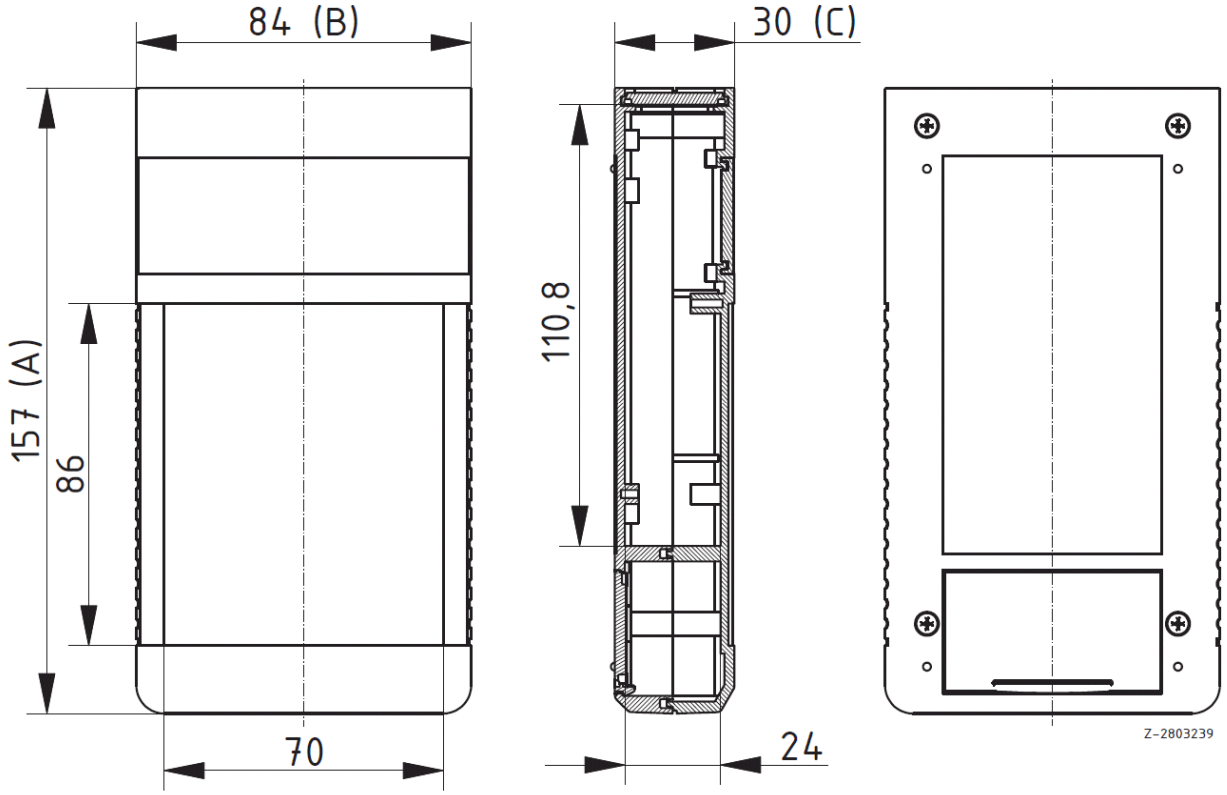


Abbildung 14: Gehäusemaße

10.1 Gehäuseumsetzung im Projekt Senso:

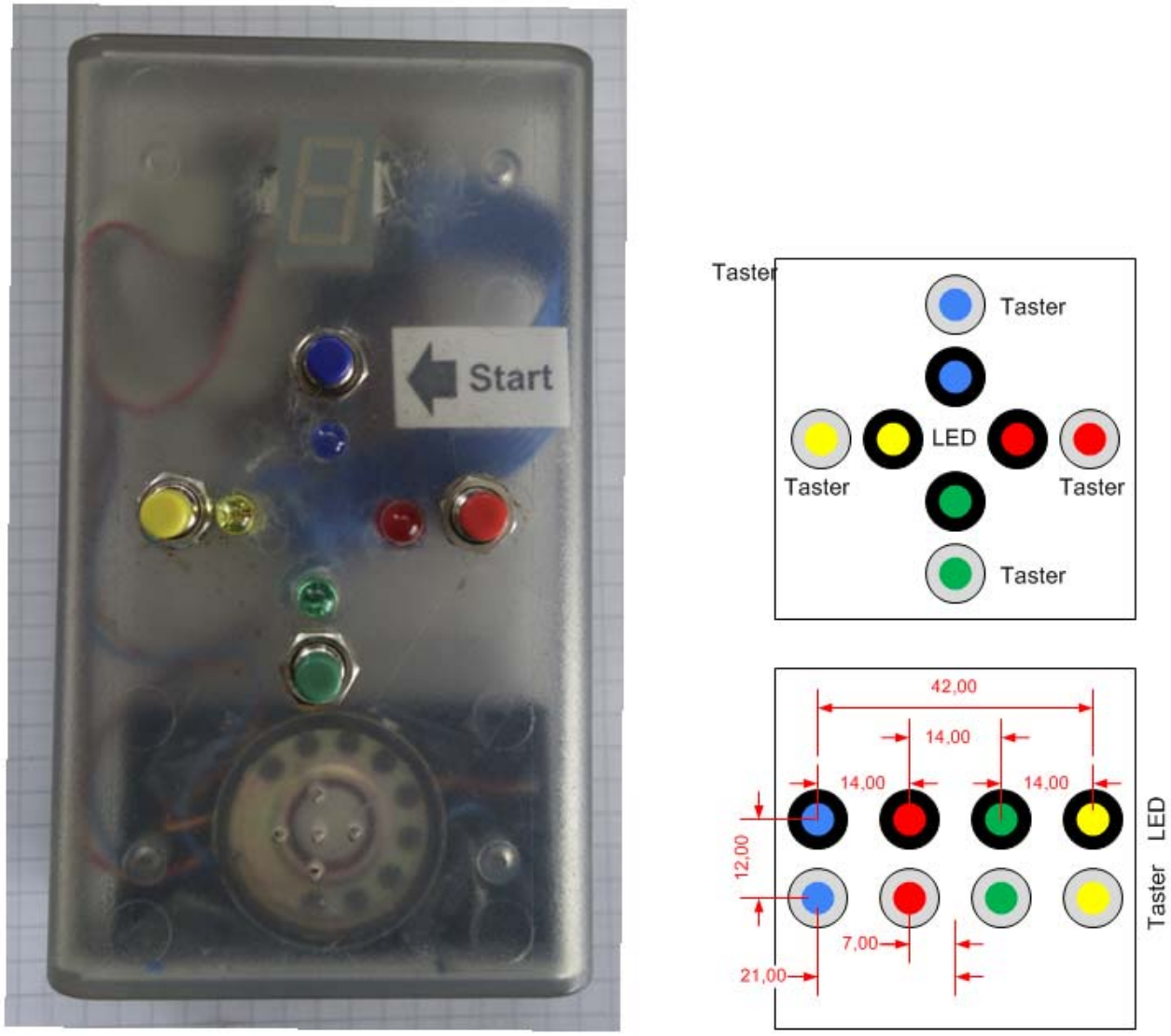


Abbildung 15: Anordnung der Tasten und LED's im Gehäuse

Obige Abbildung zeigt auf der linken Seite das originale Senso-Spiel (Vorgängervariante). Im Projekt Senso werden die Tasten und LED's im Gegensatz zur Kreuzform in einer Linienform angeordnert.

Abbildung 16: Der fertige Senso im Gehäuse

Obige Abbildung zeigt das fertig montierte Gerät im Gehäuse inkl. Beschriftung.

10.2 Bohrungsmatrix für Lautsprecherbohrungen:

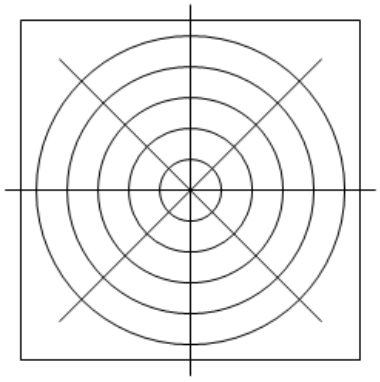


Abbildung 17: Bohrschema für Lautsprecher

10.3 Lautsprecher-Einbau:

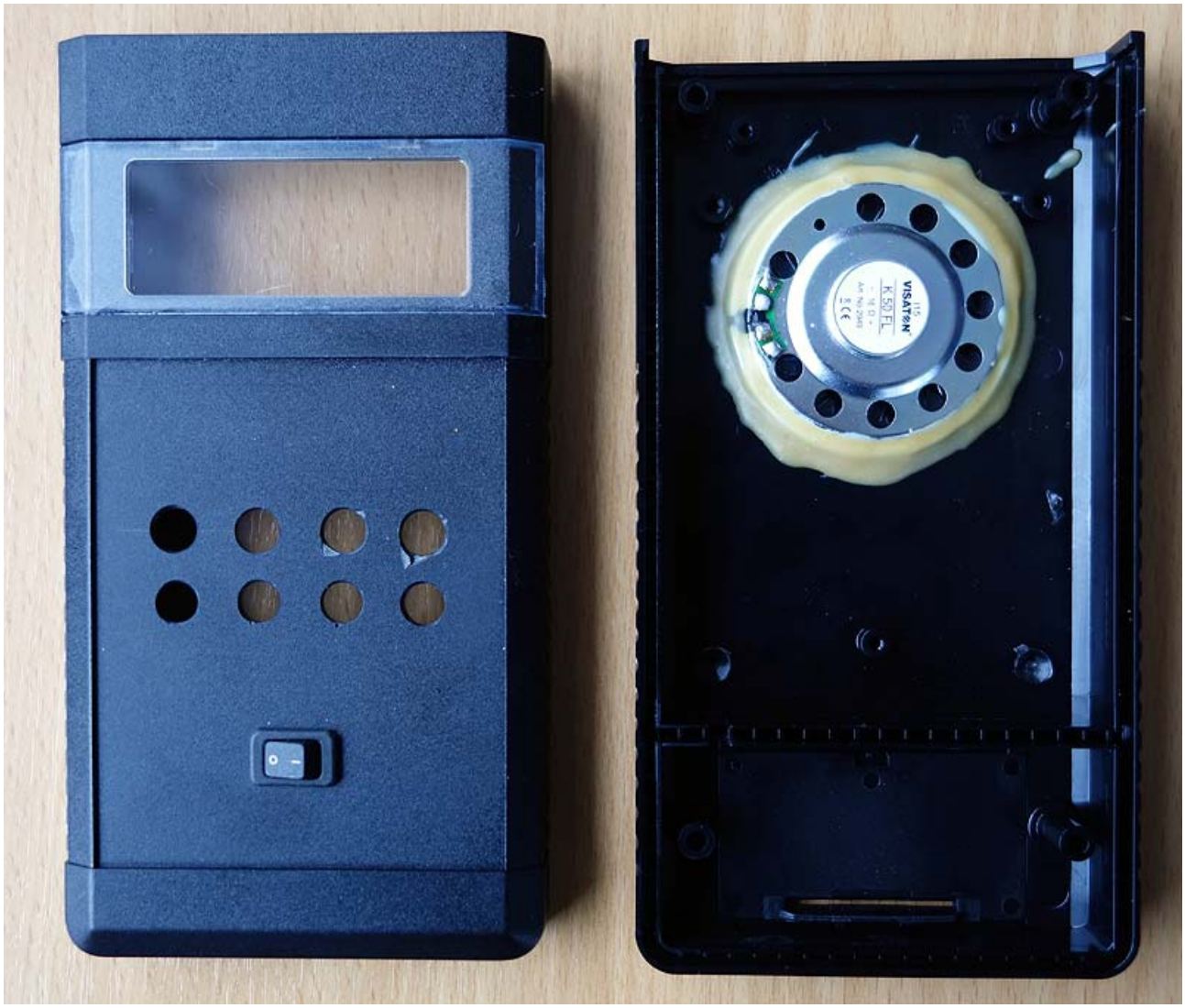


Abbildung 18: Eingebauter und verklebter Lautsprecher

11 Bauteile und Bauteilbeschaffung

Für das Prototyping und die Inbetriebnahme der im obigen Kapitel „Grundlagen“ beschriebenen Einzelthemen werden die Einzelteile in Verbindung mit dem STK500 und Labor-Lochrasterplatinen aufgebaut und in Betrieb genommen.

Die folgenden Stücklisten geben einen Überblick über die benötigten Bauelemente und ergeben gleichzeitig die Bestellisten von Conrad Electronic, Reichelt Electronic und Led-Profi-Shop.

Bauelemente Reichelt Electronik:

Stk.	*Beschreibung*	*Bestellnummer*	*Einzelpreis*	*Gesamtpreis*
		Kondensatoren		
5	Keramikkondensator 100nF 50 VDC	KERKO 100N	0,09	0,45
2	Elektrolytkondensator 100µF / 35V	RAD 100/35	0,04	0,08
		Widerstände		
2	Metallschichtwiderständ 1M Ohm	METALL 1,00M	0,082	0,164
4	Metallschichtwiderständ 150 Ohm	METALL 150	0,082	0,328
3	Metallschichtwiderständ 1k2 Ohm	METALL 1,20K	0,082	0,246
1	Metallschichtwiderständ 1k5 Ohm	METALL 1,50K	0,082	0,082
1	Metallschichtwiderständ 30 Ohm	METALL 30,0	0,082	0,082
2	Metallschichtwiderständ 68 Ohm	METALL 68,0	0,082	0,164
		Halbleiter		
1	Diode 1N4148	1N 4148	0,04	0,04
1	Spannungsregler 2A positiv, TO-220	µA 78S05	0,39	0,39
2	BC 547B Transistor NPN TO-92 45V 0,1A 0,5W	BC 547B	0,04	0,08
1	DOG LCD-Modul 08x1, Hintergrund grün	EA DOGM081E-A	10,20	10,20
1	Led-Beleuchtung für EA DOGL..Farbe: grün	EA LED55X31-G	2,40	2,40
1	LED 3mm, low-Current, grün	LED 3MM 2MA GN	0,09	0,09
1	LED 5mm, Standard, Blau	LED 5MM ST BL	0,58	0,58
1	LED 5mm, Standard, Rot	LED 5MM ST RT	0,10	0,10
1	LED 5mm, Standard, Grün	LED 5MM ST GN	0,10	0,10
1	LED 5mm, Standard, Gelb	LED 5MM ST GE	0,10	0,10
1	ATMega AVR-RISC-Controller, S-DIL-28	ATMEGA 8L8 DIP	2,10	2,10
		Sonstiges		
1	Kleinlautsprecher LSM-S36K bzw. VIS K50FL-16	LSM-S36K	2,70	2,70
1	MA03-2 für ISP06	SL 2X50G 2,54	0,85	0,85
1	MA06-02 für RS232	SL 2X50G 2,54	--	--
1	Gehäuse	BOPLA BOS 750	15,60	15,60
1	Batterieclip für 9-Volt-Block, High-Quality, vertikal	CLIP HQ9V	0,36	0,36
1	IC-Sockel, 28-polig, schmal	GS 28P-S	0,47	0,47
1	Miniatur-Drucktaster, Ein 0,5A-250VAC, T 113A, 250V / 0,5A	T113A BL	0,47	0,47
1	Miniatur-Drucktaster, Ein 0,5A-250VAC, T 113A, 250V / 0,5A	T113A RT	0,47	0,47
1	Miniatur-Drucktaster, Ein 0,5A-250VAC, T 113A, 250V / 0,5A	T113A GN	0,47	0,47
1	Miniatur-Drucktaster, Ein 0,5A-250VAC, T 113A, 250V / 0,5A	T113A GE	0,47	0,47

Tabelle 47: Bauelemente Reichelt Electronik

Bauelemente Conrad Electronic:

Stk.	Beschreibung	Bestellnummer	Einzelpreis	Gesamtpreis
Sonstiges				
1	9 V Block-Batterie Alkali-Mangan Varta Longlife 6LR61 9 V	650709 - 62	3,99	3,99
1	Wippschalter 250 V/AC 3 A 1 x Aus/Ein Eledis MR519-0F522 rastend	700039 – 62	2,06	2,06
1	Trockenmittelbeutel 5 g/Beutel Transparent Material Kieselgel	181896 – 62	1,52	1,52

Tabelle 48: Bauelemente Conrad Electronic

Bauelemente LEDProfiShop:

Stk.	Beschreibung	Bestellnummer	Einzelpreis	Gesamtpreis
Sonstiges				
4	Wasserdichte Neopren Fassungen für 5mm LEDs	09-023	0,95	3,80

Tabelle 49: Bauelemente Conrad Electronic

Die vorausberechneten Materialkosten belaufen sich auf ca. 50,924 Euro.

12 Hardware

12.1 Festlegung von Netzklassen im Projekt

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

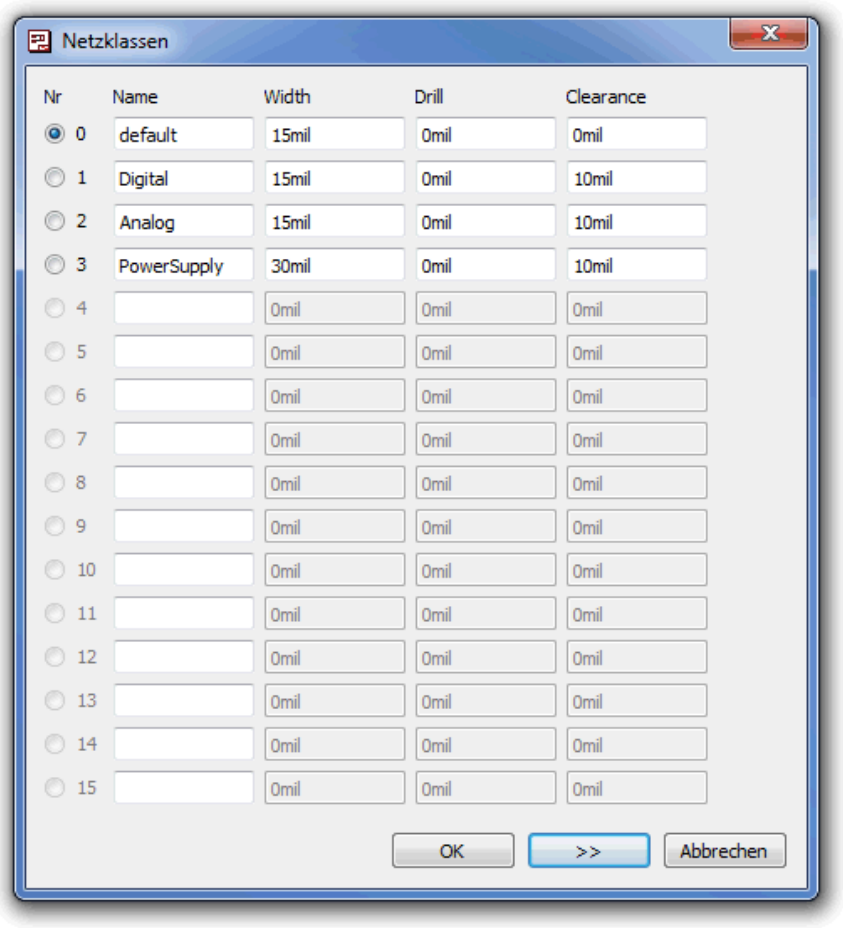
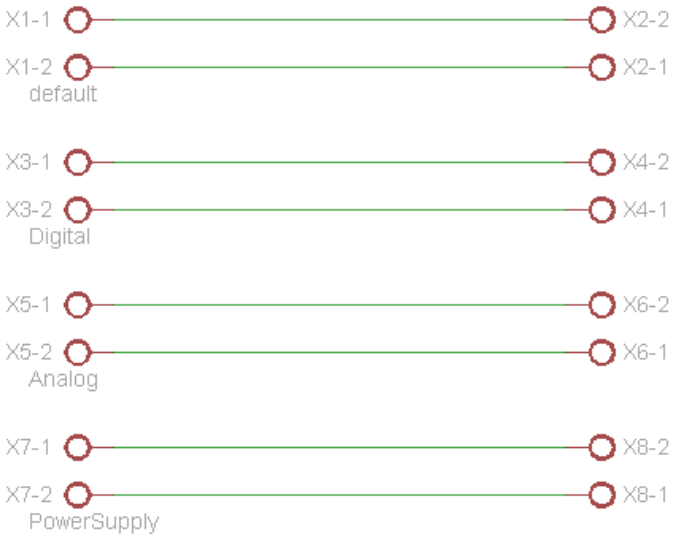


Abbildung 19: Definition der Netzklassen

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



Schaltbild 11: Schaltbild für Definition von Netzklassen

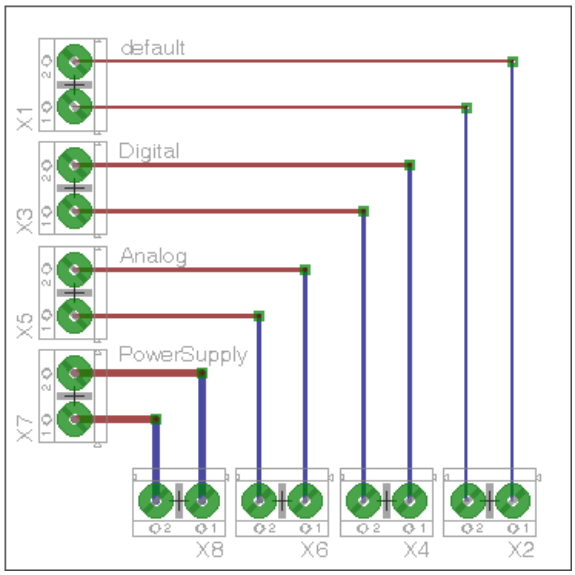
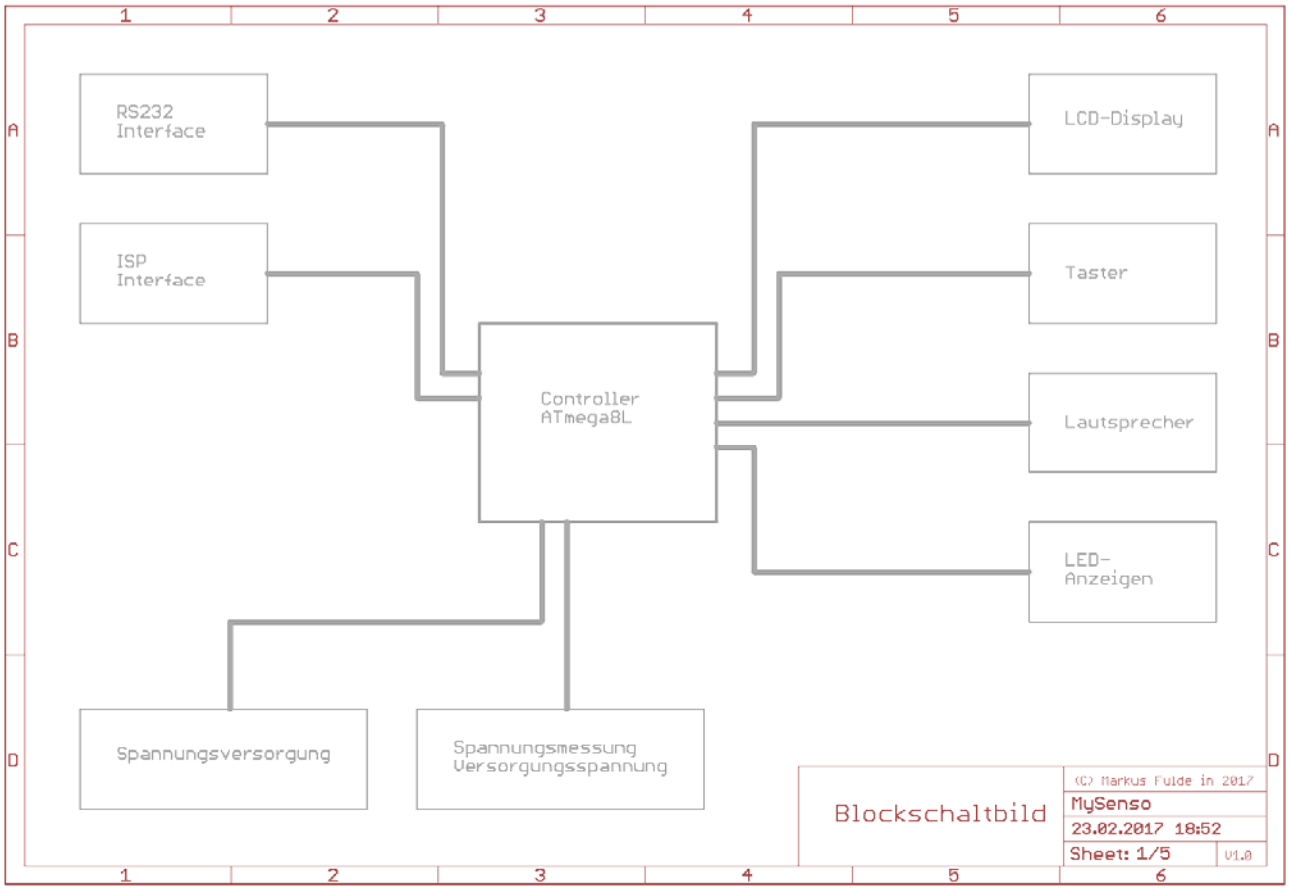


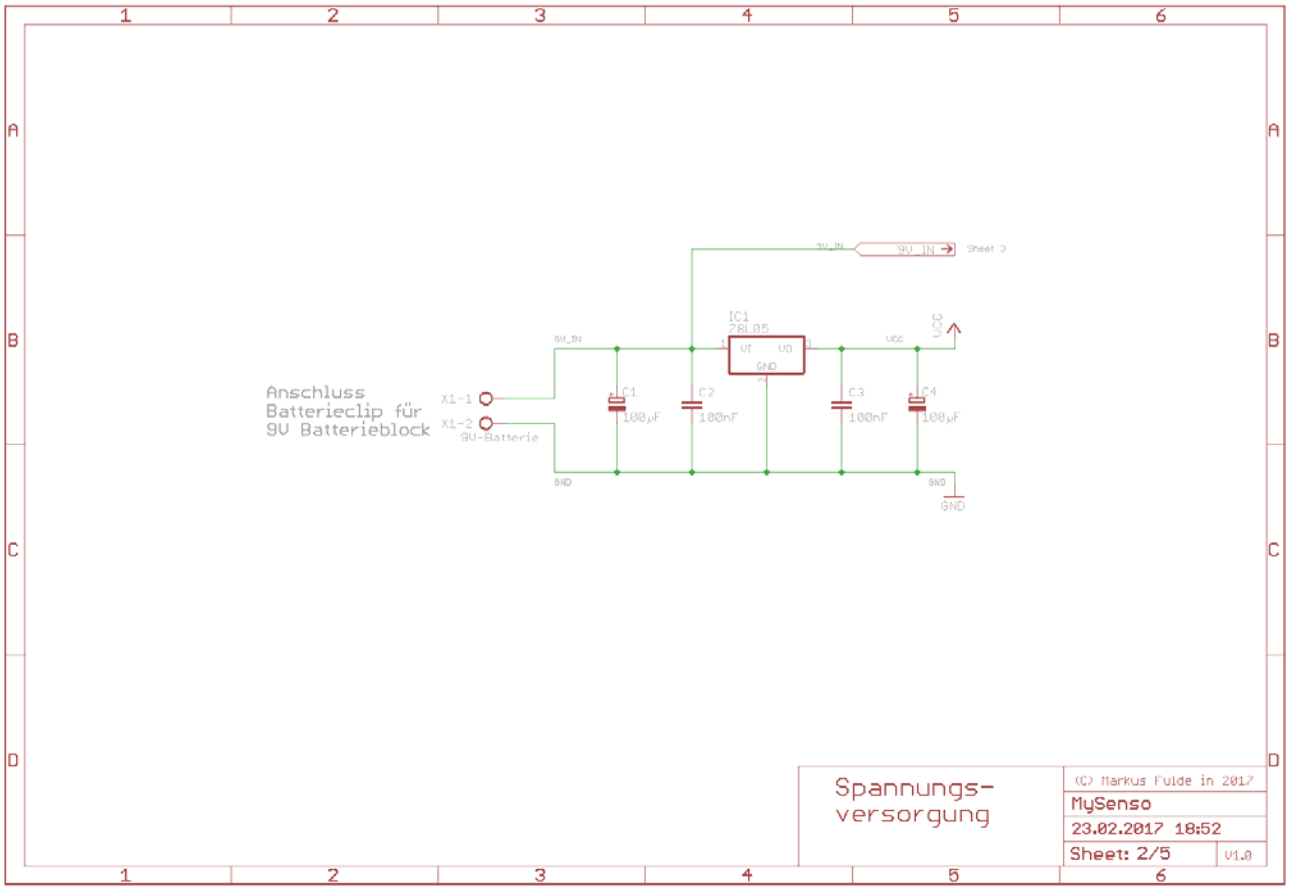
Abbildung 20: Demoboard N etzklassen

12.2 Die PCB zum Senso

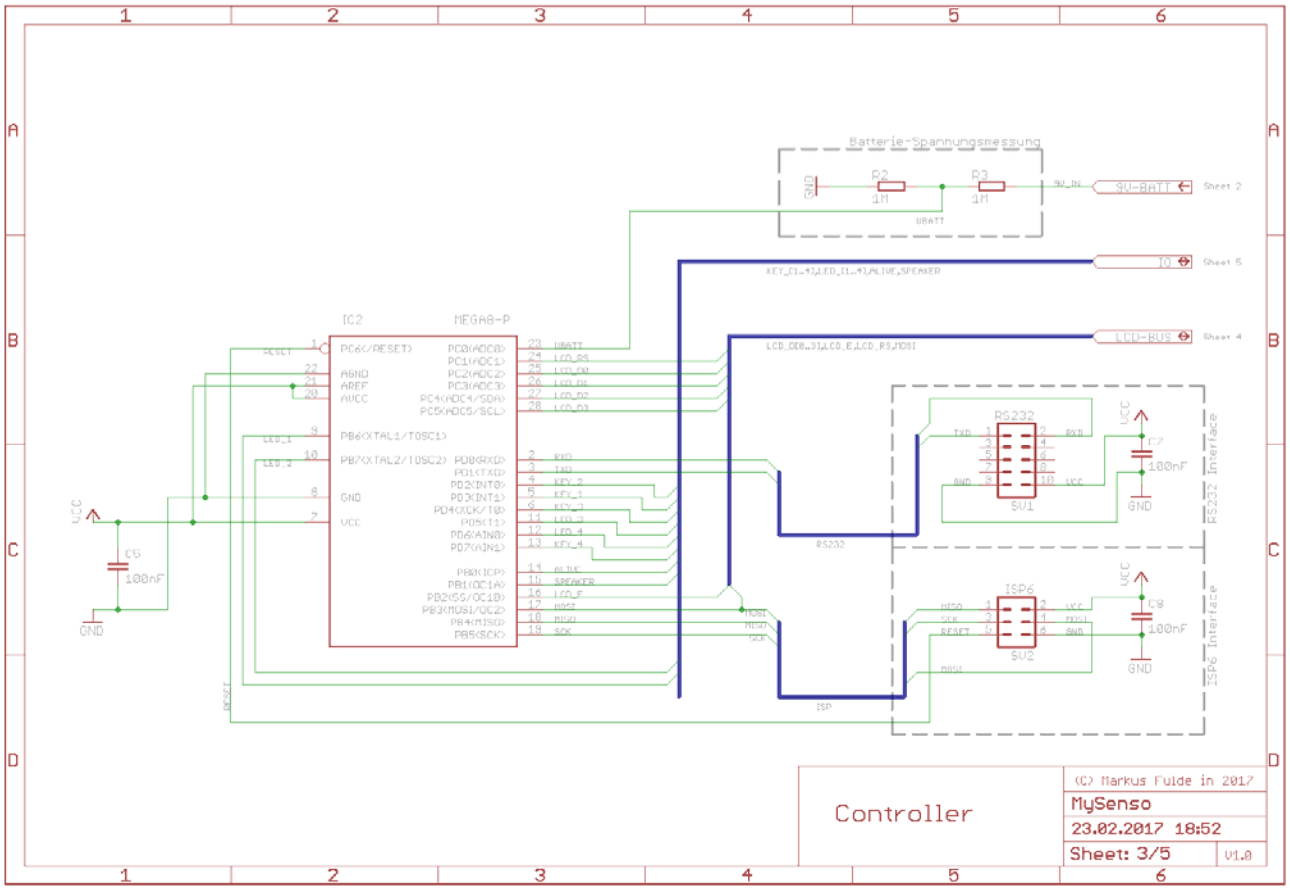
12.2.1 Schematic



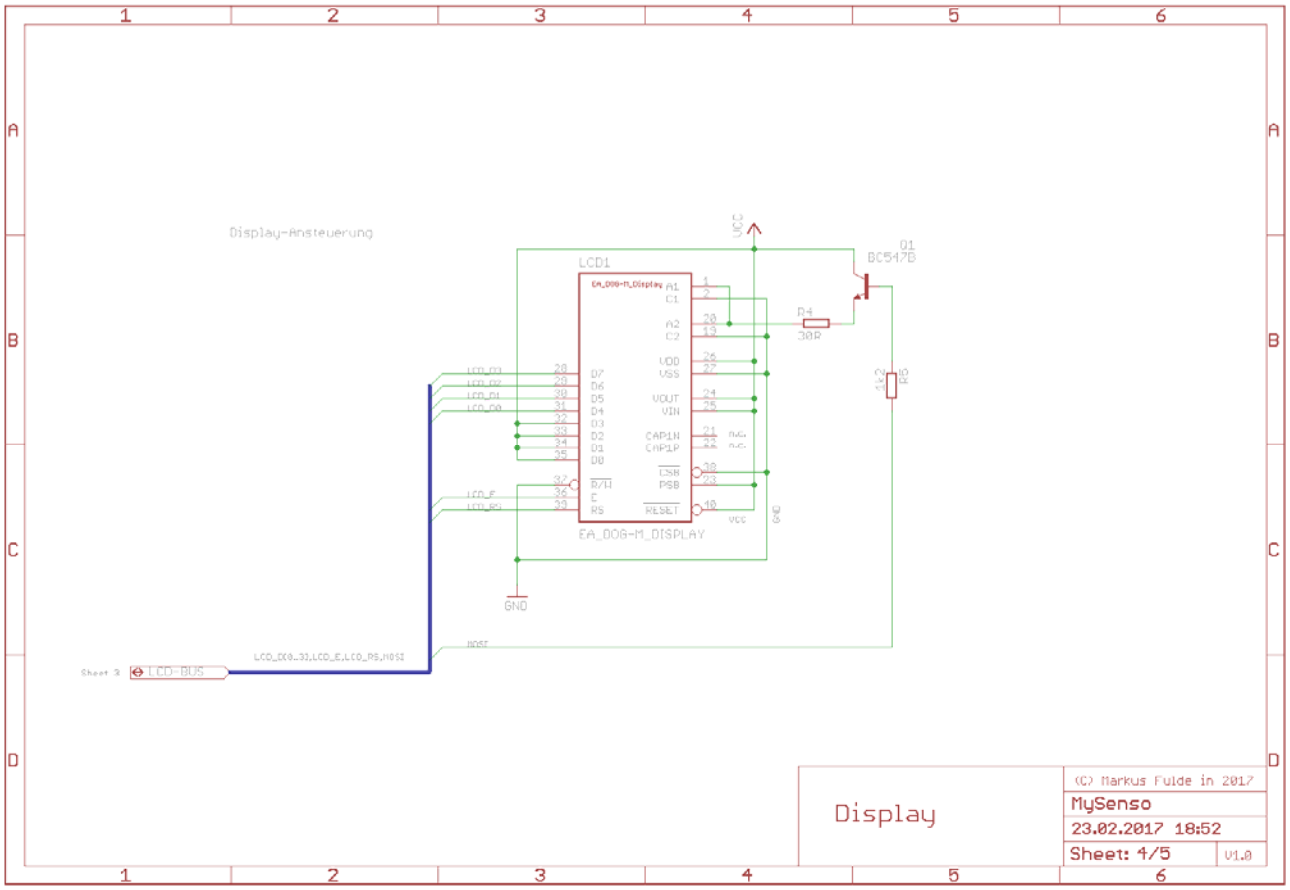
Schaltbild 12: Schaltbild Senso - Sheet 1



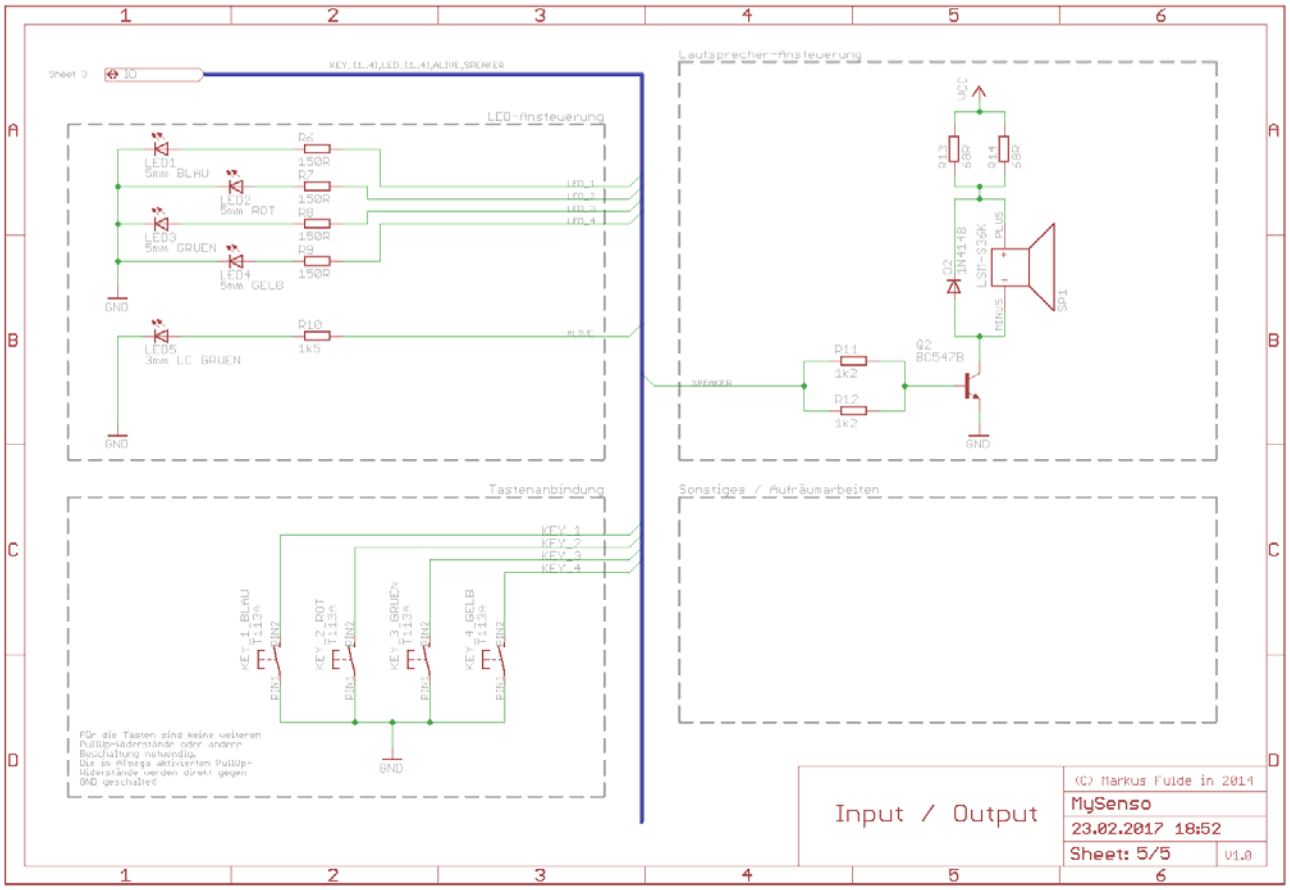
Schaltbild 13: Schaltbild Senso - Sheet 2



Schaltbild 14: Schaltbild Senso - Sheet 3



Schaltbild 15: Schaltbild Senso - Sheet 4



Schaltbild 16: Schaltbild Senso - Sheet 5

12.2.2 Layout, Layer und Bestückung

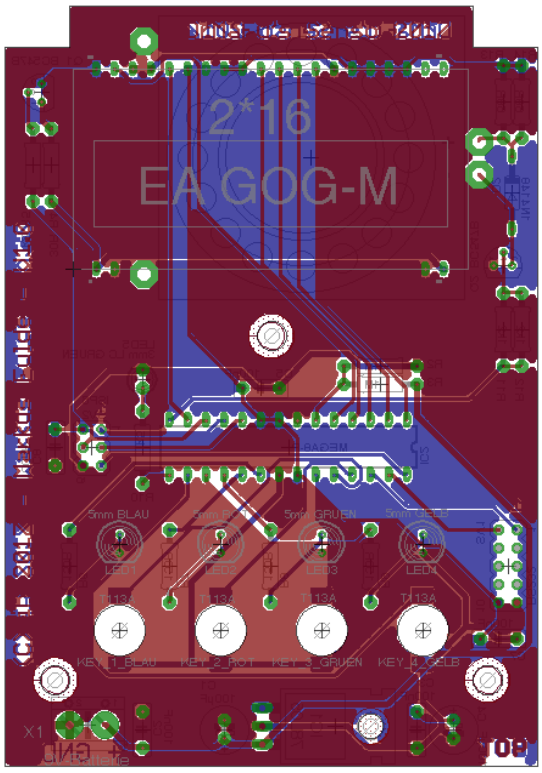


Abbildung 21: PCB Senso – Layout gesamt

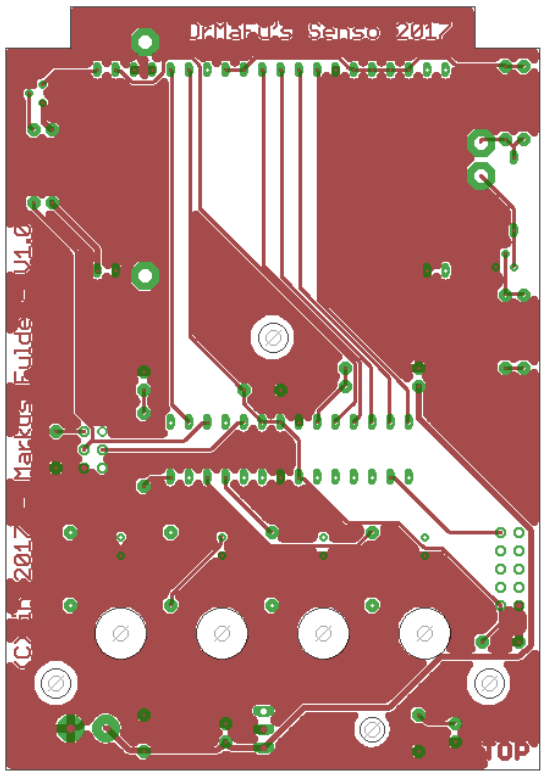


Abbildung 22: PCB Senso – Top Layer

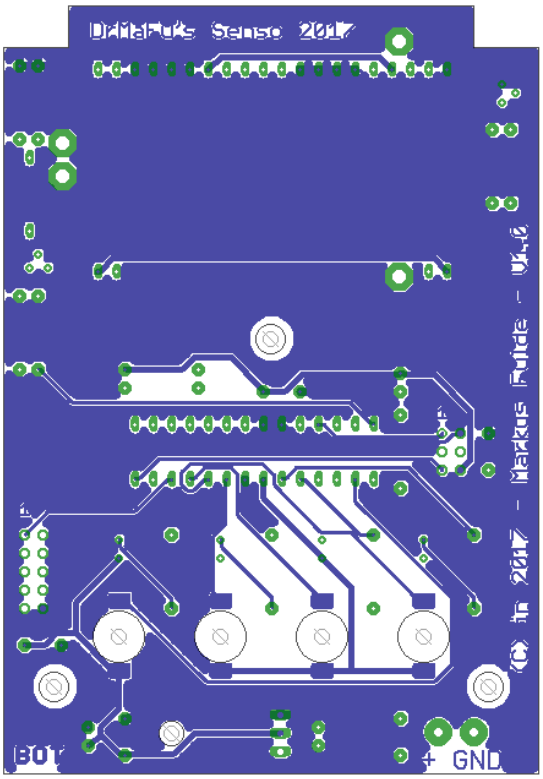


Abbildung 23: PCB Senso – Bottom Layer

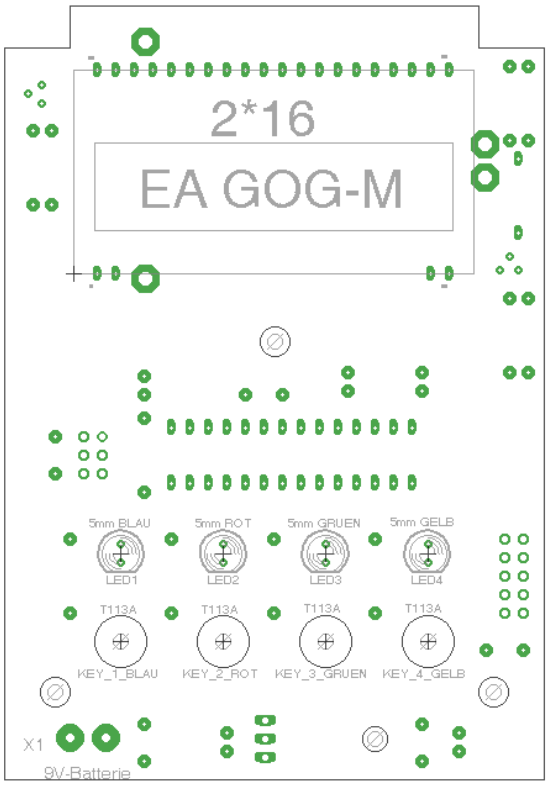


Abbildung 24: PCB Senso – Bestückung Top Layer

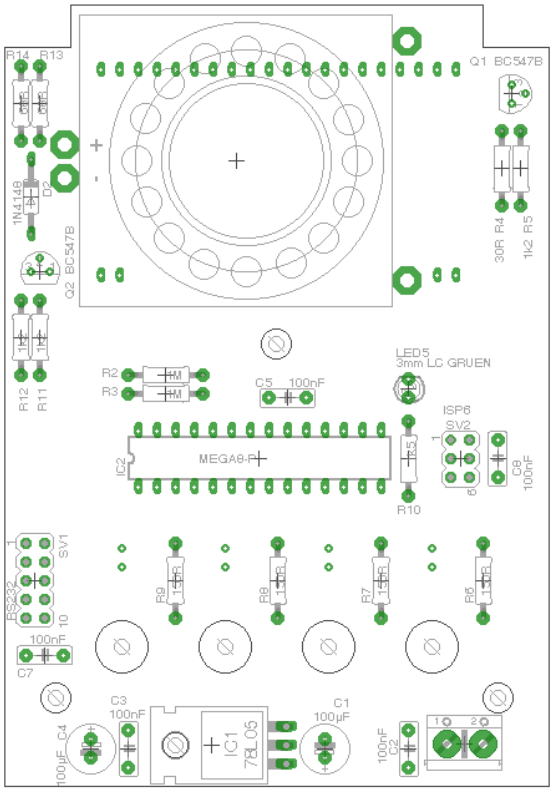


Abbildung 25: PCB Senso – Bestückung Bottom Layer

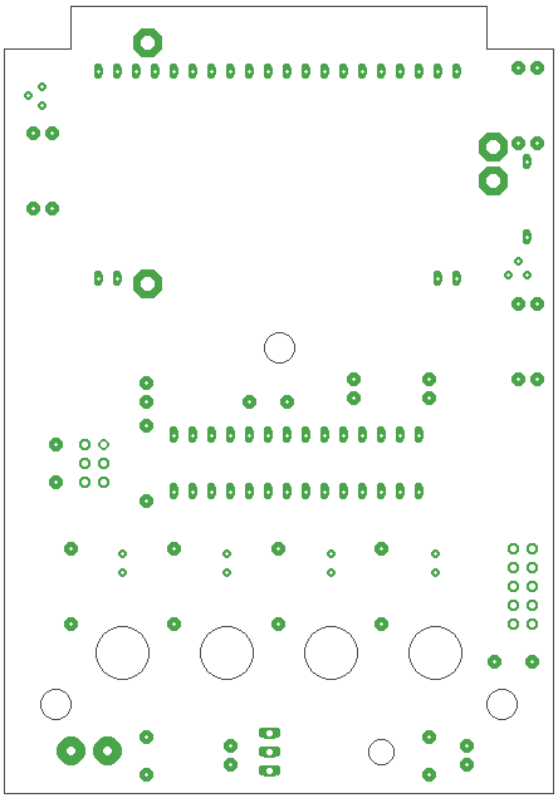


Abbildung 26: PCB Senso – Pads und Vias

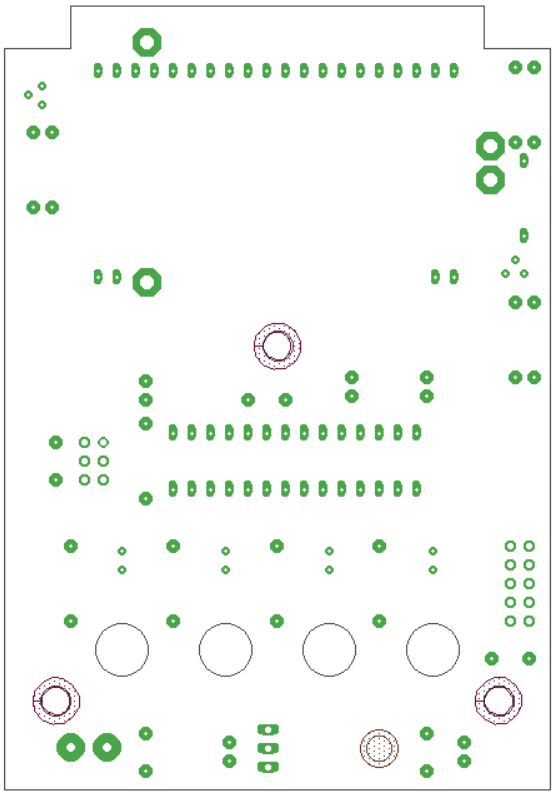


Abbildung 27: PCB Senso – Restricted Areas

12.2.3 Bestückungen in Groß

Zur besseren Lesbarkeit und Ekennung hier nochmals die Bestückungsdrucke in Groß

Die TOP-Bestückung

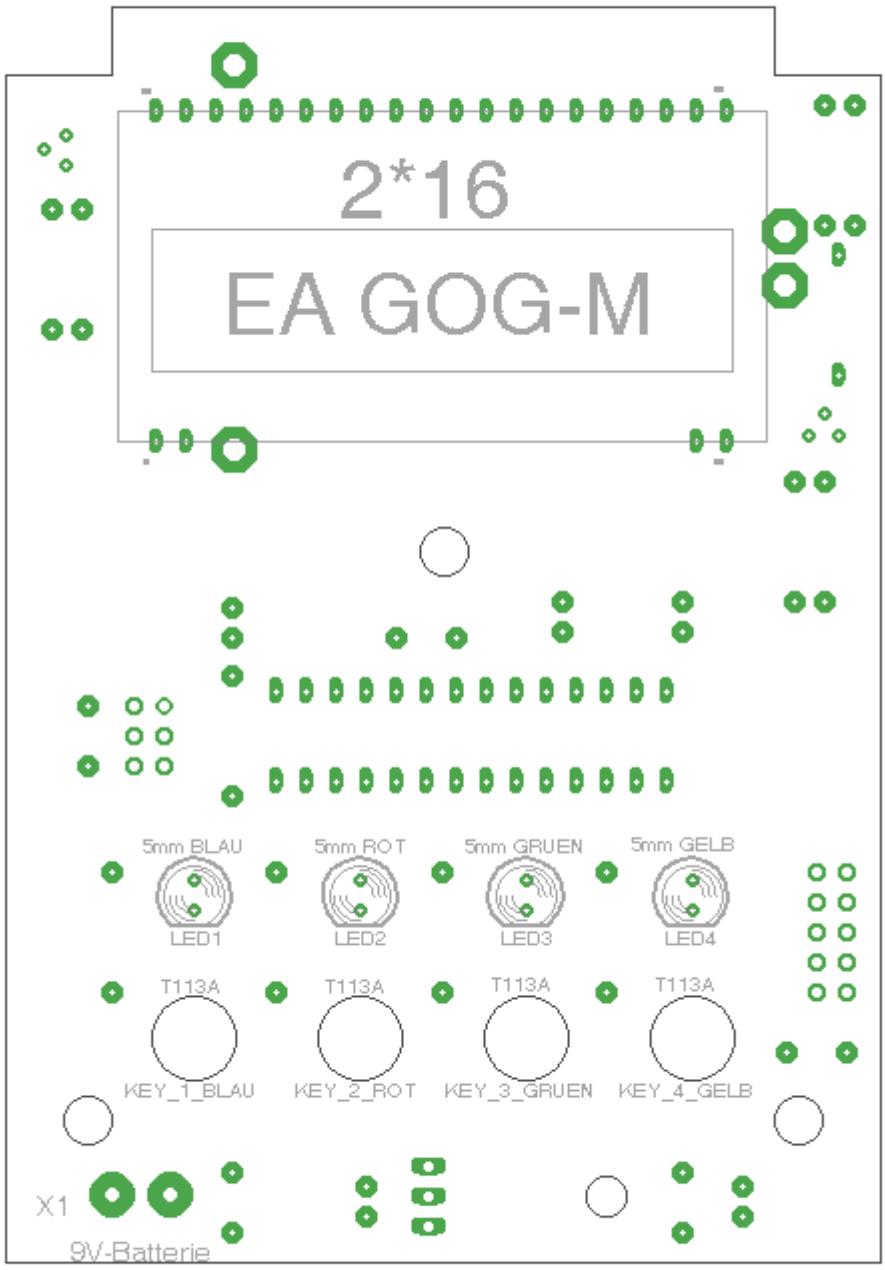


Abbildung 28: PCB Senso – TOP Bestückung

Die BOTTOM-Bestückung

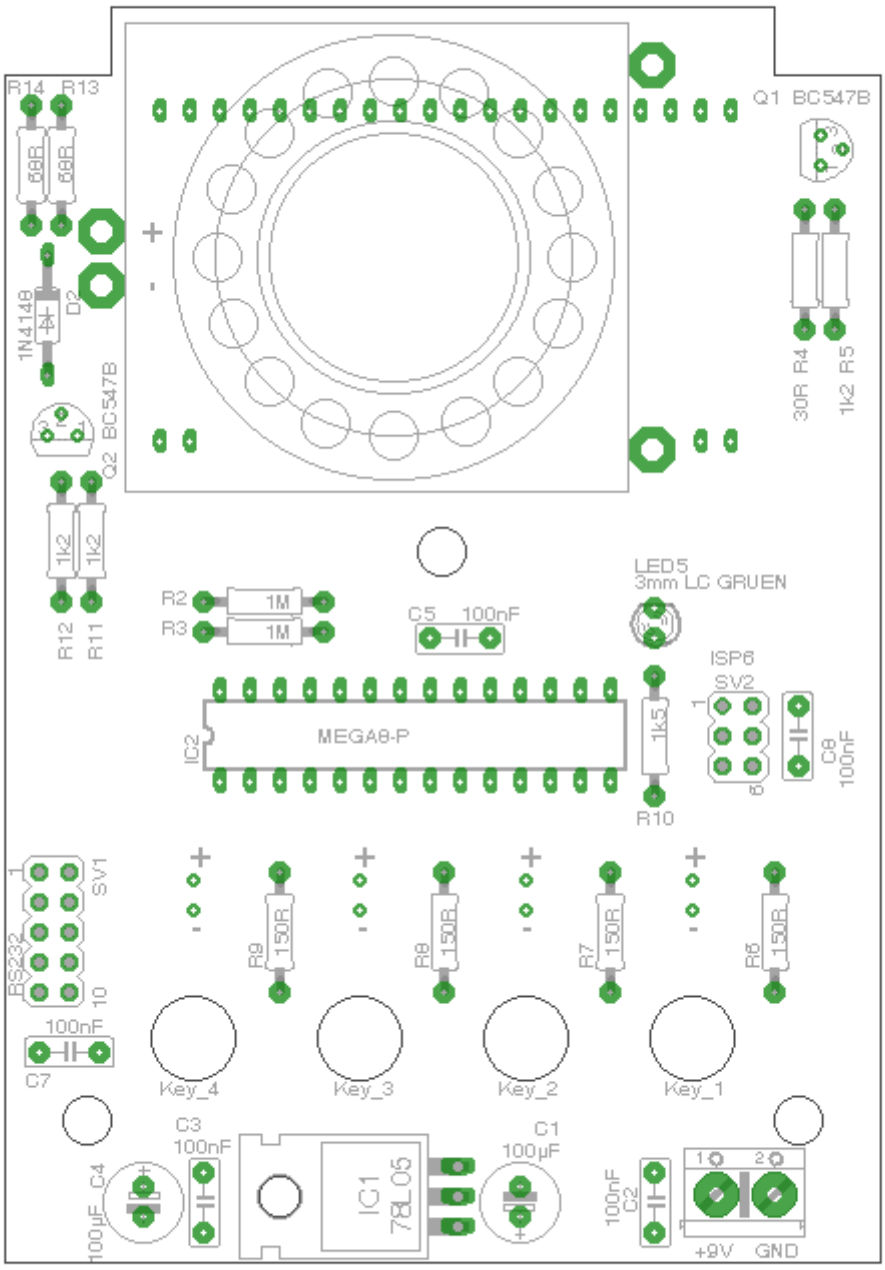


Abbildung 29: PCB Senso – TOP Bestückung

12.2.4 Eagle-BOM

Pos.	Bauteile	Menge	Wert	Device
1.	C2, C3, C5, C7, C8	5	100nF	CAPACITOR050-025X075
2.	C1, C4	2	100µF	POLARIZEDCAPACITORE2.5-7
3.	R6, R7, R8, R9	4	150R	RESISTOR0207/10
4.	R2, R3	2	1M	RESISTOR0207/10
5.	D2	1	1N4148	1N4148
6.	R5, R11, R12	3	1k2	RESISTOR0207/10
7.	R10	1	1k5	RESISTOR0207/10
8.	R4	1	30R	RESISTOR0207/10
9.	LED5	1	3mm LC GRUEN	LED3MM
10.	LED1	1	5mm BLAU	LED5MM
11.	LED4	1	5mm GELB	LED5MM
12.	LED3	1	5mm GRUEN	LED5MM
13.	LED2	1	5mm ROT	LED5MM
14.	R13, R14	2	68R	RESISTOR0207/10
15.	IC1	1	78L05	7805T
16.	X1	1	9V-Batterie	AKL055-02
17.	Q1, Q2	2	BC547B	BC547B
18.	LCD1	1	EA_DOG-M_DISPLAY	EA_DOG-M_DISPLAY
19.	SV2	1	ISP6	MA03-2
20.	SP1	1	LSM-S36K	LSM-S36K
21.	IC2	1	MEGA8-P	MEGA8-P
22.	SV1	1	RS232	MA05-2
23.	KEY_1_BLAU	1	T113A BL	T113A
24.	KEY_2_ROT	1	T113A RT	T113A RT
25.	KEY_3_GRUEN	1	T113A GR	T113A GR
26.	KEY_4_GELB	1	T113A GE	T113A GE

Tabelle 50: Eagle BOM für das Projekt Senso

Folgende Bauteile werden noch außerhalb von Eagle benötigt:

Pos.	Bauteile	Menge	Wert	Device
1.	--	1	Gehäuse	BOPLA BOS 750
2.	--	1	9V Batterieclip	--
3.	--	1	9V Blockbatterie	--
4.	--	4	LED Gehäusedurchführungen 5mm	--
5.	--	1	LED Display Hintergrundbeleuchtung	EA LED55X31W
6.	--	1	Miniatur Wippschalter	--
7.	--	1	Trockenbeutel 5 Gramm	--

Tabelle 51: Weitere Bauteile für das Projekt Senso

12.2.5 Das Board

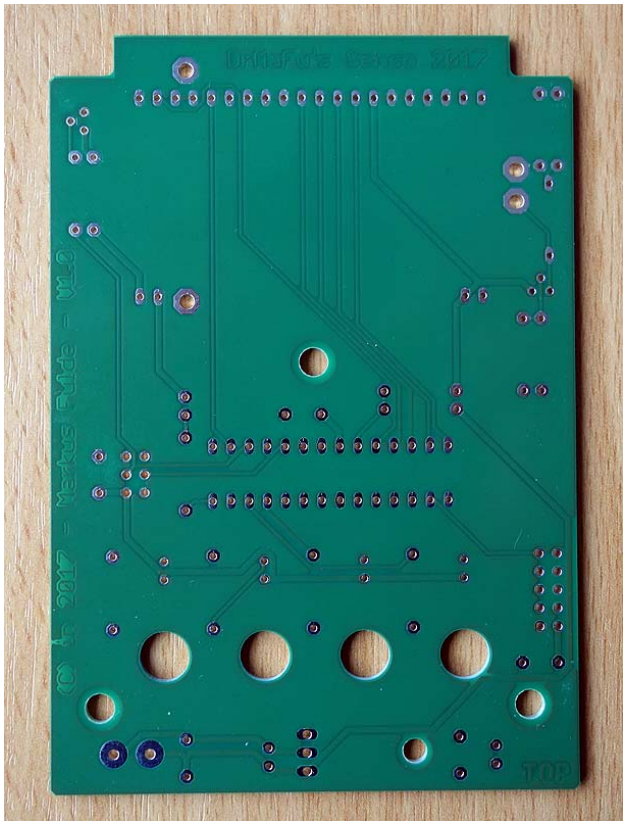


Abbildung 30: PCB Senso TOP

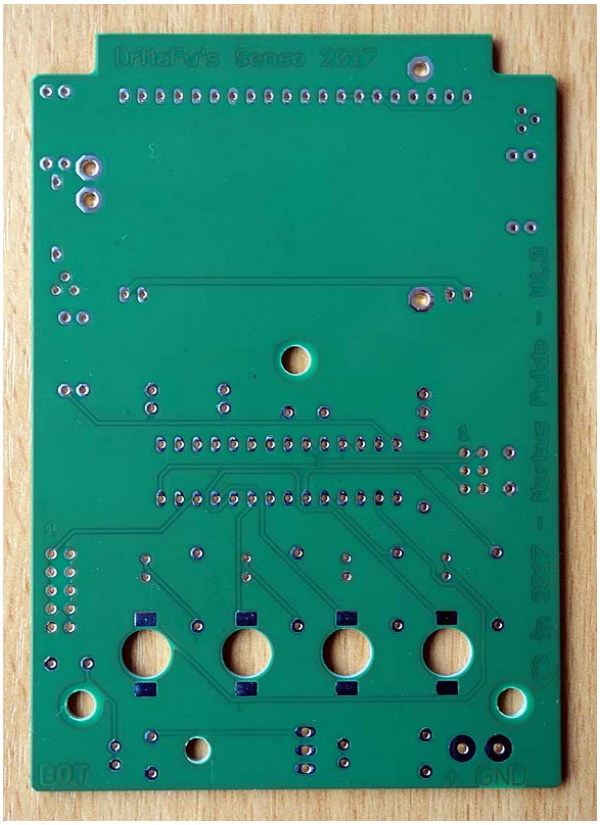


Abbildung 31: PCB Senso BOTTOM

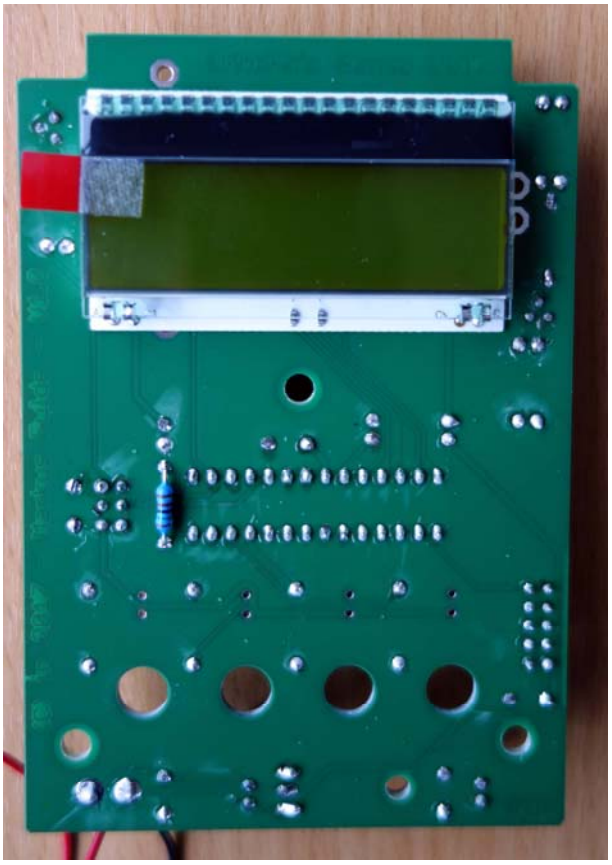


Abbildung 32: PCB TOP fertig bestückt

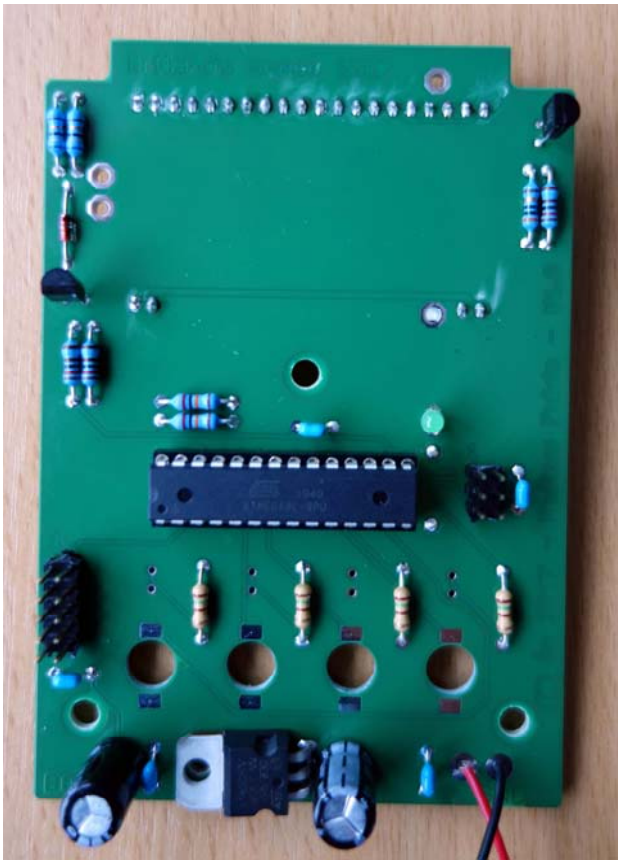


Abbildung 33: PCB BOTTOM fertig bestückt

12.3 Die fertige Hardware

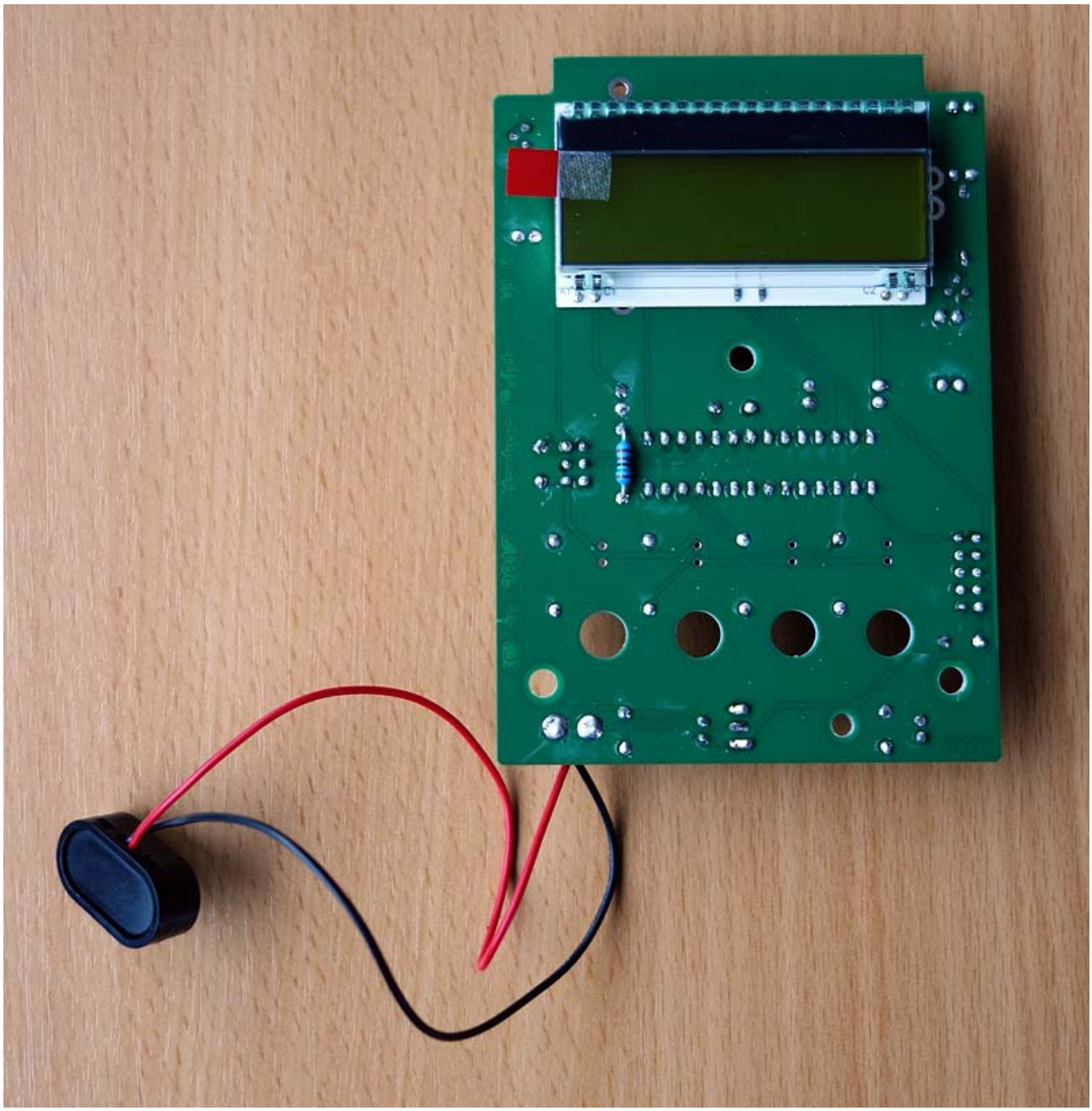


Abbildung 34: Die fertige Platine

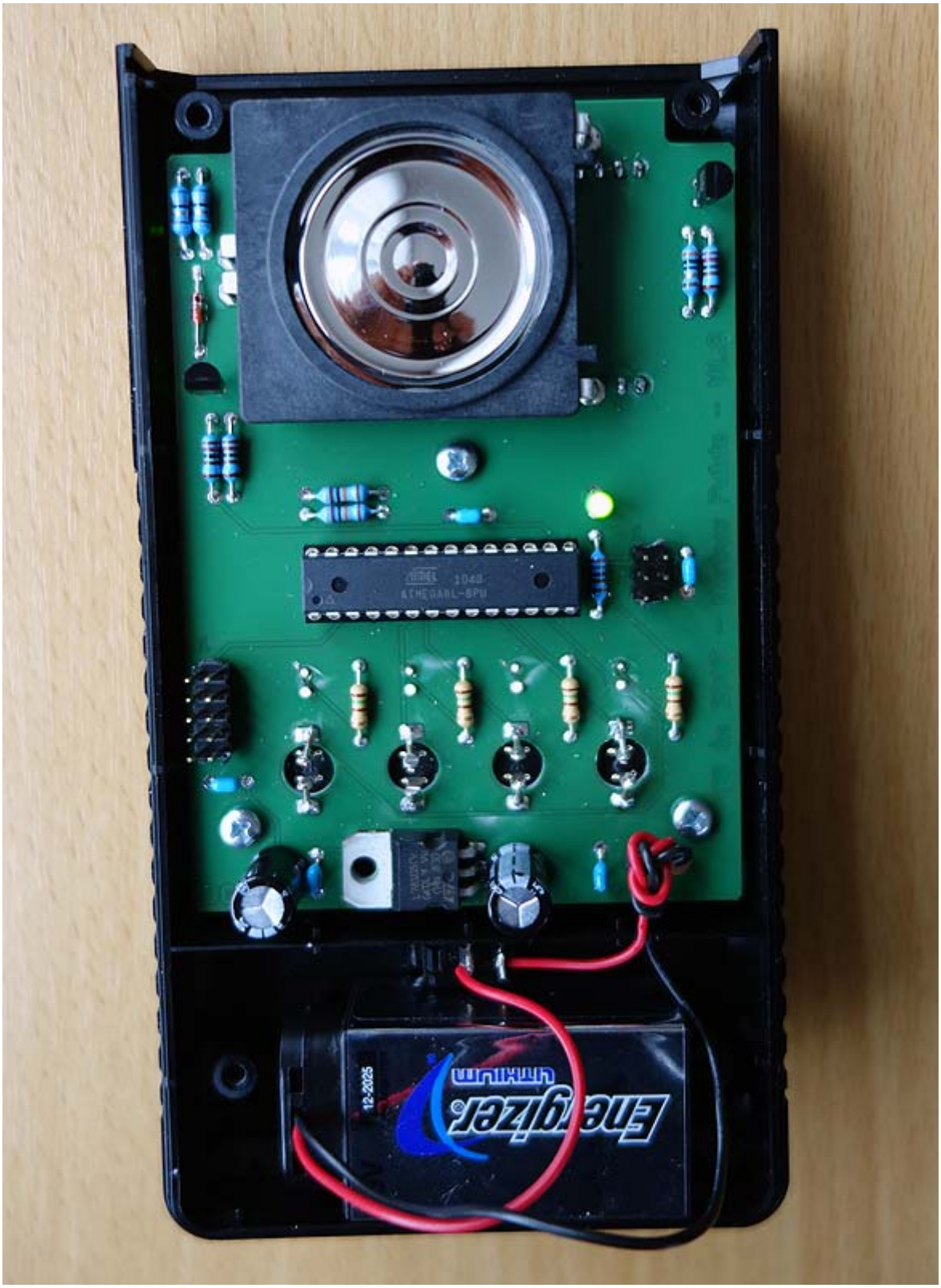


Abbildung 35: Das fertige Gerät im Gehäuse montiert

Abbildung 36: Galgenmännchen in Betrieb

13 Software

13.1 Systemfestlegungen und Definitionen

13.1.1 Timerfestlegungen

Die Timer im Gesamtsystem haben der Priorität nach 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 BetriebsLED.

Timer2:

Mit dem Timer 2 wird die PWM für die Hintergrundbeleuchtung realisiert. Dieser Time rist ein 8 Bit Timer.

13.2 KnowHow: PWM-Signale mit Bascom erzeugen

13.2.1 Grundbegriffe

Bei der Puls-Weiten-Modulation (PWM) wird ein digitales Ausgangssignal erzeugt, dessen Tastverhältnis moduliert wird.
Das Tastverhältnis gibt das Verhältnis der Länge des eingeschalteten Zustands zur Periodendauer an. Dabei bleiben die Frequenz und der Pegel des Signals immer gleich! Es ändert sich nur die Länge von High zu Low.

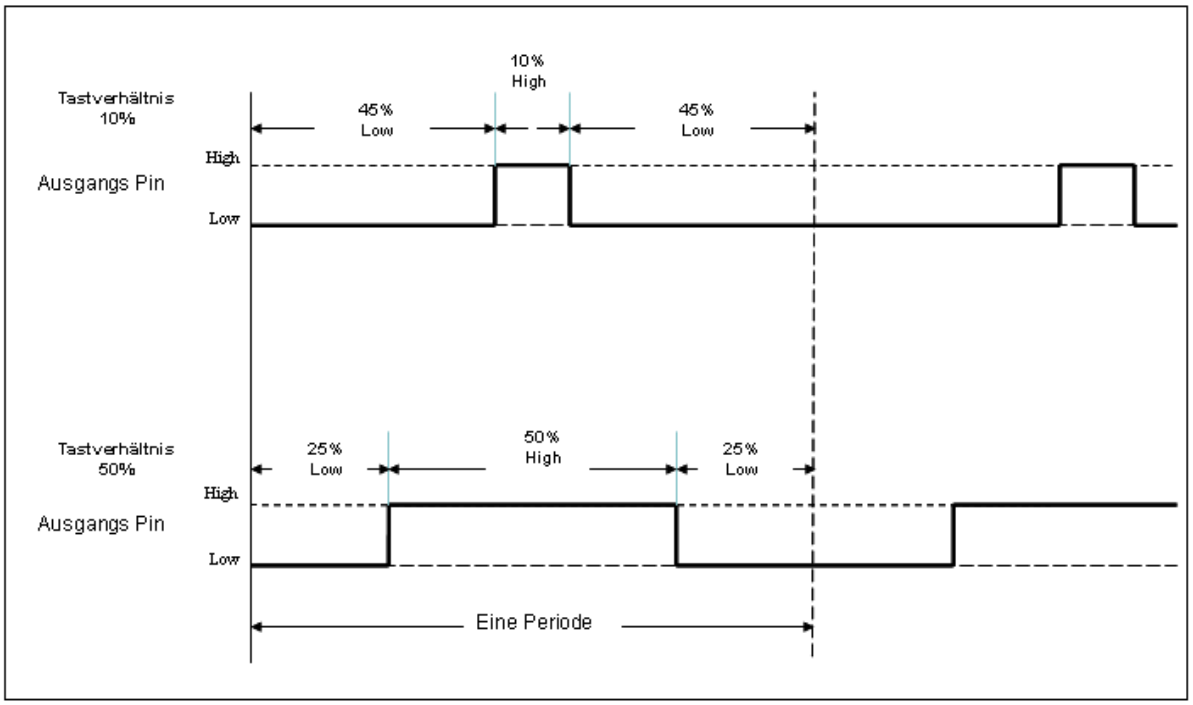


Abbildung 37: PWM Tastverhältnis einmal von 10% und einmal von 50%

Man könnte das in etwa mit einem Wasserhahn vergleichen, den man z.B. alle Minuten betätigt. Wenn man ihn in jeder Minute nur kurz aufdreht und dann gleich wieder zumacht, kommt in Summe nur wenig Wasser raus. Wenn man ihn aber in jeder dieser Minuten länger offen lässt, kommt mehr Wasser raus.

Der Rhythmus bleibt immer gleich, aber es ändert sich in Summe die Wassermenge, die raus kommt.

Mit dem PWM-Signal kann man nun tolle Sachen machen.
Zum Beispiel:

- eine LED (oder über einen Transistor auch eine Lampe) an den PWM-Ausgang anschliessen und mit der Länge des PWM-Signal's die Helligkeit der LED verändern.
- einen Motor in der Geschwindigkeit regeln.
- mittels nachgeschaltetem RC-Filter, welcher das PWM-Signal glättet, kann eine Gleichspannung erzeugt werden die zwischen 0V und 5V geregelt werden kann.

13.2.2 PWM-Arten

Es gibt zwei Arten PWM-Signale in Bascom zu erzeugen:

Software PWM

Vorteile:

- es kann (fast) jeder Ausgabe-Pin des AVR benutzt werden.
- unter zuhilfename eines (freien) Timers können sogar mehrere verschiedene PWM-Signale auf verschiedene Pins erzeugt werden.

Nachteil:

- Etwas grösserer Programmaufwand, da der PortPin per Software verändert werden muss.

Hardware PWM

Vorteile:

- Sehr schnell (Maximal die Quarzfrequenz / Periode)
- unabhängig vom Programmablauf des AVR

Nachteile:

- Je nach AVR können nur bestimmte Timer mit bestimmten Ausgangspins dafür verwendet werden.
- belegt den Timer, der für keine weiteren funktionen verwendet werden kann.

Beim einem ATMega8 stehen drei Hardware-PWM-Ausgänge verteilt auf zwei Timer zur Verfügung.

Mit Timer1 können zwei PWM Signale erzeugt werden (Compare A => OC1A - Pin 15 und Compare B => OC1B - Pin 16).

Die Auflösung kann auf 8, 9 und 10 Bit eingestellt werden, also max. 1024 Abstufungen.

Timer2 kann ein PWM-Signal mit einer Auflösung von 8 Bit erzeugen (Compare Register => OC2 - Pin 17)

13.2.3 PWM-Ablauf

Das folgende Bild zeigt den Ablauf bei Timer1. Als Taktquelle dient die CPU-Frequenz, dessen Frequenz im Prescaler (Vorteiler) nochmal verkleinert werden kann. Je nach eingestelltem Wert in den Output Compare Registern wird der Status des Ausgangs-Pin entsprechend oft umgeschaltet, und erzeugt somit das PWM-Signal.

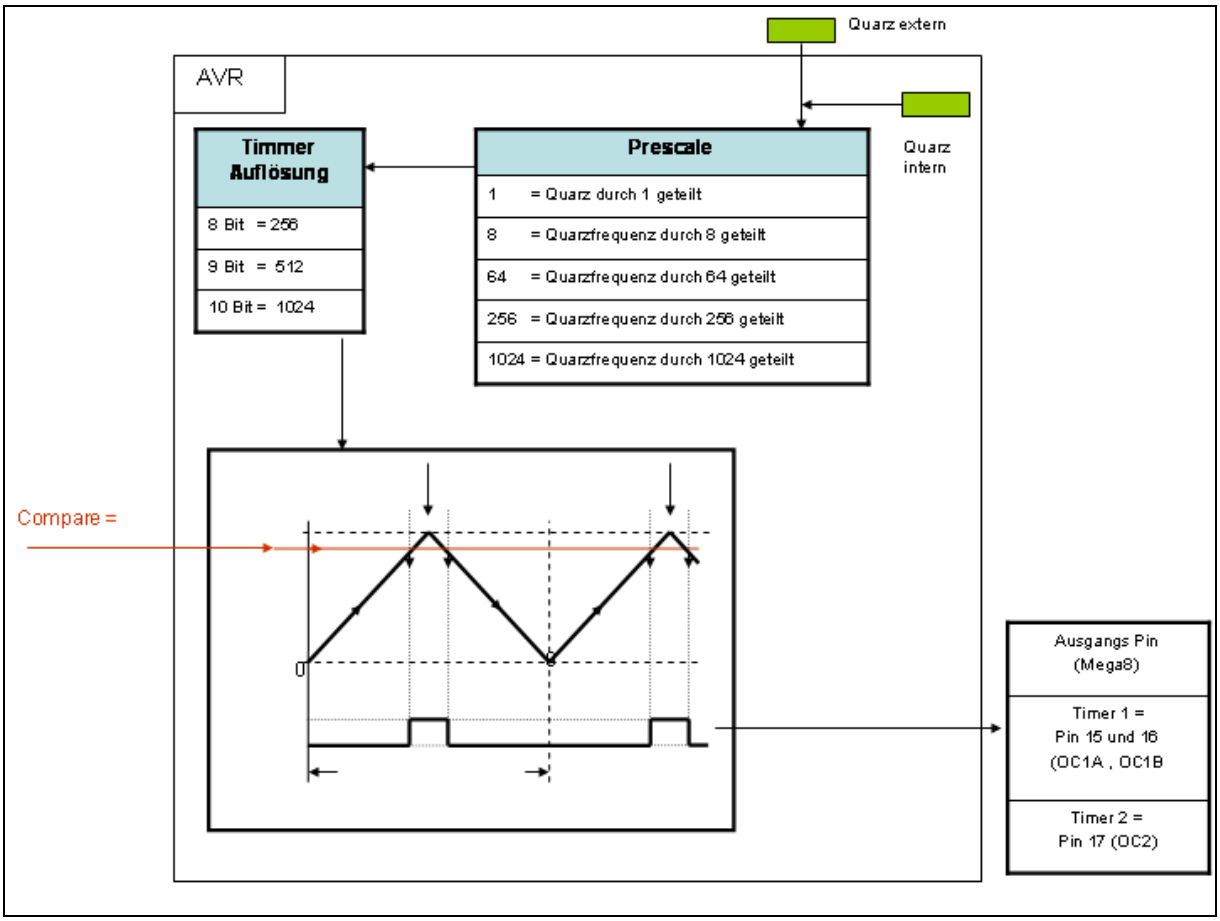


Abbildung 38: PWM Konfiguration des ATmega

13.2.4 Genauere Erklärung

Der Prescaler teilt die Frequenz die vom Quarz kommt! Bei Teilung 1 geht die vollständige Frequenz zum Timer. Bei Teilung 8 nur ein Achtel der Quarzfrequenz. (Also bei 8 MHz Quarz kommen zum Timer dann nur 1 MHz) Bei Teilung 1024 sind es dann z.B. 8 MHz/1024 = 7,8125kHz

Der Timer ist hier der Zähler für die PWM. Er zählt mit der Frequenz, die vom Prescaler kommt, einmal von 0 bis zu der eingestellten Timerauflösung rauf, dann wieder auf 0 zurück. (dann wieder von 0 auf Timerauflösung

u.s.w.) Einmal rauf- und runterzählen, ergibt ein Periode. Die Periode ist gleich die Ausgangsfrequenz des PWM-Signals. Ausgangsfrequenz = (Quarzfrequenz/Prescale) /(Timerauflösung*2)

z.B.: Quarz = 8 MHz ; Prescaler = 1 ; Timer = 8 Bit ergibt: (8000000Hz/1) / (256*2) = 15,625 kHz

oder: Quarz = 8 MHz ; Prescaler = 8 ; Timer = 10 Bit ergibt: (8000000Hz/8) / (1024*2) = 244,14 Hz

Mit dem Compare Register definiert man nun das Tastverhältnis! Überall, wo nun der Timer diese Compare Linie schneidet, schaltet der Ausgang! Beim raufzählen des Timers auf EIN, beim runterzählen auf AUS.

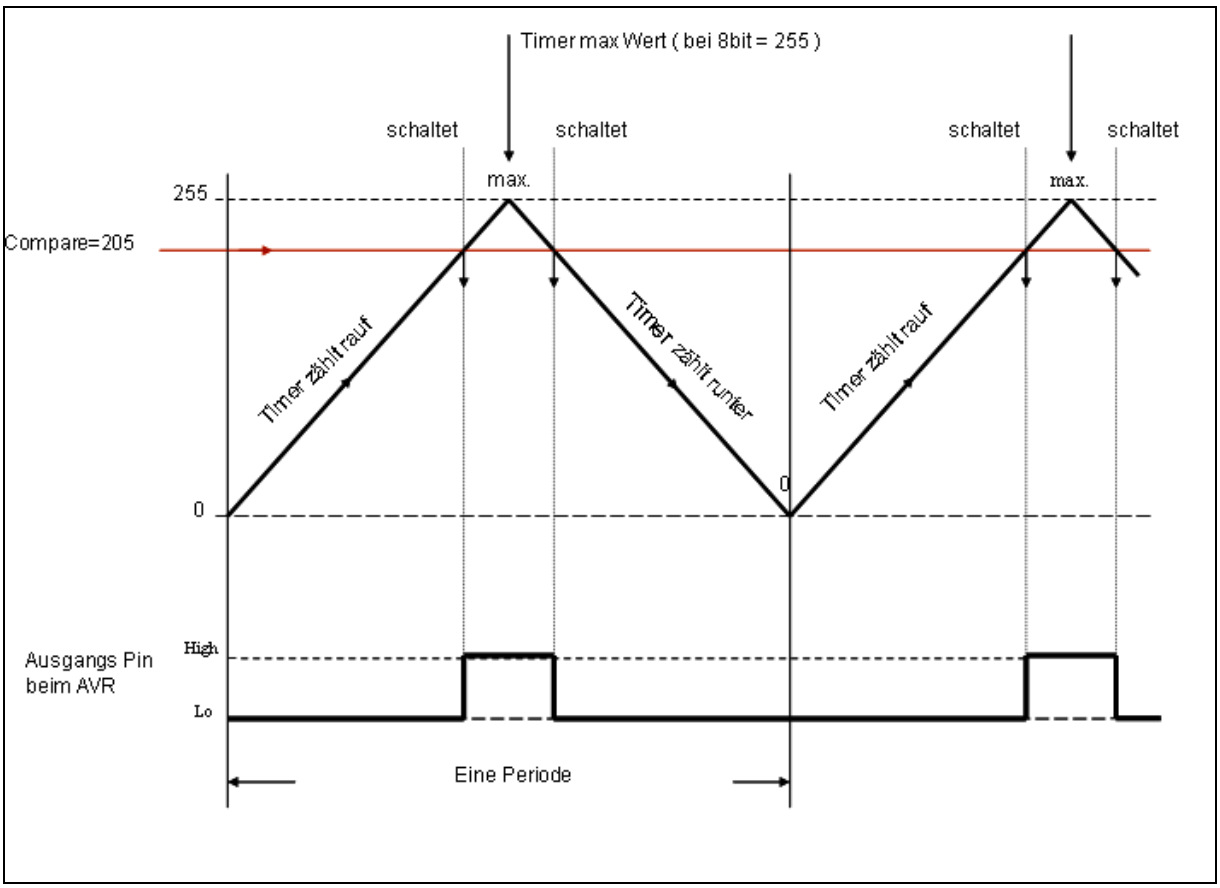


Abbildung 39: PWM mit einem Tastverhältnis 20%

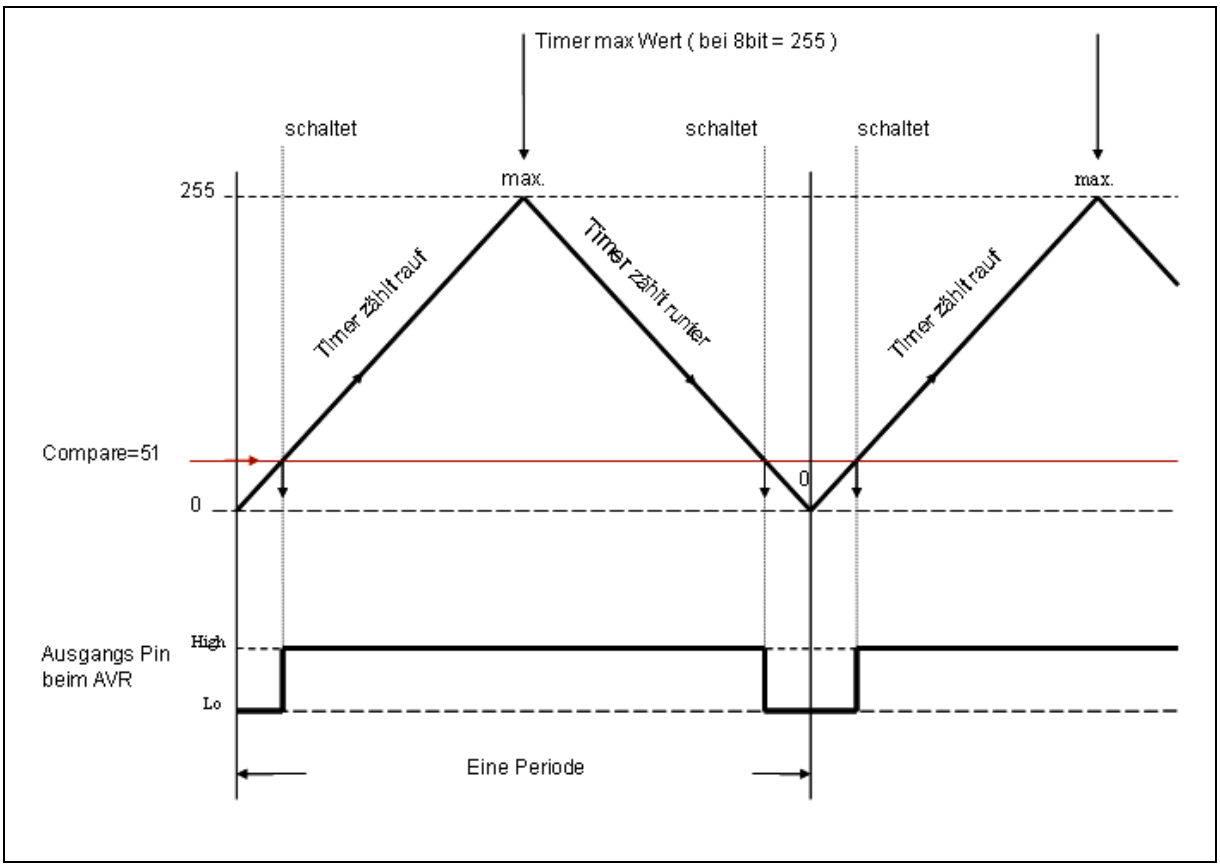


Abbildung 40: PWM mit einem Tastverhältnis von 80%

Hier sieht man, wie die Signale auf einem Oszilloskop ausschauen. Oben das Signal von Pin15 (Compare A), unten das von Pin 16 (Compare B)

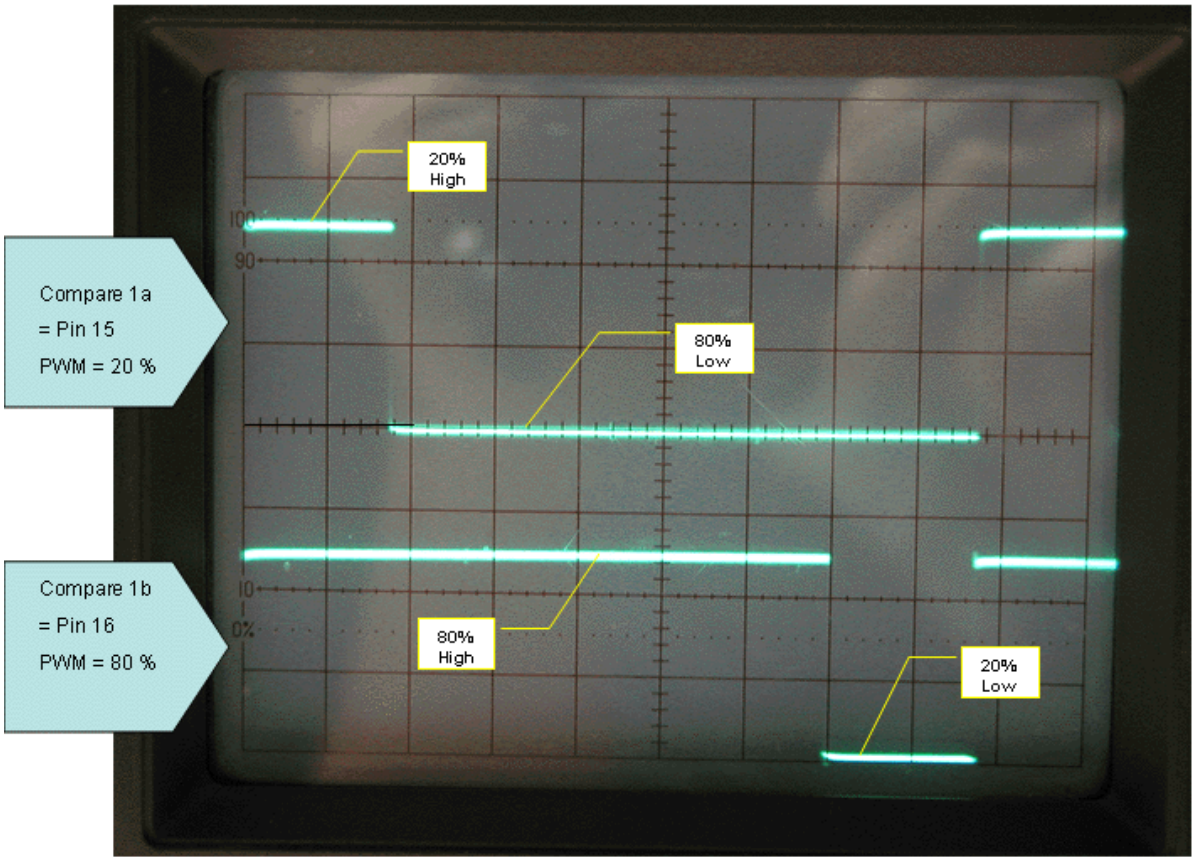


Abbildung 41: Oszillogramm der PWM mit Time 1a und 1b

13.2.5 Grundprogramm

Hier nun ein Grundprogramm für die Ausgabe von zwei PWM Signalen mit dem Timer1
' Hardware PWM mit Timer1

```
$regfile = "m8def.dat"
$crystal = 4000000

Config Portb.1 = Output
Config Portb.2 = Output

Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm =
Clear Up , Prescale = 1

Do
 Compare1a = 205
 Compare1b = 51
Loop

End
```

Erklärung:

```
$regfile = "m8def.dat"
$crystal = 8000000
```

Definiert den Mega8 und den 8MHz Quarz

```
Config Portb.1 = Output
Config Portb.2 = Output
```

Definiert die zwei Ausgänge von Timer1 auf Ausgabe.
Portb.1 = für Compare1a (= Compare A) = Pin 15
Portb.2 = für Compare1b (= Compare B) = Pin 16

```
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Up , Compare B Pwm =
Clear Up , Prescale = 1
```

Config Timer1 = Pwm
Timer1 auf PWM einstellen

Pwm = 8
Timer Auflösung = 8 Bit einstellen

Compare A Pwm = Clear Up
Definiert, wie der Compare A schalten soll. Bei „Clear Up" schaltet der Ausgang beim erreichen des Compare-Wertes zuerst auf High und dann auf Low. Bei „Clear Down", umgekehrt.

Compare B Pwm = Clear Up
Das gleiche noch mal mit Compare B

Prescale = 1
Hier wird der Prescaler auf 1 eingestellt.
(Wert 1 heißt, direkte Frequenz vom Quarz zum Timer.)
Weitere Teilungen, wie z.B.: 8, 64, 256 und 1024 sind möglich.

```
Compare1a = 205
Compare1b = 51
```

Hier kann man nun die Werte für das Tastverhältnis, in dem Register Compare1a und Compare1b übergeben. Oder man kann, statt Compare1a und 1b, auch die Bezeichnungen Pwm1a und Pwm1b verwenden, Bascom nimmt beides.

Mit diesem kurzen Programm, hat man nun zwei PWM Signale erzeugt, bei dem eines ein Tastverhältnis von 20% (Compare1a) und das andere 80 % hat. :-)

13.3 Verwendete SW

Zur Erstellung dieses Projekts kam folgende Software zum Einsatz:

- Workstation DELL XPS420: Betriebsystem Windows 7 Ultimate 64 Bit
- Notebook DELL Inspiron 17R SW: Betriebsystem Windows 8.1 Profesional 64 Bit

- BASCOM-AVR Basic Compiler BASCOM 2.0.7.7
  MCS Electronics
- EAGLE 6.6.0 Standard NON-PROFIT
- ATMEL AVR Studio 4.0 und ATMEL AVR Studio 6.0

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

- ATMEL STK500
- ATMEL ISP-Programmer AVRISP mkII
- Eigenes Prototyping auf Lochraster
- Fertiges PlatinenLayout mit Hilfe von Eagle und Herstellung durch Leiton Berlin

13.4 Bedienungsstruktur und Spielablauf

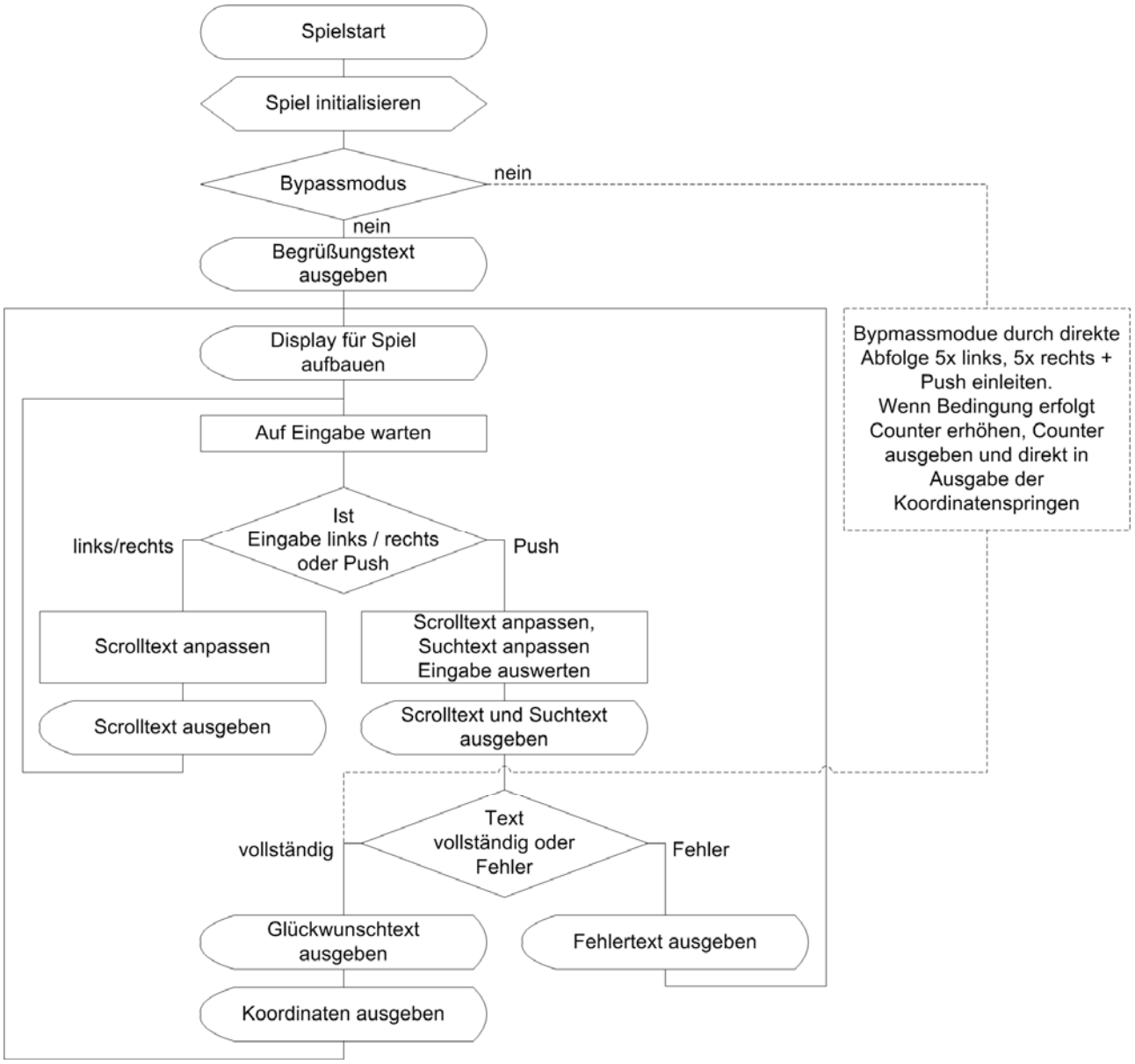


Abbildung 42: Spielablauf

13.5 Kurzanleitung

Einschalten
Das Spiel wird durch den kleinen Hauptschalter auf der Gehäuseoberseite eingeschaltet. Danach läuft die Begrüßung automatisch ab und es erfolgt der automatische Start des Spiels.

Allgemeines
Das Display besteht aus 2 Zeilen a 16 Zeichen. Auf der linken Seite werden mit jeweils 2 Zeichen in Zeile 1 und Zeile 2 der Galgen und ggf. das hängende Galgenmännchen abgebildet.

In Zeile 1 befindet sich außerdem ein Buchstaben-Auswahlbalken über den die gesuchten Buchstaben für den 13-stelligen Begriff ausgewählt werden können.

In Zeile 2 befindet sich das gesuchte 13-stellige Wort das zunächst nur durch Unterstriche verdeckt dargestellt wird. Bei Auswahl der Buchstaben wird nach und nach das Wort – sollte es sich um den richtigen Buchstaben handeln – dargestellt.

Bedienung
- Das Spiel wird mittels Drück- Drehsteller (DDS), auch Inkrementaldrehgeber genannt, gesteuert.
- Durch drehen nach links scrollt der Buchstaben-Auswahlbalken der sich in der 1. Textzeile gefindet nach links.
- Durch drehen nach rechts scrollt der Buchstaben-Auswahlbalken der sich in der 1. Textzeile gefindet nach rechts.
- Durch drücken des Drehknopfes wird der ausgewählte Buchstabe bestätigt.

Spielverlauf
Es geht darum ein 13-stelliges Nomen der deutschen Sprache zu erraten welches zu Beginn beim Spielstart automatisch aus einer Datenbank von insgesamt 100 Wörtern ermittelt wurde.

Bei jedem Spielstart und nach jedem erfolgreichen oder misslungenen Versuch wird ein neues Wort ermittelt.

Ist der ausgewählte Buchstabe korrekt ertönt ein kurzer Doppelpiep und der ausgewählt Buchstabe wird in dem gesuchten aber noch ausgeblendeten Wort in der 2. Textzeile angezeigt. Der Ausgewählte Buchstabe wird dann im Buchstaben-Auswahlbalken durch ein Leerzeichen ersetzt und somit ist er nicht mehr erneut anwählbar.

Ist der ausgewählte Buchstabe falsch so erklingt ein langezogener Piep und das Galgenmännchen beginnt am Galgen zu baumeln. Der Spieler hat so 5 Versuche das Wort richtig herauszufinden.

Werden alle 13 Buchstaben des Wortes richtig erraten ertont ein dreifacher Piep und das Spiel gibt die Geokoordinaten für das Versteck des Caches preis. Hierbei zeigt das Display repetierend die Koordinaten und den Hinweistext an.

Dieser Modus kann durch erneutes Betätigen oder Drehen des DDS verlassen werden. Danach startet das Spiel erneut.

Hängt das Galgenmännchen vollständig am Galgen so ist nach 5 Versuchen das Spiel automatisch beendet. Nach einem kurzen Trosttext startet das Spiel erneut mit einem neuen Begriff.

Bypassmodus
Für Hilfen und Telefonjoker ist im Spiel noch ein Bypassmodus implementiert. Hierzu muss beim Einschalten des Spiels der DDS-Knopf gedrückt werden. Beim ersten Pieps muss dieser losgelassen werden. Nun betätigt man 5x DDS links, 5x DDS rechts und drück sofort nochmals den DDS Knopf.

Danach wird ein Zähler ausgegeben wie oft der Bypassmodus bereits ausgeführt wurde. Dieser Counter wird persistent im EEPROM abgespeichert so dass der Owner Kontroller darüber hat ob jemand den Bypassmodus entdeckt oder weitergegeben hat und wie oft das schon passiert ist.

Nach der Ausgabe des Counter wechselt das Spiel automatisch in die Ausgabe von Koordinaten und Hinweistext wie im Spielverlauf beschrieben.

Weicht man von der Sequenz zum erreichen des Bypassmodus ab so befindet man sich sofort ohne optische Veränderung direkt im Spiel und der Bypassmodus kann nicht aktiviert werden.

13.6 Der Source-Code zum Projekt Senso

```
'################################################################################
'
' DrMaFu_Senso.BAS                                          Stand 22.03.2017
' ----------------                                          (C) Markus Fulde
'
' Version 1.1
'
' Gesamtsoftware Steuerprogramm für das Geocaching-Spiel Senso
'
' Das Projekt verfüg über die folgenden Funktions- und Teilkomponenten:
' - Spannungsversorgung via 9V Batterie und 5V über Festspannungsregler
' - Mikrocontroller ATmega8L (inkl. ISP, RS232 und Reset)
' - LCD-Display 1x8 EA DOGM081L-A von Electronic Assembly
' - Die Helligkeit des Display soll über PWM einstellbar sein
' - Bei Innaktivität soll das Display abgeschaltet werden können
' - Bei Innaktivität soll der ATmega in einen Sleep-Mode gehen
' - Hauptschalter für Spannungsversorgung
' - 4 Tasten (ROT, GELB, GRÜN, BLAU) zur Bedienung des Spiels
' - 4 LED's (ROT, GELB, GRÜN, BLAU) zur Bedienung des Spiels
' - LED-Anzeigen für Betriebsanzeige
' - Gehäuse mit Batteriefach und Displayfenster
' - Soundausgabe über Lautsprecher
'
'################################################################################

'--------------------------------------------------------------------------------
' Allgemeines zum Spiel:
'--------------------------------------------------------------------------------
'
' Die Wikipedia schreibt zum Senso folgendes:
'
' Senso (im englischen Sprachraum Simon) ist ein von Ralph Baer - dem Entwickler
' der 1972 erschienenen ersten TV-Spielekonsole Magnavox Odyssey - und
' Howard J. Morrison entwickeltes und in mehreren Ländern patentier-tes
' elektronisches Spiel, welches 1978 bei Milton Bradley veröffentlicht wurde und
' sich besonders in den 1980er Jahren großer Beliebtheit erfreute. 1979 befand
' es sich auf der Auswahlliste zum Spiel des Jahres. Neuerdings wird das Spiel
' auch in Deutschland als Simon verkauft.
'
' Idee:
' Senso kann allein oder mit mehreren Personen gespielt werden. Das Spiel
' besteht aus vier großen Feldern in den Farben Rot, Blau, Gelb und Grün. Diese
' leuchten abwechselnd auf und geben dabei für jede Farbe einen kurzen,
' individuellen Signalton von sich. Der Spieler muss sich diese Reihenfolge
' merken und nach Abschluss der Vorgabe durch das Spiel wiederholen. Mit jeder
' Runde kommt eine weitere Farb-Ton-Kombination hinzu. Mit steigendem
' Schwierigkeitsgrad leuchten die Felder in schnellerer Reihenfolge auf.
'
' In diesem Geocaching-Projekt Senso wird das Spielkonzept dazu benutzt, eine
' kleine Elektronik zu entwickeln welche im Gelände versteckt wird, die nach
' jedem erfolgreichem Spiel die finalen Geokoordinaten der finalen Dose anzeigt.
' Es muss eine Reihenfolge von insgesamt 10 Vorgaben richtig absolviert werden,
' damit das Spiel sein Ge-heimnis und damit die finalen Koordinaten Preis gibt.
'
' Im Wesentlichen geht es bei diesem Projekt aber auch umn den Ersatz des
' originalen Geocaching-Spiels das schon seit vielen Jahren Vorort liegt und
' durch Vandalismus zerstört wurde. Wasser und Elektronik vertragen sich einfach'
' zu schlecht.
'
' Anmerkung:
' Der Source-Code für Senso entsteht mit diesem Projekt neu. Es werden neue
' Funktionen und eine andere Display-Technologier wie im Original eingesetzt.
' Teile des Source-Codes von DF1SAM wurden dabei übernommen.
'
' --------------------------------------------------------------------------------


'--------------------------------------------------------------------------------
' Compilerinstruktionen und Compilerdirektiven
'--------------------------------------------------------------------------------
$regfile = "m8def.dat"                                  ' Definitionsdatei für ATmega128 laden
$crystal = 8000000                                      ' Quarzfrequenz für 16 MHz festlegen

$hwstack = 128                                          ' HW-Stack auf 128 Bytes erweitern
$swstack = 64                                           ' SW-Stack auf 64 Bytes erweitern
$framesize = 80                                         ' Framesize auf 80 Byte festlegen
```

```
$baud = 19200                                              ' Baudrate für RS232 Traceausgabe defi-
nieren


'--------------------------------------------------------------------------------
' Allgemeine Zusatzinformatonen zu Programmbeginn
'--------------------------------------------------------------------------------

' -- Fuses aus ATmega8 ausgelesen ------------------------------------------------
'   HIGH 0xD9
'   LOW  0xE4 = Int. RC Osc. 8 MHz; Start-up time: 6 CK + 64 ms


' -- Allgemeine Informationen zur Sounderzeugung ----------------------------
'     | |C||D| | |F||G||A| | |C||D| | |F||G||A| | |C||D| | |F||G||A| |
'     | |1||1| | |1||1||1| | |2||2| | |2||2||2| | |3||3| | |3||3||3| | |
'     | |#||#| | |#||#||#| | |#||#| | |#||#||#| | |#||#| | |#||#||#| | |
'     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
'     |C1|D1|E1|F1|G1|A1|H1|C2|D2|E2|F2|G2|A2|H2|C3|D3|E3|F3|G3|A3|H3| |
'     |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__| |
'
' (1/1)   - 2 sec
' (1/2)   - 1 sec
' (1/4)   - 0,5 sec
' (1/8)   - 0,25 sec
' (1/16)  - 0,125 sec
' (1/32)  - 0,0625 sec
'
' *****   FORMAT TO FUNCTIONS SOUND  *****
'
' Sound Speaker , Pulses , Periods
' Speaker - port for generations of sound
' Periods - sound frequency (1-65535)
' Pulses  - duration of sound (1-65535)
'
' Periods = Abc[F_crystal / (k * F_nota)]
' Pulses  = Abc[(T_period * F_crystal) / (k * Periods)]
'
' F_crystal - clockrate of controller, Hz
' F_nota    - frequency a notes, Hz
' k = 12    - amount of tacts, for which is formed one period of sound
' T_period  - duration of sounding a notes, sec
' Abc       - function of truncation whole number
'
'================================================================================
' Table of values Pulses,Periods for 3 octaves at frequency of quartz 8000000 Hz
'================================================================================
'Note|Freqvency|Periods|Pulses 1/1|Pulses 1/2|Pulses 1/4|Pulses 1/8|Pulses 1/16|
'================================================================================
' C1 |  261,63 |  2548 |    523   |    262   |    131   |    65    |     33    |
'Cis1|  277,18 |  2405 |    554   |    277   |    139   |    69    |     35    |
' D1 |  293,66 |  2270 |    587   |    294   |    147   |    73    |     37    |
'Dis1|  311,13 |  2143 |    622   |    311   |    156   |    78    |     39    |
' E1 |  329,63 |  2022 |    659   |    330   |    165   |    82    |     41    |
' F1 |  349,23 |  1909 |    698   |    349   |    175   |    87    |     44    |
'Fis1|  369,99 |  1802 |    740   |    370   |    185   |    92    |     46    |
' G1 |  392,00 |  1701 |    784   |    392   |    196   |    98    |     49    |
'Gis1|  415,30 |  1605 |    831   |    415   |    208   |   104    |     52    |
' A1 |  440,00 |  1515 |    880   |    440   |    220   |   110    |     55    |
'Ais1|  466,16 |  1430 |    932   |    466   |    233   |   117    |     58    |
' B1 |  493,88 |  1350 |    988   |    494   |    247   |   124    |     62    |
'================================================================================
' C2 |  523,25 |  1274 |   1047   |    523   |    262   |   131    |     65    |
'Cis2|  554,36 |  1203 |   1109   |    554   |    277   |   139    |     69    |
' D2 |  587,32 |  1135 |   1175   |    587   |    294   |   147    |     73    |
'Dis2|  622,26 |  1071 |   1245   |    622   |    311   |   156    |     78    |
' E2 |  659,26 |  1011 |   1319   |    659   |    330   |   165    |     82    |
' F2 |  698,46 |   954 |   1397   |    698   |    349   |   175    |     87    |
'Fis2|  739,98 |   901 |   1480   |    740   |    370   |   185    |     92    |
' G2 |  784,00 |   850 |   1568   |    784   |    392   |   196    |     98    |
'Gis2|  830,60 |   803 |   1661   |    831   |    415   |   208    |    104    |
' A2 |  880,00 |   758 |   1720   |    880   |    440   |   220    |    110    |
'Ais2|  932,32 |   715 |   1865   |    932   |    466   |   233    |    117    |
' B2 |  987,75 |   675 |   1976   |    988   |    494   |   247    |    124    |
'================================================================================
' C3 | 1046,50 |   637 |   2093   |   1047   |    523   |   262    |    131    |
'Cis3| 1108,70 |   601 |   2218   |   1109   |    554   |   277    |    139    |
' D3 | 1174,60 |   566 |   2350   |   1175   |    587   |   294    |    147    |
'Dis3| 1244,50 |   536 |   2490   |   1245   |    622   |   311    |    156    |
```

```
' E3 | 1318,50 |   483 |   2638   |   1319   |    659   |   330    |    165    |
' F3 | 1396,90 |   477 |   2794   |   1397   |    698   |   349    |    175    |
'Fis3| 1480,00 |   450 |   2960   |   1480   |    740   |   370    |    185    |
' G3 | 1568,00 |   425 |   3136   |   1568   |    784   |   392    |    196    |
'Gis3| 1661,20 |   401 |   3322   |   1661   |    831   |   415    |    208    |
' A3 | 1720,00 |   388 |   3440   |   1720   |    880   |   440    |    220    |
'Ais3| 1864,60 |   358 |   3730   |   1865   |    932   |   466    |    233    |
' B3 | 1975,50 |   337 |   3952   |   1976   |    988   |   494    |    247    |
'================================================================================


'--------------------------------------------------------------------------------
' Anmerkungen zum Programm:
'--------------------------------------------------------------------------------

' [1] Zufallszahlengenerator in BASCOM
' Die Rnd() Funktion Arbeitet Nicht Wirklich Zufällig. Hierzu Sagt Mcv Folgendes:
' The Rnd() Function Returns An Integer / Word And Needs An Internal Storage Of 2 Bytes.
' (___rseed). Each New Call To Rnd() Will Give A New Positive Random Number.
' notice Notice that it is a software based generated number.
' And each time you will restart your program the same sequence will be created.
'
' You can use a different SEED value by dimensioning and assigning ___RSEED yourself:
' Dim ___rseed as word : ___rseed = 10234
' Dim I as word : I = rnd(10)
'
' When your application uses a timer you can assign ___RSEED with the timer value.
' This will give a better random number.
'
' Aus diesem Grund wird hier in der Software beim Auslesen der Batteriespannung
' die SEED-Variable des Zufallszahlengenerators neu gesetzt um mehr
' Zufälligkeit zu erzeugen.
'

' [2] Nach umfangreichen Tests stell sich nach wie vor heraus, dass die
' Zufallszahl bzw. das Pattern nicht ausreichend zufällig ist und sich Reihen-
' folgen dauerhaft / oft wiederholen. Aus diesem Grund wurden die Pattern für
' die Sensowerte offline erzeugt und im Data-Bereich gehalten.
' Mit einer Zufallszahl von 1 -100 wird nun willkürlich ein Pattern aus dem
' Databereich geladen.
' Eine andere Möglichkeit hätte - wie in der originalen Senso-Implementierung
' dadurch bestanden, die Zeit welche zwischen 2 Tastendrücken verstreicht in
' einen Zufallswert umzurechnen. Dieses Verfahren kam hier nicht zum Einsatz
' da ich von Anfang an die Kombinationen berechnet haben wollte.

' [3] Die originale Senso-Implementierung basiert auf vielen GOTO's
' Da ich ein Gener von GOTO's bin wurde die Implementierung komplett über-
' arbeitet und auf "saubere" Programmabläufe umgestellt.


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

' ----- LED's -----
Alive_led_pin Alias Pinb.0                                  ' GPIO für Alive-LED (für DDR oder In-
put)
Alive_led Alias Portb.0                                     ' GPIO für Alive-LED für Output oder
Pullup)


' Oberste LED im Kreutz ist die Blaue LED und zugleich die Start-LED bei Standby

Led_blau_pin Alias Pinb.6                                   ' LED1: GPIO für blaue LED (für DDR oder
Input)
Led_blau Alias Portb.6                                      ' LED1: GPIO für blaue LED (für Output
oder Pullup)

' im Uhrzeigersinn folgen dann

Led_rot_pin Alias Pinb.7                                    ' LED2: GPIO für rote LED (für DDR oder
Input)
Led_rot Alias Portb.7                                       ' LED2: GPIO für rote LED (für Output
oder Pullup)

Led_gruen_pin Alias Pind.5                                  ' LED3: GPIO für gruene LED (für DDR
oder Input)
Led_gruen Alias Portd.5                                     ' LED3: GPIO für gruene LED (für Output
oder Pullup)
```

```
Led_gelb_pin Alias Pind.6                                   ' LED4: GPIO für gelbe LED (für DDR oder
Input)
Led_gelb Alias Portd.6                                      ' LED4: GPIO für gelbe LED (für Output
oder Pullup)


' ----- LCD-Display -----
' LCD-Display
Lcd_bit_0 Alias Portc.2                                     ' GPIO für LCD Bit 0
Lcd_bit_1 Alias Portc.3                                     ' GPIO für LCD Bit 1
Lcd_bit_2 Alias Portc.4                                     ' GPIO für LCD Bit 2
Lcd_bit_3 Alias Portc.5                                     ' GPIO für LCD Bit 3
Lcd_e Alias Portb.2                                         ' GPIO für LCD E
Lcd_rs Alias Portc.1                                        ' GPIO für LCD RS

' ----- Key / Tasten -----

' BLAU
Key_taster1_pin Alias Pind.3                                ' KEY1: GPIO für Taste 1 BLAU (für DDR
oder Input)
Key_taster1 Alias Portd.3                                   ' KEY1: GPIO für Taste 1 BLAU (für Out-
put oder Pullup)

' ROT
Key_taster2_pin Alias Pind.2                                ' KEY2: GPIO für Taste 2 ROT (für DDR
oder Input)
Key_taster2 Alias Portd.2                                   ' KEY2: GPIO für Taste 2 ROT (für Output
oder Pullup)

' GRÜN
Key_taster3_pin Alias Pind.4                                ' KEY3: GPIO für Taste 3 GRÜN (für DDR
oder Input)
Key_taster3 Alias Portd.4                                   ' KEY3: GPIO für Taste 3 GRÜN (für Out-
put oder Pullup)

' GELB
Key_taster4_pin Alias Pind.7                                ' KEY4: GPIO für Taste 4 GELB (für DDR
oder Input)
Key_taster4 Alias Portd.7                                   ' KEY4: GPIO für Taste 4 GELB (für Out-
put oder Pullup)


' ----- Sound -----
Speaker_pin Alias Pinb.1                                    ' GPIO für Sound-Ausgabe (für DDR oder
Input)
Speaker Alias Portb.1                                       ' GPIO für Sound-Ausgabe (für Output
oder Pullup)



'--------------------------------------------------------------------------------
' Definition von Konstaten
'--------------------------------------------------------------------------------

' ----- Für Testumgebung bzw. Traceausgaben -----
Const Main_testmodus = 0                                    ' Flag für Testmodus Allgemeinsystem
Const Batterie_testmodus = 0                                ' Flag für Batteriemanagement
Const Lcd_testmodus = 0                                     ' Flag für Testmodus rund um das LCD-
Display
Const Game_testmodus = 0                                    ' Flag für Testmodus der Spielsteuerung
Const Bypass_testmodus = 0                                  ' Flag für Testmodus des Bypass-Modes
Const Eeprom_testmodus = 0                                  ' Flag für Testmodus der Eeprom-Daten
Const Key_testmodus = 0                                     ' Flag für Testmodus der Taster


' ----- Allgemeine Systemkonstanten -----

' Tatsächliches Allgemeines
Const Led_aus = 0                                           ' LED Konstake für LED aus im echten
Spiel
Const Led_ein = 1                                           ' LED Konstake für LED ein im echten
Spiel

' Const Led_aus = 1                                           ' Achtung !! bei STK500 ist Logik
gedreht!!
' Const Led_ein = 0                                           ' Achtung !! bei STK500 ist Logik
gedreht!!

Const False = 0                                             ' Boolsche Variable für Falsch
Const True = 1                                              ' Boolsche Variable für Richtig
```

```
Const Pullup_aus = 0                                    ' Konstante für PullUp AUS
Const Pullup_ein = 1                                    ' Konstante für PullUp EIN

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


' ----- Tasten -----

' Konstanten für Überprüfung der Tasten
'                     G  GBR
'                     E  NLT
'                     |  |||
'                     76543210
Const Key_all_key = &B10011100
Const Key_ge__key = &B10000000
Const Key_gn__key = &B00010000
Const Key_bl__key = &B00001000
Const Key_rt__key = &B00000100

Const Key_all_key_not = &B01100011
Const Key_ge__key_not = &B01111111
Const Key_gn__key_not = &B11101111
Const Key_bl__key_not = &B11110111
Const Key_rt__key_not = &B11111011

Const Key_all_key_pressed = &B00000000
Const Key_masked_key_aktive = &B00000000

Const Key_entprellzeit = 20                             ' Entprellzeit für Tasten in ms

Const Key_pressed = 0                                   ' Signalzustand für Taster gedrückt

Const Key_bl_num = 1                                    ' Nummerierung der Tasten - Taste 1 ist
BLAU
Const Key_rt_num = 2                                    ' Nummerierung der Tasten - Taste 2 ist
ROT
Const Key_gn_num = 3                                    ' Nummerierung der Tasten - Taste 3 ist
GRUEN
Const Key_ge_num = 4                                    ' Nummerierung der Tasten - Taste 4 ist
GELB


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

' Anmerkung: Wert von 35 ist der beste Wert welcher durch Versuche ermittelt wurde!!
Const Lcd_kontrast_default = 35                         ' LCD-Default-Kontrast

' Anmerkung: Wert von 70 ist der beste Wert welcher durch Versuche ermittelt wurde!!
Const Lcd_helligkeit_default = 70                       ' LCD-Default-Helligkeit

Const Lcd_helligkeit_aus = 0                            ' Wert für Display aus
Const Lcd_abschaltzeit = 10                             ' Defaultwert auf 10 Sekunden festgelegt

Const Lcd_beleuchtung_status_ein = 1                    ' Statuswert für LCD Beleuchtung ein
Const Lcd_beleuchtung_status_aus = 0                    ' Statuswert für LCD Beleuchtung aus

Const Lcd_shift_speed = 500                             ' Scrollgeschwindigkeit für Text-Shift
Const Lcd_countdown_speed = 250                         ' Countdowngeschwindigkeit nach Spiel-
start

' ----- EEPROM -----
Const Ee_default_version_value = 1                      ' Version für persistente Daten


' ----- SOUND -----
Const Sound_ton1 = 637                                  ' Tonlage BLAU Ton
Const Sound_dauer1 = 262                                ' Tondauer BLAU Ton

Const Sound_ton2 = 483                                  ' Tonlage ROT Ton
Const Sound_dauer2 = 330                                ' Tondauer ROT Ton

Const Sound_ton3 = 425                                  ' Tonlage GRÜN Ton
Const Sound_dauer3 = 392                                ' Tondauer GRÜN Ton

Const Sound_ton4 = 337                                  ' Tonlage GELB Ton
Const Sound_dauer4 = 494                                ' Tondauer GELB Ton

Const Sound_falschton = 715                             ' Tonlage FALSCH Ton
```

```
Const Sound_falschdauer = 233                                   ' Tondauer FALSCH Ton


' ----- ADC: Batteriespannung -----
Const Adc_channel_number = 0                                    ' Auswahl von ADC0 an dem Batteriespan-
nung gemessen werden soll

' ----- Spielsteuerung ----

Const Game_anzahl_text = 100                                    ' Anzahl der Text im DATA Bereich

Const Game_mode_idle = 0                                        ' Konstante für Spielsteuerung - Idle
Mode
Const Game_mode_start = 1                                       ' Konstante für Spielsteuerung - Start
Mode
Const Game_mode_spiel = 2                                       ' Konstante für Spielsteuerung - Respon-
se Mode
Const Game_mode_aktion = 3                                      ' Konstante für Spielsteuerung - Request
Mode

Const Game_play_sensolevel_speed = 250                          ' Geschwindigkeit mit der Sensolevel
abgespielt werden woll in msec
Const Game_play_winsound_speed = 30                             ' Geschwindigkeit des Abspielens es
Winsounds in msec

Const Game_play_pause_delay = 500                               ' Delaygeschwindigkeit während Spiel
zwischen Tasten und Feedback  in msec
Const Game_play_response_delay = 100                            ' Delayzeit nach Request-Ausgabe bis
Response in msec

Const Game_play_loop_counter = 200                              ' Schleifenanzahl bildet Timeout-Zeit
'Const Game_play_loop_counter = 800                              ' Schleifenanzahl bildet Timeout-Zeit -
for debug only

Const Game_play_timeout_delay = 10                              ' Delay in Tastenabfrage für Timeout in
msex
Const Game_timeout_delay = 2                                    ' Delay wie lange Timeout angezeigt
werden soll in sec

Const Game_show_finaly_delay = 2                                ' Delayzeit für Ausgabe der Finalen
Koordinaten in sec

Const Game_play_level_pause = 1000                              ' Delayzeit zwischen erolgreicher rich-
tiger Eingabe und nextem Level in msec

' ----- LED's -----
Const Led_speed_testroutine = 100                               ' Geschwindigkeit der LED-Testroutine in
msec

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

' ----- Temporäre Hilfsvariablen -----
Dim Temp_index As Byte                                          ' Temporäre Zahlvariable
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 Variable 3

Dim Temp_word_1 As Word                                         ' Temporäre Word Variable 1
Dim Temp_word_2 As Word                                         ' Temporäre Word Variable 2
Dim Temp_word_3 As Word                                         ' Temporäre Word Variable 2

Dim Temp_string_1 As String * 8                                 ' Temporäre String Variable mit 8
Zeichen + Delimiter



' ----- System -----
Dim Sectick_counter As Word                                     ' Globaler Sekundenzähler


' ----- Arbeitsvariablen für Spieleablauf -----
Dim Game_ablaufstatus As Byte                                   ' Arbeitsvariable für System-Status


' ----- Variablen für LCD-Display -----
Dim Lcd_kontrastwert As Byte                                    ' Arbeitsvariable für Kontrastwert
Dim Lcd_helligkeit As Byte                                      ' Arbeitsvariable für Displayhelligkeit
Dim Lcd_abschaltcounter As Word                                 ' Variable für Abschaltzeitpunkt
```

```
Dim Lcd_beleuchtung_status As Byte                          ' Arbeitsvariable für Zustand der
Hintergrundbeleuchtugn


' ----- EEPROM -----
Dim Ee_data_version_value As Eram Byte At &H0000            ' EEPROM Datenstruktur internes EEPROM -
Init-Index / Version
Dim Ee_data_bypass_counter As Eram Word At &H0001           ' EEPROM Datenstruktur internes EEPROM -
Counter
Dim Ee_version_value As Byte                                ' BYTE Arbeitsvariable für Daten-Version
Dim Ee_bypass_counter As Word                               ' WORD Arbeitsvariable für Bypasscounter


' ----- GAME: Spielsteuerung -----
Dim Game_senso(10) As Byte                                  ' Matrix für Sensowerte
Dim Game_sensolevel As Byte                                 ' Index-Counter für Spielschleife


' ----- ADC: Batteriespannung -----
' Batteriespannung wird zwar nicht überwacht aber für ___seed verwendet
Dim Adc_channel As Byte                                     ' Arbeitsvariable für ADC-Kanalauswahl


'-------------------------------------------------------------------------------
' Prototyping
'-------------------------------------------------------------------------------

' ----- LCD und Print -----
Declare Sub Lcd_print_level(byval Value As Byte)            ' Funktion zum Schreiben der Rundezahl
auf dem Display


' ----- KEY -----
Declare Function Key_check_all_key() As Byte                ' Prototyp für Funktion zur Prüfung
einer Taste
Declare Function Key_check_any_key() As Byte                ' Prototyp für Funktion zur Prüfung
beliebiger Tasten


' ----- GAME -----
Declare Sub Game_eingabe(byval Value_key As Byte , Byval Value_gameplay As Byte )       ' Funktion
zur Auswertung der Tasteneingabe
Declare Sub Game_play_zu_langsam(byval Value_gameplay As Byte )       ' Funktion zur Ausgabe von ZU
LANGSAM


'-------------------------------------------------------------------------------
' 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 freigeben

' Konfiguration Timer 2 für Hardware-PWM an OC2 (D.7)
Config Timer2 = Pwm , Prescale = 128 , Compare Pwm = Clear Up       ' Timer 2 verwenden
Enable Timer2                                               ' Interrupt für PWM Timer 2 freigeben


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

' Konfiguration LCD Display
Config Lcdpin = Pin , Db4 = Lcd_bit_0 , Db5 = Lcd_bit_1 , Db6 = Lcd_bit_2 , Db7 = Lcd_bit_3 , E =
Lcd_e , Rs = Lcd_rs
Config Lcd = 16 * 1 , Chipset = Dogm162v5                   ' DOG-M Treiber laden
Config Lcdbus = 4                                           ' LCD arbeitet über 4-Bit

Gosub Lcd_init                                              ' LCD grundlegend initialisieren

'                  12345678
Locate 1 , 1 : Lcd "********"


' Definition benutzerdefinierter Zeichen
```

```
Deflcdchar 0 , 32 , 27 , 32 , 4 , 4 , 17 , 17 , 14          ' Smily lachen
Deflcdchar 1 , 27 , 32 , 4 , 4 , 32 , 14 , 17 , 17          ' Smily weinen

' ----- ADC: Batterieueberwachung -----
' Konfiguration ADC Single-Mode und automatische Prescaler Setting
' Der Single-Mode wir bei BASCOM in Verbindung mit der Funktion GETADC() verwendet
' Der Prescaler teilt den internen Tackt durch 2, 4, 8,16,32,64 or 128 da der ADC
' einen Takt zwischen 50-200 kHz benötigt.
' Das AUTO Feature von BASCOM, setzt automatisch die höchste mögliche Taktrate
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc

' ------------------------------ Port's und Pin's ---------------------------

' ----- LED-Konfigurationen -----
Config Alive_led_pin = Output                               ' GPIO für Alive-LED ist Output


' ----- LED-Konfigurationen -----
Config Led_blau_pin = Output                                ' GPIO für blaue LED auf Output schalten
Config Led_gruen_pin = Output                               ' GPIO für grüne LED auf Output schalten
Config Led_rot_pin = Output                                 ' GPIO für rote LED auf Output schalten
Config Led_gelb_pin = Output                                ' GPIO für blaue LED auf Output schalten


' ----- KEY-Tasten-Konfiguration -----
Config Key_taster1_pin = Input                              ' GPIO für Taste 1 auf Inupt schalten
und PullUp aktivieren
Key_taster1 = Pullup_ein

Config Key_taster2_pin = Input                              ' GPIO für Taste 2 auf Inupt schalten
und PullUp aktivieren
Key_taster2 = Pullup_ein

Config Key_taster3_pin = Input                              ' GPIO für Taste 3 auf Inupt schalten
und PullUp aktivieren
Key_taster3 = Pullup_ein

Config Key_taster4_pin = Input                              ' GPIO für Taste 4 auf Inupt schalten
und PullUp aktivieren
Key_taster4 = Pullup_ein


' ------------------------------ Variablen und Werte -------------------------

' ----- LEDs -----
Led_blau = Led_aus
Led_gruen = Led_aus
Led_rot = Led_aus
Led_gelb = Led_aus


' ----- System -----
Sectick_counter = 0                                         ' Globaler Sekundenzähler initialisieren


' ----- LCD-Display -----
Lcd_kontrastwert = Lcd_kontrast_default                     ' Kontrastwert
Lcd_helligkeit = Lcd_helligkeit_default                     ' Displayhelligkleit


' ----- ADC_ Batteriespannungsueberwachung -----
Adc_channel = Adc_channel_number                            ' Spannungsteiler zur
Batterispannungsmessung hängt an ADC0


' ----- Spielesteuerung -----
Game_ablaufstatus = Game_mode_idle                          ' Steuervariable für Spielablaufsteue-
rung

' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Game_testmodus
   Print "Set Game Mode = 0 - Neustart"
#endif



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

```
' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Main_testmodus
   Print ""
   Print ""
   Print "----------------------------------------"
#endif


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


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

Gosub Lcd_kontrast_set                                     ' LCD Kontrast einstellen

Gosub Lcd_helligkeit_set                                   ' Displayhelligkeit einstellen
Lcd_beleuchtung_status = Lcd_beleuchtung_status_ein        ' Hintergrundbeleuchtung ist EIN

Gosub Adc_check_batterie                                   ' Batteriespannung überprüfen

Gosub Ee_init                                              ' Persistente Daten aus EEPROM auslesen


' ###############################################################################
'
'                              Hauptprogramm Senso
'
' ###############################################################################


' ----- Bypassmodus -----

' An dieser Stelle wird sofort der Status für den Bypassmodus abgefragt.
' Der Bypassmodus kann erreicht werden indem alle Tasten gleichzeitig betätigt
' und dann der Reihe nach 1-4 gedrückt werden

' Sind alle Tasten betätigt?
Temp_byte_1 = Pind And Key_all_key

If Temp_byte_1 = Key_all_key_pressed Then

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Bypass Step 1 erkannt"
   #endif

   Gosub Game_bypass_mode                                  ' Bypassmode

   Cls

   ' Nicht weitermachen bevor Taste losgelassen
   Do
      Temp_byte_1 = Key_check_all_key()

   Loop Until Temp_byte_1 = False

   ' So tun als wenn nichts gewesen wäre und weiter im Spiel

End If



' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Main_testmodus
   Print "** OK, let's GO **"
#endif

' LED Testroutine
Gosub Led_test_routine

' Soundcheck
Sound Speaker , Sound_ton2 , Sound_dauer2
Sound Speaker , Sound_ton3 , Sound_dauer3
Sound Speaker , Sound_ton2 , Sound_dauer2

' Begrüßungstext ausgeben bis Taste betätigt wird
Cls
```

```
Locate 1 , 1 : Lcd "Willkommen zu *SENSO* - GC2ACC4"
Do
   Waitms Lcd_shift_speed
   Shiftlcd Left

   Temp_byte_2 = Key_check_all_key()

Loop Until Temp_byte_2 = True


' Achtung: shiftlcd lässt Cursor willkürlich im Speicher stehen.
'          Damit wieder geziel auf Position 1,1 gearbeitet werden kann ist
'          Initlcd notwendig. Danach muss aber nochmals Kontrast eingestellt
'          werden


Gosub Lcd_init                                          ' When the display is initialized, the
display content is cleared also.
Gosub Lcd_kontrast_set                                  ' Kontrast nochmals einstellen, geht
durch Initlcd verloren


' Bildschirm löschen und Start-Aufforderung auf Display ausgeben
Cls
Locate 1 , 1 : Lcd Chr(&B01111110) ; "  START"

' Spielablaufvariable wurde mit IDLE vorbelegt, damit geht Spiel in
' Warteposition und aktiviert ggf. auch Bildschirm-Schoner


' Logik für Abschaltung Hintergrundbeleuchtung "aufziehen"
Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit


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


' Erklärung Spielstati - Game_ablaufstatus:
' 0 = Game_mode_idle
'     Spiel macht eigentlich nichts. Es wird ggf. die Hintergrundbeleuchtung
'     abgeschaltet und auf eine Tasteneingabe gewartet

' 1 = Game_mode_start
'     Spiel wurde neu gestartet oder nach erfolgreichem Spiel wiederholt.
'     Status 0 bedeutet es wurde die Start-Taste erkannt mit der das Spiel
'     gestartet werden kann. Es müssen alle Arbeitsvorbereitungen getroffen
'     werden und dann wird Spielstatus weitergeschaltet

' 2 = Game_mode_spiel

' 3 = Game_mode_aktion



Do                                                      ' Hauptschleife

   ' -------------------- Testaufgaben waehrend Entwicklung -------------------


   ' ------------------------------ Systemaufgaben ----------------------------


   ' --------------------------- Spielsteuerung -------------------------------


   ' -----
   ' ----- Spiel - Idle - Spiel idelt so vor sich hin und macht nichts ----
   ' -----

   If Game_ablaufstatus = Game_mode_idle Then

      ' Nur im Idle gibt es die Möglichkeit, dass das Spiel die Hintergrund-
      ' beleuchtung des Display abschalten kann, daher muss auch hier der
      ' entsprechende Ablauf erfolgen

      ' Prüfen ob Sekundencounter für Hintergrundbeleuchtung abgelaufen ist
```

```
      If Lcd_abschaltcounter = Sectick_counter Then

         ' JA -> Hintergrundbeleuchtung abschalten und Strom sparen
         Gosub Lcd_beleuchtung_aus
         Lcd_beleuchtung_status = Lcd_beleuchtung_status_aus       ' Hintergrundbeleuchtung ist AUS

         ' Durch Verringerung des Counters wird verhindert dass Funktion mehrfach angesprungen wird.
         Decr Lcd_abschaltcounter

      End If                                                  ' LCD Abschaltzeitpunkt erreicht

      ' Prüfen ob der Bildschirm abgeschaltet ist und ggf. auf Taste überprüfen
      ' um Bildschirm wieder ein zu schalten
      If Lcd_beleuchtung_status = Lcd_beleuchtung_status_aus Then

         ' Ja, Bildschirm ist aus

         ' Überprüfung ob irgend eine Taste betätigt wurde
         Temp_byte_1 = Key_check_all_key()
         If Temp_byte_1 = True Then

            ' Ja, Bildschirm ist aus .,.. es darf nichts weiter passieren außer Bildschirm an
            Gosub Lcd_beleuchtung_ein                         ' Hintergrundbeleuchtung einschalten
            Lcd_beleuchtung_status = Lcd_beleuchtung_status_ein       ' Hintergrundbeleuchtung ist
EIN

            ' Logik für Abschaltung Hintergrundbeleuchtung erneut "aufziehen"
            Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit

         End If                                               ' Tastenprüfung

      Else                                                    ' Bildschirmprüfung

         ' Nein, Bildschirm ist nicht aus und daher darf das Spiel erneut
         ' gestartet werden

         ' Überprüfung ob Spielstartbedingung "Blaue Taste betätigen" gegeben ist

         ' Tasten einlesen
         Temp_byte_1 = Key_check_any_key()

         'Prüfen ob irgend eine Taste gedrückt wurde damit abschaltcounter erhöht werden kann
         Temp_byte_2 = Temp_byte_1 And Key_all_key
         If Temp_byte_2 <> Key_all_key Then

            ' Logik für Abschaltung Hintergrundbeleuchtung erneut "aufziehen"
            Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit

            ' Überprüfen ob zufällig auch die blaue Taste betätigt wurde
            Temp_byte_2 = Temp_byte_1 And Key_bl__key
            If Temp_byte_2 <> Key_bl__key Then

               ' Ja, Blaue Taste wurde betätigt, damit kann Spiel gestartet werden
               Game_ablaufstatus = Game_mode_start

            End If                                            ' Blaue Taste

         End If                                               ' Tastenauswertung

      End If                                                  ' Bildschimbeleuchtung aus ja/nein

   End If                                                     ' Idle Mode


   ' -----
   ' ----- Spiel - Start - Vorbereitungsarbeiten ----
   ' -----

   If Game_ablaufstatus = Game_mode_start Then


      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Game_testmodus
         Print
         Print "-- Entering Game_mode_start"
         Print "Zufallszahlen laden"
      #endif

      Gosub Game_lade_sensowerte
```

```
      ' Sensolevel neu setzen da Spiel neu begonnen wird
      Game_sensolevel = 1

      ' Ablaufsteuerung Spiel weiterschalten
      Game_ablaufstatus = Game_mode_spiel

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Game_testmodus
         Print "Set Game Mode = 1 - Spiel"
      #endif

      Cls                                                  ' Bildschirm löschen

      ' Nachdem nun der Start des Spiel eingeleitet wurde wir noch ein Countdown
      ' 8-1 abgespielt
      For Temp_byte_1 = 1 To 8 Step 1
         Temp_byte_2 = 9 - Temp_byte_1
         Lcd Temp_byte_2;
         Waitms Lcd_countdown_speed
      Next Temp_byte_1

      Cls                                                  ' Bildschirm löschen

      ' Spielstart-Wartezeit
      Waitms Game_play_pause_delay

   End If                                                  ' Game_mode_start


   ' -----
   ' ----- Spiel - Request - Spiel gibt zu betätigende Tasten vor (komplette Serie) -----
   ' -----

   If Game_ablaufstatus = Game_mode_spiel Then

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Game_testmodus
         Print
         Print "Entering Game_mode_spiel"
         Print "Sensolevel ist " ; Game_sensolevel
      #endif

      Gosub Game_play_sensolevel

      ' Nun ist eine Benutzereingabe gefordert - Ablaufsteuerung Spiel weiterschalten
      Game_ablaufstatus = Game_mode_aktion

      ' Verzögerungs-Wartezeit bis Response
      Waitms Game_play_response_delay

   End If                                                  ' Game_mode_spiel


   ' -----
   ' ----- Spiel - Response - Spiel wartet auf Benutzereingabe und macht Auswertung -----
   ' -----

   If Game_ablaufstatus = Game_mode_aktion Then

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Game_testmodus
         Print
         Print "Entering Game_mode_aktion"
         Print "Sensolevel ist " ; Game_sensolevel
      #endif


      ' Falls Flag für Debuging gesetzt debugausgaben ausgeben
      #if Game_testmodus
         Print "Sensowerte sind: " ;
         For Temp_byte_1 = 1 To 10 Step 1
            Print Game_senso(temp_byte_1) ; " " ;
         Next Temp_byte_1
         Print " *"
      #endif

      ' In einer Gesamtschleife von 1-10 wird der Spieler nun aufgefordert die
      ' entsprechenden Tasten zu betätigen. Die EIngabe wir unmittelbar kontrolliert
      ' und die Zeit überwacht.
```

```
      For Temp_byte_3 = 1 To Game_sensolevel Step 1

         ' In einer Schleife wird geprüft, ob nun eine Taste betätigt wird.
         ' Die Schleife bildet gleichzeitig nach Ablauf den Timeout der zur
         ' Fehlerausgabe und zum Neustart des Spiels führt.

         ' Hier eventuell nochmals zur Sicherheit prüfen dass keine Taste betätigt ist
         ' Do
         '   Temp_byte_1 = Key_check_all_key()
         ' Loop Until Temp_byte_1 = False

         Temp_word_1 = 1                                    ' Timeoutcounter zurücksetzen
         Temp_byte_1 = False                                ' Verwendung al Error-Flag
         Temp_byte_2 = False                                ' Schleifenflag für Auswertung

         Do

            ' Auf Taste 1 prüfen
            If Key_taster1_pin = Key_pressed Then

               ' Ggf. Traceausgabe schreiben - abhängig von Compilerschalter
               #if Key_testmodus
                  Print "Key 1 BLAU pressed"
               #endif

               ' Auswertung der Eingabe aufrufen
               Call Game_eingabe(key_bl_num , Temp_byte_3 )

               ' Schleife kann verlassenwerden, es muss nicht auf Timeout gewartet werden da Taste
betätigt
               Temp_byte_2 = True

            End If                                          ' Prüfung Taste 1

            ' Auf Taste 2 prüfen
            If Key_taster2_pin = Key_pressed Then

               ' Ggf. Traceausgabe schreiben - abhängig von Compilerschalter
               #if Key_testmodus
                  Print "Key 1 ROT pressed"
               #endif

               ' Auswertung der Eingabe aufrufen
               Call Game_eingabe(key_rt_num , Temp_byte_3 )

               ' Schleife kann verlassenwerden, es muss nicht auf Timeout gewartet werden da Taste
betätigt
               Temp_byte_2 = True

            End If                                          ' Prüfung Taste 2

            ' Auf Taste 3 prüfen
            If Key_taster3_pin = Key_pressed Then

               ' Ggf. Traceausgabe schreiben - abhängig von Compilerschalter
               #if Key_testmodus
                  Print "Key 3 GRUEN pressed"
               #endif

               ' Auswertung der Eingabe aufrufen
               Call Game_eingabe(key_gn_num , Temp_byte_3 )

               ' Schleife kann verlassenwerden, es muss nicht auf Timeout gewartet werden da Taste
betätigt
               Temp_byte_2 = True

            End If                                          ' Prüfung Taste 3

            ' Auf Taste 4 prüfen
            If Key_taster4_pin = Key_pressed Then

               ' Ggf. Traceausgabe schreiben - abhängig von Compilerschalter
               #if Key_testmodus
                  Print "Key 4 GELB pressed"
               #endif

               ' Auswertung der Eingabe aufrufen
               Call Game_eingabe(key_ge_num , Temp_byte_3 )

               ' Schleife kann verlassenwerden, es muss nicht auf Timeout gewartet werden da Taste
```

```
betätigt
               Temp_byte_2 = True

            End If                                          ' Prüfung Taste 4

            Waitms Game_play_timeout_delay
            Incr Temp_word_1

         Loop Until Temp_word_1 > Game_play_loop_counter Or Temp_byte_2 = True

         ' Wenn Funktion hier ankommt gibt es mehrere Möglichkeiten ...
         ' - Es wurde eine Taste gedrückt und die war richtig .... alles OK
         ' - Es wurde eine Taste gedrückt und die war falsch ....
         ' - Es wurde keine Taste gedrückt .... Timeout => Schleifenflag = False


         ' Wurde keine Taste betätigt?
         If Temp_byte_2 = False Then

            ' Ja, es wurde keine Taste betätigt

            Temp_byte_1 = True                              ' Fehlerflag setzen

            ' Timeout-Melodie spiele und Spiel zurücksetzen
            Call Game_play_zu_langsam(temp_byte_3)

            ' Bildschirm löschen und Start-Aufforderung auf Display ausgeben
            Cls
            Locate 1 , 1 : Lcd Chr(&B01111110) ; "  START"

            Game_ablaufstatus = Game_mode_idle

            ' Logik für Abschaltung Hintergrundbeleuchtung erneut "aufziehen"
            Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit

            Exit For                                        ' FOR-Schleife fühzeitig verlassen

         End If                                             ' Taste betätigt

         ' Wurde falsche Taste betätigt?
         If Game_ablaufstatus = Game_mode_idle Then

               ' Ja es wurde eine falsche Taste betätigt.
               ' Behandlung ist aber bereits in Funktion erfolgt, hier muss nur
               ' noch FOR-Schleife abgebrochen werden

               Temp_byte_1 = True                           ' Fehlerflag setzen

               Exit For

          End If                                            ' Falsche Taste

      Next Temp_byte_3                                      ' Hauptschleife über Game_sensolevel 1-
10


      ' Nun muss noch geprüft werden ob das Spiel bereits erfolgreich
      ' mit 10 Durchläufen abgeschlossen ist ... wenn kein Fehler aufgetreten ist.

      If Temp_byte_1 = False Then

         If Game_sensolevel < 10 Then

            Game_ablaufstatus = Game_mode_spiel             ' Zurück auf Ausgabe
            Incr Game_sensolevel                            ' Sensolevel erhöhen

            ' In Abhängigkeit der Konstante Traceausgabe schreiben
            #if Game_testmodus
               Print "Spielende noch nicht erreicht"
               Print "Incr Sensolevel ist " ; Game_sensolevel
            #endif

            Waitms Game_play_level_pause

         Else

            Gosub Game_play_winwin                          ' Gewinnersound spielen

            Waitms Game_play_pause_delay                    ' Wartezeit Übergang Playsound -
Anszeige finale Koordinaten
```

```
            Gosub Lcd_show_final_koordinates                ' Finale Koordinaten ausgeben

            ' Bildschirm löschen und Start-Aufforderung auf Display ausgeben
            Cls

            ' Nicht weitermachen bevor Taste losgelassen
            Do
               Temp_byte_1 = Key_check_all_key()

            Loop Until Temp_byte_1 = False

            Locate 1 , 1 : Lcd Chr(&B01111110) ; "  START"

            Game_ablaufstatus = Game_mode_idle

            ' Logik für Abschaltung Hintergrundbeleuchtung erneut "aufziehen"
            Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit

         End If                                             ' Spielweiterschaltung

      End If                                                ' kein Fehler aufgetreten

   End If                                                   ' Game_mode_aktion

Loop                                                        ' Hauptschleife

'## End Hauptprogramm ###########################################################

End


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

'-----------------------------------------------------------------------------
' Interrupt-Service-Routine (Timer1): Sekunden_tick
'
' Routine zur Auswertung des Timer Interrupts
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Sectick_counter = Variable für den Sekundencounter
'-----------------------------------------------------------------------------
Sekunden_tick:

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

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

   ' Timervariable erhöhen
   Incr Sectick_counter

Return
'-- End Sekunden_tick  ------------------------------------------------------



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

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

'-----------------------------------------------------------------------------
' LCD - Subroutine: Lcd_print_level
'
' Subroutine schreibt entsprechende Levelzahl auf das Display
'
' Parameter:    Value = Levelzahl
' Rückgabewert: keine
'
' Globale Variable:
' --
```

```
'--------------------------------------------------------------------------------
Sub Lcd_print_level(byval Value As Byte)

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

   Cls
   Temp_string_1 = Str(value)
   Temp_string_1 = Format(temp_string_1 , "00")
   Locate 1 , 4 : Lcd Temp_string_1

End Sub
'-- End Lcd_print_level ---------------------------------------------------------


' ********
' * GAME *
' ********

'--------------------------------------------------------------------------------
' GAME - Subroutine: Game_eingabe
'
' Subroutine verarbeitet die übergebene Taste für den Spielablauf
'
' Parameter:    Value_key      = Nummer der betätigten Taste
'               Value_gameplay = Erreichstes Level
' Rückgabewert: keine
'
' Globale Variable:
' --
'--------------------------------------------------------------------------------
Sub Game_eingabe(byval Value_key As Byte , Byval Value_gameplay As Byte )

   ' ----- lokale Variablen -----
   Local Temp_value As Byte
   Local Temp_senso_loop As Byte

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

   Temp_value = Game_senso(value_gameplay )

   ' Falls Flag für Debuging gesetzt debugausgaben ausgeben
   #if Game_testmodus
      Print "Auswertung: SOLL = " ; Temp_value ; " - IST = " ; Value_key
   #endif


   ' Prüfen ob betätigte Taste richtig war oder nicht
   If Value_key = Temp_value Then

      ' Ja, Eingabe war richtig

       Select Case Value_key
         Case 1:                                          ' LED und Taste 1 = BLAU
           Led_blau = Led_ein
           Sound Speaker , Sound_ton1 , Sound_dauer1
           Led_blau = Led_aus
         Case 2:                                          ' LED und Taste 1 = ROT
           Led_rot = Led_ein
           Sound Speaker , Sound_ton2 , Sound_dauer2
           Led_rot = Led_aus
         Case 3:                                          ' LED und Taste 1 = GRUEN
           Led_gruen = Led_ein
           Sound Speaker , Sound_ton3 , Sound_dauer3
           Led_gruen = Led_aus
         Case 4:                                          ' LED und Taste 1 = GELB
           Led_gelb = Led_ein
           Sound Speaker , Sound_ton4 , Sound_dauer4
           Led_gelb = Led_aus
       End Select

      #if Game_testmodus
         Print "++ Eingabe war richtig"
      #endif

   Else                                                   ' Tastenprüfung

      ' Nein, Eingabe war falsch

      ' Verlierersound spielen
      Gosub Game_play_lose_sound
```

```
      Wait 2

      ' Game-Spielestatus zurücksetzen
      Game_ablaufstatus = Game_mode_idle

      ' Bildschirm löschen und Start-Aufforderung auf Display ausgeben
      Cls
      Locate 1 , 1 : Lcd Chr(&B01111110) ; "  START"

      ' Logik für Abschaltung Hintergrundbeleuchtung erneut "aufziehen"
      Lcd_abschaltcounter = Sectick_counter + Lcd_abschaltzeit

      #if Game_testmodus
         Print "++ Eingabe war falsch"
      #endif

   End If                                                ' Tastenprüfung

End Sub
'-- End Lcd_print_level ------------------------------------------------------



'------------------------------------------------------------------------------
' GAME - Subroutine: Game_play_zu_langsam
'
' Subroutine gibt abhängig vom Sensolevel die zugehörige "Zu LANGSAM" Sequnz
' aus
'
' Parameter:    Value_gameplay = Erreichtes Level
' Rückgabewert: keine
'
' Globale Variable:
' --
'------------------------------------------------------------------------------
Sub Game_play_zu_langsam(byval Value_gameplay As Byte )

   ' ----- lokale Variablen -----
   Local Temp_value As Byte
   Local Temp_loop_count As Byte

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

   Temp_value = Game_senso(value_gameplay )

   ' Meldung auf Display ausgeben
   Cls
   '    12345678
   Lcd "Timeout "

   For Temp_loop_count = 1 To 6 Step 1

       Select Case Temp_value

         Case 1:                                         ' LED und Taste 1 = BLAU
           Led_blau = Led_ein
           Sound Speaker , Sound_ton1 , Sound_dauer1
           Led_blau = Led_aus

         Case 2:                                         ' LED und Taste 1 = ROT
           Led_rot = Led_ein
           Sound Speaker , Sound_ton2 , Sound_dauer2
           Led_rot = Led_aus

         Case 3:                                         ' LED und Taste 1 = GRUEN
           Led_gruen = Led_ein
           Sound Speaker , Sound_ton3 , Sound_dauer3
           Led_gruen = Led_aus

         Case 4:                                         ' LED und Taste 1 = GELB
           Led_gelb = Led_ein
           Sound Speaker , Sound_ton4 , Sound_dauer4
           Led_gelb = Led_aus

       End Select

       Waitms Game_play_sensolevel_speed

   Next Temp_loop_count                                  ' Schleife Anzahl Timeout-Signale
```

```
   Wait Game_timeout_delay                                    ' Wartezeit bis weiter

End Sub
'-- End Game_play_zu_langsam ---------------------------------------------------



'*****************************************************************************
' Functions
'*****************************************************************************

' *********
' * KEY's *
' *********

'-----------------------------------------------------------------------------
' KEY - Funktion: Key_check_all_key()
' Funktion prüft ob eine Taste betätigt wurde und liefert den boolschen Wert
' des Ereignis zurück. Wurde eine Taste betätigt wartet die Funktion eine
' definierte Zeit und prüft erneut ob die Taste bereits losgelassen wurde.
' So werden Mehrfachauslösungen einer Funktion verhindert
'
' Parameter:    keine
' Rückgabewert: Bytewert = TRUE  - es wurde eine Taste erkannt
'                          FALSE - es wurde keine Taste erkannt
' Globale Variablen: keine
'-----------------------------------------------------------------------------
Function Key_check_all_key() As Byte

   ' ----- lokale Variablen -----
   Local Temp_value As Byte
   Local Temp_return_value As Byte

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

   ' Returnvariable vorbelegen
   Temp_return_value = False

   ' Tasten einlesen
   Temp_value = Pind And Key_all_key

   ' Prüfen ob Taste betätigt wurde
   If Temp_value <> Key_all_key Then

      ' Ja, Taste wurde betätigt, daher 20ms warten und auf Loslassen der Taste prüfen
      Waitms Key_entprellzeit

      Do
            Temp_value = Pind And Key_all_key
      Loop Until Temp_value = Key_all_key

      Temp_return_value = True

   End If                                                     ' Tastenprüfung

   ' Ergebnis an Rückgabewert übergeben
   Key_check_all_key = Temp_return_value

End Function
'-- End Key_check_all_key ---------------------------------------------------


'-----------------------------------------------------------------------------
' KEY - Funktion: Key_check_any_key()
' Funktion prüft ob eine Taste betätigt wurde und liefert den Byte-Wert des
' kompletten eingelesenen Ports als Ereignis zurück.
' Wurde eine Taste betätigt wartet die Funktion eine
' definierte Zeit und prüft erneut ob die Taste bereits losgelassen wurde.
' So werden Mehrfachauslösungen einer Funktion verhindert
'
' Parameter:    keine
' Rückgabewert: Bytewert des gesamten Ports --> Auswertung muss außerhalb
'                                               der Funktion erfolgen.
' Globale Variablen: keine
'-----------------------------------------------------------------------------
Function Key_check_any_key() As Byte

   ' ----- lokale Variablen -----
   Local Temp_value_1 As Byte
   Local Temp_value_2 As Byte
```

```
   ' ----- Programmcode -----

   ' Tasten einlesen
   Temp_value_1 = Pind And Key_all_key

   ' Prüfen ob Taste betätigt wurde
   If Temp_value_1 <> Key_all_key Then

      ' Ja, Taste wurde betätigt, daher 20ms warten und auf Loslassen der Taste prüfen
      Waitms Key_entprellzeit

      Do
            Temp_value_2 = Pind And Key_all_key
      Loop Until Temp_value_2 = Key_all_key

   End If                                                ' Tastenprüfung

   ' Ergebnis an Rückgabewert übergeben
   Key_check_any_key = Temp_value_1

End Function
'-- End Key_check_any_key ------------------------------------------------------



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


' ********
' * GAME *
' ********

'-------------------------------------------------------------------------------
' GAME - Gosub-Routine: Game_lade_sensowerte
'
' Routine ermittelt via Zufallszahl den im Spiel zu verwendenden Pattern für
' die Sensowerte. Diese wurden während der Entwicklung offline erzeugt.
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Game_sensowert = Array 1 - 10 der jeweiligen Senso-Werte
'-------------------------------------------------------------------------------
Game_lade_sensowerte:

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

   ' Zufallszahl ermitteln
   Temp_byte_1 = Rnd(game_anzahl_text )                  ' Random bei z.B. 40 = 0-39

   ' Falls Flag für Debuging gesetzt debugausgaben ausgeben
   #if Game_testmodus
      Print "Zufallszahl: " ; Temp_byte_1
   #endif

   Temp_byte_1 = Temp_byte_1 * 2                         ' Korrektur Zufallszahl auf Doppelpaare

   ' Sensowerte Doppelpaare laden
   Temp_word_1 = Lookup(temp_byte_1 , Spielkombinationen_data)
   Incr Temp_byte_1
   Temp_word_2 = Lookup(temp_byte_1 , Spielkombinationen_data)

   ' Falls Flag für Debuging gesetzt debugausgaben ausgeben
   #if Game_testmodus
      Print "Wert1: " ; Temp_word_1 ; "  -  Wert2: " ; Temp_word_2
   #endif

   ' Extrahieren von Einer-Zener-Hunderter-Tausender-Zehntausender und Übernahme
   ' in Senso-Zahlenarray
   For Temp_index = 5 To 1 Step -1

      ' Sensowerte Block 1 zerlegen
      Temp_word_3 = Temp_word_1 Mod 10
      Game_senso(temp_index) = Temp_word_3
      Temp_word_1 = Temp_word_1 - Temp_word_3
      Temp_word_1 = Temp_word_1 / 10
```

```
      ' Sensowerte Block 2 zerlegen
      Temp_word_3 = Temp_word_2 Mod 10
      Game_senso(temp_index + 5) = Temp_word_3
      Temp_word_2 = Temp_word_2 - Temp_word_3
      Temp_word_2 = Temp_word_2 / 10

   Next Temp_index                                          ' Indexschleife

   ' Falls Flag für Debuging gesetzt debugausgaben ausgeben
   #if Game_testmodus
      Print "Sensowerte sind: " ;
      For Temp_byte_1 = 1 To 10 Step 1
         Print Game_senso(temp_byte_1) ; " " ;
      Next Temp_byte_1
      Print " *"
   #endif

Return
'-- End Game_lade_sensowert ------------------------------------------------


'-------------------------------------------------------------------------------
' GAME - Gosub-Routine: Game_play_sensolevel
'
' Routine gibt abhängig vom erreichten Sensolevel die LED-Signale und Sound
' aus
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Sensolevel     = erreichtes Spielelevel (1-10)
' Game_sensowert = Array 1 - 10 der jeweiligen Senso-Werte
'-------------------------------------------------------------------------------
Game_play_sensolevel:

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

   ' In einer For-Schleife je nach erreichtem Senso-Spiellevel (1-10)
   ' LED einschalten -> Sound spielen -> LED ausschalten

   For Temp_index = 1 To Game_sensolevel
       Call Lcd_print_level(temp_index)
       Select Case Game_senso(temp_index)
         Case 1:                                            ' LED und Taste 1 = BLAU
           Led_blau = Led_ein
           Sound Speaker , Sound_ton1 , Sound_dauer1
           Led_blau = Led_aus
         Case 2:                                            ' LED und Taste 1 = ROT
           Led_rot = Led_ein
           Sound Speaker , Sound_ton2 , Sound_dauer2
           Led_rot = Led_aus
         Case 3:                                            ' LED und Taste 1 = GRUEN
           Led_gruen = Led_ein
           Sound Speaker , Sound_ton3 , Sound_dauer3
           Led_gruen = Led_aus
         Case 4:                                            ' LED und Taste 1 = GELB
           Led_gelb = Led_ein
           Sound Speaker , Sound_ton4 , Sound_dauer4
           Led_gelb = Led_aus
       End Select

       Waitms Game_play_sensolevel_speed                    ' Wartepause

     Next Temp_index                                        ' Schleife über erreichtes Sensolevel

Return
'-- End Game_play_sensolevel -----------------------------------------------


'-------------------------------------------------------------------------------
' GAME - Gosub-Routine: Game_play_winwin
'
' Routine spielt Winsound ab und gibt Glückwunsch-Text auf Bildschirm aus
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
```

```
'--------------------------------------------------------------------------------
Game_play_winwin:

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

   ' Displayausgabe
   Cls
   '                  12345678
   Locate 1 , 1 : Lcd Chr(0) ; " JUHU " ; Chr(0)

   ' Sound gefunden und verwendet von:
   ' http://www.mcselec.com/index.php?option=com_content&task=view&id=221&Itemid=57

   'Europe :: Final Countdown
   Gosub Led_all_on : Sound Speaker , 124 , 675 : Gosub Led_all_off       'H2(1/16)
   Gosub Led_all_on : Sound Speaker , 110 , 758 : Gosub Led_all_off       'A2(1/16)
   Gosub Led_all_on : Sound Speaker , 248 , 675 : Gosub Led_all_off       'H2(1/8)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 165 , 1011 : Gosub Led_all_off       'E2(1/8)
   Waitms 250                                                           'P(1/8)
   Waitms 125                                                           'P(1/16)
   Gosub Led_all_on : Sound Speaker , 131 , 637 : Gosub Led_all_off       'C3(1/16)
   Gosub Led_all_on : Sound Speaker , 124 , 675 : Gosub Led_all_off       'H2(1/16)
   Gosub Led_all_on : Sound Speaker , 131 , 637 : Gosub Led_all_off       'C3(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 124 , 675 : Gosub Led_all_off       'H2(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 220 , 758 : Gosub Led_all_off       'A2(1/8)
   Waitms 125                                                            'P(1/16)
   Waitms 250                                                           'P(1/8)
   Gosub Led_all_on : Sound Speaker , 131 , 637 : Gosub Led_all_off       'C3(1/16)
   Gosub Led_all_on : Sound Speaker , 124 , 675 : Gosub Led_all_off       'H2(1/16)
   Gosub Led_all_on : Sound Speaker , 262 , 637 : Gosub Led_all_off       'C3(1/8)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 165 , 1011 : Gosub Led_all_off       'E2(1/8)
   Waitms 250                                                           'P(1/8)
   Waitms 125                                                           'P(1/16)
   Gosub Led_all_on : Sound Speaker , 110 , 758 : Gosub Led_all_off       'A2(1/16)
   Gosub Led_all_on : Sound Speaker , 98 , 850 : Gosub Led_all_off       'G2(1/16)
   Gosub Led_all_on : Sound Speaker , 110 , 758 : Gosub Led_all_off       'A2(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 98 , 850 : Gosub Led_all_off       'G2(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 92 , 901 : Gosub Led_all_off       'Fis2(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 110 , 758 : Gosub Led_all_off       'A2(1/16)
   Waitms 125                                                            'P(1/16)
   Gosub Led_all_on : Sound Speaker , 196 , 850 : Gosub Led_all_off       'G2(1/8)
   Wait 2

Return
'-- End Game_play_winwin ------------------------------------------------------



'--------------------------------------------------------------------------------
' GAME - Gosub-Routine: Game_play_lose_sound
'
' Routine spielt Verlierer-Sound ab
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
'--------------------------------------------------------------------------------
Game_play_lose_sound:

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

   ' Displayausgabe
   Cls
   '                  12345678
   Locate 1 , 1 : Lcd Chr(1) ; "SCHADE" ; Chr(1)

   'Note|Freqvency|Periods|Pulses 1/1|Pulses 1/2|Pulses 1/4|Pulses 1/8|Pulses 1/16|
   '=================================================================================
   ' C1 |  261,63 |  2548 |    523   |    262   |    131   |    65    |     33    |
   'Cis1|  277,18 |  2405 |    554   |    277   |    139   |    69    |     35    |
   ' D1 |  293,66 |  2270 |    587   |    294   |    147   |    73    |     37    |
   'Dis1|  311,13 |  2143 |    622   |    311   |    156   |    78    |     39    |
```

```
   ' E1 |  329,63 |  2022 |    659   |    330   |    165   |    82    |     41    |
   ' F1 |  349,23 |  1909 |    698   |    349   |    175   |    87    |     44    |
   'Fis1|  369,99 |  1802 |    740   |    370   |    185   |    92    |     46    |
   ' G1 |  392,00 |  1701 |    784   |    392   |    196   |    98    |     49    |
   'Gis1|  415,30 |  1605 |    831   |    415   |    208   |   104    |     52    |
   ' A1 |  440,00 |  1515 |    880   |    440   |    220   |   110    |     55    |
   'Ais1|  466,16 |  1430 |    932   |    466   |    233   |   117    |     58    |
   ' B1 |  493,88 |  1350 |    988   |    494   |    247   |   124    |     62    |

   Gosub Led_all_on : Sound Speaker , 062 , 1350 : Gosub Led_all_off       ' B1 (1/16)
   Gosub Led_all_on : Sound Speaker , 058 , 1430 : Gosub Led_all_off       ' Ais1 (1/16)
   Gosub Led_all_on : Sound Speaker , 055 , 1515 : Gosub Led_all_off       ' A1 (1/16)
   Gosub Led_all_on : Sound Speaker , 052 , 1605 : Gosub Led_all_off       ' Gis 1 (1/16)
   Gosub Led_all_on : Sound Speaker , 049 , 1701 : Gosub Led_all_off       ' G1 (1/16)
   Gosub Led_all_on : Sound Speaker , 046 , 1802 : Gosub Led_all_off       ' Fis1 (1/16)
   Gosub Led_all_on : Sound Speaker , 044 , 1909 : Gosub Led_all_off       ' F1 (1/16)
   Gosub Led_all_on : Sound Speaker , 041 , 2022 : Gosub Led_all_off       ' E1 (1/16)
   Gosub Led_all_on : Sound Speaker , 039 , 2143 : Gosub Led_all_off       ' Dis1 (1/16)
   Gosub Led_all_on : Sound Speaker , 037 , 2270 : Gosub Led_all_off       ' D1 (1/16)
   Gosub Led_all_on : Sound Speaker , 035 , 2405 : Gosub Led_all_off       ' Cis1 (1/16)
   Gosub Led_all_on : Sound Speaker , 131 , 2548 : Gosub Led_all_off       ' C1 (1/4)

Return
'-- End Game_play_lose_sound ------------------------------------------------


'--------------------------------------------------------------------------------
' GAME - Gosub-Routine: Game_bypass_mode
'
' Routine gibt nach erfolgreichem Spiel die finalen Koordinaten auf dem
' Display aus
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
'--------------------------------------------------------------------------------
Game_bypass_mode:

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

   Gosub Led_all_on                                        ' Alle LED's einschalten

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Eintritt in Bytepass-Sequenz"
   #endif

   ' Warten bis wieder alle Tasten losgelassen wurden
   Do
      Temp_byte_1 = Key_check_all_key()
   Loop Until Temp_byte_1 = False

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Alle Tasten losgelassen"
   #endif

   Waitms Key_entprellzeit

   ' Tasten einlesen und maskieren
   Do
      Temp_byte_1 = Key_check_any_key()
   Loop Until Temp_byte_1 <> Key_all_key
   Temp_byte_1 = Temp_byte_1 And Key_bl__key

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Taste 1 erkannt - " ; Bin(temp_byte_1)
   #endif

   ' Taste BLAU abfragen
   If Temp_byte_1 = Key_masked_key_aktive Then

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Bypass_testmodus
         Print "Taste 1 ist BLAU"
      #endif

      Waitms Key_entprellzeit
```

```
' Tasten einlesen und maskieren
Do
   Temp_byte_1 = Key_check_any_key()
Loop Until Temp_byte_1 <> Key_all_key
Temp_byte_1 = Temp_byte_1 And Key_rt__key

' In Abhängigkeit der Konstante Traceausgabe schreiben
#if Bypass_testmodus
   Print "Taste 2 erkannt - " ; Bin(temp_byte_1)
#endif

' Taste ROT abfragen
If Temp_byte_1 = Key_masked_key_aktive Then

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Taste 2 ist ROT"
   #endif

   Waitms Key_entprellzeit

   ' Tasten einlesen und maskieren
   Do
      Temp_byte_1 = Key_check_any_key()
   Loop Until Temp_byte_1 <> Key_all_key
   Temp_byte_1 = Temp_byte_1 And Key_gn__key

   ' In Abhängigkeit der Konstante Traceausgabe schreiben
   #if Bypass_testmodus
      Print "Taste 3 erkannt - " ; Bin(temp_byte_1)
   #endif

   ' Taste GRÜN abfragen
   If Temp_byte_1 = Key_masked_key_aktive Then

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Bypass_testmodus
         Print "Taste 3 ist GRUEN"
      #endif

      Waitms Key_entprellzeit

      ' Tasten einlesen und maskieren
      Do
         Temp_byte_1 = Key_check_any_key()
      Loop Until Temp_byte_1 <> Key_all_key
      Temp_byte_1 = Temp_byte_1 And Key_ge__key

      ' In Abhängigkeit der Konstante Traceausgabe schreiben
      #if Bypass_testmodus
         Print "Taste 4 erkannt - " ; Bin(temp_byte_1)
      #endif

      ' Taste GELB abfragen
      If Temp_byte_1 = Key_masked_key_aktive Then

         ' In Abhängigkeit der Konstante Traceausgabe schreiben
         #if Bypass_testmodus
            Print "Taste 4 ist GELB"
         #endif

         ' Counter für Bypassmode erhöhen und in EEPROM speichern
         Incr Ee_bypass_counter
         Ee_data_bypass_counter = Ee_bypass_counter

         Temp_string_1 = Str(ee_bypass_counter)
         Temp_string_1 = Format(temp_string_1 , "000")
         Cls
         '    12345678
         Lcd "BPC: " ; Temp_string_1

         Wait 3

         ' Gewinnmelodie abspielen
         Gosub Game_play_winwin

         ' Bytepassmode auslösen und finale Koordinaten anzeigen
         Gosub Lcd_show_final_koordinates
```

```
            End If                                              ' Taste GELB

         End If                                                 ' Taste GRÜN

      End If                                                    ' Taste ROT

   End If                                                       ' Taste BLAU

   Gosub Led_all_off                                            ' Alle LED's ausschalten

Return
'-- End Game_bypass_mode ------------------------------------------------------


' *********
' * LED's *
' *********

'-------------------------------------------------------------------------------
' LED - Gosub-Routine: Led_test_routine
'
' Routine dreht die 4 LED des 4 Farben 2x im Kreis als Test
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen: keine
'-------------------------------------------------------------------------------
Led_test_routine:                                               ' LED Testroutine

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

   Led_blau = Led_ein
   Waitms Led_speed_testroutine
   Led_rot = Led_ein
   Waitms Led_speed_testroutine
   Led_gruen = Led_ein
   Waitms Led_speed_testroutine
   Led_gelb = Led_ein
   Waitms Led_speed_testroutine
   Led_blau = Led_aus
   Waitms Led_speed_testroutine
   Led_rot = Led_aus
   Waitms Led_speed_testroutine
   Led_gruen = Led_aus
   Waitms Led_speed_testroutine
   Led_gelb = Led_aus
   Waitms Led_speed_testroutine
   Led_blau = Led_ein
   Waitms Led_speed_testroutine
   Led_rot = Led_ein
   Waitms Led_speed_testroutine
   Led_gruen = Led_ein
   Waitms Led_speed_testroutine
   Led_gelb = Led_ein
   Waitms Led_speed_testroutine
   Led_blau = Led_aus
   Waitms Led_speed_testroutine
   Led_rot = Led_aus
   Waitms Led_speed_testroutine
   Led_gruen = Led_aus
   Waitms Led_speed_testroutine
   Led_gelb = Led_aus

Return
'-- End Led_test_routine --------------------------------------------------------


'-------------------------------------------------------------------------------
' LED - Gosub-Routine: Led_all_on
'
' Routine schaltet alle LED's ein und wartet x ms
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen: keine
'-------------------------------------------------------------------------------
Led_all_on:
```

```
   ' ----- Programmcode -----

   Led_blau = Led_ein
   Led_rot = Led_ein
   Led_gruen = Led_ein
   Led_gelb = Led_ein
   Waitms Game_play_winsound_speed

Return
'-- End Led_all_on ------------------------------------------------------------


'--------------------------------------------------------------------------------
' LED - Gosub-Routine: Led_all_off
'
' Routine schaltet alle LED's aus und wartet x ms
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen: keine
'--------------------------------------------------------------------------------
Led_all_off:

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

   Led_blau = Led_aus
   Led_rot = Led_aus
   Led_gruen = Led_aus
   Led_gelb = Led_aus
   Waitms Game_play_winsound_speed

Return
'-- End Led_all_on ------------------------------------------------------------


' ***************
' * LCD-Dsiplay *
' ***************

'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_init
'
' Routine macht grundlegende LCD Initiualisierungen die im Programm dank
' Lauftext usw. immer wieder benötigt werden.
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
'--------------------------------------------------------------------------------
Lcd_init:                                                   ' LCD Initialisierung

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

   Initlcd                                                  ' LCD initialisieren
   Waitms 100                                               ' 100ms nach Init warten
   Cursor Off Noblink                                       ' Blinkenden Cursor abschalten
   Cls                                                      ' Clear Screen


Return
'-- End Lcd_init -------------------------------------------------------------


'--------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_contrast_set
'
' Routine berechnen neue Kontrastwerte und steuert direkt den
' Kontroller des Display an.
' Aufgrund von 6 Bit sind nur Kontrastwerte zwischen 0 und 63 möglich!
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Lcd_kontrastwert = Kontrastwert für Display
'--------------------------------------------------------------------------------
Lcd_kontrast_set:                                           ' Kontrasteinstellung Display
```

```
   ' ----- Programmcode -----

   ' Verarbeitung des Kontrastwertes für High-Byte und Low-Byte
   Temp_byte_1 = Lcd_kontrastwert And &B00001111
   Temp_byte_1 = Temp_byte_1 + &B01110000

   Temp_byte_2 = Lcd_kontrastwert
   Shift Temp_byte_2 , Right , 4
   Temp_byte_2 = Temp_byte_2 And &B00000011
   Temp_byte_2 = Temp_byte_2 + &B01010100

   ' Instruction Table 1 einstellen [0,1]
   _temp1 = &B00101001
   !rCall _Lcd_control

   ' Tempvar_1 = &B0111xxxx für Kontrast Set Instruction Table 1 - Low Byte
   _temp1 = Temp_byte_1
   !rCall _Lcd_control

   ' Temovar_2 = &B010101xx für Kontrast Set Instruction Table 1 - High Byte
   _temp1 = Temp_byte_2
   !rCall _Lcd_control

   ' Zurückschalten auf Instruction Table 0 [0,0]
   _temp1 = &B00101000
   !rCall _Lcd_control

Return
'-- End Lcd_kontrast_set -------------------------------------------------------


'-----------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_helligkeit_set
'
' Routine setzt den Helligkeitswert aus der globalen Variable Lcd_helligkeit in
' das Controllregister
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Lcd_helligkeit = Helligkeitswert für PWM
'-----------------------------------------------------------------------------
Lcd_helligkeit_set:

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

   ' Variable in Timer-Count-Register laden
   Ocr2 = Lcd_helligkeit

Return
'-- End Ir_set_helligkeit -------------------------------------------------------


'-----------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_beleuchtung_aus
'
' Routine schaltet die LCD-Hintergrundbeleuchtung aus
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'-----------------------------------------------------------------------------
Lcd_beleuchtung_aus:

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

   Lcd_helligkeit = Lcd_helligkeit_aus
   Gosub Lcd_helligkeit_set

   #if Lcd_testmodus
      Print "Beleuchtung aus"
   #endif

Return
'-- End Lcd_beleuchtung_aus -----------------------------------------------------
```

```
'-------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_beleuchtung_ein
'
' Routine schaltet die LCD-Hintergrundbeleuchtung ein
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'-------------------------------------------------------------------------------
Lcd_beleuchtung_ein:

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

   Lcd_helligkeit = Lcd_helligkeit_default
   Gosub Lcd_helligkeit_set

   #if Lcd_testmodus
      Print "Beleuchtung ein"
   #endif

Return
'-- End Lcd_beleuchtung_ein ------------------------------------------------


'-------------------------------------------------------------------------------
' LCD - Gosub-Routine: Lcd_show_final_koordinates
'
' Routine gibt wiederholend die finalen Koordinaten aus bis Taste benötigt ist
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' --
'-------------------------------------------------------------------------------
Lcd_show_final_koordinates:

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

   ' Abbruchflag initialisieren
   Temp_byte_3 = False

   Do

      Cls
      '    12345678
      Lcd "Final "

      Wait Game_show_finaly_delay

      Cls
      '    12345678
      Lcd "ist bei"

      Wait Game_show_finaly_delay

      Cls
      '    12345678
      Lcd "N 26.704"

      Wait Game_show_finaly_delay

      Cls
      '    12345678
      Lcd "E 57.961"

      Wait Game_show_finaly_delay

      Cls
      '                    2345678
      Lcd Chr(&B01111110) ; "  Taste"

' Implementierung 1 führt aber zu ggf. unnötiger Wiederholung
'      Wait Game_show_finaly_delay
'      Temp_byte_2 = Pind And Key_all_key
'   Loop Until Temp_byte_2 <> Key_all_key

      ' Implementierung 2 - Tasten werden in FOR-Schleife abgefragt so dass
```

```
      ' eventueller Tastendruck sofort erkannt werden kann

      ' FOR-Schleife mit Delay aufziehen in der der Hinweise "Taste" angezeigt wird
      For Temp_byte_2 = 0 To 200 Step 1

         ' Tasten einlesen
         Temp_byte_1 = Pind And Key_all_key

         ' Prüfen ob Taste betätigt wurde / erkannt wurde
         If Temp_byte_1 <> Key_all_key Then

            ' JA, Taste wurde erkannt

            Temp_byte_3 = True                              ' Abbruchbedingung setzen
            Exit For                                        ' FOR-Schleife abbrechen

         End If                                             ' Tastenabfrage

         Waitms Game_play_timeout_delay                     ' Verzögerung für FOR-Schleife

      Next Temp_byte_2                                      ' Hinweisanzeige

   Loop Until Temp_byte_3 = True                            ' Hinweis repetierend anzeigen

Return
'-- End Lcd_show_final_koordinates ------------------------------------------



' **********
' * EEPROM *
' **********

'-------------------------------------------------------------------------------
' EEPROM - Gosub-Routine: EE_init
'
' Routine prüft ob EEPROM schon initialisiert ist / war, holt ggf. dieses nach
' und liest aktuellen Bytepasscounter aus.
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
'    Ee_bypass_counter      = Arbeitsvariable für Bypasscounter
'    Ee_data_bypass_counter = Flash-variable für Bypasscounter
'    Ee_data_version_value  = Flash-Variable für Version
'    Ee_version_value       = Arbeitsvariable für Version
'-------------------------------------------------------------------------------
Ee_init:

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

   ' Anmerkung:
   ' Es wird davon ausgegangen dass nicht initialisierte / neue EEPROMS
   ' bzw. gelöschte EEPROMS den Wert 0xFF enthalten.

   ' Version aus Speicher auslesen
   Ee_version_value = Ee_data_version_value

   ' Prüfen ob Eeprom schon initialisiert ist, wenn NEIN tue dies
   If Ee_version_value <> Ee_default_version_value Then

      ' NEIN, noch nicht initialisiert

      ' Arbeitsvariablen vorbelegen
      Ee_version_value = Ee_default_version_value
      Ee_bypass_counter = 0

      ' Arbeitsvariablen in EEPROM schreiben
      Ee_data_bypass_counter = Ee_bypass_counter
      Ee_data_version_value = Ee_version_value

      ' Ggf. Trace ausgeben
      #if Eeprom_testmodus
         Print "EEPROM und Arbeitsvariablen wurden neu initialisiert"
      #endif

   Else                                                     ' Eeprom schon Initialisiert

      ' JA, Speicher ist initialisiert
```

```
      ' Counter in Arbeitsvariable auslesen
      Ee_bypass_counter = Ee_data_bypass_counter

      ' Ggf. Trace ausgeben
      #if Eeprom_testmodus
         Print "EEPROM ist gueltig. Bypasscounter = " ; Ee_bypass_counter
      #endif

   End If                                                  ' Eeprom schon Initialisiert

Return
'-- End Ee_init ------------------------------------------------------------


' **********************
' * ADC-Batteriestatus *
' **********************

'-------------------------------------------------------------------------------
' ADC - Gosub-Routine: Adc_check_batterie
'
' Routine liest den Spannungswert aus dem AD-Wandler, prüft gegen eine
' voreingestellte Grenze und schaltet ggf. die Power-LED als Warnsignal ein.
'
' Parameter: keine
' Rückgabe:  keine
'
' Globale Variablen:
' Temp_word_1     = temporäre WORD Arbeitsvariable
' adc_channel     = Variable beinhaltet Kanalnummer des ADC
' Ubatt_grenzwert = Konstante für Grenzwert
'-------------------------------------------------------------------------------
Adc_check_batterie:

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

   ' ADC-Wert lesen
   Temp_word_1 = Getadc(adc_channel)

   #if Batterie_testmodus
      Print "ADC-Wert Batteriespannung : " ; Temp_word_1
   #endif

   ' Hier findet keine Batterie-Spannungsüberwachung statt

   ' Zufallszahlengenerator "verdrehen" :-) Erklärung siehe ganz oben in der Einleitung!!
   ___rseed = Temp_word_1

Return
'-- End Adc_check_batterie -------------------------------------------------


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

' System halt
End                                                        'end program


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

' Anmerkung:
' Eigentlich wird eine 10-stellige ganze Zahl benötigt die in ein Senso-Patter
' aufgelöst werden soll. Dazu würde sich der Wert DWord von BASCOM eignen.
' Wertebereich 0 to 4294967295
' Anscheinend gibt es aber einen Bug in BASCOM der dien verhindert und durch den
' nur Werte bis 2147483647 möglich sind. Das ist zu wendig
' Da Werte bis 4444444444 benötigt werden wird der Zahlenwert in 2 Word-Werte
' zerlegt die nacheinander gelesen und dann zusammengesetzt werden. Unschön!

Spielkombinationen_data:
Data 34212% , 31441%                                       ' #000
Data 43124% , 43432%                                       ' #001
Data 42244% , 42442%                                       ' #002
Data 32243% , 41112%                                       ' #003
Data 33234% , 14414%                                       ' #004
```

```
Data 12421% , 41344%                    ' #005
Data 44221% , 31431%                    ' #006
Data 33211% , 41234%                    ' #007
Data 42141% , 42121%                    ' #008
Data 43433% , 31414%                    ' #009
Data 44224% , 31232%                    ' #010
Data 34144% , 31342%                    ' #011
Data 24214% , 22423%                    ' #012
Data 42424% , 44421%                    ' #013
Data 21341% , 21342%                    ' #014
Data 11244% , 24442%                    ' #015
Data 24123% , 41344%                    ' #016
Data 42212% , 34221%                    ' #017
Data 42433% , 23341%                    ' #018
Data 23131% , 13444%                    ' #019
Data 41343% , 43444%                    ' #020
Data 44112% , 31221%                    ' #021
Data 34243% , 23312%                    ' #022
Data 42412% , 12112%                    ' #023
Data 31324% , 23114%                    ' #024
Data 44242% , 32441%                    ' #025
Data 44413% , 23343%                    ' #026
Data 41243% , 21243%                    ' #027
Data 21332% , 24333%                    ' #028
Data 13224% , 24413%                    ' #029
Data 41333% , 11314%                    ' #030
Data 33113% , 31143%                    ' #031
Data 31244% , 33112%                    ' #032
Data 33313% , 13124%                    ' #033
Data 22423% , 33322%                    ' #034
Data 23411% , 14243%                    ' #035
Data 14312% , 23114%                    ' #036
Data 24212% , 44324%                    ' #037
Data 44333% , 43114%                    ' #038
Data 34221% , 44311%                    ' #039
Data 23441% , 31341%                    ' #040
Data 43112% , 11334%                    ' #041
Data 21312% , 12341%                    ' #042
Data 23133% , 44334%                    ' #043
Data 42231% , 32221%                    ' #044
Data 24241% , 11113%                    ' #045
Data 24142% , 13224%                    ' #046
Data 13432% , 44443%                    ' #047
Data 24212% , 14113%                    ' #048
Data 44132% , 22443%                    ' #049
Data 12423% , 24323%                    ' #050
Data 33111% , 13432%                    ' #051
Data 22112% , 44324%                    ' #052
Data 23414% , 12243%                    ' #053
Data 12133% , 42234%                    ' #054
Data 34444% , 22324%                    ' #055
Data 24312% , 34211%                    ' #056
Data 31414% , 24242%                    ' #057
Data 13323% , 31133%                    ' #058
Data 41113% , 24333%                    ' #059
Data 23313% , 43444%                    ' #060
Data 43121% , 22214%                    ' #061
Data 32431% , 33132%                    ' #062
Data 24224% , 33432%                    ' #063
Data 12223% , 34132%                    ' #064
Data 42143% , 33213%                    ' #065
Data 33111% , 21244%                    ' #066
Data 14321% , 14222%                    ' #067
Data 12123% , 12134%                    ' #068
Data 32424% , 44443%                    ' #069
Data 14112% , 13343%                    ' #070
Data 21242% , 32331%                    ' #071
Data 11212% , 44313%                    ' #072
Data 34431% , 24222%                    ' #073
Data 13431% , 21232%                    ' #074
Data 34212% , 12224%                    ' #075
Data 32212% , 22231%                    ' #076
Data 13121% , 42124%                    ' #077
Data 24132% , 41242%                    ' #078
Data 31112% , 22331%                    ' #079
Data 34311% , 21111%                    ' #080
Data 32322% , 21112%                    ' #081
Data 44231% , 31421%                    ' #082
Data 24322% , 43344%                    ' #083
Data 33233% , 13343%                    ' #084
```

```
Data 34213% , 42321%                                                    ' #085
Data 44443% , 41443%                                                    ' #086
Data 41243% , 12413%                                                    ' #087
Data 13314% , 22234%                                                    ' #088
Data 41242% , 21412%                                                    ' #089
Data 12342% , 32333%                                                    ' #090
Data 34412% , 22121%                                                    ' #091
Data 34421% , 33144%                                                    ' #092
Data 13133% , 43213%                                                    ' #093
Data 41412% , 22233%                                                    ' #094
Data 12412% , 14311%                                                    ' #095
Data 32244% , 42142%                                                    ' #096
Data 41241% , 34441%                                                    ' #097
Data 14423% , 21344%                                                    ' #098
Data 13442% , 32421%                                                    ' #099


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


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


'################################### Historie ###############################
'
' 25.12.2016 : Version x.x
'              Beginn der Implementierungen für DrMaFu_Senso
' ---------------------------------------------------------------------------
' 08.02.2017 : Version 1.0
'              Fertigstellung der Implementierung auf Basis STK500
' ---------------------------------------------------------------------------
' 19.03.2017 : Version 1.0
'              Umstellung GPIO auf Zielsystem und Programmierung Zielsystem
' ---------------------------------------------------------------------------
' 22.03.2017 : Version 1.1
'              - Delayzeit zwischen Eingabe und nächstem Level verkürzt
'              - WinWin-Melodie wird nun auch im Bypassmodus gespielt
'              - Bei Ausgabe der Koordinaten zyklische Tastenabfrage während
'                Hinweis Taste hinzugefügt wegen besserer Bedienbarkeit
'
'############################################################################
```

Software 3: Source-Code des Projekts Senso

14 Interessantes und wichtige Links

14.1 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, mitp 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

14.2 Internet

14.2.1 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
- Baulelemente

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
Spillbähnstraße 19a
53844 Troisdorf

Tel. 02241-3010465
Fax 02241-3010469

eMail gselectronic@gsel.de
URL: http://www.gsel.com/
- Platinenservice
- 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

14.2.2 ATmega SW und HW-Lösungen

Laser & Co. Solutions GmbH
URL: http://www.myavr.de
- Bausätze zum ATmega8
- SW-Lösungen zum Selbststudium
- Dokumente

14.2.3 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 bereitstelle, 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-grundschaltungen

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

14.3 Das STK500 und STK501

14.3.1 Das STK500 mit STK501 (ATMEL)

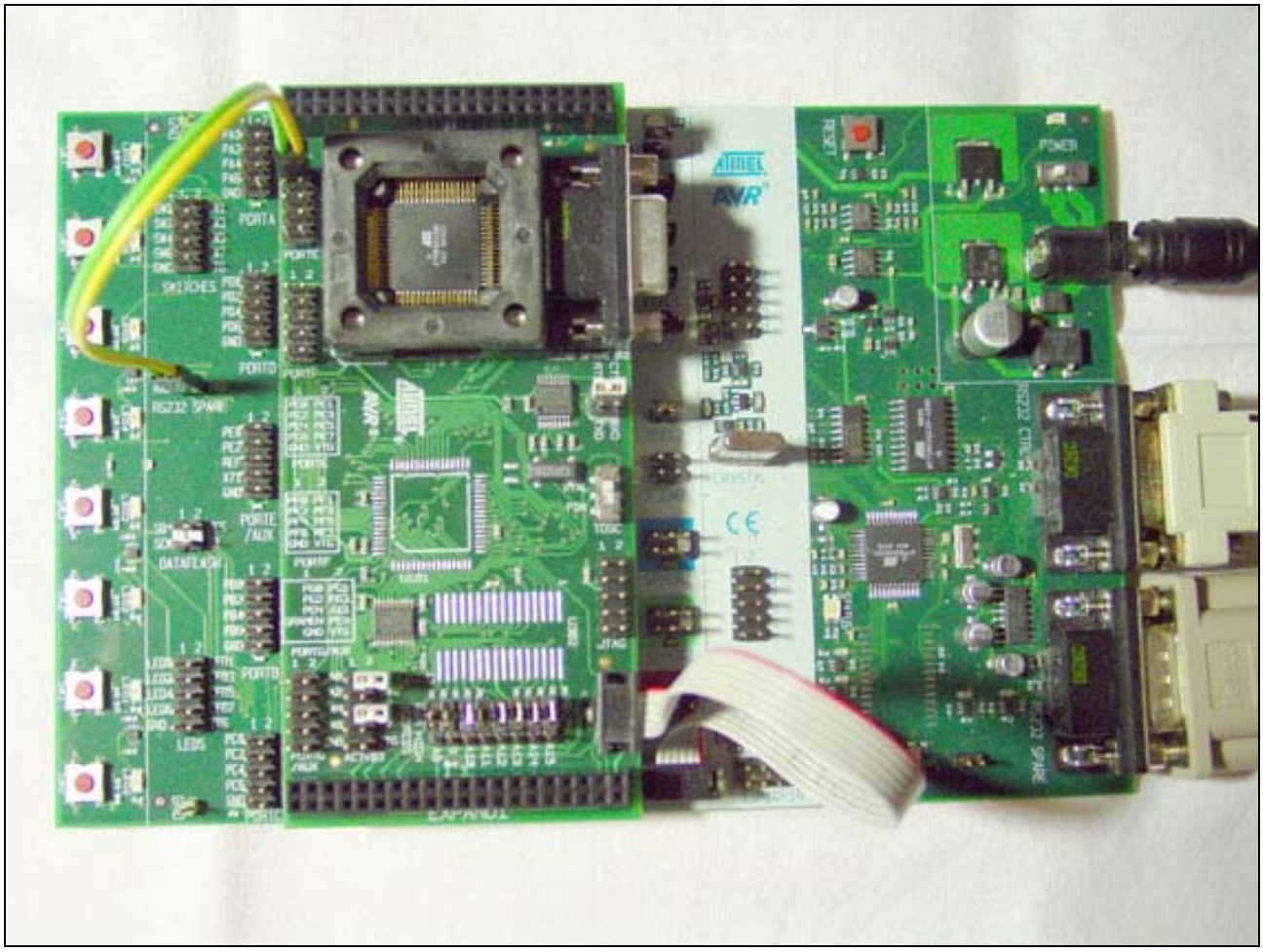


Abbildung 43: Das STK500 mit STK501 (ATMEL)

14.3.2 STK501 anschließen für RS232-Kommunikation

Der 9-polige Sub-D Stecker mit der Beschriftung „RS232 Spare“ wird mit einem freien COM-Port des PC verbunden.

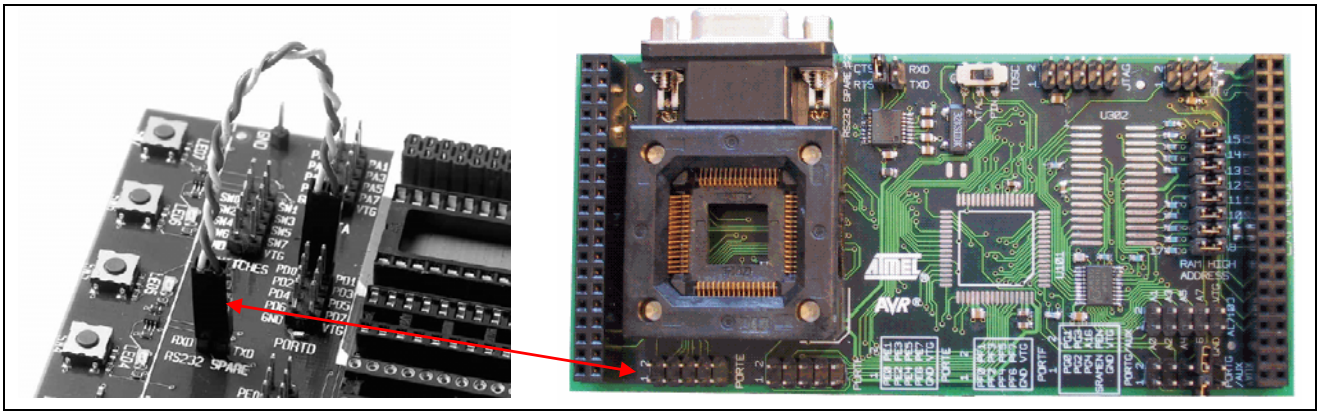


Abbildung 44: RS232-Verbindung STK500 und STK501

Der 9pol. RS232 Spare Stecker ist über einen Pegelwandler mit den beiden Pins im IO-Bereich verbunden (heist auch RS232 Spare). Diese zwei Pins (RxD und TxD) müssen mit dem mitgelieferten 2pol Kabel mit den Pins PE0 und PE1 (PortE) auf dem STK501 verbunden werden.

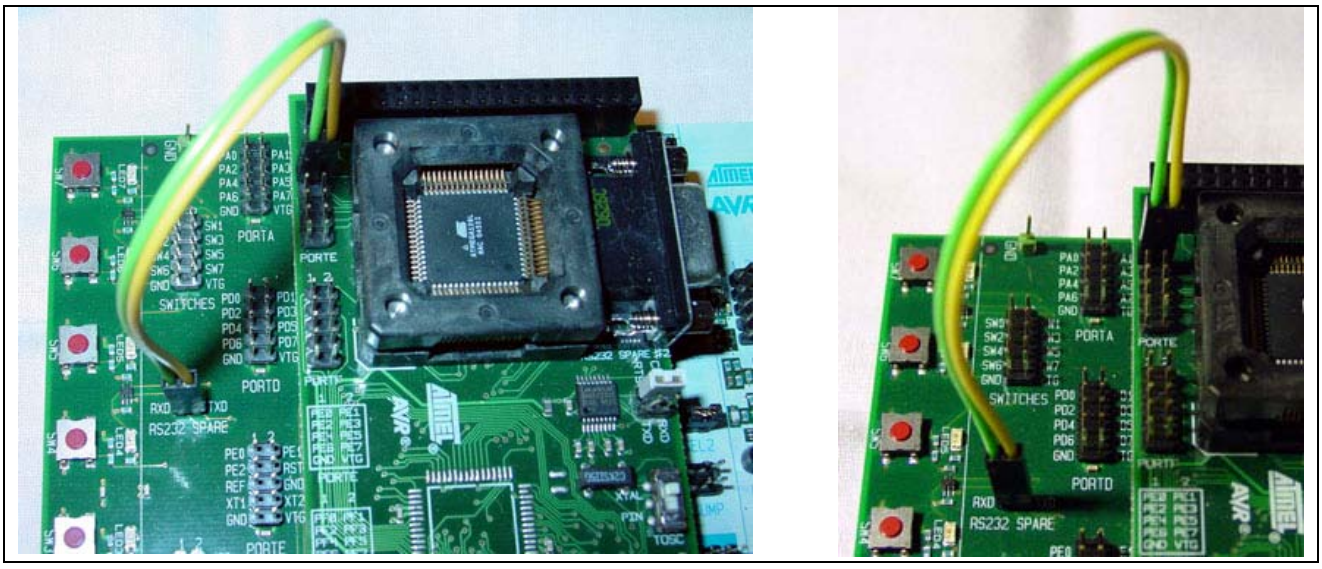


Abbildung 45: RS232-Verbindung STK500 mit STK501

14.4 Inbetriebnahme AVR ISP mkII

Für die Inbetriebnahme des STK500 + STK501 muss der RESET-Jumper auf dem STK500-Board entfernt werden (Dieser Hinweis stammt aus der Dokumentation des mkII).

Alle weiteren Einstellungen zeigen die folgenden Dialog-Boxen des AVR Studios 4.0:

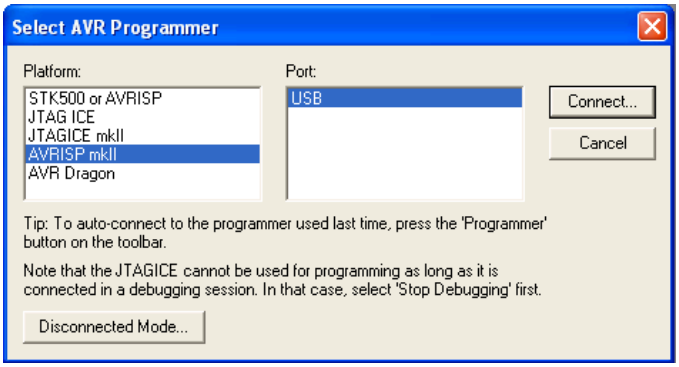


Abbildung 46: AVR Studio Programmer Selection

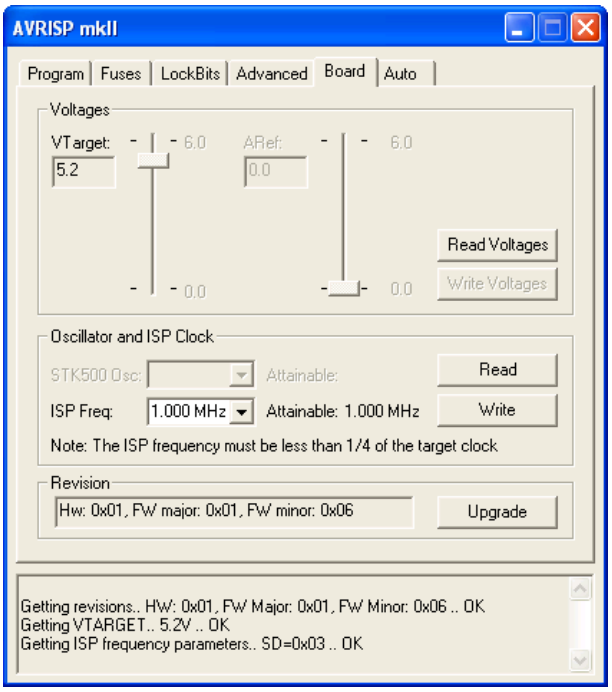


Abbildung 47: AVR Studio Board-Settings

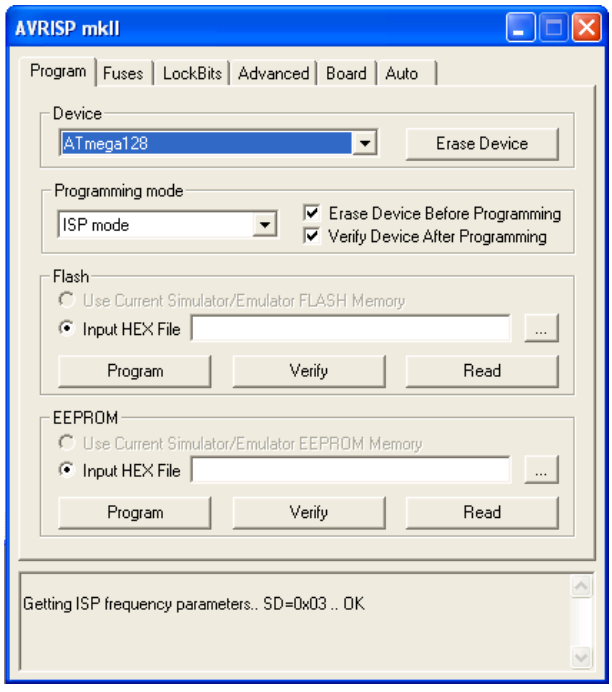


Abbildung 48: AVR Studio Program Settings

Hinweis: Die ISO Frequenz muss 1/4 bzw. 1/6 der Taktfrequenz des ATmegas betragen. Bei 16 MhZ des ATmega128 wurde der ISP Prommer erfolgreich mit 1,000 MHz in Betrieb genommen.

15 Entwicklungsbegleitende Notizen und Informationen

15.1 Projektcheckliste für AVR Systemdesigns

Diese Checkliste beinhaltet eingige grundlegende Regeln beim Design mit AVR Mikrokontrollern.

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

Dies sind zusammengefasst in Kürze:

15.1.1 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. AVRs 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!

15.1.2 Spannungsversorgung richtig angeschlossen?

Der AVCC-Pin ist der Versorgungsanschluss für den AD-Wandler und den zugehörigen Port. Er ist nicht an allen AVRs 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 AVRs 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.

15.1.3 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 AVRs 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.

15.1.4 Alle Ground-Anschlüsse beschaltet?

Bei AVRs mit mehreren Ground-Anschlüssen müssen alle Anschlüsse beschaltet werden.
Siehe

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

15.2 Datenblätter

T 113A xx: Miniatur-Drucktaster, Ein 0,5A-250VAC, xx

按钮开关

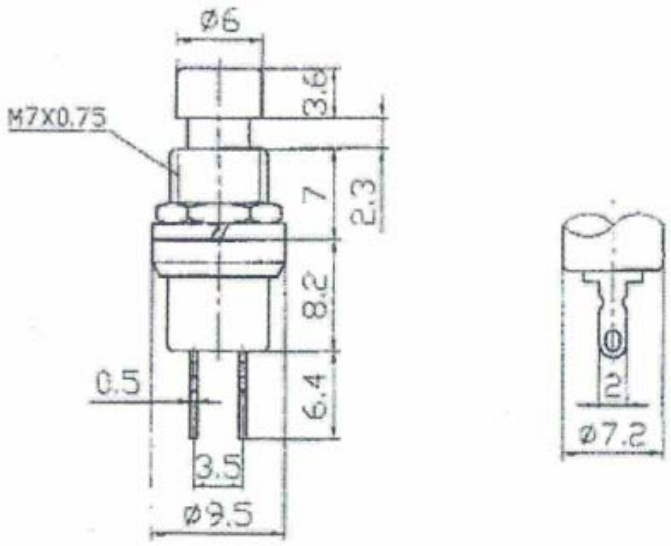
Push-Button Switch

T-113 A

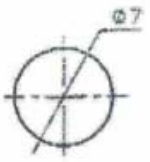
SPECIFICATION

Rating	3A 125 V AC; 1 A 250 V AC
Contact R:	20mOhm max
Insulation R	500 V DC 100 MOhm min
Dielectric Strength	1 Minute 1000V AC
Operating Temperature	-25°C ~+85°C
Electrical Life	50. 000 Cycles

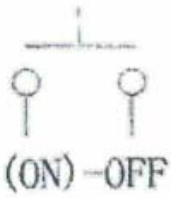
外型图 (Outline Drawing)



安装尺寸 (Mounting)



电路图 (Circuitry)



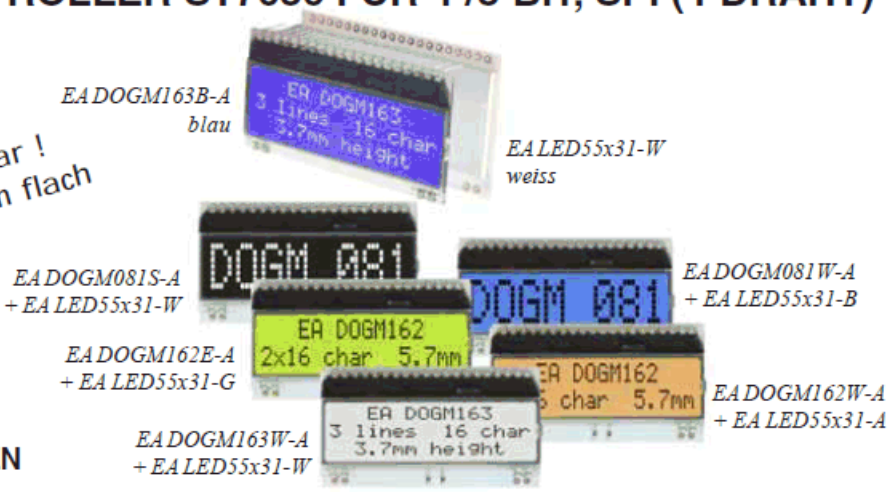
LCD Display
Electronic Assembly EA DOG162B-A

6.2005

EA DOG-M

DOG-SERIE 3,3V
SUPER FLACH / 55x27mm
INKL. KONTROLLER ST7036 FÜR 4-/8-BIT, SPI (4-DRAHT)

ab 1 Stück lieferbar !
auch mit LED: 5,8mm flach



TECHNISCHE DATEN

* KONTRASTREICHE LCD-SUPERTWIST ANZEIGE
* OPTIONALE LED-BELEUCHTUNGSKÖRPER INVERSCHIEDENEN FARBEN
* 1x8, 2x16 UND 3x16 ZEICHEN MIT 12,0 mm / 5,6 mm UND 3,6 mm SCHRIFT
* KONTROLLER ST 7036 FÜR 4-BIT, 8-BIT UND SPI (4-DRAHT) INTERFACE
* SPANNUNGSVERSORGUNG+3,3V ODER +5V SINGLE SUPPLY (typ. 250µA)
* KEINE ZUS. SPANNUNGEN ERFORDERLICH
* BETRIEBSTEMPERATURBEREICH -20..+70°C
* LED-HINTERGRUNDBELEUCHTUNG3..80mA@3,3V oder 2..40mA@5V
* KEINE MONTAGE ERFORDERLICH: EINFACH NUR IN PCB EINLÖTEN

BESTELLBEZEICHNUNG

LCD-MODUL 1x8 - 11,97mm	**EA DOGM081x-A**
LCD-MODUL 2x16 - 5,57mm	**EA DOGM162x-A**
LCD-MODUL 3x16 - 3,65mm	**EA DOGM163x-A**

x: *B = blauer Hintergrund*
E = Gelb/grüner Hintergrund
L = Gelb/grüner Hintergrund (nicht beleuchtbar)
S = schwarzer Hintergrund
W = weisser Hintergrund

LED-BELEUCHTUNG WEISS	**EA LED55X31-W**
LED-BELEUCHTUNG GELB/GRÜN	**EA LED55X31-G**
LED-BELEUCHTUNG BLAU	**EA LED55X31-B**
LED-BELEUCHTUNG ROT	**EA LED55X31-R**
LED-BELEUCHTUNG AMBER	**EA LED55X31-A**
BUCHSENLEISTE 4,8mm HOCH (2 STÜCK ERFORDERLICH)	**EA FL-20P**

ELECTRONIC ASSEMBLY GMBH
making things easy
LOCHHAMER SCHLAG 17 · D-82166 GRÄFELFING
TEL 089/8541991 · FAX 089/8541721 · http://www.lcd-module.de

EA DOG-M

ELECTRONIC ASSEMBLY

EA DOG SERIE

Mit der EA DOG-Serie präsentiert ELECTRONIC ASSEMBLY die weltweite 1. Displayserie, welche ohne zusätzlicher Hilfsspanung an 3,3V Systemen lauffähig sind. Selbstverständlich ist auch der Betrieb an einem herkömmlichen 5V System möglich.
Anders als bei üblichen LCD-Modulen bestellen Sie hier die Anzeige und die entsprechende Beleuchtung separat. Dadurch ergeben sich mannigfaltige Kombinationsmöglichkeiten.
Konzipiert für kompakte Handgeräte bietet diese moderne LCD-Serie mit und ohne Beleuchtung eine Reihe echter Vorteile:

* extrem kompakt mit 55x31mm bei marktüblicher Schriftgröße von 5,57mm (2x16) !
* superfach mit 2,0mm unbeleuchtet bzw. 5,8mm inkl. LED-Beleuchtung
* 4-Bit, 8-Bit und SPI Interface (4-Draht)
* nur typ. 250µA Stromverbrauch in vollem Betrieb (LED-Beleuchtung weiss ab 3mA)
* simple Montage durch einfaches Einlöten
* verschiedenste (63) Designvarianten ab 1 Stück lieferbar

KONTRASTEINSTELLUNG

Für alle Displays der EA DOG- Serie ist der Kontrast per Befehl einstellbar. Dies erfolgt über die Bits C0..C5 in den Befehlen "Contrast Set" und "Power/Icon Control/Contrast Set". In der Regel wird der Kontrast einmalig eingestellt und dann - dank integrierter Temperaturkompensation - über den gesamten Betriebstemperaturbereich (-20..+70°C) konstant gehalten.
Insgesamt benötigen die Displays selbst im 3,3V Betrieb keine zusätzliche negative Spannung !

LED-BELEUCHTUNGEN

Zur individuellen Hintergrundbeleuchtung sind 5 verschiedene Varianten erhältlich: weiss, gelb/grün, blau, rot und amber.

Es stehen jeweils 2 separate LED-Pfade zur Verfügung, welche zur optimalen Anpassung an die Systemspannung parallel oder in Serie geschaltet werden können. Dadurch sind alle Beleuchtungen alternativ mit 5V oder auch mit 3,3V zu betreiben!

Der Betrieb der Hintergrundbeleuchtung erfordert einen externen Vorwiderstand zur Strombegrenzung. Dieser errechnet sich aus R=U/I; die Werte entnehmen Sie aus den Tabellen nebenan. Für eine optimale Lebensdauer empfehlen wir den Einsatz einer Stromquelle.

Die Lebensdauer der gelb/grünen, roten und amber-farbigen Beleuchtung beträgt 100.000 Stunden, die der weißen und blauen Beleuchtung deutlich darunter.

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

yellow/green EA LED55x31-G	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	2,2 V	80 mA	14 ohm	35 ohm
Connected in series	4,4 V	40 mA	-	7,5 ohm

white EA LED55x31-W	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	3,2 V	60 mA	1,6 ohm	30 ohm
Connected in series	6,4 V	30 mA	-	-

blue EA LED55x31-B	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	3,2 V	60 mA	1,6 ohm	30 ohm
Connected in series	6,4 V	30 mA	-	-

amber EA LED55x31-A	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	2,4 V	80 mA	11 ohm	32 ohm
Connected in series	4,8 V	40 mA	-	5 ohm

red EA LED55x31-R	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	2,1 V	80 mA	15 ohm	36 ohm
Connected in series	4,2 V	40 mA	-	20 ohm

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 4 Pins für die Beleuchtung auch von oben verlötet werden müssen.
Achtung: Auf dem Display befinden sich 2 Schutzfolien und auf der Beleuchtung jeweils eine. Diese sollen während oder nach der Fertigung entfernt werden.

EA DOG-M

ELECTRONIC ASSEMBLY

5 VERSCHIEDENEN 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
	STN pos. gelb/grün transmissiv	Beleuchtungskörper erforderlich	auch bei abgeschalteter Bel. lesbar	dunkelgrün auf gelb/grün	schwarz auf gelb/grün	gelb/grün
	STN neg. blau transmissiv	nur beleuchtet zu verwenden	---	---	Beleuchtungsfarbe auf blauem Hintergrund	weiß, gelb/grün
	FSTN neg. transmissiv	nur beleuchtet zu verwenden	---	---	Beleuchtungsfarbe auf schwarzem Hintergrund	weiß
	STN pos. gelb/grün reflektiv	keine Beleuchtung möglich	ohne Beleuchtung bestens lesbar	dunkelgrün auf gelb/grün	---	---

3 VERSCHIEDENE DISPLAYS

Alle 3 Displays sind in den oben genannten Technologien erhältlich:

5 VERSCHIEDENE BELEUCHTUNGEN

Zur Anpassung an unterschiedlichste Designs stehen 5 verschiedene Beleuchtungsfarben zur Auswahl. Die effektivste und gleichzeitig hellste Beleuchtung ist die weiße EA LED55x31-W.

EA LED55x31-W
Weiß

EA LED55x31-G
Gelb/Grün

EA LED55x31-B
Blau

EA LED55x31-R
Rot

EA LED55x31-A
Amber

Wenn Sie auf dieser Seite nur schwarz/weiß Darstellungen sehen: das farbige Datenblatt finden Sie im Internet unter http://www.lcd-module.de/deu/pdf/doma/dogm.pdf

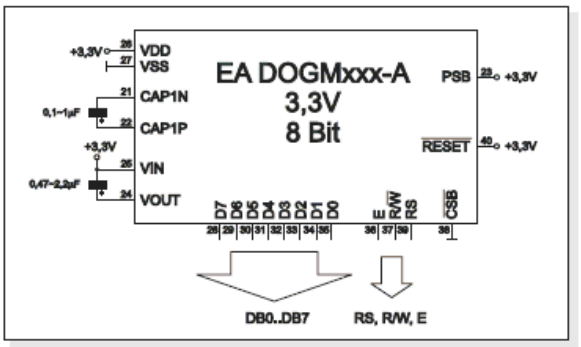
3

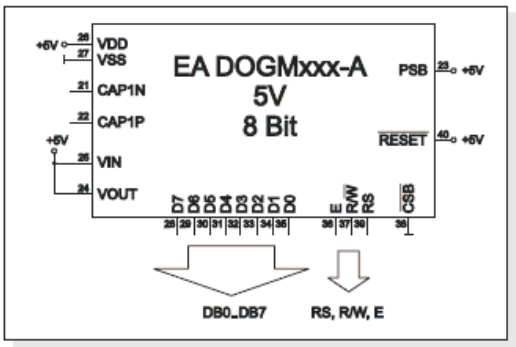
EA DOG-M

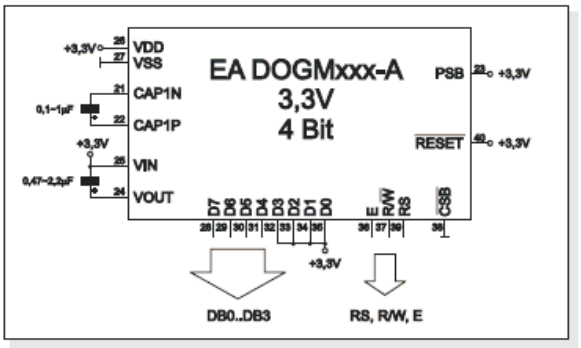
ELECTRONIC ASSEMBLY

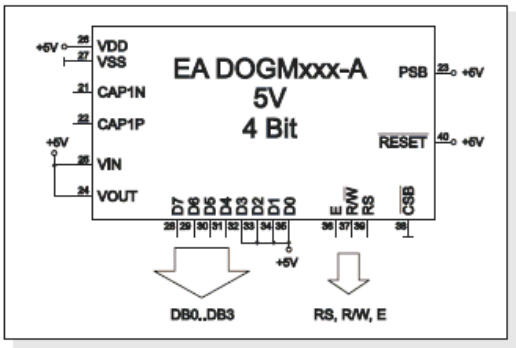
APPLIKATIONSBEISPIELE

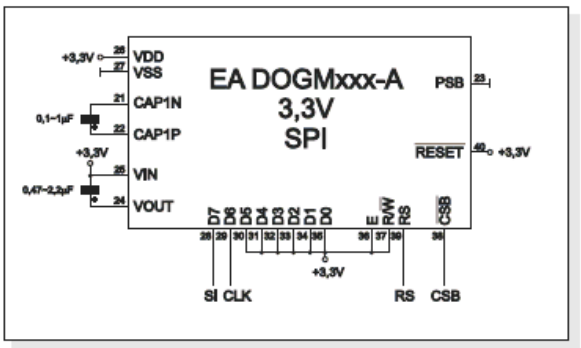
Je nach Interface und Versorgungsspannung ist eine unterschiedliche Beschaltung zu wählen. Im 3,3V Betrieb sind 2 zusätzliche Kondensatoren erforderlich.
Bitte beachten Sie, dass aufgrund der COG-Technik die Strombelastbarkeit der Ausgänge begrenzt ist. Es kann dadurch bei größerer Buslast zu Signalverschleifungen und unsauberen Pegeln kommen. Im Zweifelsfall sind zusätzliche Pull-Down Widerstände (8051) erforderlich, oder es müssen zusätzliche Waits/NOP's eingefügt werden.

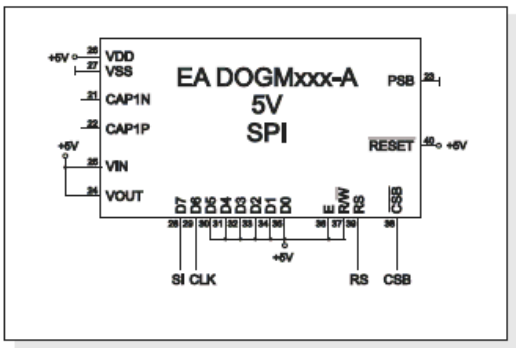












Technische Änderung vorbehalten. Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

EA DOG-M

ELECTRONIC ASSEMBLY

ZEICHENSATZ

Der unten abgebildete Zeichensatz ist integriert. Zusätzlich können 8 eigene Zeichen frei definiert werden.

Eine detaillierte Beschreibung des hier integrierten Kontrollers ST7036 finden Sie im Internet unter http://www.lcd-module.de/eng/pdf/zubehoer/st7036.pdf

EA DOG-M

ELECTRONIC ASSEMBLY

BEFEHLSTABELLEN

Instruction	Instruction Code										Description	Instruction Execution Time		
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		OSC= 380kHz	OSC= 540kHz	OSC= 700kHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.08 ms	0.76 ms	0.59 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.08 ms	0.76 ms	0.59 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	26.3 µs	18.5 µs	14.3 µs
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	26.3 µs	18.5 µs	14.3 µs
Function Set	0	0	0	0	1	DL	N	DH	IS2	IS1	DL: interface data is 8/4 bits N: number of line is 2/1 DH: double height font IS[2:1]: instruction table select	26.3 µs	18.5 µs	14.3 µs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	26.3 µs	18.5 µs	14.3 µs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0	0	0
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs
Instruction table 0(IS[2:1]=[0,0])														
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	X	X	S/C and R/L: Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	26.3 µs	18.5 µs	14.3 µs
Set CGRAM	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	26.3 µs	18.5 µs	14.3 µs
Instruction table 1(IS[2:1]=[0,1])														
Bias Set	0	0	0	0	0	1	BS	1	0	FX	BS=1:1/4 bias BS=0:1/5 bias FX: fixed on high in 3-line application and fixed on low in other applications.	26.3 µs	18.5 µs	14.3 µs
Set ICON Address	0	0	0	1	0	0	AC3	AC2	AC1	AC0	Set ICON address in address counter.	26.3 µs	18.5 µs	14.3 µs
Power/ICON Control/ Contrast Set	0	0	0	1	0	1	Ion	Bon	C5	C4	Ion: ICON display on/off Bon: set booster circuit on/off C5,C4: Contrast set for internal follower mode.	26.3 µs	18.5 µs	14.3 µs
Follower Control	0	0	0	1	1	0	Fon	Rab 2	Rab 1	Rab 0	Fon: set follower circuit on/off Rab2~0: select follower amplified ratio.	26.3 µs	18.5 µs	14.3 µs
Contrast Set	0	0	0	1	1	1	C3	C2	C1	C0	Contrast set for internal follower mode.	26.3 µs	18.5 µs	14.3 µs
Instruction table 2(IS[2:1]=[1,0])														
Double Height Position Select	0	0	0	0	0	1	UD	X	x	x	UD: Double height position select	26.3 µs	18.5 µs	14.3 µs
Reserved	0	0	0	1	X	X	X	X	X	X	Do not use (reserved for test)	26.3 µs	18.5 µs	14.3 µs

Eine detaillierte Beschreibung des hier integrierten Kontrollers ST7036 finden Sie im Internet unter http://www.lcd-module.de/eng/pdf/zubehoer/st7036.pdf

EA DOG-M

ELECTRONIC ASSEMBLY

INITIALISIERUNGSBEISPIELE

EA DOGM081

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 3.3V
EA DOGM081

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	0	1	1	0	0	1	$31	8-Bit Datenlänge, 1 Zeile, Instruction table 1
Bias Set	0	0	0	0	0	1	0	1	0	0	$14	BS: 1/5, 1-zeiliges LCD
Power Control	0	0	0	1	0	1	0	1	0	1	$55	Booster ein, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	1	0	1	$6D	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	1	1	0	0	$7C	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Initialisierung für 3,3V

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 5V
EA DOGM081

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	0	0	0	1	$31	8-Bit Datenlänge, 1 Zeile, Instruction table 1
Bias Set	0	0	0	0	0	1	1	1	0	0	$1C	BS: 1/4, 1-zeiliges LCD
Power Control	0	0	0	1	0	1	0	0	0	1	$51	Booster aus, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	0	1	0	$6A	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	0	1	0	0	$74	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Initialisierung für 5V

EA DOGM162

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 3.3V
EA DOGM162

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	0	1	0	0	$14	BS: 1/5, 2-zeiliges LCD
Power Control	0	0	0	1	0	1	0	1	0	1	$55	Booster ein, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	1	0	1	$6D	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	1	0	0	0	$78	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Initialisierung für 3,3V

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 5V
EA DOGM162

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	1	1	0	0	$1C	BS: 1/4, 2-zeiliges LCD
Power Control	0	0	0	1	0	1	0	0	1	0	$52	Booster aus, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	0	0	1	$69	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	0	1	0	0	$74	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Initialisierung für 5V

EA DOGM163

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 3.3V
EA DOGM163

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	0	1	0	1	$15	BS: 1/5, 3-zeiliges LCD
Power Control	0	0	0	1	0	1	0	1	0	1	$55	Booster ein, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	1	1	0	$6E	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	0	0	1	0	$72	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

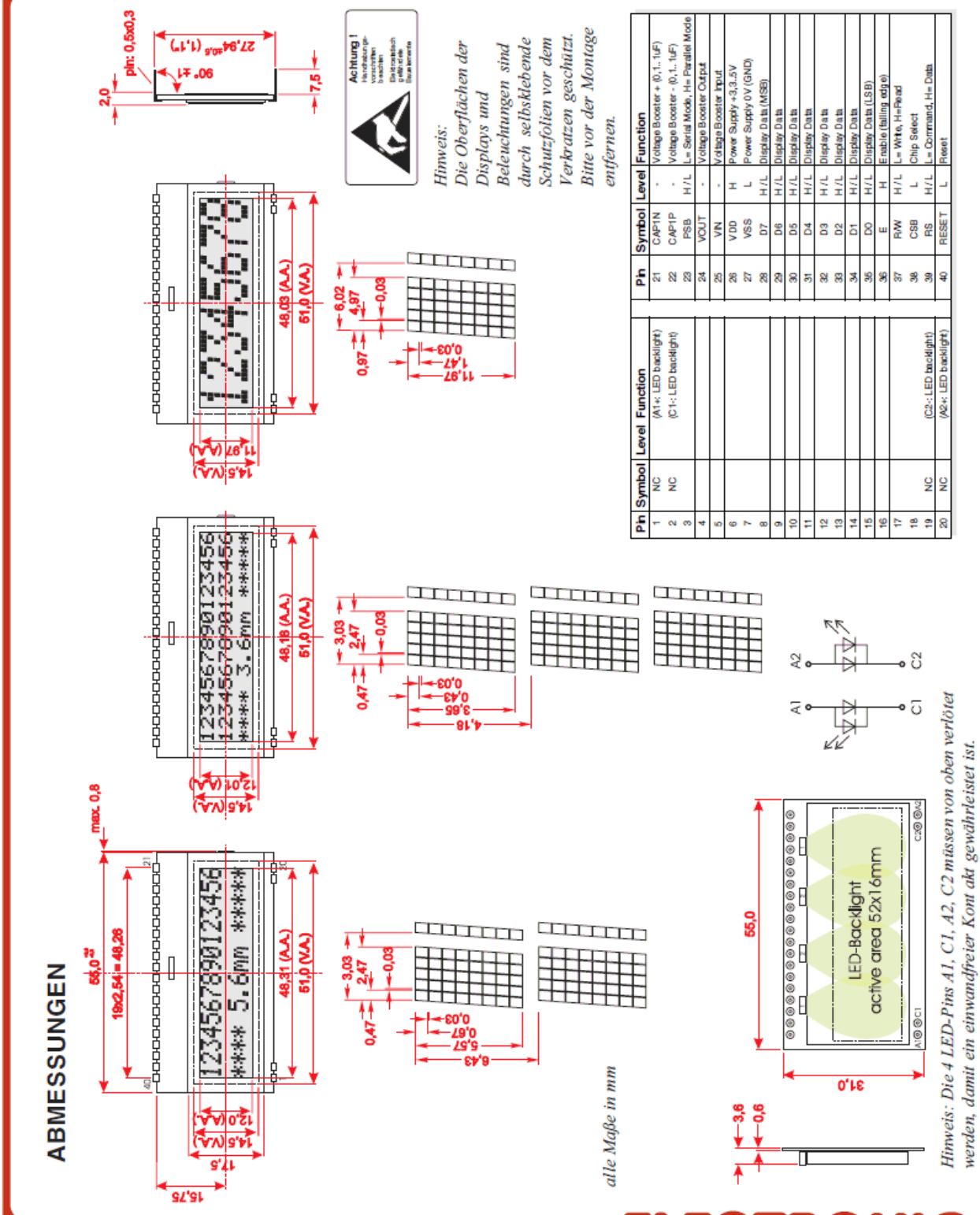
Initialisierung für 3,3V

INITIALISIERUNGSBEISPIEL FÜR 8 Bit / 5V
EA DOGM163

Befehl	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Bemerkung
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8-Bit Datenlänge, 2 Zeilen, Instruction table 1
Bias Set	0	0	0	0	0	1	1	1	0	1	$1D	BS: 1/4, 3-zeiliges LCD
Power Control	0	0	0	1	0	1	0	0	0	0	$50	Booster aus, Kontrast C5, C4setzen
Follower Control	0	0	0	1	1	0	1	1	0	0	$6C	Spannungsfolger und Verstärkung setzen
Contrast Set	0	0	0	1	1	1	1	1	0	0	$7C	Kontrast C3, C2, C1 setzen
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	Display ein, Cursor ein, Cursor blinken
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	Display löschen, Cursor Home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	Cursor Auto-Increment

Initialisierung für 5V

EA DOG-M



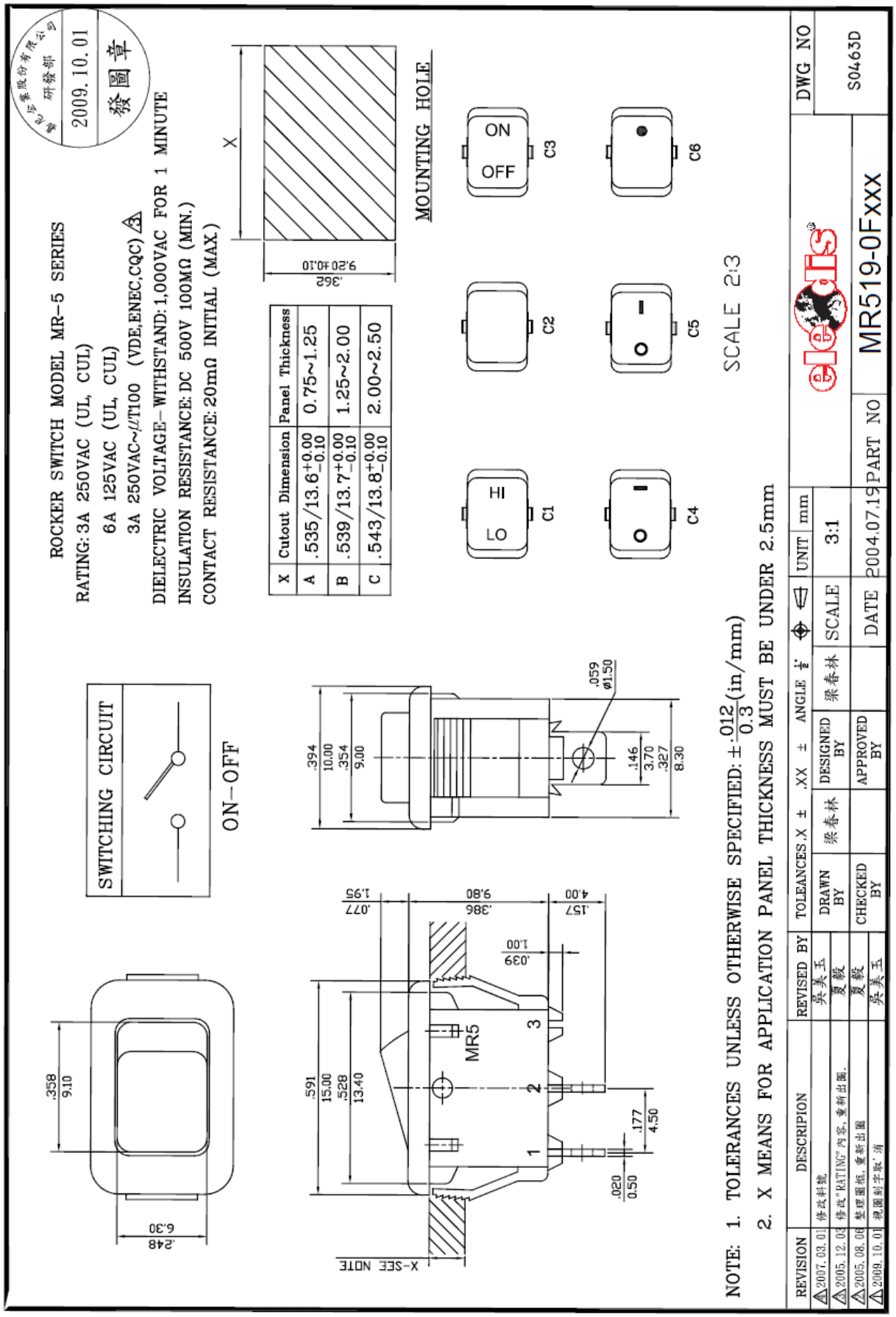
Pin	Symbol	Level	Function
1	NC		(A1+: LED backlight)
2	NC		(C1-: LED backlight)
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18	NC		(C2-: LED backlight)
19	NC		(A2+: LED backlight)

Pin	Symbol	Level	Function
21	CAP1N	-	Voltage Booster + (0,1..1uF)
22	CAP1P	-	Voltage Booster - (0,1..1uF)
23	PSB	H/L	L= Serial Mode, H= Parallel Mode
24	VOUT	-	Voltage Booster Output
25	VIN	-	Voltage Booster Input
26	VDD	H	Power Supply +3,3..5V
27	VSS	L	Power Supply 0V (GND)
28	D7	H/L	Display Data (MSB)
29	D6	H/L	Display Data
30	D5	H/L	Display Data
31	D4	H/L	Display Data
32	D3	H/L	Display Data
33	D2	H/L	Display Data
34	D1	H/L	Display Data
35	D0	H/L	Display Data (LSB)
36	E	H	Enable (falling edge)
37	R/W	H/L	L= Write, H=Read
38	CSB	L	Chip Select
39	RS	H/L	L= Command, H= Data
40	RESET	L	Reset

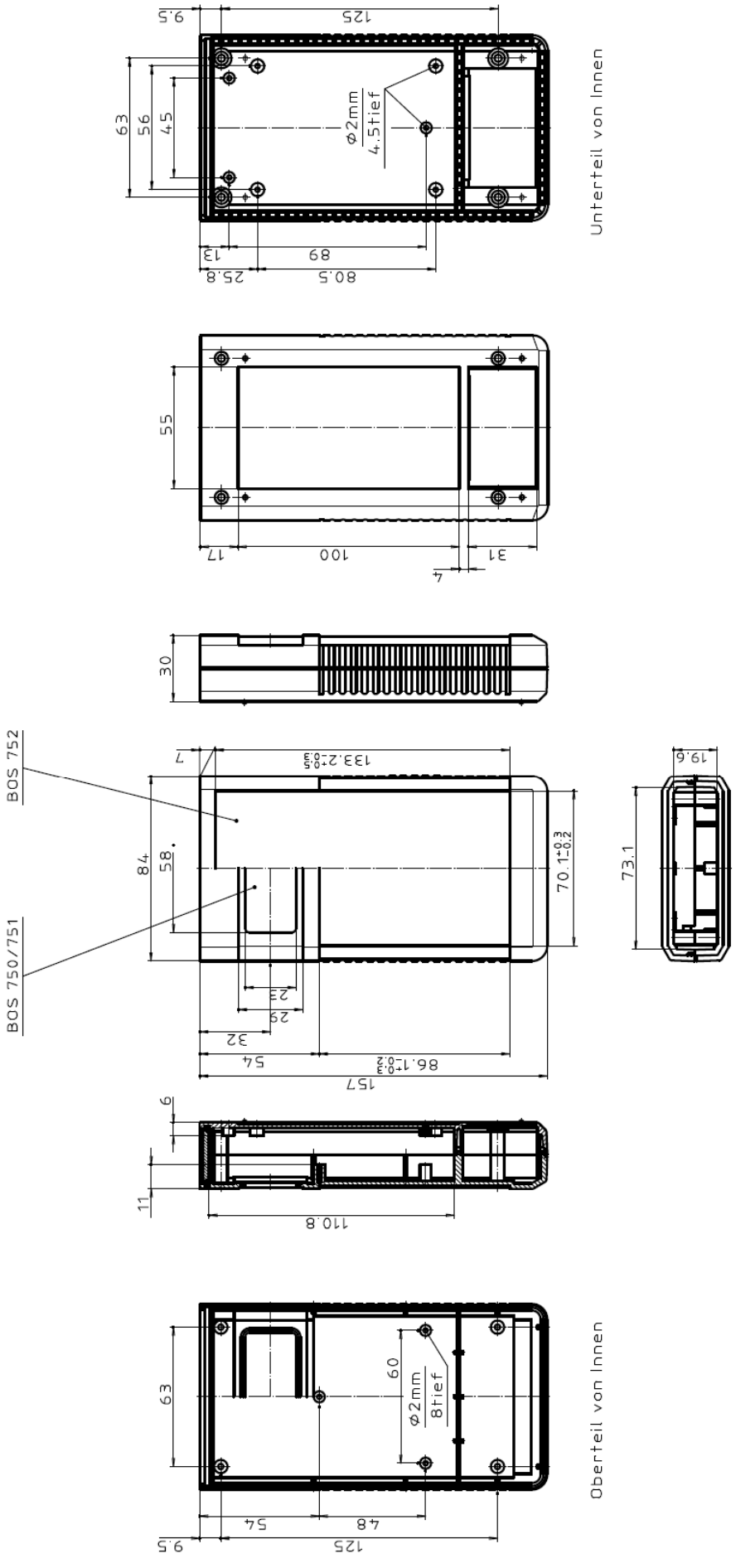
LOCHHAMER SCHLAG 17 · D-82166 GRÄFELFING
TEL 089/8541991 · FAX 089/8541721 · http://www.lcd-module.de

ELECTRONIC ASSEMBLY
making things easy

Hauptschalter:



Gehäuse:


Unterteil von Innen
Oberteil von Innen
BOS 752
BOS 750/751
Ø2mm
4.5tief
8tief
125
80.5
89
63
56
45
55
100
17
31
30
133.2
84
58
73.1
19.6
70.1
157
86.1
54
32
29
23
110.8
48
60