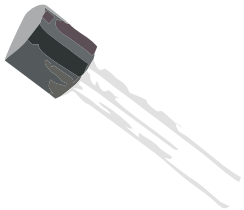
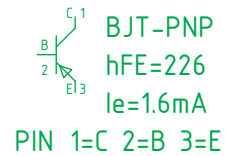
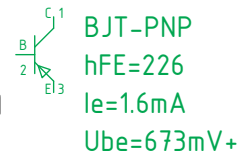


Beschreibung Component Tester

Ein Gerät zur Bestimmung und Testen
von elektronischen Bauteilen,
mit der Möglichkeit der PC
Kommunikation



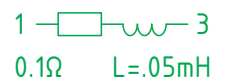
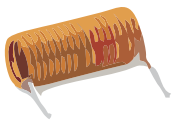
Version 1.38m



Markus Reschke

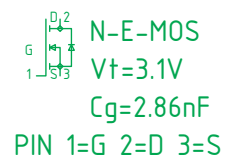
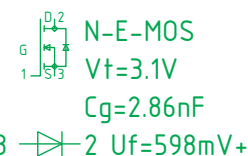
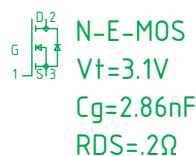
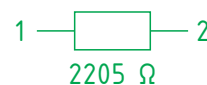
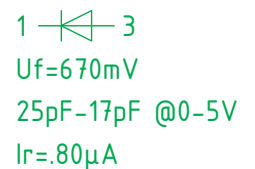
© 2012

madires@theca-tabellaria.de

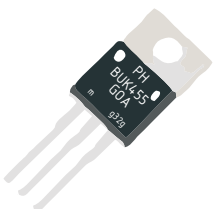


Zusammengestellt
von bm-magic

16. Januar 2020



13.05.2019/MOR



Inhaltsverzeichnis

1	Über den Tester	7
1.1	Über den Tester	7
1.2	Sicherheitshinweise	7
1.3	Lizenz	7
1.3.1	Zusätzliche HinweiseLizenz	7
1.4	Unterschiede	7
1.5	Quellcode	8
1.6	Firmware übersetzen	8
2	Hardware	9
2.1	Hardware-Optionen	9
2.2	Und natürlich die Software-Optionen	9
2.3	Verfügbare Sprachen	10
2.4	MCU Wahl	10
2.5	Anzeige Wahl	10
2.6	Busse & Schnittstellen	10
2.6.1	I2C/SPI	10
2.6.2	Serielle TTL-Schnittstelle	11
2.6.3	One Wire	11
2.7	LCD-Module	11
2.7.1	HD44780	13
2.7.2	ILI9163	13
2.7.3	ILI9341/ILI9342	13
2.7.4	PCD8544	14
2.7.5	PCF8814	14
2.7.6	SSD1306	14
2.7.7	ST7036 (ungetestet)	15
2.7.8	ST7565R	15
2.7.9	ST7735	15
2.7.10	ST7920	16
2.7.11	STE2007/HX1230	16
2.7.12	VT100 Terminal	16
2.8	Tasten und Eingabeoptionen	16
2.8.1	Die Test-Taste	16
2.8.2	Drehencoder (Hardware-Option)	17
2.8.3	Mehr/Weniger-Tasten (Hardware-Option)	17
2.8.4	Touch-Screen (Hardware-Option)	17
2.8.5	Kommunikation mit PC	18
2.8.6	Serielle Ausgabe	18
2.8.7	Automatisierung	18
2.8.8	VT100-Ausgabe	18

3	Bedienung	19
3.0.1	Einschalten	19
3.0.2	Bauteilesuche	19
3.0.3	Batterieüberwachung	19
3.0.4	Ausschalten	19
3.0.5	Menü	20
3.1	Menü	21
3.1.1	PWM-Generator	21
3.1.2	Einfache PWM	21
3.1.3	Erweiterte PWM	21
3.1.4	Rechteck-Signalgenerator	21
3.1.5	Zenertest (Hardware-Option)	21
3.1.6	ESR-Messung	22
3.1.7	Kondensatorleckstrom	22
3.1.8	R/L-Monitor	22
3.1.9	C-Monitor	22
3.1.10	Frequenzzähler (Hardware-Option)	22
3.1.11	Einfacher Zähler	22
3.1.12	Erweiterter Zähler	23
3.1.13	Ereigniszähler (Hardware-Option)	23
3.1.14	Drehencoder	23
3.1.15	Kontrast	23
3.1.16	Detektor/Decoder für IR-Fernbedienungen	24
3.1.17	IR-Fernbedienung	25
3.1.18	Opto-Koppler-Test	26
3.1.19	Modellbau-Servo-Test	26
3.1.20	DS18B20-Temperatursensor	27
3.1.21	DHTxx-Sensoren	27
3.1.22	Selbsttest	28
3.1.23	Selbstabgleich	28
3.1.24	Speichern/Laden	29
3.1.25	Werte anzeigen	29
3.1.26	Ausschalten	29
3.1.27	Exit	29
4	Messdetail	30
4.0.1	Widerstände	30
4.0.2	Kondensatoren	30
4.0.3	Induktivitäten	31
4.0.4	Bauteile entladen	31
4.0.5	ADC Oversampling	31
4.0.6	V _{BE} von Bipolartransistoren	31
4.0.7	Ergebnisanzeige	31
4.0.8	Zusätzliche Hinweise	32
4.0.9	Hilfe	33
4.0.10	Firmware-Änderungen	33
5	Code	34
5.1	Makefile	34
5.1.1	MCU-Typ	34
5.1.2	MCU-Taktfrequenz	34
5.1.3	Oszillator-Typ	34
5.1.4	Avrdude MCU-Typ	35
5.1.5	Avrdude ISP-Programmierer	35

5.2	config.h	36
5.2.1	Hardware Bedienung	36
5.2.2	Software Optionen	37
5.2.3	Benutzerschnittstelle	39
5.2.4	Energieverwaltung	40
5.2.5	Messeinstellungen und Offsets	41
5.2.6	Busse	43
5.3	Config_<MCU>.h	44
5.3.1	LCD-Module	44
5.3.2	Port- und Pinbelegung	44
5.3.3	Busse	47
6	Sammlung von Einstellungen	48
6.0.1	DIY Kit „AY-AT“	48
6.0.2	M12864 DIY Transistor Tester	49
6.0.3	T3/T4	50
6.0.4	GM328	50
6.0.5	Fish8840 TFT	51
6.0.6	Multifunktionstester TC-1	51
6.0.7	Hiland M644	53
7	Programmieren des Component Testers	55
7.1	Konfigurieren des Component Testers	55
7.2	Programmierung des Mikrocontrollers	55
7.2.1	Betrieb System Linux	55
7.2.2	Benutzung unter Linux	56
7.2.3	Programm Pakete installieren	56
7.2.4	Download der Quellen	56
7.2.5	Benutzung der Schnittstellen	56
7.2.6	Gruppen Mitgliedschaft	57
7.2.7	Arbeitsumgebung	57
8	Fernsteuerung	59
8.1	Fernsteuerungskommandos	59
8.1.1	ERR	59
8.1.2	OK	59
8.1.3	N/A	59
8.2	Basiskommandos	59
8.2.1	VER	59
8.2.2	OFF	59
8.3	Testkommandos	59
8.3.1	PROBE	59
8.3.2	COMP	59
8.3.3	MSG	59
8.3.4	QTY	59
8.3.5	NEXT	59
8.3.6	TYPE	59
8.3.7	HINT	60
8.3.8	PIN	60
8.3.9	R	60
8.3.10	C	60
8.3.11	L	60
8.3.12	ESR	60
8.3.13	I_1	60
8.3.14	V_F	61

8.3.15	V_F2	61
8.3.16	C_D	61
8.3.17	I_R	61
8.3.18	R_BE	61
8.3.19	h_FE	61
8.3.20	h_FE_r	61
8.3.21	V_BE	61
8.3.22	I_CEO	61
8.3.23	V_th	61
8.3.24	C_GS	61
8.3.25	R_DS	61
8.3.26	I_DSS	61
8.3.27	C_GE	61
8.3.28	V_GT	61
8.3.29	V_T	61
8.3.30	R_BB	62

9 Der Version Vorschritt 63

9.1	v1.01m 2012-10	63
9.2	v1.02m 2012-11	63
9.3	v1.03m 2012-11	63
9.4	v1.04m 2012-11	63
9.5	v1.05m 2012-11	63
9.6	v1.06m 2013-03	63
9.7	v1.07m 2013-06	64
9.8	v1.08m 2013-07	64
9.9	v1.09m 2013-07	64
9.10	v1.10m 2013-10	64
9.11	v1.11m 2014-03	64
9.12	v1.12m 2014-03	65
9.13	v1.13m 2014-07	65
9.14	v1.14m 2014-08	65
9.15	v1.15m 2014-09	65
9.16	v1.16m 2014-09	65
9.17	v1.17m 2015-02	65
9.18	v1.18m 2015-07	65
9.19	v1.19m 2015-11	66
9.20	v1.20m 2015-12	66
9.21	v1.21m 2016-01	66
9.22	v1.22m 2016-03	66
9.23	v1.23m 2016-07	66
9.24	v1.24m 2016-08	67
9.25	v1.25m 2016-09	67
9.26	v1.26m 2016-12	67
9.27	v1.27m 2017-02	67
9.28	v1.28m 2017-04	68
9.29	v1.29m 2017-07	68
9.30	v1.30m 2017-10	68
9.31	v1.31m 2017-12	68
9.32	v1.32m 2018-02	68
9.33	v1.33m 2018-05	69
9.34	v1.34m 2018-10	69
9.35	v1.35m 2019-02	69
9.36	v1.36m 2019-05	70
9.37	v1.37m 2019-09	70

9.38 v1.38m 2019-12	70
-------------------------------	----

1.1. Über den Tester

Der Component Tester basiert auf dem Projekt von Markus Frejek [1] und [2] und der Weiterführung von Karl-Heinz Kübbeler [3] und [4].

Es ist eine alternative Firmware für die aktuelle Testerschaltung von Karl-Heinz und bietet einige Änderungen der Benutzerschnittstelle und Mess- und Testverfahren.

Während die Firmware von Karl-Heinz die offizielle Version ist und auch ältere ATmega MCUs unterstützt, ist diese Version zum Ausprobieren und Testen neuer Ideen.

Außerdem ist sie auf ATmegas mit mindestens 32kB Flash beschränkt.

Die primären Sprachen für die Bedienung sind Englisch und Deutsch, können aber leicht um weitere Sprachen ergänzt werden.

Hinweis: Bitte den Selbstabgleich bei brandneuen Testern oder nach Firmware-aktualisierung laufen lassen. Oder auch bei Benutzung anderer Messkabel.

1.2. Sicherheitshinweise

Der Component Tester ist kein Multimeter!

Es ist ein einfacher Tester für Bauteile, der Verschiedenes messen kann.

Die Eingänge sind nicht geschützt und werden durch Spannungen über 5V beschädigt.

Den Tester nicht für Schaltungen in Betrieb nutzen, sondern nur für einzelne Bauteile!

Bei Kondensatoren darauf achten, dass sie entladen sind, bevor der Tester angeklemmt wird.

Benutzung auf eigene Gefahr!

1.3. Lizenz

Der Autor der Ursprungsversion hat bzgl. der Lizenzbedingungen nur zwei Vorgaben gemacht.

Zum einen ist das Projekt Open Source, und zum anderen sollen sich kommerzielle Benutzer beim Autor melden.

Unglücklicherweise haben wir, Karl-Heinz und ich, den Autor bisher nicht erreichen können. Um das Problem des Fehlens einer vernünftigen Open-Source-Lizenz zu lösen, habe ich am 1.1.2016 eine Standard-Open-Source-Lizenz ausgewählt, nachdem der ursprüngliche Autor ausreichend Zeit hatte uns seine Wünsche bzgl. einer Lizenz mitzuteilen.

Da diese Firmwareversion eine komplett neue Version ist, die lediglich ein paar Ideen der ursprünglichen Firmware aufgreift, aber keinen Code teilt, sollte dieses Vorgehen gerechtfertigt sein.

Lizenziert unter der EUPL V.1.1

1.3.1. Zusätzliche HinweiseLizenz

Produkt- oder Firmennamen können geschützte Marken der jeweiligen Eigentümer sein.

1.4. Unterschiede

Karl-Heinz hat eine richtig gute Dokumentation für den Tester geschrieben. Unbedingt lesen! Daher zähle ich nur die Hauptunterschiede zur k-Firmware auf:

- Benutzerschnittstelle
 - + Keine Panik! ;-)
 - + Touch Screen
 - + Automatisierung (Fernsteuerkommandos)
- Adaptive Entladefunktion - Widerstandsmessung
 - + zusätzliche Methode für Widerstände <10 Ohm (anstatt ESR-Messung)
- Kapazitätsmessung + ab 5pF

- + zusätzliche Methode für Kondensatoren zwischen $4,7\mu\text{F}$ und $47\mu\text{F}$
 - + anderes Verfahren zur Korrektur/Kompensation
 - kein SamplingADC() für sehr niedrige Kapazitäten oder Induktivitäten
 - Dioden
 - + Erkennungslogik
 - Bipolartransistoren
 - + V_f wird für praxisnahe Testströme interpoliert
 - + Erkennung von Germanium-Transistoren mit höherem Leckstrom
 - JFETs
 - + Erkennung von JFETs mit sehr niedrigem I_{DSS}
 - TRIACs
 - + Erkennung von MT1 und MT2
 - Detektor/Dekoder für IR-Fernsteuerungen
 - IR-Fernbedienung
 - Test von Opto-Kopplern
 - Test von Modellbau-Servos
 - OneWire (DS18B20)
 - Ereigniszähler
 - Strukturierter Quellcode
 - Und Manches mehr, was mir gerade nicht einfällt.
- Mehr Details dazu findest Du in den nachfolgenden Abschnitten.

1.5. Quellcode

Die erste m-Firmware basierte auf dem Quellcode von Karl-Heinz. Eine Menge wurde aufgeräumt, Kommentare, Variablen umbenannt, Funktionen umstrukturiert, große Funktionen in mehrere kleine aufgeteilt und mehr. Danach entwickelte sich die m-Firmware immer weiter zu einer eigenständigen Version. Es sind u.A. einfache Frameworks für Anzeige und Schnittstellen dazu gekommen. Ich hoffe, dass der Quellcode einfach zu lesen und gut verständlich ist.

Du findest die aktuelle Firmware auf folgenden Webseiten: [6] oder [7].

1.6. Firmware übersetzen

Als erstes solltest Du das **Makefile** siehe Kapitel 5.1 ab Seite 34 editieren und hier MCU-Modell, Frequenz, Oszillator-Typ und die Programmieradaptoreinstellungen anpassen.

In der **config.h** siehe Kapitel 5.2 ab Seite 36 werden Bedienung und Menü Optionen eingestellt,

wie wähle Hardware- und Software-Optionen, die Sprache für die Bedienung, und ändere Standardwerte, wenn notwendig

und schließlich in der **config_328.h** und/oder **config_644.h** im Kapitel 5.3 ab Seite 44 die MCU-spezifische globale Konfiguration.

In der Datei „Clones“ siehe Kapitel 6 ab Seite 36, findest Du Einstellungen zu verschiedenen Testversionen bzw. Clonen.

Ist Dein Tester nicht dabei, schicke bitte die Einstellungen per EMail an den Autor [8] um anderen Benutzern damit zu helfen.

Alle Einstellungen und Werte sind in der Datei selber erklärt, daher folgt hier nur eine kurze Übersicht der wichtigsten Punkte.

Nach dem Editieren vom Makefile, config.h oder config-<MCU>.h bitte "make" ausführen oder was-auch-immer Dein IDE will, um die Firmware zu übersetzen.

Siehe dazu das Kapitel 7 ab Seite 55. Das Makefile bietet folgende zusätzliche Targets:

- clean alle Objektdateien löschen
- fuses Fuse Bits setzen
- upload Firmware brennen

2.1. Hardware-Optionen

- zusätzliche Tasten bzw. Eingabeoptionen
- Drehencoder
- Mehr/Weniger-Tasten
- Touch-Screen
- externe 2,5V Spannungsreferenz
- Schutz-Relais zur Kondensatorentladung
- Messung von Zenerdioden (DC-DC-Konverter)
- Frequenzzähler (einfache und erweiterte Version)
- Ereigniszähler
- Test von IR-Fernbedienungen (festes IR-Empfängermodul)
- fester Kondensator für Selbstabgleich von Spannungsoffsets
- SPI-Bus (Bit-Bang und Hardware)
- I2C-Bus (Bit-Bang und Hardware)
- TTL-Serielle (Bit-Bang und Hardware)
- OneWire Bus (Bit-Bang)

Die externe 2,5V Spannungsreferenz sollte nur genutzt werden, wenn sie um den Faktor 10 genauer als der Spannungsregler ist.

Ansonsten würde sie die Messergebnisse eher verschlechtern als verbessern.

Wenn Du einen MCP1702 mit einer typischen Genauigkeit von 0,4% hast, brauchst Du eigentlich keine zusätzliche Spannungsreferenz mehr.

2.2. Und natürlich die Software-Optionen

- PWM Generator (2 Varianten)
- Messung von Induktivität
- ESR-Messung und In-Circuit ESR
- Test von Drehencodern
- Rechtecksignalgenerator (braucht zusätzliche Tasten)
- Test von IR-Fernbedienungen (IR-Empfängermodul an Testpins)
- IR-Fernbedienung (IR-LED mit Treibertransistor)
- Test von Opto-Kopplern
- Test von Modellbau-Servos (braucht zusätzliche Tasten, Display >2 Zeilen)
- Erkennung von UJTs
- Test für Kondensatorleckstrom
- DS18B20 Temperatursensor
- Farbkodierung für Testpins (benötigt Farbdisplay)
- Ausgabe der gefundenen Bauteile parallel über TTL-Serielle, z.B auf PC
- Fernsteuerkommandos über TTL-Serielle.
- Ausgabe des umgekehrten hFE (C & E verdreht) für Bipolartransistoren

Bitte die Optionen entsprechend Deinen Wünschen und den begrenzten Ressourcen der MCU, d.h. RAM, EEPROM und Flash-Speicher, auswählen.

Sollte die Firmware zu groß werden, versuche für Dich nicht so wichtige Optionen wieder zu deaktivieren.

2.3. Verfügbare Sprachen

- Dänisch
 - von glennndk@mikrocontroller.net (benötigt kleine Änderungen im Font)
- Deutsch
- Englisch
- Italienisch
 - von Gino_09@EEVblog
- Polnisch
 - von Szpila
- Russisch
 - von indman@EEVblog und hapless@EEVblog
 - Zeichensatz mit kyrillischen Zeichen basierend auf Windows-1251
- Russisch 2
 - von hapless@EEVblog
 - Zeichensatz mit kyrillischen Zeichen basierend auf Windows-1251
- Spanisch
 - von pepe10000@EEVblog
- Tschechisch
 - von Kapa
 - Zeichensatz basierend auf ISO 8859-1
- Tschechisch 2
 - von Bohu
 - Zeichensatz basierend auf ISO 8859-2

Bei Zahlenwerten werden die Dezimalstellen standardmäßig durch einen Punkt gekennzeichnet, kann aber auf Komma umgestellt werden.

2.4. MCU Wahl

Für die MCU-spezifischen Einstellungen, wie Pin-Zuordnungen und Display, editiere `config_MCU`: Kapitel 5.1 Sektion 5.1.1 auf Seite 34.

- ATmega 328 `config_328.h`
- ATmega 324/644/1284 `config_644.h`

Siehe dazu Kapitel 5.3 ab der Seite 44

2.5. Anzeige Wahl

Das LCD-Modul sollte min. 2 Zeilen mit min. 16 Zeichen haben.
Für Grafikmodule einen Zeichensatz wählen, der die obige Bedingung erfüllt.
Die Einstellungen unten 2.7 ab Seite 11 und im Kapitel 5.3.1 auf Seite 44.

2.6. Busse & Schnittstellen

2.6.1. I2C/SPI Manche LCD-Module and andere Komponenten benötigen I2C oder SPI als Schnitt- stelle zur MCU. Daher hat der Firmware Treiber für beide Bussysteme. Um unterschiedliche Beschaltungen zu unterstützen haben die Bustreiber einen Bit-Bang und einen Hardware-Modus. Beim Bit-Bang-Modus können beliebige IO Pins an dem gleichen Port genutzt werden, während der Hardware-Modus die fest vorgegeben Bus-Pins der MCU nimmt. Der Nachteil des Bit-Bang-Modus ist seine Geschwindigkeit, er ist langsam. Der Hardware-Modus dagegen ist deutlich schneller. Du kannst den Unterschied leicht bei Farb-LCD-Modulen mit hoher Auflösung sehen.

Für Tester mit ATmega 328 wird fast immer der Bit-Bang-Modus aufgrund der Beschaltung benötigt. Der ATmega 324/644/1284 hat mehr I/O Pins, plus die veränderte Beschaltung erlauben es, die festen Bus-Pins für den Hardware- Modus zu nutzen.

Da SPI oder I2C primär vom LCD-Modul genutzt wird, können beide im Abschnitt fuer LCD-Module in `config-<MCU>.h` Seite 44 direkt konfiguriert werden. Alternativ kannst Du auch I2C

bzw. SPI in config.h Seite 43 aktivieren und Ports & Pins in config-<MCU>.h festlegen (schaue nach I2C_PORT Seite 47 bzw. SPI_PORT) auf Seite 47.

2.6.2. Serielle TTL-Schnittstelle Der Tester kann auch eine optionale serielle TTL-Schnittstelle haben. Wird diese zur Kommunikation mit einem PC genutzt, sollte sie mit einem USB-zu-TTL Konverter oder einem RS-232 Treiberbaustein kombiniert werden. Die Firmware kann den UART der MCU oder einen Software-UART (Bit-Bang) nutzen. Die TTL- Serielle wird in config.h 43 aktiviert siehe Abschnitt "Busses", und Port & Pins sind in config-<MCU>.h definiert (schaue nach SERIAL_PORT) Seite 47.

Der Software-UART hat den Nachteil, daß das TX-Signal nicht ständig „high“ ist, wenn die Schnittstelle ruht. Ursache dafür ist benutze Methode wie Port-Pins angesteuert werden. Ein Umschreiben der Ansteuerung würde die Firmware deutlich vergrößern. Dieses Problem scheint aber keine Auswirkung auf die meisten USB-zu-TTL Konverter zu haben. Sollte es doch Probleme geben, kannst Du einen Pull-up-Widerstand (10-100k) am TX-Pin probieren, um den Signalpegel im Ruhemodus auf „high“ zu halten.

Die Standardeinstellung der seriellen Schnittstelle ist 9600 8N1:

- 9600 bps
- 8 Datenbits
- keine Parität
- 1 Stopbit
- keine Flussteuerung

2.6.3. One Wire Ein weiterer unterstützter Bus ist OneWire, welcher entweder die Test-Pins (ONEWIRE_PROBES) oder einen festen MCU-Pin (ONEWIRE_IO_PIN) Seite 43 benutzen kann. Der Treiber ist für Standard-Busgeschwindigkeit und Clients mit externer Stromversorgung (nicht parasitär versorgt) ausgelegt.

Beschaltung von Test-Pins:

Probe #1: Gnd

Probe #2: Vcc (Strom durch 680 Ω Widerstand begrenzt)

Probe #3: DQ (Daten)

Ein externer Pull-Up-Widerstand von 4,7 k Ω zwischen DQ and Vcc wird benötigt!

2.7. LCD-Module

Im Augenblick werden folgende LCD-Module bzw. Controller unterstützt:

- HD44780	(textbasiertes Display, 2-4 Zeilen mit je 16-20 Zeichen)	S.13
- ILI9163	(grafisches Farb-Display 128x160)	S.13
- ILI9341/ILI9342	(grafisches Farb-Display 240x320 oder 320x240)	S.13
- PCD8544	(grafisches Display 84x48)	S.14
- PCF8814	(grafisches Display 96x65)	S.14
- SSD1306	(grafisches Display 128x64)	S.14
- ST7036	(textbasiertes Display, 3 Zeilen mit je 16 Zeichen)	S.15
- ST7565R	(grafisches Display 128x64)	S.15
- ST7735	(grafisches Farb-Display 128x160)	S.15
- ST7920	(grafisches Display bis zu 256x64)	S.16
- STE2007/HX1230	(grafisches Display 96x68)	S.16
- VT100 Terminal		S.16

Beachte die Versorgungsspannung und die Logikpegel des LCD-Moduls! Benutze Pegelumsetzer, sofern notwendig. Wenn das Display trotz korrekter Beschaltung nichts anzeigt, versuche den Kontrast zu ändern (config-<MCU>.h) siehe 44.

Bei den meisten LCD-Modulen kannst Du die /CS und /RES Signale per Pullup/down-Widerstände fest verdrahten und die entsprechenden IO-Pins auskommentieren, sofern nur das LCD-Modul am Bus hängt.

Hinweis zu ATmega 328: Wenn Du einen Drehencoder an PD2/PD3 hängst, dann verbinde /CS vom LCD-Modul mit PD5 und setze LCD_CS in config_328.h (nur für grafische LCD-Module). Anderenfalls würde der Drehencoder den Datenbus durcheinander bringen und zu fehlerhaften Ausgaben führen.

2.7.1. HD44780 Der HD44780 wird im 4-Bit-Modus betrieben.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
DB4	LCD_DB4	PD0	
DB5	LCD_DB5	PD1	
DB6	LCD_DB6	PD2	
DB7	LCD_DB7	PD3	
RS	LCD_RS	PD4	
R/W	LCD_RW	Gnd	
E	LCD_EN1	PD5	

Tabelle 2.1. Pinbelegung für den Parallelbus von HD44780.

Das LCD-Modul kann auch über einen I2C-Adapter mit PCF8574 betrieben werden. Dazu muss zum einen I2C aktiviert werden, und zum anderen wird auch die I2C-Adresse des PCF8574 benötigt.

Modul	config-<MCU>.h	Standard	Hinweis
DB4	LCD_DB4	PCF8574_P4	
DB5	LCD_DB5	PCF8574_P5	
DB6	LCD_DB6	PCF8574_P6	
DB7	LCD_DB7	PCF8574_P7	
RS	LCD_RS	PCF8574_P0	
R/W	LCD_RW	PCF8574_P1	
E	LCD_EN1	PCF8574_P2	
LED	LCD_LED	PCF8574_P3	

Tabelle 2.2. Das Pinout für das LCD-Modul mit dem PCF8574.

2.7.2. ILI9163 Der ILI9163 wird mittels 4-Draht-SPI gesteuert.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
/RESX	LCD_RES	PD4	
/CSK	LCD_CS	PD5	
D/CX	LCD_DC	PD3	
SCL	LCD_SCL	PD2	
SDIO	LCD_SDA	PD1	

Tabelle 2.3. Pinbelegung für ILI9163.

2.7.3. ILI9341/ILI9342 Der ILI9341/ILI9342 wird mittels SPI gesteuert.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
/RES	LCD_RES	PD4	optional
/CS	LCD_CS	PD5	optional
D/C	LCD_DC	PD3	
SCK	LCD_SCK	PD2	
SDI	LCD_SDI	PD1	
SDO	LCD_SDO	PD0	nur ILI9341 *
* noch nicht benutzt			

Tabelle 2.4. Pinbelegung für ILI9341/ILI9342.

2.7.4. PCD8544

Der PCD8544 wird mittels SPI gesteuert.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
/RES	LCD_RES	PD4	optional
/SCE	LCD_SCE	PD5	optional
D/C	LCD_DC	PD3	
SCLK	LCD_SCLK	PD2	
SDIN	LCD_SDIN	PD1	
SDO	LCD_SDO	PD0	

Tabelle 2.5. Pinbelegung für PCD8544.

Da das Display nur 84 Punkte in X-Richtung hat, ergeben sich max. 14 Zeichen pro Zeile bei einem 6x8-Zeichensatz. Also werden bis zu zwei Zeichen verschluckt. Wenn das stört, kannst Du in variables.h die Texte etwas kürzen.

2.7.5. PCF8814

Der PCF8814 wird typischerweise per 3-Draht-SPI gesteuert.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
/RES	LCD_RES	PD4	
/CS	LCD_CS	PD5	optional
SCLK	LCD_SCLK	PD2	
SDIN	LCD_SDIN	PD1	

Tabelle 2.6. Pinbelegung für PCD8814.

Bei Bedarf kann die Ausgabe über die Y-Flip-Einstellung und den MX-Pin (X-Flip) des PCF8814 (Pull-Up/Down) rotiert werden.

2.7.6. SSD1306

Der SSD1306 wird mittels 3-Line-SPI, 4-Line-SPI oder I2C gesteuert. 3-Line-SPI benötigt den Bit-Bang-Modus, und SPI_9 muss aktiviert sein.

Modul	config-<MCU>.h	Standard	Hinweis
/RES	LCD_RES	PD4	optional
/CS	LCD_CS	PD5	optional
DC	LCD_DC	PD3	
SCLK (D0)	LCD_SCLK	PD2	
SDIN (D1)	LCD_SDIN	PD1	

Tabelle 2.7. Pinbelegung für SSD1306 4-Line-SPI.

Modul	config-<MCU>.h	Standard	Hinweis
/RES	LCD_RES	PD4	optional
/CS	LCD_CS	PD5	optional
SCLK (D0)	LCD_SCLK	PD2	
SDIN (D1)	LCD_SDIN	PD1	

Tabelle 2.8. Pinbelegung für SSD1306 3-Line-SPI (nur Bit-Bang).

Modul	config-<MCU>.h	Standard	Hinweis
/RES	LCD_RES	PD4	optional
/SCL (D0)	I2C_SCL	PD1	optional
SDA (D1&2)	I2C_SDA	PD0	
SA0 (D/C)		Gnd	(0x3c) / 3.3V (0x3d)

Tabelle 2.9. Pinbelegung für SSD1306 I2C.

Mit den X/Y-Flip Einstellungen kannst Du die Orientierung der Anzeige anpassen, wenn notwendig.

2.7.7. ST7036 (ungetestet) Der ST7036 wird per 4-bit-Parallel-Schnittstelle oder 4-Draht-SPI angesprochen.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
DB4	LCD_DB4	PD0	
DB5	LCD_DB5	PD1	
DB6	LCD_DB6	PD2	
DB7	LCD_DB7	PD3	
RS	LCD_RS	PD4	
R/W	LCD_RW	Gnd	optional LCD_RW
E	LCD_EN	PD5	
XRESET		Vcc	optional LCD_RESET

Tabelle 2.10. Pinbelegung für den 4-bit-Parallel-Schnittstelle von ST7036.

Modul	config-<MCU>.h	Standard	Hinweis
XRESET	LCD_RESET	PD4	optional
CSB	LCD_CS	PD5	optional
RS	LCD_RS	PD3	
SCL (DB6)	LCD_SCL	PD2	
SI (DB7)	LCD_SI	PD1	

Tabelle 2.11. Pinbelegung für den 4-Draht-SPI von ST7036.

Der ST7036i spricht I2C, wird aber (noch) nicht unterstützt. Ein spezielles Merkmal des ST7036 ist ein Pin zum Aktivieren eines erweiterten Befehlssatzes (Pin EXT), welcher bei den meisten Modulen eingeschaltet ist. Sollte er abgeschaltet sein, sind die Einstellungen LCD_EXTENDED_CMD und LCD_CONTRAST auszukommentieren.

2.7.8. ST7565R Der ST7565R wird mittels 4/5-Line-SPI gesteuert.

Modul	config-<MCU>.h	Standard	Hinweis
/RES	LCD_RESET	PD0	optional
/CS1	LCD_CS	PD5	optional
A0	LCD_A0	PD1	
SCL (DB6)	LCD_SCL	PD2	
SI (DB7)	LCD_SI	PD3	

Tabelle 2.12. Pinbelegung für den 4/5-Line-SPI von ST7565R.

Für eine korrekte Anzeige musst Du evtl. mit den Einstellungen X/Y-Flip und X-Offset experimentieren.

2.7.9. ST7735 Der ST7735 wird mittels 4-Draht-SPI gesteuert.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
/RESX	LCD_RES	PD4	optional
/CSK	LCD_CS	PD5	optional
D/CX	LCD_DC	PD3	
SCL	LCD_SCL	PD2	
SDIO	LCD_SDA	PD1	

Tabelle 2.13. Pinbelegung für ST7735.

Für eine korrekte Anzeige musst Du evtl. mit den Einstellungen X/Y-Flip experimentieren. Wenn LCD_LATE_ON aktiviert ist, startet der Tester mit einem gelöschten Display, was zu einer kurzen Verzögerung beim Einschalten führt. Ansonsten sind beim Anschalten kurz zufällige Pixel zu sehen.

2.7.10. ST7920 Der ST7920 kann per 4-bit-Parallel-Modus oder SPI gesteuert werden.

Modul	config-<MCU>.h	Standard ATmega 328	Hinweis
DB4	LCD_DB4	PD0	
DB5	LCD_DB5	PD1	
DB6	LCD_DB6	PD2	
DB7	LCD_DB7	PD3	
RS	LCD_RS	PD4	
E	LCD_EN	PD5	
R/W	LCD_RW	Gnd	optional LCD_RW
XRESET		Vcc	optional LCD_RESET

Tabelle 2.14. Pinbelegung für den 4-bit-Parallel-Schnittstelle von ST7920.

Modul	config-<MCU>.h	Standard	Hinweis
/XRESET	LCD_RES	PD4	optional
/CS	LCD_CS	PD5	optional
SCLK (E)	LCD_SCLK	PD2	
SID (RW)	LCD_SID	PD1	

Tabelle 2.15. Pinbelegung für SPI mit ST7920.

Wegen dem schlechten Design des ST7920 können nur Zeichensätze mit einer Breite von 8 Pixeln verwendet werden. Zur Handhabung der horizontalen Adressierung in 16-Bit Schritten musste ich einen Bildschirmpuffer für Zeichen einrichten.

2.7.11. STE2007/HX1230 Der STE2007 wird typischerweise per 3-Draht-SPI gesteuert.

Modul	config-<MCU>.h	Standard	Hinweis
/RES	LCD_RES	PD4	optional
/CS	LCD_CS	PD5	optional
SCLK	LCD_SCLK	PD2	
SDIN	LCD_SDIN	PD1	

Tabelle 2.16. Pinbelegung für SPI mit ST7920.

Falls nötig, kannst Du die Ausgabe über die X/Y-Flip-Einstellungen rotieren.

2.7.12. VT100 Terminal Der VT100-Treiber ersetzt ein LCD-Modul, und die komplette Ausgabe erfolgt über ein seriell VT100-Terminal. Der Konfigurationsabschnitt für VT100 aktiviert die serielle Schnittstelle gleich mit. Bitte beachte, dass der VT100-Treiber andere Optionen für die serielle Schnittstelle deaktiviert, welche die Ausgabe beeinträchtigen können.

2.8. Tasten und Eingabeoptionen

Der Tester wird über primär über die Test-Taste bedient, erlaubt aber zusätzliche Eingabeoptionen, welche die Bedienung erleichtern oder für manche Funktion auch notwendig sind.

2.8.1. Die Test-Taste schaltet den Tester **ein** und dient zur **Bedienung**.

Dazu unterscheidet der Tester zwischen:

1. **kurzen Tastendruck**, der üblicherweise zum Fortfahren einer Funktion oder zur Auswahl des nächsten Menüpunktes benutzt wird,
2. **langen Tastendruck** (> 0,3s), der eine kontextabhängige Aktion ausführt und
3. **Doppelklick** der die Aktion beendet.

Wenn der Tester einen Tastendruck zum Fortfahren der aktuellen Aktion erwartet, zeigt es dies durch einen Cursor rechts unten auf dem LCD-Modul an.

Ein statischer Cursor signalisiert, dass weitere Informationen folgen,
und ein blinkender Cursor bedeutet, dass mit der Bauteilesuche weiter gemacht wird.

Für Menüs und einige Extrafunktionen wird der Cursor nicht angezeigt, da die erwartete Eingabe hier klar sein sollte.

Optional kannst du Bedienungshinweise einschalten, sofern dein Tester zusätzliche Eingabeoptionen und eine Anzeige mit ausreichend vielen Textzeilen hat (siehe `UI_KEY_HINTS` in `config.h`). Dann zeigt der Tester eine Bedienungshilfe statt des Cursors an, falls eine vorhanden ist. Im Augenblick gibt es nur eine solche Hilfe für die Bauteilesuche (Menü/Test).

2.8.2. Drehencoder (Hardware-Option) Mit einem Drehencoder erhält die Bedienung zusätzliche Funktionalität, die kontextabhängig ist.

Die Details werden in den weiteren Abschnitten erklärt. Manche Funktionen erlauben über die Drehgeschwindigkeit größere Änderungen oder Sprünge von Werten.

Der Lese-Algorithmus berücksichtigt die Anzahl der Gray-Code-Pulse pro Schritt (`ENCODER_PULSES`) und auch die Anzahl der Schritte für eine volle 360° Umdrehung (`ENCODER_STEPS`). Mit dem letzteren Wert kannst Du auch eine Feineinstellung der Erkennung der Drehgeschwindigkeit vornehmen. Beide auf Seite 36

Ein höherer Wert verlangsamt die Drehgeschwindigkeit, ein niedriger Wert erhöht sie. Sollte die Drehrichtung verkehrt herum sein, einfach die Pin-Definitionen für A und B in `config_<MCU>.h` vertauschen.

Die Erkennung der Drehgeschwindigkeit misst die Zeit von zwei Schritten. Also solltest Du den Encoder mindestens um zwei Schritte für mittlere Geschwindigkeiten drehen.

Für höhere Geschwindigkeiten sind es drei Schritte.

Ein einzelner Schritt resultiert immer in der niedrigsten Geschwindigkeit.

2.8.3. Mehr/Weniger-Tasten (Hardware-Option) Wenn Dir Tasten lieber als ein Drehencoder sind, dann kannst Du alternativ auch zwei Tasten nutzen. Siehe Seite 36. Die Tasten werden genauso wie ein Drehencoder angeschlossen (Pull-Up Widerstände, Logikpegel Low bei Betätigung).

Für eine Beschleunigung, ähnlich der Drehgeschwindigkeit beim Drehencoder, einfach die Taste lange drücken. Je länger, desto höher wird die Beschleunigung.

2.8.4. Touch-Screen (Hardware-Option) Normalerweise wird der ST7565- oder der SSD1306-Controller mit einer 4-Wire SPI-Schnittstelle angeschlossen. Als weitere Eingabeoption ist ein Touch-Screen möglich. Dazu sollte das LCD-Modul groß genug sein und mindestens 8 Textzeilen je 16 Zeichen oder mehr unterstützen.

Um wertvollen Platz auf dem LCD zu sparen, verzichten wir auf Symbole zum Berühren. Stattdessen gibt es unsichtbare Leisten links und rechts (je 3 Zeichen breit), oben und unten (je 2 Zeilen hoch) und eine Fläche in der Mitte.

Die Leisten links und oben sind für „weniger“ oder „Menüpunkt hoch“, rechts und unten entsprechend für „mehr“ oder „Menüpunkt runter“.

Also die gleiche Funktion wie beim Drehencoder.

Ein langes Berühren beschleunigt Änderungen in manchen Funktionen, ähnlich der Drehgeschwindigkeit beim Drehencoder.

Die Fläche in der Mitte ist eine Software-Version der Test-Taste, d.h. damit wird z.B. **nicht** die Zener-Test-Option mit Strom versorgt.

Für die Benutzung des Touch-Screen ist ein Abgleich notwendig. Dieser wird automatisch nach dem Einschalten des Testers gestartet, wenn noch keine Abgleichwerte im EEPROM gespeichert sind.

Man kann ihn auch über das Haupt-Menü starten. Die Prozedur ist recht einfach.

Wenn Du ein Sternchen (gelbes * bei Farb-LCDs) siehst, drauf drücken. Der Tester zeigt nach jeder Berührung die native x/y-Position an. Den Abgleich kannst Du jeder Zeit mit der Test-Taste abbrechen.

Wenn Du Probleme mit dem Abgleich, wie z.B. seltsame x/y-Positionen, hast, überprüfe bitte die Orientierung des Touch-Screen gegenüber dem LCD-Modul.

Der Treiber hat Schalter zum Verdrehen und Vertauschen der Orientierung.

Nach einem erfolgreichen Abgleich nicht vergessen, die Offset-Werte zu speichern.

(Hauptmenü: Speichern).

Unterstützte Touch-Screen-Controller: - ADS7843 / XPT2046

Du findest die Konfiguration dazu unterhalb des Bereichs für LCD-Module in config-<MCU>.h (momentan nur config_644.h, auf Seite ?? da der 328 zu wenig unbenutzte IO-Pins hat).




2.8.5. Kommunikation mit PC Der Tester kann eine serielle TTL-Schnittstelle zur Kommunikation mit einem PC nutzen.

Dies kann eine unidirektionale Verbindung (nur Senden) zur seriellen Ausgabe von gefundenen Bauteilen sein, oder auch eine bidirektionale zur Automatisierung.

In beiden Fällen muss die serielle Schnittstelle aktiviert werden (siehe Abschnitt „Busses“ auf Seite 43 in config.h).

Spezielle Zeichen werden durch Standardzeichen ersetzt, z.B. wird das Ω (Ohm) zu einem R.

Konvertierungstabelle:

	> <
	
	μ
Ω	R
μ	u

Hinweise:

- 9600 8N1

- Newline ist <CR><LF>

2.8.6. Serielle Ausgabe Der Tester gibt gefundene Bauteile zusätzlich über die serielle Schnitt- stelle aus, wenn dies aktiviert ist (siehe UI_SERIAL_COPY im Abschnitt „Benutzerschnittstelle“ auf Seite 40 in config.h).

Dazu reicht ein einfaches Terminalprogramm auf dem PC.

Die Ausgabe folgt der Ausgabe auf dem LCD-Display, aber nur für gefundene Bauteile.

Es erfolgt keine Ausgabe von Menüs und Funktionen über die Serielle, außer für Ergebnisse vom Opto-Koppler-Test.

2.8.7. Automatisierung Die Automatisierung ermöglicht die Fernsteuerung des Testers per Kommandos über eine bidirektionale serielle Verbindung.

Zum Aktivieren dieser Funktion siehe bitte UI_SERIAL_COMMANDS im Abschnitt „Benutzerschnittstelle“ auf Seite 39 in config.h.

Das Verhalten des Testers ändert sich etwas.

Die Automatisierung erzwingt den Auto-Hold-Modus, und der Tester sucht nach dem Einschalten **nicht** automatisch nach einem Bauteil.

Die Kommandoschnittstelle ist recht einfach.

Du sendest ein Kommando und der Tester antwortet.

Die Kommunikation basiert auf ASCII-Textzeilen, und bei den Kommandos ist auf Groß- und Kleinschreibung zu achten.

Jede Kommandozeile wird mit einem <CR><LF> oder <LF> Newline abgeschlossen.

Der Tester nimmt Kommandos nur während des Wartens auf den Benutzer nach dem Einschalten, der Ausgabe eines Bauteils oder der Ausführung einer Menüfunktion an.

Antwort-zeilen enden mit einem <CR><LF> Newline.

Für die Liste der Kommandos und ihrer Beschreibung siehe Abschnitt „Fernsteuerungskommandos“ 8 ab Seite 59.

2.8.8. VT100-Ausgabe Anstatt einer LCD-Anzeige kann die komplette Ausgabe über ein VT100-Terminal erfolgen (siehe VT100 im Abschnitt LCD-Module 2.7 auf Seite 16). Um ein Durcheinander im Layout der Ausgabe zu vermeiden, werden die anderen Optionen für die serielle Schnitt- stelle deaktiviert.

3.0.1. Einschalten Ein langer Tastendruck beim Einschalten aktiviert den Auto-Hold-Modus. In diesem Modus wartet der Tester nach einer Ausgabe auf einen kurzen Tastendruck, um mit der Bauteilesuche weiter zu machen.

Ansonsten läuft der Tester im kontinuierlichen Modus. Die Auswahl des Modus lässt sich per `config.h` (`UI_AUTOHOLD`) umdrehen.

Nach dem Einschalten wird kurz die Firmwareversion angezeigt.

Mit einem sehr langen Tastendruck (2s) beim Einschalten, kannst Du die Abgleich-werte auf ihre Standards zurück setzen.

- Das kann praktisch sein, wenn z.B. der Kontrast vom LCD-Modul so verstellt ist, dass man nichts mehr sieht.

Wenn der Tester ein Problem mit den gespeicherten Abgleich-werten entdeckt (Problem mit dem EEPROM), zeigt er einen Prüfsummenfehler an und benutzt stattdessen die Standardwerte.

3.0.2. Bauteilesuche Nach dem Einschalten sucht Tester automatisch nach Bauteilen.

Im kontinuierlichen Modus wiederholt der Tester die Suche nach einer kurzen Wartepause.

Wenn mehrfach hintereinander kein Bauteil gefunden wurde, schaltet sich der Tester aus.

Im Auto-Hold-Modus (durch Cursor signalisiert) führt der Tester einen Suchvorgang aus und wartet dann auf einen Tastendruck bzw. Rechtsdrehung vom Drehencoder bevor er die nächste Suche startet.

Die Wartepause und das automatische Abschalten im kontinuierlichen Modus kann mittels `CYCLE_DELAY` Seite 37 und `CYCLE_MAX` Seite 39 in `config.h` geändert werden.

Für den Auto-Hold-Modus gibt es eine optionale automatische Abschaltung (`POWER_OFF_TIMEOUT`) Seite 40, welche nur während der Bauteilesuche und Ausgabe aktiv ist.

In beiden Modi kannst Du das Hauptmenü aufrufen (siehe weiter unten).

3.0.3. Batterieüberwachung Jeder Zyklus der Bauteilesuche beginnt mit der Anzeige der Batteriespannung und des Status (ok, schwach, leer). Bei Unterschreiten der Schwellspannung für eine leere Batterie schaltet sich der Tester aus. Die Batterie wird regelmäßig während des Betriebs überprüft.

Die Standardkonfiguration der Batterieüberwachung ist für eine 9V-Batterie ausgelegt, kann aber an fast jede andere Stromversorgung angepasst werden. Im Abschnitt „power management“ in `config.h` findest Du alle Einstellungen dazu auf Seite 40.

Die Batterieüberwachung kann mittels `BAT_NONE` deaktiviert werden, auf direkte Messung für Batterien mit weniger als 5V per `BAT_DIRECT` konfiguriert werden, oder auf indirekte Messung über einen Spannungsteiler (definiert durch `BAT_R1` und `BAT_R2`) gesetzt werden.

Manche Tester unterstützen zwar eine optionale externe Stromversorgung, aber erlauben keine Überwachung dieser.

In diesem Fall kannst Du per `BAT_EXT_UNMONITORED` Probleme mit dem automatischen Abschalten bei zu niedriger Batteriespannung umgehen.

Bei externer Strom-versorgung wird der Batteriestatus dann auf „ext“ (für extern) gesetzt. Die Schwellwerte für eine schwache und leere Batterie werden über `BAT_WEAK` und `BAT_LOW` gesetzt, während `BAT_OFFSET` den Spannungsverlust durch die Schaltung definiert, z.B. Verpolungsschutzdiode und PNP-Transistor zum Schalten der Stromversorgung.

3.0.4. Ausschalten Während das Ergebnis der letzten Bauteilesuche angezeigt wird, schaltet ein langer Tastendruck den Tester aus. Dabei zeigt der Tester ein kurzes „Auf Wiedersehen“ oder „Ciao!“ und schaltet sich dann selbst ab. Allerdings bleibt der Tester solange noch

eingeschaltet, wie die Taste gedrückt gehalten wird. Das liegt am Design des Schaltungsteils der Stromversorgung.

3.0.5. Menü Durch zweimaliges kurzes Drücken der Test-Taste nach der Ausgabe des letztes Ergebnisses gelangt man in das Menü. Einfach zweimal kurz hintereinander drücken. Kann vielleicht etwas Übung am Anfang benötigen ;)

Wenn ein Drehencoder vorhanden ist, startet zusätzlich eine Linksdrehung das Menü. Die alte Methode über den Kurzschluss der drei Testpins kann ebenfalls aktiviert werden (UI_SHORT_CIRCUIT_MENU).

Im Menü wählt ein kurzer Tastendruck den nächsten Punkt aus und ein langer Tastendruck führt den ausgewählten Punkt aus. Bei einem LCD-Modul mit 2 Zeilen wird unten rechts eine Navigationshilfe angezeigt. Ein „>“, wenn weitere Punkte folgen, oder ein „<“ beim letzten Punkt.

Geht man weiter als der letzte Punkt, gelangt man wieder zum ersten. Bei einem LCD-Modul mit mehr als 2 Zeilen wird der ausgewählte Punkt mit einem „*“ davor gekennzeichnet.

Ist ein Drehencoder vorhanden, wird mit dem Drehen der vorherige bzw. nächste Punkt ausgewählt. Hier gibt es auch wieder einen Überlauf, d.h. vom ersten zum letzten Punkt.

Ein kurzer Tastendruck führt den Punkt aus, im Gegensatz zu oben. Manche Punkte/Extras zeigen beim Start das Pin-out der benutzten Testpins kurz an.

Die Info wird für ein paar Sekunden gezeigt, kann aber mit einem kurzen Tastendruck übersprungen werden.

Funktionen, welche Signale erzeugen, geben ihr Signal standardmäßig auf Testpin #2 aus. Dabei werden die Pins #1 und #3 auf Masse gesetzt.

Ist Dein Tester für die Signalausgabe auf einem eigenen Ausgang (OC1B) konfiguriert, werden die Testpins nicht genutzt und es erfolgt auch keine Ausgabe des Pinout.

3.1. Menü

3.1.1. PWM-Generator Macht genau das, was Du erwartest :-) Vor dem Übersetzen der Firmware bitte entweder den PWN-Generator mit einfacher Bedienung oder den mit erweiterter Bedienung auswählen. Letzterer benötigt einen Drehencoder und ein größeres Display. Beschaltung bei Signalausgabe über die Testpins:

Pin #2: Ausgang (680Ω Widerstand zur Strombegrenzung)
Pin #1 und #3: Masse

3.1.2. Einfache PWM Zuerst muss man aus einer vorgegeben Liste die Frequenz wählen. Kurzer Tasten- druck für die nächste Frequenz und langer Tastendruck zum Starten, wie beim Menü.

Mit Drehencoder ein kurzer Tastendruck zum Starten.
Das Tastverhältnis startet bei 50% und kann in 5%-Schritten geändert werden.

Ein kurzer Tastendruck für +5% und ein langer für -5%.
Zum Beenden die Test-Taste zweimal kurz hintereinander drücken.
Ist ein Drehencoder vorhanden, lässt sich das Tastverhältnis in 1%-Schritten ändern.

3.1.3. Erweiterte PWM Mit einem kurzen Tastendruck schaltest Du zwischen Frequenz und Tastverhältnis um.

Der ausgewählte Wert wird durch ein Sternchen markiert.
Mit dem Dreh-encoder änderst Du den Wert, rechts für höher, links für niedriger.
Und mit einem langen Tastendruck wird auf den Standardwert zurück gestellt (Frequenz: 1kHz, Tastverhältnis: 50%).
Mit zwei kurzen Tastendrücken wird der PWM-Generator beendet.

3.1.4. Rechteck-Signalgenerator Der Signalgenerator gibt ein Rechtecksignal mit variabler Frequenz bis zu einem 1/4 des MCU-Taktes aus (2MHz bei 8MHz Takt). Die Startfrequenz liegt bei 1000Hz und kann mit dem Drehencoder geändert werden. Die Dreh- -geschwindigkeit bestimmt den Grad der Änderung, d.h. langsames Drehen ergibt kleine Änderungen und schnelles Drehen große.

Da die Signalerzeugung auf der internen PWM-Funktion der MCU basiert, können nicht beliebige Frequenzen generiert werden, sondern nur in Schritten. Für niedrige Frequenzen ist die Schrittweite recht klein, erst bei hohen Frequenzen wird sie signifikant.

Ein langer Tastendruck stellt die Frequenz zurück auf 1kHz und zwei kurze Tastendrucke beenden den Signalgenerator.
Beschaltung bei Signalausgabe über die Testpins:

Pin #2: Ausgang (680Ω Widerstand zur Strombegrenzung)
Pin #1 und #3: Masse

Hinweis: Drehencoder oder andere Eingabeoption notwendig!

3.1.5. Zenertest (Hardware-Option) Mit Hilfe eines DC-DC-Konverters wird eine Testspannung von bis zu 50V zum Testen von Zenerdioden generiert. Der Anschluss erfolgt über zusätzliche Testpins. **Solange die Test-Taste gedrückt wird, erzeugt der Konverter die Testspannung und die aktuelle Spannung wird angezeigt.**

Nach dem Loslassen der Taste wird die kleinste gemessene Spannung angezeigt, sofern der Test ausreichend lange für eine stabile Testspannung lief.

Dieser Vorgang kann beliebig oft wiederholt werden. Zum Beenden die Test-Taste zweimal kurz hintereinander drücken.

Beschaltung für Zenerdiode:

Pin +: Kathode
Pin -: Anode

3.1.6. ESR-Messung Die ESR-Messung kann den Kondensator in der Schaltung messen und zeigt neben der Kapazität den ESR an, wenn ein Kondensator tatsächlich entdeckt wird.

Stelle sicher, dass der Kondensator **vor dem Anschließen** entladen wurde!

Die gemessenen Werte können von einer Messung außerhalb der Schaltung wegen parallel geschalteter Bauteile abweichen.

Um die Messung zu starten, kurz die Test-Taste drücken.

Zum Beenden die Test-Taste zweimal kurz hintereinander drücken.

Beschaltung für Kondensator:

Pin #1: Plus

Pin #3: Minus

3.1.7. Kondensatorleckstrom Der Test auf Leckstrom lädt einen Kondensator auf und zeigt dabei den Strom und die Spannung über den Messwiderstand an. Das Laden beginnt mit Rl (680Ω) und schaltet auf Rh (470kΩ) um, sobald der Strom einen bestimmten Grenzwert unterschreitet.

Jeder Testzyklus beginnt mit der Anzeige der Belegung der Testpins.

Nach dem Verbinden des Kondensators startet ein Druck der Testtaste das Laden (oder Rechtsdrehung bei einem Drehencoder).

Ein weiterer Druck beendet das Laden, und der Tester entlädt den Kondensator während die Restspannung angezeigt wird.

Sobald der Entladegrenzwert erreicht ist, startet der Tester einen neuen Testzyklus.

Zum Verlassen des Tests zweimal kurz die Testtaste drücken.

Hinweis: **Auf Polarität von Elkos achten!**

Beschaltung für Kondensator:

Pin #1: Plus

Pin #3: Minus

3.1.8. R/L-Monitor Der R/L-Monitor misst ständig Widerstand und ggf. Induktivität eines Bauteils an den Pins #1 und #3. Zwischen den Messungen gibt es jeweils eine kurze Pause von 2 Sekunden, die durch einen Cursor unten rechts signalisiert wird. Während der Pause lässt sich der Monitor durch zwei kurze Tastendrucke (Testtaste) beenden.

3.1.9. C-Monitor Der C-Monitor misst ständig Kapazität und ggf. den ESR eines Kondensators an den Pins #1 und #3. Zwischen den Messungen gibt es jeweils eine kurze Pause von 2 Sekunden, die durch einen Cursor unten rechts signalisiert wird. Während der Pause lässt sich der Monitor durch zwei kurze Tastendrucke (Testtaste) beenden.

3.1.10. Frequenzzähler (Hardware-Option) Den Frequenzzähler gibt es in zwei Versionen.

Der Einfache besteht aus einem passiven Eingang auf den T0-Pin (F-in) der MCU.

Und der Erweiterte hat neben einem Eingangspuffer auch zwei Oszillatoren zum Testen von Quarzen (für niedrige und hohe Frequenzen) und einen zusätzlichen Frequenzvorteiler.

Beide Schaltungen sind in der Dokumentation von Karl-Heinz [5] beschrieben.

3.1.11. Einfacher Zähler Ist die Zusatzschaltung für den einfachen Frequenzzähler eingebaut, kannst Du damit Frequenzen von ca. 10Hz bis zu 1/4 der MCU-Taktfrequenz mit einer Auflösung von 1Hz bei Frequenzen unterhalb von 10kHz messen.

Die Frequenz wird ständig gemessen und angezeigt, bis Du die Messung durch zwei kurze Tastendrucke beendest. Die automatische Bereichswahl setzt die Torzeit auf Werte zwischen 10ms und 1000ms, je nach Frequenz. Der T0-Pin kann parallel zum Ansteuern einer Anzeige verwendet werden.

3.1.12. Erweiterter Zähler Der erweiterte Frequenzzähler hat einen zusätzlichen Vorteiler, welcher die Messung höherer Frequenzen erlaubt.

Das theoretische Maximum liegt bei $1/4$ des MCU-Taktes multipliziert mit dem Vorteiler (16:1 or 32:1). Die Steuer- signale werden in `config_<mcu>.h` definiert, und bitte nicht vergessen, in `config.h` den korrekten Vorteiler auszuwählen.

Der Signaleingang (gepufferter Eingang, Quarz-Oszillator für niedrige Frequenzen, Quarz-Oszillator für hohe Frequenzen) wird über die Testtaste oder den Drehencoder geändert.

Zwei kurze Tastendrucke beenden den Frequenzzähler.

3.1.13. Ereigniszähler (Hardware-Option) Der Ereigniszähler nutzt den T0-Pin (F-in) als festen Eingang und reagiert auf die steigend Flanke eines Signals. Der T0-Pin kann nicht parallel zum Ansteuern einer Anzeige verwendet werden. Eine einfache Eingangsstufe wird empfohlen.

Der Zähler wird über ein kleines Menü gesteuert, welches auch die Zählerwerte anzeigt. Die Menüpunkte werden über einen kurzen Tastendruck ausgewählt, und die Einstellungen über den Drehencoder oder zusätzliche Tasten geändert.

Der erste Menüpunkt ist der Zählermodus:

- Zählen zähle Zeit und Ereignisse
- Zeit zähle Ereignisse für eine vorgegebene Zeit
- Ereignisse zähle Zeit für eine vorgegebene Anzahl von Ereignissen

Der zweite Menüpunkt „n“ ist die Anzahl der Ereignisse. Im Zählermodus „Ereignisse“ wird der Stopwert angezeigt, welcher geändert werden kann. Ein langer Tastendruck stellt den Stopwert auf einen Vorgabewert (100). In anderen Zählermodi ist dieser Menüpunkt blockiert.

Der nächste Menüpunkt „t“ ist die Zeitperiode in Sekunden (Vorgewert: 60s).
Gleiches Spiel, nur für den Zeit-Modus.

Und der letzte Menüpunkt startet oder stoppt den Zähler über einen langen Tastendruck. Während der Zähler läuft, werden die Anzahl der Ereignisse und die vergangene Zeit jede Sekunde aktualisiert, und nach dem Stoppen die Ergebnisse angezeigt.

Der Grenzwert für die Zeit ist 43200s (12h) und für die Ereignisse $4 \cdot 10^9$.
Sobald einer der Grenzwerte überschritten wird, stoppt der Zähler automatisch.
Der Grenz- oder Stopwert für Ereignisse wird alle 200ms überprüft. Daher ist bei mehr als 5 Ereignissen/s ein Übersteigen des Wertes möglich.

- Triggerausgang

Optional kannst Du den Triggerausgang aktivieren (EVENT_COUNTER_TRIGGER_OUT), um ein anderes Gerät über die Testpins zu steuern. Der Triggerausgang wird während des Zählens auf High gesetzt, d.h. steigende Flanke beim Start und fallende Flanke bei Stop.
Beschaltung für Triggerausgang über die Testpins:

- Pin #1: Masse
- Pin #2: Ausgang (680 Ohm Widerstand zur Strombegrenzung)
- Pin #3: Masse

3.1.14. Drehencoder Diese Funktion testet Drehencoder und bestimmt das Pin-out.

Deine Aufgabe ist es, die Testpins an den Drehencoder (A, B, Common) anzuschließen und den Encoder nach rechts (also Uhrzeigersinn) zu drehen.

Der Algorithmus benötigt 4 Grey-Code-Schritte zur Erkennung.

Die Drehrichtung ist wichtig zur Erkennung von A und B, da eine falsche Richtung zur Verdrehung der Pins führen würde.

Wenn ein Drehencoder entdeckt wird, gibt der Tester die Pinbelegung aus und wartet auf einen Tastendruck beim Auto-Hold-Modus oder wartet kurz beim kontinuierlichen Modus.

Zum Beenden die Test-Taste kurz während eines Suchlaufs drücken.

3.1.15. Kontrast Für manche grafische LCD-Module kannst du den Kontrast stellen.

Ein kurzer Tastendruck erhöht den Wert, ein langer verkleinert ihn.

Zum Beenden die Test-Taste zweimal kurz hintereinander drücken.

Ist ein Drehencoder vorhanden, kann der Kontrastwert damit ebenfalls geändert werden.

3.1.16. Detektor/Decoder für IR-Fernbedienungen Diese Funktion erkennt und dekodiert Signale von IR-Fernbedienungen und benötigt ein

IR-Empfängermodul, wie z.B. aus der TSOP-Serie.

Beim Übersetzen der Firmware kannst Du zwischen zwei Anschlussvarianten wählen.

Bei der ersten Variante wird das Modul mit den normalen Testpins verbunden.

Die zweite Variante ist ein festes Modul, welches mit einem bestimmten MCU-Pin verbunden ist.

Wenn ein bekanntes Protokoll erkannt wird, gibt der Tester das Protokoll, Adresse (sofern verfügbar), Kommando und ggf. zusätzliche Informationen hexadezimal aus.

Das Ausgabeformat ist: <Protokoll> <Datenfeld(er)>

Bei einem defekten Datenpaket wird „?“ als Datenfeld ausgegeben.

Ist das Protokoll unbekannt, zeigt der Tester die Anzahl der Pausen & Pulse und die Dauer des ersten Puls und der ersten Pause in Einheiten von 50µs an: ? <Pulse>:<erster Pulse>-<erste Pause>

Wenn die Anzahl der Pulse bei verschiedenen Tasten der Fernbedienung gleich bleibt, ist die Modulation sehr wahrscheinlich PDM oder PWM.

Eine sich ändernde Anzahl von Pulsen weist auf Bi-Phase-Modulation hin.

Zum Beenden die Test-Taste einmal kurz drücken.

Unterstützte Protokolle und ihre Datenfelder:

- JVC <Adresse>:<Kommando>
- Kaseikyo (Japancode, 48 Bit) <Herstellercode>:<System>-<Produkt>:<Funktion>
- Matsushita (Panasonic MN6014, C6D6 / 12 bits) <Gerätecode>:<Datencode>
- Motorola <Kommando>
- NEC (Standard & Erweitert) <Adresse>:<Kommando> R für Wiederholsequenz
- Proton / Mitsubishi (M50560) <Adresse>:<Kommando>
- RC-5 (Standard) <Adresse>:<Kommando>
- RC-6 (Standard) <Adresse>:<Kommando>
- Samsung / Toshiba (32 Bit) <Gerätecode>:<Datencode>
- Sharp <Adresse>:<Kommando>
- Sony SIRC (12, 15 & 20 Bit) 12 & 15: <Kommando>:<Adresse>

20: <Kommando>:<Adresse>:<Erweitert>

Optionale Protokolle (SW_IR_RX_EXTRA):

- IR60 (SDA2008/MC14497) <Kommando>
- Matsushita (Panasonic MN6014, C5D6 / 11 bits) <Gerätecode>:<Datencode>
- NEC µPD1986C <Datencode>
- RECS80 (Standard & Erweitert) <Adresse>:<Kommando>
- RCA <Adresse>:<Kommando>
- Sanyo (LC7461) <Gerätecode>:<Taste>
- Thomson <Gerät>:<Funktion>

Die Trägerfrequenz vom TSOP IR-Empfängermodul muss nicht genau zur Fernsteuerung passen.

Es verringert sich eigentlich nur die Reichweite, was für unseren Zweck aber kein Problem darstellt.

- IR-Empfängermodul an Testpins

Das IR-Empfängermodul bitte erst im IR-Fernbedienungsdetektor anschließen!

Beschaltung für das TSOP-Modul:

Probe #1: Masse/Gnd

Probe #2: Vs (680Ω Widerstand zur Strombegrenzung)

Probe #3: Data/Out

Hinweis: Der Widerstand zur Strombegrenzung setzt ein IR-Empfängermodul mit einem Versorgungsspannungsbereich von ca. 2,5 - 5V voraus.

Wenn Du ein 5V-Modul hast, kannst Du in config.h den Widerstand auf eigene Gefahr abschalten. Ein Kurzschluss kann allerdings die MCU zerstören.

- Festes IR-Empfängermodul

Für das feste Modul bitte Port und Daten-Pin in config_<MCU>.h passend setzen.

3.1.17. IR-Fernbedienung Die IR-Fernbedienung sendet Fernbedienungs-codes, welche Du zuvor eingegeben hast, und dient zum Testen von IR-Empfängern bzw. von Geräten mit IR- Fernbedienung.

Diese Funktion benötigt eine zusätzliche Eingabeoption, wie z.B. ein Drehencoder, ein Display mit mehr als vier Textzeilen und eine einfache Treiberschaltung für die IR-LED.

Der Tester zeigt Dir das Protokoll, die Trägerfrequenz, das Tastverhältnis des Trägers und ein paar Datenfelder.

Mit einem kurzen Druck der Test-Taste schaltest Du zwischen den Punkten hin und her. Der ausgewählte Punkt wird durch ein „*“ gekennzeichnet.

Über den Drehencoder (oder andere Eingabeoption) änderst Du die Einstellung bzw. den Wert eines Punktes.

Bei einem langen Druck der Test-Taste sendet der Tester den IR-Code solange die Test-Taste gedrückt bleibt. Und wie üblich beenden zwei kurze Tastendrucke die Funktion.

Wenn Du das Protokoll änderst, werden Trägerfrequenz und Tastverhältnis auf die Standardwerte des jeweiligen Protokolls gesetzt.

Du kannst diese aber nach Belieben ändern.

Die Trägerfrequenz kann auf 30 bis 56 kHz gestellt werden, und das Tastverhältnis auf 1/2 (50%), 1/3 (33%) oder 1/4 (25%).

Die Datenfelder sind die Teile des Fernbedienungs-codes, die Du setzen kannst.

Sie werden weiter unten erklärt und sind meistens nur die Adresse und das Kommando.

Unterstützte Protokolle und ihre Datenfelder:

- JVC <Adresse:8> <Kommando:8>
- Kaseikyo (Japanese Code) <Hersteller:16> <System:4> <Produkt:8> <Funktion:8>
- Matsushita (Panasonic, MN6014 12 bits) <Gerät:6> <Taste:6>
- Motorola <Kommando:9>
- NEC Standard <Adresse:8> <Kommando:8>
- NEC Extended <Adresse:16> <Kommando:8>
- Proton / Mitsubishi (M50560) <Adresse:8> <Kommando:8>
- RC-5 Standard <Adresse:5> <Kommando:6>
- RC-6 Standard, Mode 0 <Adresse:8> <Kommando:8>
- Samsung / Toshiba (32 bits) <Gerät:8> <Taste:8>
- Sharp / Denon <Adresse:5> <Kommando:8> <Maskierung:1>
- Sony SIRC-12 <Kommando:7> <Adresse:5>
- Sony SIRC-15 <Kommando:7> <Adresse:8>
- Sony SIRC-20 <Kommando:7> <Adresse:5> <Erweitert:8>

Optionale Protokolle (SW_IR_RX_EXTRA):

- Thomson <Gerät:4> <Funktion:7>

Die Datenfelder sind durch Leerzeichen getrennt und ihre Syntax ist: <Feldname>:<Anzahl Bits>

Beschaltung bei Signalausgabe über die Testpins:

Pin #2: Ausgang (680Ω Widerstand zur Strombegrenzung)

Pin #1 und #3: Masse

Der Signalausgang (Test-Pin #2) hat einen Widerstand zur Strombegrenzung und kann eine IR-LED mit nur etwa 5mA direkt schalten, was für eine typische IR-LED mit einem If von 100mA nicht ausreichend ist.

Daher wird ein einfacher Treiber auf Basis eines Transistors, der IR-LED und einem Widerstand zur Strom- begrenzung benötigt.

Die Abbildung 3.1 zeigt einen Treiber, welcher die IR-LED (Vf 1.5V, If 100mA) mit 50mA schaltet.

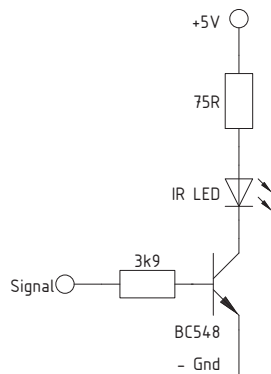


Abbildung 3.1. Beispiel 50mA IR Treiber mit (V_f 1.5V, I_f 100mA,)

Hinweis: Falls das Timing der Pulse/Pausen nicht passen sollte, bitte die alternative Warteschleifenmethode `SW_IR_TX_ALTDELAY` auf Seite 37 aktivieren.

Dies ist notwendig, wenn der C-Compiler die Standardwarteschleife trotz Anweisung, den Inline-Assembler-Code beizubehalten, optimiert.

3.1.18. Opto-Koppler-Test Dieser Test prüft Opto-Koppler und gibt V_f der LED, den CTR-Wert (auch I_f) und t_{on} bzw. t_{off} Zeiten (für Transistortypen) aus.

Unterstützt werden Standard-NPN-Transistoren, NPN-Darlington-Stufen und TRIACs. Für die CTR-Messung wird der I/O-Pin der MCU kurzzeitig für ca. 3ms überlastet.

Das Datenblatt gibt einen maximalen Ausgangsstrom vom 20mA an, wir überlasten den Pin aber bis zu ca. 100mA.

Daher ist der maximale CTR-Wert begrenzt, und Werte über 2000% sollte man mit Vorsicht genießen.

Der maximale Strom für die LED ist 5mA, was bei TRIAC-Typen zu beachten ist. Relais-Typen (MOSFET back to back) werden als Transistor erkannt und der CTR-Wert ist dann bedeutungslos. Typen mit anti-parallelen LEDs werden ignoriert.

Zum Testen brauchst Du einen einfachen Adapter mit folgenden drei Testpunkten:

Transistor-Typ:

- Anode der LED
- Kathode der LED und Emitter vom Transistor miteinander verbunden
- Kollektor vom Transistor

TRIAC-Typ:

- Anode der LED
- Kathode der LED und MT1 vom TRIAC miteinander verbunden
- MT1 vom TRIAC

Du kannst den Adapter nach Belieben mit den drei Testpins vom Tester verbinden.

Der Tester findet die Anschlussbelegung dann automatisch.

Nach dem Starten bitte den Adapter mit den Testpins vom Tester verbinden und kurz die Taste zum Prüfen drücken.

Wenn ein Opto-Koppler gefunden wurde, zeigt der Tester den Typen und verschiedene Infos an.

Wurde keiner erkannt, erfolgt die Anzeige von „keiner“.

Ein blinkender Cursor weist darauf hin, dass ein Tastendruck für die nächste Prüfung erwartet wird.

Zwei kurze Tastendrucke beenden, wie üblich, den Test.

3.1.19. Modellbau-Servo-Test Diese Funktion erzeugt ein PWM-Signal für Modellbau-Servos, welche mit einem 1-2 ms PWM-Puls gesteuert werden.

Die typischen PWM-Frequenzen von 50, 125, 250 und 333Hz werden unterstützt, und die Pulselänge ist im Bereich von 0,5 bis 2,5 ms einstellbar.

Zusätzlich gibt es einen Sweep-Modus für Pulse von 1 - 2ms und wählbarer Geschwindigkeit. Die Puls-breite stellst Du mit dem Drehencoder ein. Links für kürzere Pulse, und rechts für längere.

Mit einem langen Tastendruck wird die Pulsweite auf 1,5ms zurück gesetzt (mittlere Position vom Servo).

Mit einem kurzen Tastendruck wechselst Du zwischen Puls- und Frequenzauswahl (durch ein Sternchen markiert).

In der Frequenzauswahl schaltest Du mit dem Drehencoder zwischen den Frequenzen um. Mit einem langen Tastendruck wird der Sweep-Modus ein- bzw. ausgeschaltet (durch ein „<->“ markiert).

Solange der Sweep-Modus eingeschaltet ist, wird die Pulslänge durch die Sweep-Zeit ersetzt, welche mittels dem Drehencoder geändert werden kann.

Wie üblich beenden zwei kurze Tastendrucke die Funktion.
Beschaltung bei Signalausgabe über die Testpins:

Pin #2: PWM-Ausgang (680Ω Widerstand zur Strombegrenzung)

Pin #1 und #3: Masse

Hinweis: Für den Servo benötigst Du eine zusätzliche Stromversorgung.

Hersteller	Pin 1	Pin 2	Pin3
Airtronics	PWM weiss/schwarz	Gnd schwarz	Vcc rot
Futaba	PWM weiss	Vcc rot	Gnd schwarz
hitec	PWM gelb	Vcc rot	Gnd schwarz
JR Radios	PWM orange	Vcc rot	Gnd braun

Tabelle 3.1. Pinbelegungen für typische 3-Pin-Servo-Stecker

3.1.20. DS18B20-Temperatursensor Hiermit wird der OneWire-Temperatursensor DS18B20 ausgelesen. Zum Einrichten des OneWire-Busses Seite 43 siehe bitte den Abschnitt „Busse“. Bei Benutzung der Test-Pins informiert der Tester über die Beschaltung und wartet bis ein externer Pull-Up-Widerstand erkannt wurde. Mit einem Tastendruck lässt sich dies überspringen. Nach den Verbinden des DS18B20 als einziger Client am Bus startet ein Tastendruck das Auslesen der Temperatur (kann fast eine Sekunde dauern). Zum Beenden zweimal kurz die Test-Taste drücken.

Beschaltung für das DS18B20 Sensor:

Probe #1: Masse/Gnd

Probe #2: VSS (680Ω Widerstand zur Strombegrenzung)

Probe #3: Data/In

Hinweis: Parallel zum Sensor muss noch ein (4,7kΩ Widerstand zwischen #2 (VSS) und #3 (Data/In) eingesetzt werden.

Zum Beenden zweimal kurz die Test-Taste drücken.

3.1.21. DHTxx-Sensoren Zum Lesen von DHT11, DHT22 und kompatiblen Temperatur & Luftfeuchte-Sensoren.

Zuerst zeigt der Tester die Beschaltung der Test-Pins und wartet auf den externen Pull-Up-Widerstand.

Danach wird der ausgewählte Sensortyp angezeigt (Standard: DHT11), welcher durch einen kurzen Druck der Testtaste gelesen wird.

Bei erfolgreichem Lesen gibt der Tester die Messwerte aus, bei einem Fehler nur ein ".

Ein langer Tastendruck ändert den Sensortypen, und zwei kurze Tastendrucke beenden die Funktion. Beim Ändern des Sensortyps hast Du die Möglichkeit, den automatischen Lesemodus (jede Sekunde) zu aktivieren. Dieser wird durch ein "*" nach dem Sensornamen signalisiert.

Unterstützte Sensoren:

DHT11: DHT11, RHT01

DHT22: DHT22, RHT03, AM2302

DHT21, RHT02, AM2301, HM2301

DHT33, RHT04, AM2303

DHT44, RHT05

Beschaltung von Test-Pins:

Probe #1: Gnd

Probe #2: Data

Probe #3: Vdd (Strom nicht begrenzt)

Ein externer Pull-Up-Widerstand von 4,7kOhm zwischen Data und Vdd wird benötigt! Manche Sensormodule haben bereits einen 10kOhm Pull-Up-Widerstand integriert, welcher ebenfalls gut mit kürzeren Kabeln funktioniert.

Hinweis:

Wegen des Strombedarfs des Sensor kann der 680 Ohm Testwiderstand nicht zur Strombegrenzung genutzt werden. Also Vorsicht, ein Kurzschluss kann die MCU beschädigen.

3.1.22. Selbsttest Wenn Du den Selbsttest über das Menü gestartet hast, bittet dich der Tester die drei Testpins kurz zu schließen und wartet solange, bis er dies erkennt.

Bei Problemen kannst Du das Warten mit einem Tastendruck abbrechen.

Der Selbsttest führt jeden Test 5-mal aus.

Mit einem kurzen Tastendruck wird der aktuelle Test übersprungen,
mit einem langen Tastendruck der komplette Test.

In Test #4 ist der Kurzschluss wieder zu entfernen. Der Tester wartet dann so lange.

Die Testschritte sind:

- T1 interne Spannungsreferenz (in mV)
- T2 Vergleich der Rl-Widerstände (Offset in mV)
- T3 Vergleich der Rh-Widerstände (Offset in mV)
- T4 Entfernen des Kurzschlusses der Testpins/kabel
- T5 Leckstromtest für Testpins mit Gnd-Pegel (Spannung in mV)
- T6 Leckstromtest für Testpins mit Vcc-Pegel (Spannung in mV)

3.1.23. Selbstabgleich Der Selbstabgleich misst den Widerstand und die Kapazität der Messkabel, d.h. von Platine, interner Verkabelung und dem Messkabel als Summe, um einen Nulloffset zu bestimmen. Auch wird der interne Widerstand der MCU-Portpins im Pull-Up und Pull-Down Modus bestimmt. Wenn der Abgleich übersprungen wird oder unplausible Werte gemessen werden, werden die Standardwerte der Firmware angenommen.

Wenn alles sauber durch läuft, werden die neuen Werte angezeigt, aber **nicht** im EEPROM gespeichert (siehe Speichern).

Der Spannungsoffset des Analogkomparators wird automatisch bei der Messung eines Kondensators bestimmt (bei der normalen Bauteilesuche), wenn der Kondensator einen Wert zwischen 100nF und 3,3µF hat.

Außerdem wird gleichzeitig der Offset der internen Spannungsreferenz gemessen.

Bevor der Selbstabgleich ausgeführt wird, solltest Du einen Folienkondensator mit einer Kapazität zwischen 100nF und 3,3µF min. 3-mal messen, damit die oben erwähnten Offsets bestimmt werden können.

Typischerweise liefert die erste Messung einen zu niedrigen Wert, die zweite einen zu hohen und erst die dritte einen korrekten Wert.

Das wird durch die sich selbst abgleichenden Offsets verursacht.

Mit einem festen Kondensator zum Selbstabgleich wird der automatische Abgleich in der Kapazitätsmessung durch eine eigene Funktion ersetzt, welche während des Selbstabgleichs ausgeführt wird.

Somit brauchst Du nicht mehr vorher einen Folienkondensator zu messen.

Falls der Kapazitätsoffset zwischen den Test-Pin-Paaren zu sehr variiert, kannst Du in config.h auf Test-Pin spezifische Offsets umschalten (CAP_MULTIOFFSET) Seite 41.

Der Selbstabgleich ist dem Selbsttest vom Ablauf und der Bedienung her sehr ähnlich.

Die Schritte des Selbstabgleichs sind:

- A1 Offsets für interne Spannungsreferenz und Analogkomparator

- (nur bei festem Abgleichkondensator)
- A2 Widerstand der Testpins/Kabel (in $10\text{m}\Omega$)
 - A3 Entfernen des Kurzschlusses der Testpins/Kabel
 - A4 interner Widerstand der Ports-pins für Gnd (Spannung über RiL)
 - A5 interner Widerstand der Ports-pins für Vcc (Spannung über RiH)
 - A6 Kapazität der Testpins/Kabel (in pF)
- Erlaubte Maximalwerte: - Widerstand Testpin/Kabel $< 1,50\ \Omega$ (zwei in Reihe)
- Kapazität Testpin/Kabel $< 100\text{pF}$

Hinweis: Wenn der Widerstandswerte der Testpins zu sehr variieren, könnte ein Kontaktproblem vorliegen.

Merke: Abgleich ist nicht Kalibrierung!

Kalibrierung ist die Prozedur, Messergebnisse mit verfolgbaren Standards zu vergleichen und die Abweichungen zu notieren. Der Zweck ist die Überwachung der Abweichungen über die Zeit.

Der Abgleich ist die Prozedur, ein Messgerät so einzustellen, dass es seine Vorgaben bzgl. Genauigkeit und anderer Parameter einhält.

3.1.24. Speichern/Laden Beim Brennen der Firmware wird ein Satz vordefinierter Standardwertwerte in das EEPROM geschrieben.

Nach dem Selbstabgleich kannst mit dieser Funktion die Standardwerte durch die korrekten Werte überschreiben.

Beim nächsten Neustart vom Tester werden dann diese Werte (Profil #1) automatisch geladen und benutzt.

Zur Bequemlichkeit stehen zwei Profile zum Speichern bzw. Laden zu Verfügung, z.B. für zwei unterschiedliche Sätze an Messkabeln.

Die Idee hinter der manuellen Speichern-Funktion ist, dass man z.B. beim temporären Wechsel der Messkabel nur einen Selbstabgleich macht und nach dem Neustart wieder die Werte für die Haupt-Messkabel hat.

Ansonsten müsste man für die alten Kabel wieder einen neuen Selbstabgleich machen.

3.1.25. Werte anzeigen Diese Funktion zeigt die aktuellen Abgleich-werte an. Die Nutzung einer externen 2.5V Spannungsreferenz wird mit einem „*“ nach Vcc signalisiert.

3.1.26. Ausschalten Hiermit kannst Du den Tester abschalten, sofern die Funktion über SW_POWER_OFF auf Seite 40 aktiviert wurde.

3.1.27. Exit Damit kannst Du das Menü verlassen, wenn Du z.B. aus Versehen reingegangen bist.

4.0.1. Widerstände Widerstände werden zweimal gemessen. d.h. in beide Richtungen, und die Werte dann verglichen. Wenn die Werte zu unterschiedlich sind, nimmt der Tester an, daß es zwei Widerstände sind und nicht nur einer.

In dem Fall zeigt die Ausgabe zwei Widerstände mit gleichen Pins in der Form „1 – 2 – 1“ mit den beiden Werten an.

Für Widerstände kleiner als 10 Ohm wird eine zusätzliche Messung mit höherer Auflösung durchgeführt.

In seltenen Fällen kann der Tester sehr kleine Widerstände nicht erkennen. Am besten dann die Messung einfach wiederholen.

4.0.2. Kondensatoren Die Messung von Kondensatoren ist in drei Methoden aufgeteilt.

Große Kondensatoren $>47\mu\text{F}$ werden mittels Ladezyklen Methode mit Pulsen von 10ms gemessen. Mittelgroße zwischen $4,7\mu\text{F}$ and $47\mu\text{F}$ werden ebenfalls mit der Ladezyklen Methode gemessen, aber mit Pulsen von 1ms.

Kleine Kondensatoren laufen über den analogen Komparator.

Auf dies Weise wird die Genauigkeit der Kapazitätsmessung optimiert.

Die Messwerte großer Kondensatoren benötigen eine eine Korrektur.

Ohne Korrektur wären die Werte zu hoch. Ich denke, dass dies durch die Messmethode verursacht wird, da die Analog-Digital-Wandlung nach dem Ladepuls eine gewisse Zeit benötigt und der Kondensator in dieser Zeit etwas Ladung durch Leckströme verliert und natürlich auch durch die Analog-Digital-Wandlung selber.

So dauert es dann länger bis der Kondensator geladen ist, und die Kapazität erscheint höher. Eine spätere Selbstentlademessung versucht dies zu kompensieren, kann es aber nur teilweise.

Die Korrekturfaktoren (CAP_FACTOR_SMALL, CAP_FACTOR_MID und CAP_FACTOR_LARGE in config.c) auf Seite 41, sind so gewählt, daß sie für die meisten Testermodelle passen. In manchen Fällen kann aber eine Änderung notwendig sein.

Eine Logik zum Verhindern, dass große Kondensatoren als Widerstände erkannt werden, wurde hinzugefügt. Widerstände kleiner als 10Ω werden zusätzlich auf Kapazität geprüft.

Der Tester versucht den ESR bei Kondensatoren größer als 18nF zu messen.

Alternativ kannst Du auch die alte Messmethode aktivieren, welche ab 180nF misst.

Da aber die Messung nicht per Wechselstrom mit einer bestimmten Frequenz durchgeführt wird, bitte keine super genauen Ergebnisse erwarten.

Die benutzte Methode entspricht vielleicht der Messung mit 1kHz.

Für die Prüfung von Elkos ist die Messung mehr als ausreichend.

Bei Filmkondensatoren mit kleinen Werten, können je nach MCU-Taktrate unterschiedliche Werte zustande kommen.

Ich denke, Herr Fourier könnte dies erklären.

Ein gemessener Kapazitätswert von mehr als 5pF (incl. Nulloffset) wird als gültig angesehen. Niedrigere Werte sind zu ungenau und könnten durch Verschiedenes, wie z.B. anderes Platzieren der Messkabel, verursacht werden.

Eine weitere Messung ist der Selbstentladungsleckstrom für Kondensatoren größer als $4,7\mu\text{F}$. Der Wert gibt z.B. einen Hinweis auf den Zustand eines Elkos.

Von meinen Tests her scheinen folgende Werte typisch für Elkos zu sein:

- 10-220 μF	1-3 μA
- 330-470 μF	4-5 μA
- 470-820 μF	4-7 μA
- >1000 μF	5-7 μA pro 1000 μF

4.0.3. Induktivitäten Die Induktivitätsmessung ist nicht sonderlich genau, und Dinge wie z.B. der MCU-Takt und die Platine haben Auswirkungen auf das Ergebnis.

Die Messung selber basiert auf der Bestimmung der Zeit zwischen dem Einschalten des Stroms und dem Erreichen eines bestimmten Stroms.

Für große Induktivitäten gibt es eine Niedrig-Strom Messung, und für kleine Induktivitäten eine Hoch-Strommessung, welche das Stromlimit der MCU-Pins kurzzeitig überschreitet (bis zu etwa 25 Mikrosekunden).

Bei der Untersuchung der Effekte des MCU-Takts und anderer Dinge habe ich ein Muster an Abweichungen gefunden, welches zum Kompensieren genutzt werden kann.

Je nach Tester kann eine Anpassung notwendig sein. In der Datei inductance.c in der Funktion MeasureInductor() gibt es die Variable Offset für Kompensation.

Sie ist ein Offset für die Referenzspannung.

Ein positiver Wert verkleinert die Induktivität, und ein negativer Wert vergrößert die Induktivität.

Die Kompensation für die Hochstrommessung basiert auf den MCU-Takt und ist in drei Zeitbereiche unterteilt. Für die Niedrig-Strom Messung gibt es momentan nur eine einfache Kompensation, da hier noch weitere Tests notwendig sind.

Wenn Du größere Abweichungen beim Vergleich mit einem „richtigen“ LCR-Meter siehst, kannst Du die Offset werte entsprechend deinem Tester anpassen.

Hinweis: Bei unerwarteten Messwerten bitte die Messung wiederholen.

4.0.4. Bauteile entladen Der Tester versucht vor und während des Messens das angeschlossene Bauteil zu entladen. Wenn er das Bauteil nicht auf einen vorgegebenen Nullwert (CAP_DISCHARGED) entladen kann, gibt er einen Fehler mit Angabe des Testpins und der Restspannung aus.

Die Entladefunktion basiert nicht auf einem festen Timeout, sondern passt sich automatisch dem Entladefortschritt an. Auf diese Weise wird eine Batterie schneller erkannt und große Kondensatoren erhalten mehr Zeit zum Entladen. Sollte ein großer Elko als Batterie erkannt werden, bitte nochmal versuchen. In einer Umgebung mit vielen elektrischen Störungen könnte auch der Nullwert CAP_DISCHARGED mit 2mV zu niedrig sein (ggf. anpassen). Die angezeigte Restspannung hilft beim Finden des optimalen Nullwertes.

4.0.5. ADC Oversampling Die ADC-Funktion unterstützt ein variables Oversampling (1 - 255).

Der Standardwert ist 25 Samples.

Du kannst versuchen, durch Erhöhen des Oversamplingwerts die Genauigkeit des Testers zu erhöhen.

Allerdings steigt mit einem höheren Wert auch die benötigte Zeit, d.h. die Messungen werden langsamer.

4.0.6. V_{BE} von Bipolartransistoren Beim Test auf Dioden wird Vf zum einen mit Rl (hoher Teststrom) und zum anderen mit Rh (niedriger Teststrom) gemessen.

Die Ausgabefunktion für Bipolartransistoren interpoliert aus den beiden Vf-Werten V_{BE} abhängig vom hFE für einen virtuellen Teststrom.

Somit erhält man praxisnahe Werte, da V_{BE} eines Kleinsignaltransistors mit einem anderen Strom gemessen wird als bei einem Leistungstransistor.

4.0.7. Ergebnisanzeige Einige Namen und Abkürzungen wurden geändert. Die Ausgabe mancher Bauteile wird auf mehrere Seiten aufgeteilt, wenn das LCD-Modul nicht ausreichend viele Zeilen hat.

Bei einer einzelnen Diode wird das Vf der Messung mit niedrigem Teststrom (10μA) in Klammern angezeigt, wenn der Wert unter 250mV liegt.

Damit erhält man einen Hinweis auf eine Germanium-diode.

Die meisten Datenblätter von Germaniumdioden geben einen Messstrom von 0,1mA für Vf an.

Leider unterstützt der Tester nicht diesen Messstrom.

Und für höhere Ströme liegt Vf bei etwa 0,7V, was eine Unterscheidung zu Silizium Dioden schwierig macht.

Der Leckstrom I_R für eine einzelne Diode bzw. I_{CEO} für einen Bipolar-Transistor wird ausgegeben, sofern er höher als 50nA ist.

Germanium-transistoren haben einen Leckstrom im Bereich von wenigen μA bis etwa $500\mu A$. Germaniumdioden liegen üblicherweise bei ein paar μA .

Für manche Bauteile wird ein Kapazitätswert angezeigt. Liegt die Kapazität unterhalb von 5pF oder die Messung schlug fehl, wird 0pF ausgegeben.

Wenn ein Verarmungs-FET mit symmetrischem Drain und Source gefunden wird, zeigt die Pinbelegung ein 'x' statt 'D' oder 'S', da beide vom Tester nicht unterschieden werden können, sie sind funktional identisch.

In diesem Fall bitte im Datenblatt nach Details der Pinbelegung schauen.

Die Pinbelegung eines Triac wird mit den Pins 'G', '1' and '2' angezeigt.

'1' ist MT1 und '2' ist MT2.

Und für einen UJT, sofern die Erkennung aktiviert ist, ist es

'1' für B1, '2' für B2 und 'E' für den Emitter.

Wenn die „Fancy Pinout“ Funktion aktiviert wurde (über das Setzen einer Symboldatei in config.h), wird das Bauteilsymbol mit den entsprechenden Testpins für 3-Pin-Halbleiter angezeigt.

Sollte nicht genügend Platz auf dem Display sein, wird die Ausgabe des Symbols übersprungen.

4.0.8. Zusätzliche Hinweise

Bipolartransistoren Bei einem Bipolartransistor mit Basis-Emitter-Widerstand wird der Widerstand angezeigt.

Beachte, dass der B-E-Widerstand Einfluss auf V_{BE} und hFE hat. Wenn der Transistor zusätzlich eine Schutzdiode hat, kann er als Transistor oder zwei Dioden erkannt werden, je nach Wert des Basis-Emitter-Widerstands. Im letzteren Fall werden zwei Dioden plus Widerstand mit dem Hinweis auf einen möglichen Transistor angezeigt. Ein niedriger Basis-Emitter-Widerstand verhindert leider die eindeutige Erkennung des Transistors.

Ein weiterer Spezialfall ist ein Bipolartransistor mit integrierter Schutzdiode, die sich auf dem gleichen Substrat befindet.

Der integrierte PN-Übergang erzeugt einen parasitären zweiten Transistor.

Ein NPN bekommt somit einen parasitären PNP und umgekehrt.

Wird ein solcher Transistor entdeckt, wird er mit einem "+" hinter der Typenangabe gekennzeichnet.

TRIACs s werden in drei bzw. vier Modi, auch als Quadranten bekannt, betrieben. Typischerweise unterscheiden sich manche Parameter je nach Quadrant, wie z.B. der Triggerstrom I_{GT} . In manchen Fällen passiert es, dass der Teststrom vom Tester ausreichend ist, den TRIAC in einem Quadranten zu triggern, aber nicht in einem anderen. Da der Tester zwei Testläufe benötigt, um die Pins für MT1 und MT2 zu bestimmen, kann der Tester in solchen Fällen die Pins nicht unterscheiden, d.h. sie könnten vertauscht sein.

Manche TRIACs können vom Tester getriggert werden, haben aber einen zu hohen Haltestrom (I_H), wodurch sie nicht korrekt erkannt werden können. Wenn bei einem TRIAC der Triggerstrom zu hoch für den Tester ist, wird er meistens als Widerstand erkannt.

CLDs Die Diodenprüfung erkennt eine CLD (Current Limiting Diode) als normale Diode und gibt ihren Strom I_F als Leckstrom aus.

- Zu beachten ist, daß bei einer CLD Anode und Kathode gegenüber einer normalen Diode vertauscht sind.

- Eine gesonderte Erkennung von CLDs ist schwierig, da der Leckstrom einer Germanium oder Leistungs-Schottky-Diode im Bereich von I_F (ab ca. $33\mu A$) liegt.

- Wenn eine Diode ein ungewöhnliches V_f hat, ein niedriges V_f beim Niedrigstromtest (zweiter Wert in Klammern) und keine Kapazität gemessen werden konnte, dann ist es wahrscheinlich eine CLD.

Nicht unterstützte Bauelemente Alle Halbleiter, welche einen hohen Steuerstrom benötigen, können nicht erkannt werden, da der Tester max. ca. 7mA Strom zum Schalten hat. Auch liefert der Tester nur 5V, was z.B. nicht ausreichend für DIACs mit einem V_{BO}

von 20-200V ist.

Bekannte Probleme - Ein Speicher- bzw. Superkondensator, wie z.B. Panasonic NF Serie, wird als Diode oder zwei anti-parallele Dioden erkannt.

Die Kapazitätsmessung kann keinen brauchbaren Wert bestimmen.

- Bei Verwendung eines Schaltnetztes oder DC-DC-Konverters zur Strom- versorgung gibt der Tester manchmal fälschlicherweise einen Elko um die $50\mu\text{F}$ aus, obwohl kein Bauteil angeschlossen ist.

- Der ESR kann bei Kondensatoren mit 180 - 220nF je nach MCU-Takt variieren.

4.0.9. Hilfe Hilfe findest Du in folgenden zwei Foren: [10] und [11].

4.0.10. Firmware-Änderungen Findest du im Kapitel 9 ab Seite 63.

Wie bereits erwähnt kann die Firmware für verschiedene Tester und Zusatzfunktionen angepaßt werden. Dazu gibt es einige Einstellungen im Makefile, in config.h und config_ <MCU>.h. Dieses Kapitel erklärt die Einstellungen. Das Makefile steuert das Übersetzen des Quellcodes und enthält grundsätzliche Dinge, wie z.B. den MCU-Typen und ISP-Programmierer. In der Datei config.h befinden sich allgemeine Einstellungen zu Bedienung und Funktionen. Und config_ <MCU>.h ist für Dinge auf der Hardwareebene zuständig, also für LCD-Module und die Zuordnung der Pins.

5.1. Makefile

Im Makefile werden Einstellungen durch das Setzen von bestimmten Variablen durchgeführt. Zum Anpassen einfach den Wert oder die Zeichenkette hinter der Variablen ändern. Für manche Variablen gibt es mehrere Vorschläge, die mittels des #-Symbols auskommentiert sind. Dort bitte die gewünschte Einstellung Einkommentieren (# löschen), und ggf. die Standardeinstellung auskommentieren (# einfügen).

5.1.1. MCU-Typ

```
# avr-gcc: MCU model
# - ATmega 328/328P : atmega328
# - ATmega 324P/324PA : atmega324p
# - ATmega 644/644P/644PA : atmega644
# - ATmega 1284/1284P : atmega1284
MCU = atmega328
```

Listing 5.1. Worgewählt ist atmega328

5.1.2. MCU-Taktfrequenz

```
# MCU frequency:
# - 1MHz : 1
# - 8MHz : 8
# - 16MHz : 16
# - 20MHz : 20
FREQ = 8
```

Listing 5.2. Worgewählt ist 8MHz

5.1.3. Oszillator-Typ

```
# oscillator type
# - internal RC oscillator : RC
# - external full swing crystal : Crystal
# - external low power crystal : LowPower
OSCILLATOR = Crystal
```

Listing 5.3. Worgewählt ist Crystal

5.1.4. Avrdude MCU-Typ

```
# avrdude: part number of MCU
# - ATmega 328 : m328
# - ATmega 328P : m328p
# - ATmega 324P : m324p
# - ATmega 324PA : m324pa
# - ATmega 644 : m644
# - ATmega 644P : m644p
# - ATmega 644PA : m644p
# - ATmega 1284 : m1284
# - ATmega 1284P : m1284p
PARTNO = m328p
```

Listing 5.4. Worgewählt ist m328p

5.1.5. Avrdude ISP-Programmierer

Bei dem Programmierer sind notwendig:

- Name
- BitClock
- Port

```
#
# avrdude: ISP programmer
#
# Vorwahl Buspirate
# PROGRAMMER = buspirate
# BITCLOCK=10
# PORT = /dev/bus_pirate
# Vorwahl USBasp von Fischl
# PROGRAMMER = USBasp
# BITCLOCK=20
# PORT = usb
# Vorwahl USBtiny ISP
# PROGRAMMER = usbtiny
# BITCLOCK=5
# PORT = usb
# Vorwahl Pololu
# PROGRAMMER=stk500v2
# BITCLOCK=1.0
# PORT = /dev/ttyACM0
# Vorwahl Diamex
PROGRAMMER = avrispmkII
BITCLOCK=5.0
PORT = usb
```

Listing 5.5. Worgewählt ist Diamex

Die Liste der Programmer wurde hier schon editiert und einige bekannte Programmer und erprobte Einstellungen zugefügt.

Falls Dein Programmierer nicht aufgeführt ist, bitte die passenden Werte eintragen.

Weitere Informationen findest Du im Avrdude-Handbuch oder der Online-Dokumentation [9].

5.2. config.h

Diese Datei dient zum Einstellen von Bedienung und Funktionen. Da es eine normale C-Header-Datei ist, werden die bekannten Kommentar regeln für C verwendet. Um etwas zu aktivieren, ist das //äm Zeilenanfang zu löschen. Zum Deaktivieren wird ein //äm Zeilenanfang eingefügt. Manche Einstellungen benötigen einen Zahlenwert, der ggf. anzupassen ist.

5.2.1. Hardware Bedienung

Drehencoder zur Bedienung

- Standard-Pins: PD2 & PD3 (ATmega 328)
- könnte parallel zum LCD-Modul liegen
- siehe ENCODER _PORT für Port-Pins (config- <MCU> .h)
- //#define HW_ENCODER - zum Aktivieren einkommentieren

Anzahl der Grey-Code-Impulse pro Schritt oder Rastung

- Ein Drehencoderimpuls ist die vollständige Folge von 4 Gray-Code-Impulsen
- typische Werte: 2 oder 4, selten 1
- #define ENCODER_PULSES ... 4 - Passe den Wert an Deinen Drehencoder an.

Anzahl der Rastungen oder Schritte

- wird zur Erfassung der Drehgeschwindigkeit des Drehencoders verwendet
- muss nicht genau übereinstimmen und ermöglicht Dir die Feinabstimmung (höherer Wert: langsamer, niedrigerer Wert: schneller)
- typische Werte: 20, 24 oder 30
- #define ENCODER_STEPS ... 24 - Passe den Wert an Deinen Drehencoder an

Mehr/Weniger-Tasten zur Bedienung

- Alternative zu Drehencoder
- siehe KEY _PORT für Port-Pins (config- <MCU> .h)
- //#define HW_INCDEC_KEYS - zum Aktivieren einkommentieren

2,5-V-Spannungsreferenz für Vcc-Prüfung

- Standard-Pin: PC4 (ATmega 328)
- sollte mindestens 10-mal genauer als der Spannungsregler sein
- siehe TP _REF für den Port-Pin (config- <MCU> .h)
- ggf. UREF _25 weiter unten für Deine Spannungsreferenz anpassen
- //#define HW_REF25 - zum Aktivieren einkommentieren

Typische Spannung von 2,5 V Referenzspannung (in mV)

- siehe Datenblatt der Spannungsreferenz
- oder nehme >= 5,5-stelliges DMM, um die Spannung zu messen
- #define UREF_25 ... 2495 / -Bei Bedarf Wert ändern

Schutzrelais zum Entladen von Kondensatoren

- Standard-Pin: PC4 (ATmega 328)
- Low-Signal: Testpins kurzschließen
- High-Signal über externe Referenz: Kurzschluss beseitigen
- //#define HW_DISCHARGE_RELAY - zum Aktivieren einkommentieren

Spannungsmessung bis 50V DC / Zenerdioden Test

- Standard-Pin: PC3 (ATmega 328)
- 10:1 Spannungsteiler
- für Zenerdioden
- DC-DC-Boostkonverter, der über den Testtaster gesteuert wird
- Siehe TP_ZENER für den Port-Pin
- //#define HW_ZENER - zum Aktivieren einkommentieren

Hohe Auflösung für den Zener-Check

- 10 mV anstelle von 0,1 V
- `//#define ZENER_HIGH_RES` - zum Aktivieren einkommentieren

Fester Signalausgang

- falls der OC1B-Pin der MCU als dedizierter Signalausgang verdrahtet ist, anstatt den RL-Widerstand von Testpin # 2 anzusteuern
- `//#define HW_FIXED_SIGNAL_OUTPUT` - zum Aktivieren einkommentieren

Einfacher Frequenzzähler

- Standard-Pin: T0 (PD4 ATmega 328) direkt als Frequenzeingang
- zählt bis zu 1/4 der MCU-Taktfrequenz
- Möglicherweise parallel zum LCD-Modul
- `//#define HW_FREQ_COUNTER_BASIC` - zum Aktivieren einkommentieren

Erweiterter Frequenzzähler

- Nieder- und Hochfrequenz-Quarzoszillatoren und gepufferter Frequenzeingang
- Vorkalierer 1:1 und 16:1 (32:1)
- siehe COUNTER_PORT für Port-Pins (config- <MCU> .h)
- erfordert eine Anzeige mit mehr als 2 Textzeilen
- setze die Prescaler-Einstellung der Schaltung: entweder 16:1 oder 32:1
- `//#define HW_FREQ_COUNTER_EXT` - zum Aktivieren einkommentieren
- `#define FREQ_COUNTER_PRESCALER ... 16 / 16:1 /` - vorgewählt
- `//#define FREQ_COUNTER_PRESCALER ... 32 / 32:1 /`

Ereigniszähler

- Standard-Pin: T0 (PD4 ATmega 328)
- verwendet T0 direkt als Ereignis- / Impulseingang (steigende Flanke)
- kein gemeinsamer Betrieb mit Anzeigen für T0 möglich
- erfordert zusätzliche Tasten (z. B. Drehgeber) und ein Display mit mehr als 5 Zeilen
- nur für MCU-Takt von 8, 16 oder 20 MHz
- `//# definiere HW_EVENT_COUNTER` - zum Aktivieren einkommentieren

Ausgang für Ereigniszähler auslösen

- Verwendet Pin # 2 als Triggerausgang, Pins # 1 und # 3 sind Gnd
- setzt den Trigger-Ausgang während des Zählens auf High
- `//# definiere EVENT_COUNTER_TRIGGER_OUTR` - zum Aktivieren einkommentieren

IR-Detektor/Dekoder (über dedizierten MCU-Pin)

- erfordert ein IR-Empfängermodul, z.B. TSOP-Serie
- das Modul ist an einen festen E/A-Pin angeschlossen
- siehe IR_PORT für den Port-Pin (config- <MCU> .h)
- für zusätzliche Protokolle aktiviere SW_IR_RX_EXTRA
- `//#define HW_IR_RECEIVER` - zum Aktivieren einkommentieren

Fester Kondensator für Selbstabgleich

- Siehe TP_CAP und ADJUST_PORT für Port-Pins (config- <MCU> .h)
- `//#define HW_ADJUST_CAP` - zum Aktivieren einkommentieren

Relais für Parallelkondensator (Sampling-ADC)

- `//#define HW_CAP_RELAY` - zum Aktivieren einkommentieren

5.2.2. Software Optionen

PWM-Generator mit einfacher Bedienung

- `#define SW_PWM_SIMPLE` - zum Deaktivieren auskommentieren

PWM-Generator mit erweiterter Bedienung

- erfordert zusätzliche Tasten und Anzeige mit mehr als 2 Textzeilen
- `//#define SW_PWM_PLUS` - zum Aktivieren einkommentieren

Induktivitätsmessung

- `#define SW_INDUCTOR` - zum Deaktivieren auskommentieren

ESR-Messung und In-Circuit-ESR-Messung

- MCU-Takt >= 8 MHz erforderlich
- `#define SW_ESR` - zum Deaktivieren auskommentieren
- Wähle `SW_OLD_ESR` für die alte Messmethode ab 180nF
- `//#define SW_OLD_ESR` - zum Aktivieren einkommentieren

Drehencoder-Prüfung

- `//#define SW_ENCODER` - zum Aktivieren einkommentieren

Rechtecksignalgenerator

- erfordert zusätzliche Tasten oder Drehencoder
- `#define SW_SQUAREWAVE` - zum Deaktivieren auskommentieren

IR-Detektor/Dekoder (über Testpins)

- erfordert ein IR-Empfängermodul, z. TSOP-Serie
- Das Modul wird an die Testpins angeschlossen
- `#define SW_IR_RECEIVER` - zum Deaktivieren auskommentieren

Strombegrenzungswiderstand für IR-Empfängermodul

- nur für 5V-Module
- Warnung: Jeder Kurzschluss kann die MCU zerstören
- `//#define SW_IR_DISABLE_RESISTOR` - zum Aktivieren einkommentieren

Zusätzliche Protokolle für den IR-Detektor/Dekoder

- seltenere Protokolle, die die Nutzung des Flash-Speichers erhöhen;)
- `//#define SW_IR_RX_EXTRA` - zum Aktivieren einkommentieren

IR-Fernbedienungssender

- erfordert zusätzliche Tasten und Anzeige mit mehr als 4 Textzeilen
- erfordert auch eine IR-LED mit einem einfachen Treiber
- `//#define SW_IR_TRANSMITTER` - zum Aktivieren einkommentieren

Alternative Verzögerungsschleife für IR-Fernbedienungssender

- für den Fall, dass der C-Compiler die Standardverzögerungsschleife vermässelt und falsche Puls-/Pausen Zeiten verursacht
- `//#define SW_IR_TX_ALTDELAY` - zum Aktivieren einkommentieren

Zusätzliche Protokolle für IR-Fernbedienungssender

- seltenere Protokolle, die die Nutzung des Flash-Speichers erhöhen;)
- `//#define SW_IR_TX_EXTRA` - zum aktivieren Auskommentieren

Optokoppler-Test

- `#define SW_OPTO_COUPLER` - zum Deaktivieren einkommentieren

Unijunction-Transistoren prüfen

- `#define SW_UJT` - zum Deaktivieren auskommentieren

Servo-Test

- erfordert zusätzliche Tasten und Anzeige mit mehr als 2 Textzeilen
- `//#define SW_SERVO` - zum Aktivieren einkommentieren

DS18B20

- aktiviere auch ONEWIRE _PROBES oder ONEWIRE _IO _PIN (siehe Abschnitt 'Busse')
- `//#define SW_DS18B20` - zum Aktivieren einkommentieren

Kondensatorleckstromtest

- erfordert eine Anzeige mit mehr als zwei Zeilen
- `//#define SW_CAP_LEAKAGE` - zum Aktivieren einkommentieren

Anzeige der umgekehrten hFE für BJTs

- hFE für Kollektor und Emitter vertauscht
- `#define SW_REVERSE_HFE` - zum Deaktivieren aus kommentieren

Überwachen von Widerstand und Induktivität auf Pins # 1 und # 3

- `//#define SW_MONITOR_RL` - zum Aktivieren einkommentieren

Überwachen von Kapazität auf Pins # 1 und # 3

- `//#define SW_MONITOR_C` - zum Aktivieren einkommentieren

DHT11, DHT22 und kompatible Feuchtigkeits- und Temperatursensoren

- `//# definiere SW_DHTXX` - zum Aktivieren einkommentieren

Für einige Tester Deaktivierung der hFE-Messung mit gemeinsamer Kollektorschaltung und Rl als Basiswiderstand

- Problem:
hFE-Werte sind zu hoch, weil die Basisspannung zu niedrig gemessen wird
- betroffene Tester:
Hiland M664 (in Bearbeitung)
- `//#define NO_HFE_C_RL` - zum Aktivieren einkommentieren

Startzyklen des Oszillators (nach dem Aufwecken aus dem Stromsparmodes):

- typische Werte
- Interner RC-Oszillator: 6
- Quarzkristall: ... 16384 (auch 256 oder 1024 je nach Fuse-Einstellungen)
- Resonator: ... 16384 (auch 256 oder 1024, je nach Fuse-Einstellungen)
- Bitte den Wert ändern, wenn er nicht zu Deinem Tester passt!

```
#ifndef OSC_STARTUP
#define OSC_STARTUP 16384
#endif
```

Listing 5.6. Wert ändern falls er NICHT zum Tester passt!

5.2.3. Benutzerschnittstelle

Sprachwahl

```
#define UI_ENGLISH
//#define UI_CZECH
//#define UI_CZECH_2
//#define UI_DANISH
//#define UI_GERMAN
//#define UI_ITALIAN
//#define UI_POLISH
//#define UI_SPANISH
//#define UI_RUSSIAN
//#define UI_RUSSIAN_2
```

Listing 5.7. Wunschsprache auswählen

Komma anstelle eines Punktes , um einen Dezimalbruch anzugeben.

- `//#define UI_COMMA` - zum Aktivieren einkommentieren

Temperaturen in Fahrenheit anstelle von Celsius anzeigen.

`//#define UI_FAHRENHEIT` - zum Aktivieren auskommentieren

Standard-Betriebsmodus Auto-Hold - anstelle des kontinuierlichen Modus

`//#define UI_AUTOHOLD` - zum Aktivieren einkommentieren

Menüaufruf durch Kurzschluss aller drei Testpins.

- altes Standardverhalten

`//#define UI_SHORT_CIRCUIT_MENU` - zum Aktivieren einkommentieren

Hinweise anstelle des Cursors, falls verfügbar.

- zur Zeit nur "Menü / Test"

- erfordert zusätzliche Tasten und ein Display mit einer ausreichenden Anzahl von Textzeilen (empfohlen: > = 8 Zeilen)

`//#define UI_KEY_HINTS` - zum Aktivieren einkommentieren

Ausgabe über serielle TTL-Schnittstelle

- aktiviere ebenfalls SERIAL_BITBANG oder SERIAL_HARDWARE (siehe Abschnitt 'Busse')

`//#define UI_SERIAL_COPY` - zum Aktivieren einkommentieren

Bedienung des Testers über serielle TTL-Schnittstelle

- aktiviere ebenfalls SERIAL_BITBANG oder SERIAL_HARDWARE und SERIAL_RW

`//#define UI_SERIA_COMMANDS` - zum Aktivieren einkommentieren

Maximale Wartezeit nach dem Testen (in ms)

- gilt nur für den kontinuierlichen Modus

- Zeit zwischen der Ergebnisausgabe und dem Starten eines neuen Testlaufs.

`#define CYCLE_DELAY ... 3000` - ggf. Zeit ändern

Maximale Anzahl von Testläufen ohne gefundene Bauteile

- gilt nur für den kontinuierlichen Modus

- Wenn diese Zahl erreicht ist, schaltet sich der Tester aus.

`#define CYCLE_MAX ... 5` - ggf. Anzahl ändern

Automatisches Abschalten wenn eine Weile keine Taste gedrückt wird (in s).

- gilt nur für den Auto-Hold-Modus

`//#define POWER_OFF_TIMEOUT ... 60` - zum Aktivieren einkommentieren

Farbcodierung für Testpins

- erfordert Farb-LCD

- editiere colors.h, um die passenden Farben auszuwählen

`#define SW_PROBE_COLORS` - zum Deaktivieren auskommentieren

Menüpunkt Ausschalten

`//#define SW_POWER_OFF` - zum Aktivieren einkommentieren

Runden der Werten für DS18B20

- DS18B20 (0.1 °C/F)

`//#define UI_ROUND_DS18B20` - zum Aktivieren einkommentieren

5.2.4. Energieverwaltung

Batterieüberwachungsmodus

- BAT_NONE deaktiviert die Batterieüberwachung komplett

- BAT_DIREKTE direkte Messung der Batterie-Spannung (<5V)

- BAT_DIVIDER Messung über Spannungsteiler

`//#define BAT_NONE`

`//#define BAT_DIRECT`

`#define BAT_DIVIDER` -ausgewählt

Optionale externe Stromversorgung ohne Überwachung

- Manche Tester unterstützen eine zusätzliche externe Stromversorgung, die aber aufgrund der Schaltung keine Spannungsmessung erlaubt. Dies würde zum Abschalten des Testers wegen zu geringer Batteriespannung führen. Der Schalter unten verhindert das Abschalten, wenn die gemessene Spannung unter 0,9V liegt (verursacht durch den Leckstrom der Diode).
- ```
//#define BAT_EXT_UNMONITORED
```
- zum Aktivieren einkommentieren

### Spannungsteiler zur Batterieüberwachung

- BAT\_R1: oberer Widerstand in  $\Omega$
  - BAT\_R2: unterer Widerstand in  $\Omega$
- ```
#define BAT_R1 ... 10000
```
- ggf. Wert anpassen
-
- ```
#define BAT_R2 ... 3300
```
- ggf. Wert anpassen

### Spannungsabfall durch Verpolungsschutzdiode und Power-Management-Transistor (in mV)

- oder einen anderen Schaltungsteil in der Stromversorgung
  - Nimm Dein DMM und messe den Spannungsabfall!
  - Schottky-Diode ungefähr 200 mV / PNP BJT ungefähr 100 mV.
- ```
#define BAT_OFFSET ... 290
```
- ggf. Wert anpassen

Spannung für schwache Batterie (in mV)

- Tester warnt, wenn BAT_WEAK erreicht ist.
 - Spannungsabfall BAT_OFFSET wird bei der Berechnung berücksichtigt.
- ```
#define BAT_WEAK dots 7400
```
- ggf. Wert anpassen

### Spannung für leere Batterie (in mV)

- Der Tester schaltet sich aus, wenn BAT\_LOW erreicht ist.
  - Spannungsabfall BAT\_OFFSET wird bei der Berechnung berücksichtigt
- ```
#define BAT_LOW ... 6400
```
- ggf. Wert anpassen

Schlafmodus für geringeren Stromverbrauch

- ```
#define SAVE_POWER
```
- zum Deaktivieren auskommentieren

## 5.2.5. Messeinstellungen und Offsets

### ADC-Spannungsreferenz basierend auf Vcc (in mV)

- ```
#define UREF_VCC dots 5001
```
- ggf. Wert anpassen

Offset für die interne Spannungsreferenz (in mV): -100 bis 100

- Zum Ausgleich von Abweichungen zwischen Realwert und Messwert.
 - Der ADC hat eine Auflösung von ca. 4,88 mV für $V_{ref} = 5V$ (V_{cc}) und 1,07 mV für $V_{ref} = 1,1 V$ (Bandgap).
 - Wird zur gemessenen Spannung der Bandgap-Referenz addiert.
- ```
#define UREF_OFFSET ... 0
```
- ggf. Wert anpassen

### Genaue Werte der Testwiderstände

- Standardwert für Rl ist 680  $\Omega$
  - Der Standardwert für Rh beträgt 470 k $\Omega$
- ```
/ Rl in  $\Omega$  /
```
- ```
#define R_LOW ... 680
```
- ggf. Wert anpassen
- 
- ```
/ Rh in  $\Omega$  /
```
- ```
#define R_HIGH ... 470000
```
- ggf. Wert anpassen

### Offset für systematische Fehler der Widerstandsmessung mit Rh (470k) in $\Omega$

- Wenn Widerstände >20k zu hoch oder zu niedrig sind, entsprechend den Offset einstellen.

- Der Standardoffset beträgt 350Ω #define RH\_OFFSET ...3500 - ggf. Wert anpassen

**Widerstand der Testpins/-kabel** (in 0,01 Ω) - Standardoffset für Leiterbahnen und Testkabel

- Widerstand von zwei in Reihe geschalteten Testpins
- vorausgesetzt alle Testpins haben den gleichen / ähnlichen Widerstand
- wird durch Selbstabgleich aktualisiert

```
#define R_ZERO ...20
```

- ggf. Wert anpassen

**Kapazität der Testpins/-kabel** (in pF) - Standardoffset für MCU, Platine und Testkabel

- wird durch Selbstabgleich aktualisiert
- Beispiele von Kapazitäten für unterschiedliche Kabellängen:
  - 3pF      ungefähr 10cm
  - 9pF      ungefähr 30cm
  - 15pF     ungefähr 50cm
- Maximalwert      ...100

```
#define C_ZERO ...43
```

- ggf. Wert anpassen

**Testpin-spezifische Kapazität** anstelle von Durchschnittswert für alle Testpins

```
// #define CAP_MULTIOFFSET
```

- zum Aktivieren einkommentieren

**Maximale Endladespannung** für Kondensatoren (in mV)

- unterhalb welcher Spannung wir den Kondensator als entladen ansehen

```
#define CAP_DISCHARGED ...2
```

- ggf. Wert anpassen

**Korrekturfaktoren für Kondensatoren** (in 0,1%)

- positiver Faktor erhöht den Kapazitätswert
- negativer Faktor verringert den Kapazitätswert

```
CAP_FACTOR_SMALL für Kondensatoren < 4,7 μF
CAP_FACTOR_MID für Kondensatoren 4,7 - 47 μF
CAP_FACTOR_LARGE für Kondensatoren > 47 μF
```

```
#define CAP_FACTOR_SMALL 0 keine Korrektur - ggf. Wert anpassen
#define CAP_FACTOR_MID ...-40 -4.0% - ggf. Wert anpassen
#define CAP_FACTOR_LARGE ...-90 -9.0% - ggf. Wert anpassen
```

**Anzahl der ADC-Runden für jede Messung**

- Gültige Werte liegen im Bereich von 1 bis 255.

```
#define ADC_SAMPLES ...25
```

- ggf. Wert anpassen

## 5.2.6. Busse

**I2C-Bus** wird möglicherweise von bestimmter Hardware benötigt

- könnte bereits über die Anzeigeneinstellung aktiviert sein (config\_<MCU>.h)
- für Bit-Bang-Port und -Pins siehe I2C \_PORT (config\_<MCU>.h)
- Hardware I2C (TWI) verwendet automatisch die richtigen MCU-Pins
- Zum Aktivieren entweder I2C \_BITBANG oder I2C \_HARDWARE auskommentieren
- einen der Bustakte auskommentieren

```
// #define I2C_BITBANG bit-bang I2C
// #define I2C_HARDWARE MCUs Hardware-TWI
// #define I2C_STANDARD_MODE 100kHz Bustakt
// #define I2C_FAST_MODE 400kHz Bustakt
// #define I2C_RW Leseunterstützung aktivieren (ungetestet)
```

**SPI-Bus** wird möglicherweise von bestimmter Hardware benötigt

- könnte bereits über die Anzeigeneinstellung aktiviert sein (config\_<MCU>.h)
- für Bit-Bang-Port und -Pins siehe SPI \_PORT (config\_<MCU>.h)
- Hardware-SPI verwendet automatisch die richtigen MCU-Pins
- Zum Aktivieren entweder SPI \_BITBANG oder SPI \_HARDWARE auskommentieren

```
// #define SPI_BITBANG bit-bang SPI
// #define SPI_HARDWARE Hardware-SPI
// #define SPI_RW Leseunterstützung aktivieren
```

**Serielle TTL-Schnittstelle** könnte bereits über die Anzeigeeinstellung aktiviert sein (config\_<MCU>.h)

- für Bit-Bang-Port und -Pins siehe SERIAL \_PORT (config\_<MCU>.h).
- Hardware-Serielle verwendet automatisch die richtigen MCU-Pins
- Zum Aktivieren entweder SERIAL \_BITBANG oder SERIAL \_HARDWARE auskommentieren

```
// #define SERIAL_BITBANG Bit-bang-Serielle
// #define SERIAL_HARDWARE Hardware-Serielle
// #define SERIAL_RW Leseunterstützung aktivieren
```

**OneWire-Bus**

- Informationen zum dedizierten I/O-Pin findest Du unter ONEWIRE \_PORT (config\_<MCU>.h).
- Zum Aktivieren entweder ONEWIRE \_PROBES oder ONEWIRE \_IO \_PIN auskommentieren

```
// #define ONEWIRE_PROBES via Testpins
// #define ONEWIRE_IO_PIN via dediziertem I/O-Pin
```

### 5.3. Config\_<MCU>.h

Die Config\_<MCU>.h enthält hardwarenahe Einstellungen für Anzeigen, Tasten und so weiter. Da Pin-Zuordnungen natürlich vom MCU-Typen abhängig sind, gibt es für den ATmega328 und die Familie um den ATmega644 jeweils eine eigene Datei mit den jeweiligen Standardzuordnungen. Beim Übersetzen der Firmware wird die passende Datei entsprechend der MCU automatisch eingebunden. Es handelt sich auch wieder um eine C-Header-Datei, d.h. es gelten die Kommentar-regeln für C. Neben dem “//” für einzelne Zeilen werden auch Blockkommentare mittels “#if 0 ... #endif” verwendet. Um einen Block ein kommentieren, einfach ein “//” vor dem entsprechenden “#if 0” und “#endif” einfügen; zum auskommentieren der umgekehrte Weg. Man kann einen Block auch ein kommentieren, indem man die Zeilen mit dem “#if 0” und “#endif” löscht.

**5.3.1. LCD-Module** Als Beispiel wird ein HD44780-Modul mit Parallelbus als Anzeige ausgewählt in dem #if 0 und #endif mit “//” auskommentiert wurden.

#### HD44780, 4-Bit-Parallelschnittstelle

- Aktiviere LCD\_DB\_STD, um die Port-Pins 0-3 für LCD\_DB4/5/6/7 zu verwenden.
- Bei Bedarf kann man auch die Schriftart ändern

```
//#if 0
#define LCD_HD44780 /* display controller HD44780 */
#define LCD_TEXT /* character display */
#define LCD_PAR_4 /* 4 bit parallel interface */
#define LCD_PORT PORTB /* port data register */
#define LCD_DDR DDRB /* port data direction register */
//#define LCD_DB_STD /* use standard pins 0-3 for DB4-7 */
#define LCD_DB4 PB4 /* port pin used for DB4 */
#define LCD_DB5 PB5 /* port pin used for DB5 */
#define LCD_DB6 PB6 /* port pin used for DB6 */
#define LCD_DB7 PB7 /* port pin used for DB7 */
#define LCD_RS PB2 /* port pin used for RS */
#define LCD_EN1 PB3 /* port pin used for E */
#define LCD_CHAR_X 16 /* characters per line */
#define LCD_CHAR_Y 2 /* number of lines */
/* HD44780 has an internal 5x7 font */
#define FONT_HD44780_INTERNATIONAL /* 5x7 font: international version */
//#define FONT_HD44780_CYRILLIC /* 5x7 font: cyrillic version */
//#endif
```

Listing 5.8. HD44780 4-Bit Parallel ausgewählt

#### HD44780, PCF8574 (Hardware-I2C)

- Wenn Du LCD\_DB4/5/6/7 änderst, kommentiere LCD\_DB\_STD aus!
- Hardware-I2C wählt automatisch SDA- und SCL-Pins aus
- PCF8574T ist 0x27, PCF8574AT ist 0x3f

```
#define LCD_DB_STD /* use standard pins 4-7 for DB4-7 */
```

Listing 5.9. LCD DB STD aktiv

Die Displays und ihre Einstellungen werden im Kapitel 2.7. auf Seite 11 behandelt.

### 5.3.2. Port- und Pinbelegung

#### Testpins/Prüfspitzen

- Die ersten 3 Pins des Analogports müssen für die Testpins verwendet werden.
- Bitte die Definitionen von TP1, TP2 und TP3 nicht ändern!
- Diesen Port nicht mit POWER\_CTRL oder TEST\_BUTTON teilen!

```
#define ADC_PORT PORTA /* ADC port data register */
#define ADC_DDR DDRA /* ADC port data direction register */
#define ADC_PIN PINA /* port input pins register */
#define TP1 PA0 /* test pin 1 */
#define TP2 PA1 /* test pin 2 */
#define TP3 PA2 /* test pin 3 */
```

Listing 5.10. Pins nicht ändern

## Testwiderstände

- Für die PWM- / Rechteckwellenausgabe über Testpin 2 muss R\_RL\_2 PD4 / OC1B sein.
- Diesen Port nicht mit POWER\_CTRL oder TEST\_BUTTON teilen!

```
#define TOUCH_PEN PB1 /* port pin used for /PENIRQ */
// #define TOUCH_FLIP_X /* enable horizontal flip */
// #define TOUCH_FLIP_Y /* enable vertical flip */
// #define TOUCH_ROTATE /* switch X and Y (rotate by 90Grad) */
#define SPI_HARDWARE /* hardware SPI */
#define SPI_RW /* enable SPI read support */
#endif
```

Listing 5.11. ggf. anpassen

## dedizierte Signalausgabe über OC1B - bitte nicht ändern!

```
#define SIGNAL_PORT PORTD /* port data register */
#define SIGNAL_DDR DDRD /* port data direction register */
#define SIGNAL_OUT PD4 /* MCU's OC1B pin */
```

Listing 5.12. nicht ändern!

## Stromschalter - Kann nicht derselbe Port sein wie ADC\_PORT oder R\_PORT.

```
#define POWER_PORT PORTC /* port data register */
#define POWER_DDR DDRC /* port data direction register */
#define POWER_CTRL PC6 /* controls power (1: on / 0: off) */
```

Listing 5.13. ggf. anpassen

## Testtaster - Kann nicht derselbe Port sein wie ADC\_PORT oder R\_PORT.

```
#define BUTTON_PORT PORTC /* port data register */
#define BUTTON_DDR DDRC /* port data direction register */
#define BUTTON_PIN PINC /* port input pins register */
#define TEST_BUTTON PC7 /* test/start push button (low active) */
```

Listing 5.14. ggf. anpassen

## Drehenkoder

```
#define ENCODER_PORT PORTC /* port data register */
#define ENCODER_DDR DDRC /* port data direction register */
#define ENCODER_PIN PINC /* port input pins register */
#define ENCODER_A PC4 /* rotary encoder A signal */
#define ENCODER_B PC3 /* rotary encoder B signal */
```

Listing 5.15. ggf. anpassen

## Tasten Mehr/Weniger

```
#define KEY_PORT PORTC /* port data register */
#define KEY_DDR DDRC /* port data direction register */
#define KEY_PIN PINC /* port input pins register */
#define KEY_INC PC4 /* increase push button (low active) */
#define KEY_DEC PC3 /* decrease push button (low active) */
```

Listing 5.16. ggf. anpassen

## Frequenzzähler

- Einfache und erweiterte Version
- Eingang muss Pin PB0 / T0 sein

```
#define COUNTER_PORT PORTB /* port data register */
#define COUNTER_DDR DDRB /* port data direction register */
#define COUNTER_IN PB0 /* signal input T0 */
```

Listing 5.17. nicht ändern!

## Steuerung für erweiterten Frequenzzähler

```
#define COUNTER_CTRL_PORT PORTC /* port data register */
#define COUNTER_CTRL_DDR DDRC /* port data direction register */
#define COUNTER_CTRL_DIV PC0 /* prescaler */
#define COUNTER_CTRL_CH0 PC1 /* channel addr #0 */
#define COUNTER_CTRL_CH1 PC2 /* channel addr #1 */
```

Listing 5.18. ggf. anpassen

## IR-Detektor/Decoder

- festes Modul, welches an einen dedizierten E/A-Pin angeschlossen ist

```
#define IR_PORT PORTC /* port data register */
#define IR_DDR DDRC /* port data direction register */
#define IR_PIN PINC /* port input pins register */
#define IR_DATA PC2 /* data signal */
```

Listing 5.19. ggf. anpassen

### 5.3.3. Busse

- SPI** - Hardware-SPI verwendet PB7, PB5 und PB6  
- Bit-Bang-SPI könnte bereits im Anzeigebereich eingestellt sein

```
#ifndef SPI_PORT
#define SPI_PORT PORTB /* port data register */
#define SPI_DDR DDRB /* port data direction register */
#define SPI_PIN PINB /* port input pins register */
#define SPI_SCK PB7 /* pin for SCK */
#define SPI_MOSI PB5 /* pin for MOSI */
#define SPI_MISO PB6 /* pin for MISO */
#endif
```

Listing 5.20. ggf. anpassen

- I2C** - Hardware-I2C (TWI) verwendet PC1 und PC0  
- Bit-Bang-I2C könnte bereits im Anzeigebereich eingestellt sein

```
#ifndef I2C_PORT
#define I2C_PORT PORTC /* port data register */
#define I2C_DDR DDRC /* port data direction register */
#define I2C_PIN PINC /* port input pins register */
#define I2C_SDA PC1 /* pin for SDA */
#define I2C_SCL PC0 /* pin for SCL */
#endif
```

Listing 5.21. ggf. anpassen

#### serielle TTL-Schnittstelle

- Hardware-USART0 verwendet PD0 & PD1, USART1 verwendet PD2 & PD3

```
/* for hardware TTL serial */
#define SERIAL_USART 0 /* use USART0 */
/* for bit-bang TTL serial */
#define SERIAL_PORT PORTD /* port data register */
#define SERIAL_DDR DDRD /* port data direction register */
#define SERIAL_PIN PIND /* port input pins register */
#define SERIAL_TX PD1 /* pin for Tx (transmit) */
#define SERIAL_RX PD0 /* pin for Rx (receive) */
#define SERIAL_PCINT 24 /* PCINT# for Rx pin */
```

Listing 5.22. ggf. anpassen

#### OneWire - dedizierter I/O-Pin

```
#define ONEWIRE_PORT PORTC /* port data register */
#define ONEWIRE_DDR DDRC /* port data direction register */
#define ONEWIRE_PIN PINC /* port input pins register */
#define ONEWIRE_DQ PC2 /* DQ (data line) */
```

Listing 5.23. ggf. anpassen

#### Eingebauter Kondensator für Selbstabgleich - Der ADC-Pin ist TP\_CAP von weiter oben

- Einstellungen gelten für den 470k Widerstand  
- sollte ein Folienkondensator zwischen 100nF und 1000nF sein

```
#define ADJUST_PORT PORTC /* port data register */
#define ADJUST_DDR DDRC /* port data direction register */
#define ADJUST_RH PC5 /* Rh (470k) for fixed cap */
```

Listing 5.24. ggf. anpassen

#### Relais für Parallelkondensator (Sampling-ADC)

- TP1 und TP3  
- Kondensator sollte zwischen 10nF und 27nF haben

```
#define CAP_PORT PORTC /* port data register */
#define CAP_DDR DDRC /* port data direction register */
#define CAP_RELAY PC2 /* control pin */
```

Listing 5.25. ggf. anpassen

---

## Kapitel 6      Sammlung von Einstellungen

---

Hier findest Du Einstellungen für verschiedene Tester-Modelle. Falls Du einen nicht aufgeführten Tester zum Laufen gebracht hast, schicke bitte eine kurze Beschreibung vom Tester und die dazugehörigen Einstellungen per EMail an den Autor [8], um anderen Benutzern zu helfen.

### 6.0.1. DIY Kit „AY-AT“

- ATmega328
- Farb-LCD-Modul ST7735 (Bit-Bang-SPI)
- Dreheencoder (PD1 & PD3, parallel zur Anzeige)
- Externe 2,5-V-Spannungsreferenz (TL431)
- Einfacher Frequenzzähler mit dediziertem Eingang (PD4)
- Messung einer externen Spannung bis 45V (PC3)
- Einstellungen stammen von flywheelz@EEVBlog

```
#define HW_ENCODER
#define ENCODER_PULSES 4 /* usually 4 pulses per step */
#define ENCODER_STEPS 20 /* usually 20 detents */
#define HW_REF25
#define HW_ZENER
#define HW_FREQ_COUNTER_BASIC
```

Listing 6.1. HW optionen

```
#define LCD_ST7735
#define LCD_GRAPHIC /* graphic display */
#define LCD_COLOR /* color display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTD /* port data register */
#define LCD_DDR DDRD /* port data direction register */
#define LCD_RES PD0 /* port pin used for /RESX */
#define LCD_CS PD5 /* port pin used for /CSX (optional) */
#define LCD_DC PD1 /* port pin used for D/CX */
#define LCD_SCL PD2 /* port pin used for SCL */
#define LCD_SDA PD3 /* port pin used for SDA */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 160 /* number of vertical dots */
#define LCD_FLIP_X /* enable horizontal flip */
//#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90 Grad) */
//#define LCD_OFFSET_X 4 /* enable x offset of 2 or 4 dots */
//#define LCD_OFFSET_Y 2 /* enable y offset of 1 or 2 dots */
//#define LCD_LATE_ON /* turn on LCD after clearing it */
#define FONT_10X16_HF /* 10x16 font */
#define SYMBOLS_24X24_HF /* 24x24 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SDA /* port pin used for MOSI */
```

Listing 6.2. LCD Modul

Wenn der Tester mit einer leeren Anzeige starten soll, entferne das Kommentarzeichen vor LCD\_LATE\_ON.



```

#define ENCODER_PORT PORTD /* port data register */
#define ENCODER_DDR DDRD /* port data direction register */
#define ENCODER_PIN PIND /* port input pins register */
#define ENCODER_A PD1 /* rotary encoder A signal */
#define ENCODER_B PD3 /* rotary encoder B signal */

```

Listing 6.3. Drehkoder

Der Eingang für den Frequenzzähler ist PD4 (T0)

Induktivitätskompensations-Offsets für 20-MHz-Modell bereitgestellt von indman@EEVBlog

- Abschnitt für den Hochstrommodus in der Funktion MeasureInductor() in inductor.c

```

#if CPU_FREQ == 20000000
/* 20 MHz */
if (Temp < 1500) /* < 1.5us / < 100uH */
{
 Offset = -10;
}
else if (Temp < 5000) /* 1.5-5us / 100-330uH */
{
 Offset = -10;
}
else /* > 5us / > 330uH */
{
 Offset = -30;
}
#endif

```

Listing 6.4. Induktivitätskompensations-Offsets

### 6.0.2. M12864 DIY Transistor Tester - ATmega328

- ST7565-Anzeige (Bit-Bang-SPI)

- Drehencoder (PD1 & PD3, parallel zur Anzeige)

- Externe 2,5-V-Spannungsreferenz (TL431)

```

#define HW_ENCODER
#define ENCODER_PULSES 4 /* not confirmed yet, could be also 2 */
#define ENCODER_STEPS 24 /* not confirmed yet */
#define HW_REF25

```

Listing 6.5. Hardware Optionen

```

#define LCD_ST7565R
#define LCD_GRAPHIC /* graphic display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTD /* port data register */
#define LCD_DDR DDRD /* port data direction register */
#define LCD_RESET PD0 /* port pin used for /RES */
#define LCD_A0 PD1 /* port pin used for A0 */
#define LCD_SCL PD2 /* port pin used for SCL */
#define LCD_SI PD3 /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 64 /* number of vertical dots */
// #define LCD_OFFSET_X /* enable x offset of 4 dots */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 11 /* default contrast (0-63) */
#define FONT_8X8_VF /* 8x8 font */
#define SYMBOLS_24X24_VFP /* 24x24 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SI /* port pin used for MOSI */

```

Listing 6.6. LCD-Modul

```

#define ENCODER_PORT PORTD /* port data register */
#define ENCODER_DDR DDRD /* port data direction register */
#define ENCODER_PIN PIND /* port input pins register */
#define ENCODER_A PD1 /* rotary encoder A signal */
#define ENCODER_B PD3 /* rotary encoder B signal */

```

Listing 6.7. Drehkoder

6.0.3. T3/T4 - ATmega328, 8 MHz Takt  
- ST7565-Anzeige (Bit-Bang-SPI)  
- Einstellungen bereitgestellt von tom666@EEVblog

```

#define LCD_ST7565R
#define LCD_GRAPHIC /* graphic display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTD /* port data register */
#define LCD_DDR DDRD /* port data direction register */
#define LCD_RESET PD4 /* port pin used for /RES */
#define LCD_A0 PD3 /* port pin used for A0 */
#define LCD_SCL PD2 /* port pin used for SCL */
#define LCD_SI PD1 /* port pin used for SI (LCD's data input) */
#define LCD_CS PD5 /* port pin used for /CS1 (optional) */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 64 /* number of vertical dots */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 11 /* default contrast (0-63) */
#define FONT_8X8_VF /* 8x8 font */
#define SYMBOLS_24X24_VFP /* 24x24 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SI /* port pin used for MOSI */

```

Listing 6.8. LCD Modul

6.0.4. GM328 - ATmega328, 8 MHz Takt  
- ST7565-Anzeige (Bit-Bang-SPI)  
- Einstellungen bereitgestellt von rddube@EEVblog

```

#define LCD_ST7565R
#define LCD_GRAPHIC /* graphic display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTD /* port data register */
#define LCD_DDR DDRD /* port data direction register */
#define LCD_RESET PD0 /* port pin used for /RES (optional) */
#define LCD_A0 PD1 /* port pin used for A0 */
#define LCD_SCL PD2 /* port pin used for SCL */
#define LCD_SI PD3 /* port pin used for SI (LCD's data input) */
#define LCD_CS PD5 /* port pin used for /CS1 (optional) */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 64 /* number of vertical dots */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 11 /* default contrast (0-63) */
#define FONT_8X8_VF /* 8x8 font */
#define SYMBOLS_24X24_VFP /* 24x24 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SI /* port pin used for MOSI */

```

Listing 6.9. LCD Modul

#### 6.0.5. Fish8840 TFT - ATmega328, 8 MHz Takt

- ST7565 Farbdisplay (Bit-Bang-SPI)
- Externe 2,5-V-Spannungsreferenz (TL431)
- Einstellungen von indman@EEVBlog / bdk100@vrtp.ru

```
LCD module:
#define LCD_ST7735
#define LCD_GRAPHIC /* graphic display */
#define LCD_COLOR /* color display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTD /* port data register */
#define LCD_DDR DDRD /* port data direction register */
#define LCD_RES PD3 /* port pin used for /RESX (optional) */
// #define LCD_CS PD5 /* port pin used for /CSX (optional) */
#define LCD_DC PD2 /* port pin used for D/CX */
#define LCD_SCL PD0 /* port pin used for SCL */
#define LCD_SDA PD1 /* port pin used for SDA */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 156 /* number of vertical dots */
#define LCD_FLIP_X /* enable horizontal flip */
// #define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90 Grad) */
#define LCD_OFFSET_X /* enable x offset of 4 dots */
#define LCD_OFFSET_Y /* enable y offset of 2 dots */
#define LCD_LATE_ON /* turn on LCD after clearing it */
#define FONT_10X16_HF /* 10x16 font */
#define SYMBOLS_30X32_HF /* 30x32 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SDA /* port pin used for MOSI */
```

Listing 6.10. LCD Modul

#### 6.0.6. Multifunktionstester TC-1 - ATmega324 (sehr schlechte Pinbelegung),

- 16-MHz-Takt
- ST7735 Farbdisplay (Bit-Bang-SPI)
- Externe 2,5-V-Spannungsreferenz (TL431)
- festes IR-Empfangsmodul
- Boostkonverter für Zenertest
- Fester Kondensator für Selbstabgleich
- Stromversorgung durch Li-Ion-Akku 3.7V
- Testexemplare von jellytot@EEVblog und joystik@EEVblog
- erste Informationen von indman@EEVblog

Hinweise: - Die Steuer-MCU U4 muss durch eine einfache Schaltung ersetzt werden (TC1-Mod, siehe Quell-Repository für Hardware / Markus / TC1-Mod.kicad.tgz, 5µA Standby-Strom insgesamt) oder mit einer modifizierten Firmware neu programmiert werden

(siehe <https://github.com/atar-axis/tc1-u4>)

- Extended Fuse Byte auf 0xfd setzen (Brown-Out Detection)
- Wenn D2 (Gleichrichterdiode für Zenertestspannung) heiß wird, diese ersetzen durch eine Schottky-Diode, die für eine Sperrspannung von 80 V oder höher ausgelegt ist, z.B. SS18
- C11 und C12 (Filterkondensatoren für Zenertestspannung) ersetzen durch einen 10 oder 22µF Elko mit niedrigem ESR und 100V Spannungsfestigkeit. MLCCs haben das bekannte Problem, daß die Kapazität in Abhängigkeit einer Bias-Gleichspannung sinkt.
- Abhängig vom verwendeten LCD-Modul musst Du möglicherweise LCD\_FLIP\_X anstelle von LCD\_FLIP\_Y nehmen.

```

#define HW_REF25
#define HW_ZENER
#define HW_IR_RECEIVER
#define HW_ADJUST_CAP

```

Listing 6.11. Hardware Optionen

```

#define BAT_DIRECT
#define BAT_OFFSET 0
#define BAT_WEAK 3600
#define BAT_LOW 3400

```

Listing 6.12. Verschiedene Einstellungen

```

#define LCD_ST7735
#define LCD_COLOR /* color graphic display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTB /* port data register */
#define LCD_DDR DDRB /* port data direction register */
#define LCD_RES PB4 /* port pin used for /RESX (optional) */
// #define LCD_CS PB? /* port pin used for /CSX (optional) */
#define LCD_DC PB5 /* port pin used for D/CX */
#define LCD_SCL PB7 /* port pin used for SCL */
#define LCD_SDA PB6 /* port pin used for SDA */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 160 /* number of vertical dots */
// #define LCD_FLIP_X /* enable horizontal flip */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90 Grad) */
#define LCD_LATE_ON /* turn on LCD after clearing it */
#define LCD_OFFSET_X 2 /* enable x offset of 2 or 4 dots */
#define LCD_OFFSET_Y 1 /* enable y offset of 1 or 2 dots */
#define FONT_10X16_HF /* 10x16 font */
| #define SYMBOLS_30X32_HF /* 30x32 symbols */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SDA /* port pin used for MOSI */

```

Listing 6.13. LCD Modul

```

#define TP_ZENER PA4 /* test pin with 10:1 voltage divider */
#define TP_REF PA3 /* test pin with 2.5V reference */
#define TP_BAT PA5 /* test pin with 4:1 voltage divider */
#define TP_CAP PA7 /* test pin for self-adjustment cap */

```

Listing 6.14. Testpin Belegung

```

#define R_PORT PORTC /* port data register */
#define R_DDR DDRC /* port data direction register */
#define R_RL_1 PC0 /* Rl (680R) for test pin #1 */
#define R_RH_1 PC1 /* Rh (470k) for test pin #1 */
#define R_RL_2 PC2 /* Rl (680R) for test pin #2 */
#define R_RH_2 PC3 /* Rh (470k) for test pin #2 */
#define R_RL_3 PC4 /* Rl (680R) for test pin #3 */
#define R_RH_3 PC5 /* Rh (470k) for test pin #3 */

```

Listing 6.15. Pinbelegung für Sondenwiderstände

```

#define POWER_PORT PORTD /* port data register */
#define POWER_DDR DDRD /* port data direction register */
#define POWER_CTRL PD2 /* controls power (1: on / 0: off) */

```

Listing 6.16. Pinbelegung zur Leistungsregelung

```

#define BUTTON_PORT PORTD /* port data register */
#define BUTTON_DDR DDRD /* port data direction register */
#define BUTTON_PIN PIND /* port input pins register */
#define TEST_BUTTON PD1 /* test/start push button (low active) */

```

Listing 6.17. Pinbelegung für Testknopf

```

#define IR_PORT PORTD /* port data register */
#define IR_DDR DDRD /* port data direction register */
#define IR_PIN PIND /* port input pins register */
#define IR_DATA PD3 /* data signal */

```

Listing 6.18. Pinbelegung für festen IR-Detektor / Decoder

```

#define ADJUST_PORT PORTC /* port data register */
#define ADJUST_DDR DDRC /* port data direction register */
#define ADJUST_RH PC6 /* Rh (470k) for fixed cap */

```

Listing 6.19. Pinbelegung für Festkondenzator zur Selbsteinstellung

#### 6.0.7. Hiland M644 - ATmega 644, 8 MHz Takt

- ST7565-Anzeige (Bit-Bang-SPI)
- Drehencoder (PB7 & PB5, parallel zum Display)
- Externe 2,5-V-Spannungsreferenz (TL431)
- Boostkonverter für Zenertest
- Erweiterter Frequenzzähler
- Fester Kondensator für Selbstabgleich
- Einstellungen von Horst O. (obelix2007@mikrocontroller.net)

```

#define HW_ENCODER
#define ENCODER_PULSES 4 /* 4 */
#define HW_REF25
#define HW_ZENER
#define HW_FREQ_COUNTER_EXT
#define FREQ_COUNTER_PRESCALER 16 /* 16:1 */
#define HW_ADJUST_CAP

Workarounds:
#define NO_HFE_C_RL /* if hFE values are too high */

```

Listing 6.20. Hardware Optionen

```

#define LCD_ST7565R /* display controller ST7565R */
#define LCD_GRAPHIC /* graphic display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTB /* port data register */
#define LCD_DDR DDRB /* port data direction register */
#define LCD_RESET PB4 /* port pin used for /RES (optional) */
// #define LCD_CS PB2 /* port pin used for /CS1 (optional) */
#define LCD_A0 PB5 /* port pin used for A0 */
#define LCD_SCL PB6 /* port pin used for SCL */
#define LCD_SI PB7 /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128 /* number of horizontal dots */
#define LCD_DOTS_Y 64 /* number of vertical dots */
// #define LCD_FLIP_X /* enable horizontal flip */
// #define LCD_OFFSET_X /* enable x offset of 4 dots */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 3 /* default contrast (0-63) */
/* font and symbols: vertically aligned & flipped, bank-wise grouping */
#define FONT_6X8_VF /* 6x8 font */
// #define FONT_8X8_VF /* 8x8 font */
// #define FONT_8X16_VFP /* 8x16 font */
// #define FONT_8X8_CYRILLIC_VF /* 8x8 cyrillic font */

```



```

// #define FONT_8X8T_CYRILLIC_VF /* thin 8x8 cyrillic font */
// #define FONT_8X12T_CYRILLIC_VFP /* thin 9x12 cyrillic font */
// #define FONT_8X16_CYRILLIC_VFP /* 8x16 cyrillic font */
// #define FONT_6X8_CZECH_VF /* 6x8 Czech font */
// #define FONT_8X8_CZECH_VF /* 8x8 Czech font */
// #define FONT_8X16_CZECH_VFP /* 8x16 Czech font */
#define SYMBOLS_24X24_VFP /* 24x24 symbols */
// #define SPI_HARDWARE /* hardware SPI */
#define SPI_BITBANG /* bit-bang SPI */
#define SPI_PORT LCD_PORT /* SPI port data register */
#define SPI_DDR LCD_DDR /* SPI port data direction register */
#define SPI_SCK LCD_SCL /* port pin used for SCK */
#define SPI_MOSI LCD_SI /* port pin used for MOSI */

```

Listing 6.21. LCD Modul

```

#define TP_ZENER PA3 /* test pin with 10:1 voltage divider */
#define TP_REF PA4 /* test pin with 2.5V reference */
#define TP_BAT PA5 /* test pin with 4:1 voltage divider */
#define TP_CAP PA7 /* test pin for self-adjustment cap */

```

Listing 6.22. Testpin Belegung

```

#define R_PORT PORTD /* port data register */
#define R_DDR DDRD /* port data direction register */
#define R_RL_1 PD2 /* Rl (680R) for test pin #1 */
#define R_RH_1 PD3 /* Rh (470k) for test pin #1 */
#define R_RL_2 PD4 /* Rl (680R) for test pin #2 */
#define R_RH_2 PD5 /* Rh (470k) for test pin #2 */
#define R_RL_3 PD6 /* Rl (680R) for test pin #3 */
#define R_RH_3 PD7 /* Rh (470k) for test pin #3 */

```

Listing 6.23. Pinbelegung für Sondenwiderstände (Standard)

```

#define POWER_PORT PORTB /* port data register */
#define POWER_DDR DDRB /* port data direction register */
#define POWER_CTRL PB1 /* controls power (1: on / 0: off) */

```

Listing 6.24. Pinbelegung zur Leistungsregelung

```

#define BUTTON_PORT PORTC /* port data register */
#define BUTTON_DDR DDRC /* port data direction register */
#define BUTTON_PIN PINC /* port input pins register */
#define TEST_BUTTON PC7 /* test/start push button (low active) */

```

Listing 6.25. Pinbelegung für die Testschaltfläche (Standard)

```

#define COUNTER_PORT PORTB /* port data register */
#define COUNTER_DDR DDRB /* port data direction register */
#define COUNTER_IN PB0 /* signal input T0 */
#define COUNTER_CTRL_PORT PORTC /* port data register */
#define COUNTER_CTRL_DDR DDRC /* port data direction register */
#define COUNTER_CTRL_DIV PC0 /* prescaler */
#define COUNTER_CTRL_CH0 PC1 /* channel addr #0 */
#define COUNTER_CTRL_CH1 PC2 /* channel addr #1 */

```

Listing 6.26. Pinbelegung für erweiterten Frequenzzähler (Standard)

```

#define ADJUST_PORT PORTC /* port data register */
#define ADJUST_DDR DDRC /* port data direction register */
#define ADJUST_RH PC6 /* Rh (470k) for fixed cap */

```

Listing 6.27. Pinbelegung für Festkondenzator zur Selbsteinstellung

---

## ***Kapitel 7 Programmieren des Component Testers***

---

Um die Verzweigungen und „schlaflose Nächte“, die Schreiber dieses Kapitels erlitten hatte, nach dem er, ohne jegliche AVR Erfahrung, einen Clonetester erworben hatte und diesem die deutsche Sprache „beibringen“ wollte, anderen Kollegen zu ersparen, wurde dieses Kapitel geschrieben.

Die hier bei erworbene Erfahrungen sollten anderen “willigen“ unerfahrenen helfen, ERFOLGREICH ihr Tester zu programmieren.

Diese Gelegenheit wird benutzt, dem Autor und Entwickler des Transistortester

Karl-Heinz Kübbeler siehe [3] zu danken für sein Engagement und Geduld, denn die folgenden Seiten hätten ohne seiner Hilfe nie entstanden.

Damit das übersetzen der Firmware und Brennen ins MCU gelingt und gleichzeitig ... „das Rad nicht neu erfunden sein müsste“, wurde ein Teil der folgenden Seiten aus dem Beschreibung des Transistor Tester von Karl-Heinz Kübbeler siehe [3] übernommen.

Also noch einmal ... **EINEN RIESEN DANK.**

### **7.1. Konfigurieren des Component Testers**

Dazu bitte Kapitel 1.6 auf Seite 8 lesen.

### **7.2. Programmierung des Mikrocontrollers**

Die Programmierung des Testers wird über die Datei Makefile gesteuert.

Die Makefile stellt sicher, dass die Software entsprechend der vorher in der Makefile eingestellten Optionen übersetzt wird.

Das Ergebnis der Übersetzung hat die Dateierweiterung .hex und .eep.

Üblicherweise heißen die Dateien ComponentTester.hex und ComponentTester.eep.

Die .hex-Datei enthält die Daten für den Programmspeicher (Flash) des ATmega-Prozessors.

Die .eep-Datei enthält die Daten für den EEPROM des ATmega.

Beide Dateien müssen in den richtigen Speicher geladen werden.

Zusätzlich muss der ATmega mit den „fuses“ richtig konfiguriert werden.

Wenn Sie das Makefile zusammen mit dem Programm avrdude [9] benutzen, brauchen Sie keine genaue Kenntnis über die Einzelheiten der fuses.

Sie brauchen nur „make fuses“ aufrufen, wenn Sie keinen Quarz benutzen oder Sie müssen „make fuses-crystal“ aufrufen, wenn Sie einen 8MHz Quarz auf der Baugruppe installiert haben.

Wenn Sie sich nicht sicher mit den fuses sind, lassen Sie diese erst einmal wie vom Werk gesetzt und bringen Sie den Tester in diesem Zustand zum Laufen.

Es kann sein, dass das Programm zu langsam läuft, wenn Sie die für den 8MHz-Betrieb erzeugten Programmdateien benutzen, aber das kann man später korrigieren!

Aber falsch gesetzte fuses können die spätere ISP-Programmierung verhindern.

**7.2.1. Betrieb System Linux** Die Programmierung unten Linux bringt viele Vorteile, weil dieses OS von Experten entwickelt wurde, die sich auf den Wünschen der Benutzer orientieren. Zu dem ist die Umgebung kostenlos erhältlich und perfekt gewartet.

Ein weiterer Vorteil ist die Sicherheit von OS selbst aber auch beim nutzen des Internets.

Die heutigen Editionen lassen sich noch viel leichter bedienen als die Mitbewerber OS.

Diese Anleitung soll alle “nicht“ Linux Benutzer dazu animieren es NUN zu testen in dem sie ihr Tester damit programmieren.

Als Beispiel wird hier Linux Mint in der aktueller Version benutzt, die im Internet zu erhalten ist. Die Installation ist auf verschiedene Arten möglich.

### 7.2.2. Benutzung unter Linux als neu aufgesetzten OS.

Für diejenigen, die nicht gerne schreiben, bietet sich eine einfache Möglichkeit.

Dieses Handbuch auf einen USB-Stick kopieren und in Linux öffnen.

Danach die Maus zum Namen des Dokumentes, also zum Text (ctester.pdf) führen, hier linke Maustaste drücken und das Dokument bis zum linken Rand des Bildschirms ziehen, bis ein mögliches Rahmen angezeigt wird. Nun wird die Maus losgelassen.

Die Anleitung nimmt nun die linke Hälfte des Bildschirms ein.

### 7.2.3. Programm Pakete installieren Um den Tester zu programmieren, müssen zuerst Programmpakete installiert werden:

'binutils-avr', 'avrdude', 'avr-libc' und 'gcc-avr'.

Jetzt zu dieser Seite navigieren, bis zu diesem Text:

```
sudo apt-get install avrdude avr-libc binutils-avr gcc-avr
```

Im nächsten Schritt wird [Strg] + [Alt] + [t] gleichzeitig gedrückt, um das Befehlsfenster zu öffnen. Dies wird nun auf gleicher Weise zum rechten Rand des Bildschirmes bewegt.

Den oben genannten Text im linkem Fenster, bei gedrückter linken Maustaste markieren, die Maus auf das Cursor des rechten Befehlsfenster führen und durch drücken

der mittleren Maustaste (Scroll-Rad) **weiter als [MT] abgekürzt** wieder einfügen.

Nach der Bestätigung mit [Enter], wird von 'sudo' noch Benutzerpasswort verlangt.

Anders als bei Windows wird das Passwort **Blind** eingegeben und mit [Enter] bestätigt.

Damit werden nun alle Software Pakete durch 'apt' installiert.

U.U. muss man dazwischen, Fragen durch [J] bestätigen.

Bitte darauf achten, dass im Linux zwischen Groß und Kleinschreibung unterschieden wird.

Also nicht mit [j] sondern mit [J] antworten!

### 7.2.4. Download der Quellen Zum Download der Quellen und der Dokumentation aus dem SVN-Archiv wird das Paket

'subversion' gebraucht. Dies wird erreicht mit einer Anweisung:

```
sudo apt install subversion
```

und nach dem das Paket installiert wurde mit:

```
svn checkout svn://www.mikrocontroller.net/transistortester
```

wird das komplette Archiv heruntergeladen. Wenn Sie dieses Archiv bereits heruntergeladen wurde, werden mit diesem Befehl nur neue Updates heruntergeladen.

Die Dateien sind nun in dem Linux [Persönlicher Ordner] auf (/home/„user“) unten dem Namen „transistortester“.

Die Kontrolle des Vorhandenseins. Terminal Fenster öffnen, „ls“ eingeben und bestätigen.

### 7.2.5. Benutzung der Schnittstellen für den Nutzer (user) vorbereiten.

USB-Geräte können durch Eingabe von 'lsusb' im Befehlsfenster erkannt werden. Geben Sie 'lsusb' zuerst ohne und dann mit angeschlossenem USB-Programmierer ein.

Ein Vergleich der Ergebnisse lokalisiert den USB-Programmierer.

Das Ergebnis von lsusb kann so aussehen:

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 046d:c050 Logitech, Inc. RX 250 Optical Mouse
Bus 002 Device 058: ID 03eb:2104 Atmel Corp. AVR ISP mkII
Bus 002 Device 059: ID 2341:0042 Arduino SA Mega 2560 R3 (CDC ACM)
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub}
```

Hier wurde als Device 58 ein AVR ISP mkII erkannt (DIAMEX ALL-AVR).

Die ID 03eb ist eine Herstellerkennung und die ID 2104 eine Produktkennung.

Diese beiden Kennungen werden der Datei /etc/udev/rules.d/90-atmel.rules benötigt und erstellt

mit Hilfe von:

```
sudo xed /etc/udev/rules.d/90-atmel.rules
```

In diesem Beispiel besteht die Datei 90-atmel.rules aus einer Zeile:



```
SUBSYSTEM=="usb", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2104", MODE="0660",
GROUP="plugdev"
```

Dieser Eintrag erlaubt den Zugriff auf das Gerät für Mitglieder der Gruppe „plugdev“.

Um die meisten Programmierer verwenden zu können, wird folgendes Text in 90-atmel.rules empfohlen:

```
Copy this file to /etc/udev/rules.d/90-atmel.rules
AVR ISP mkII - DIAMEX ALL-AVR
SUBSYSTEM=="usb", ATTRS {idVendor}=="03eb", ATTS {idProduct}=="2104", MODE="0660",
 GROUP = "plugdev",
USB ISP-programmer für Atmel AVR
SUBSYSTEM=="usb", ENV {DEVTYPE}=="usb_device", SYSFS {idVendor}=="16c0", MODE="0666",
 SYSFS {idProduct} == "05dc",
USB asp programmer
ATTRS {idVendor}=="16c0", ATTRS {idProduct}=="05dc", GROUP="plugdev", MODE="0660"
USBtiny programmer
ATTRS {idVendor}=="1781", ATTRS {idProduct}=="0c9f", GROUP="plugdev", MODE="0660"
Pololu programmer
SUBSYSTEM=="usb", ATTRS {idVendor}=="1fffb", MODE="0666"
```

Nach dem die Datei erstellt wurde, kann die Erstellung und Inhalt kontrollieren mit:

```
less /etc/udev/rules.d/90-atmel.rules
```

Das ebenfalls als Device 59 erkannte USB Gerät Arduino SA Mega 2560 System erzeugt eine Zugriffsmöglichkeit auf das serielle Gerät „/dev/ttyACM0“ für Mitglieder der Gruppe 'dialout'.

**7.2.6. Gruppen Mitgliedschaft** Deswegen sollte die eigene Benutzerkennung sowohl Mitglied der Gruppe 'plugdev' als auch der Gruppe 'dialout' sein. Das Kommando:

```
sudo usermod -a -G dialout,plugdev $USER
```

sollte die Zugehörigkeit sicherstellen. Jetzt sollte ein Zugriff mit avrdude auf beide Geräte möglich sein. Kontrollieren kann man es mit dem Kommando: 'id'.

Sollten Probleme auftreten kann man die Mitgliedschaft auch über:

Menü/Systemverwaltung/Benutzer und Gruppen/<Passwort>/ nun erscheint ein Fenster mit zwei Reitern.

Wenn man nun im Reiter Benutzer auf sein Name klickt, sieht man rechts sein Profil und die Gruppen Zugehörigkeiten. Mit dem Button <ADD> ist nun Möglich neue Gruppen hinzuzufügen.

**7.2.7. Arbeitsumgebung** Vorbereitung.

Zu erst in der Taskleiste mit grünem Ordnersymbol zum /transistortester/Software/Markus/ navigieren

nun mit der rechten Maustaste auf ComponentTester-1.38m.tgz klicken und in der Auswahl <hier entpacken> der Ordner Dekomprimieren.

Damit das Original erhalten bleibt und weil sich das Terminalfenster immer in ../home/ "user" öffnet, wird empfohlen, dort sein Arbeitsverzeichnis mit dem Namen **Mytester** zu verlegen.

Mit der schon bekannten Methode folgendes Verzeichnis markieren, und im Terminal Fenster mir mittlerer Taste einfügen.

```
cd transistortester/Software/Markus/
```

Nach dem Bestätigen, sieht man alle gepackten Ordner (Endung.tgz) und nur ein Ordner wo diese Endung fehlt -> also unserer (vorher ausgepackter) Ordner.

Bei den folgenden zwei Kommandos, diese zuerst **NUR** einfügen, **ohne [Enter]** zu drücken: !

```
cp -r 'MyT' Mytester/
```

Das Verzeichnis des benötigten Models mit der Maus markieren.

Nun den blinkender Cursor mit Hilfe von [Pfeiltaste-links] zur letztem Zeichen des Text 'MyT' positionieren und diese Zeichen löschen. Nach dem das letzte Zeichen gelöscht wurde, die [MT] Taste der Maus drücken. Erst jetzt [Enter] benutzen. Damit ist die Arbeitsumgebung erstellt. Die Kontrolle der Existenz und Inhalts ist möglich mit:

```
diff 'MyT' Mytester/
```

vobei muss 'MyT', wie vorher, für das Verzeichnis des „benötigten Testers Models“ ausgetauscht werden. Mit dem letzten Anweisung:

In -s ~/transistortester/Software//Markus/Mytester ~/Mytester  
wird die Verknüpfung zum Arbeitsverzeichnis erstellt.

Ab jetzt erreicht man dieses Verzeichnis sehr einfach mit:

[Strg] + [Alt] + [t], cd [Leertaste] My [Tab] [Enter]

und schon ist man in benötigten Verzeichnis. Mit 'ls' sieht man den Inhalt.

Weiter geht es zum bearbeiten von Makefile mit, inzwischen bekanntem Anweisung:

xed Ma [Tab] [Enter]

**Hier ist am wichtigsten, den VORHANDENEN USB-Programmer anmelden.**  
Siehe dazu im Kapitel 5.1.5, auf der Seite 35, Thema PROGRAMMER.

Folgende Aufrufe sind zu empfehlen:

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| make clean         | zum reinigen des Arbeitsumgebung                              |
| make               | zum Übersetzen des Programms                                  |
| make fuses         | zum setzen von ATmega „fuses“ ohne Quarz                      |
| make fuses-crystal | zum setzen von ATmega „fuses“ NUR mit Version mit 8MHz Quarz! |
| make upload        | zum laden des Programms über die ISP Schnittstelle in ATmega  |

Nun bleibt nur die Freude über das erreichter Erfolg.

## 8.1. Fernsteuerungskommandos

Wenn der Tester ein Kommando akzeptiert, antwortet er mit spezifischen Daten oder einem der folgenden Standardtexte:

### 8.1.1. ERR - unbekanntes Kommando

- Kommando im aktuellen Komponentenkontext nicht unterstützt
- Pufferüberlauf

### 8.1.2. OK - Kommando ausgeführt

(manche Kommandos benötigen etwas Zeit zum Ausführen)

### 8.1.3. N/A - Information/Wert nicht verfügbar

Antworten mit Daten beginnen nie mit einem der obigen Standardtexte, um mögliche Unklarheiten zu vermeiden.

## 8.2. Basiskommandos

### 8.2.1. VER - gibt Version der Firmware zurück

- to verify connectivity and to determine command set based on version
- Beispielantwort: "1.33m"

### 8.2.2. OFF - schaltet den Tester aus

- Tester antwortet mit „OK“ vor dem Ausschalten
- Beispielantwort: „OK“ <Tester schaltet ab>

## 8.3. Testkommandos

### 8.3.1. PROBE - sucht nach Bauteil und überspringt alle Pausen für Benutzereingaben

- Tester antwortet mit einem „OK“ nach dem Beenden der Suche
- Beispielantwort: <some time elapses for probing> „OK“

### 8.3.2. COMP - gibt ID der Bauteilart zurück

- für IDs siehe COMP\_\* in common.h
- Beispielantwort für BJT: „30“

### 8.3.3. MSG - gibt Fehlermeldung zurück

- nur falls ein Fehler aufgetreten ist (COMP: 1)
- Antwort kann von der Sprache des Benutzerinterfaces abhängen
- Beispielantwort: „Battery? 1:1500mV“

### 8.3.4. QTY - gibt Anzahl der gefundenen Bauteile zurück

- Beispielantwort für BJT: „1“

### 8.3.5. NEXT - wählt zweites Bauteil aus

- nur wenn zwei Bauteile gefunden wurden (QTY: 2)
- Beispielantwort: „OK“

### 8.3.6. TYPE

- gibt den Typen eines Bauteils zurück
- nur für BJT, FET und IGBT
- verfügbare Typen:

- NPN           NPN (BJT)
- PNP           PNP (BJT)
- JFET           JFET (FET)
- MOSFET       MOSFET (FET)
- N-ch           n-Kanal (FET, IGBT)
- P-ch           p-Kanal (FET, IGBT)- enh. Anreicherungstyp (FET, IGBT)

- dep. Verarmungstyp (FET, IGBT)

- Beispielantwort für BJT: „NPN“
- Beispielantwort für FET (MOSFET): „MOSFET n-ch enh.“

### 8.3.7. HINT

- gibt Hinweise zu speziellen Merkmalen zurück
- nur für Diode, BJT, FET und IGBT
- verfügbare Merkmale:
  - NPN möglicherweise ein NPN BJT (Diode)
  - PNP möglicherweise ein PNP BJT (Diode)
  - R\_BE Basis-Emitter-Widerstand (Diode, BJT)
  - BJT+ integrierte Freilaufdiode auf gleichem Substrat  
erzeugt parasitären zweiten BJT (BJT)
  - D\_FB Body/Freilauf-Diode (BJT, FET, IGBT)
  - SYM Drain und Source symmetrisch (FET)
- Beispielantwort für BJT: „D\_FB R\_BE“
- Beispielantwort für FET (MOSFET): „D\_FB“

### 8.3.8. PIN

- gibt die Anschlußbelegung zurück
- benutzte Kennungen:
 

|               |                                       |                     |
|---------------|---------------------------------------|---------------------|
| - Widerstand  | x = verbunden,                        | - = nicht verbunden |
| - Kondensator | x = verbunden,                        | - = nicht verbunden |
| - Diode       | A = Anode, C = Kathode,               | - = nicht verbunden |
| - BJT         | B = Basis, C = Kollektor, E = Emitter |                     |
| - FET         | G = Gate, S = Source, D = Drain,      | x = Drain/Source    |
| - IGBT        | G = Gate, C = Kollektor, E = Emitter  |                     |
| - SCR         | G = Gate, A = Anode, C = Kathode      |                     |
| - TRIAC       | G = Gate, 2 = MT2, 1 = MT1            |                     |
| - PUT         | G = Gate, A = Anode, C = Kathode      |                     |
| - UJT         | E = Emitter, 2 = B2, 1 = B1           |                     |

- Format der Antwort:  
<Test-Pin #1 Kennung><Test-Pin #2 Kennung><Test-Pin #3 Kennung>
- Beispielantwort für Widerstand: „xx-“
- Beispielantwort für Diode: „C-A“
- Beispielantwort für BJT: „EBC“

- 8.3.9. R** - gibt Widerstandswert zurück
- nur für Widerstand (beinhaltet auch Induktivität)
- Beispielantwort: „122R“

- 8.3.10. C** - gibt Kapazitätswert zurück
- nur für Kondensator
- example responses: „98nF“ „462uF“

- 8.3.11. L** - gibt Induktivitätswert zurück
- nur für Widerstand (beinhaltet auch Induktivität)
- Beispielantwort: „115uH“

- 8.3.12. ESR** - gibt ESR-Wert zurück (Equivalent Series Resistance)
- requires ESR measurement to be enabled
- nur für Kondensator
- Beispielantwort: „0.21R“

- 8.3.13. I\_l** - gibt I\_leak zurück (Leckstromequivalent zur Selbstentladung)
- nur für Kondensator
- Beispielantwort: „3.25uA“

- 8.3.14. V\_F** - gibt V\_F zurück (forward voltage)  
 - nur für Diode und PUT  
 - auch für Body  
 - Diode von MOSFET und Freilaufdiode von BJT oder IGBT  
 - Beispielantwort: „654mV“
- 8.3.15. V\_F2** - gibt V\_F der Niedrigstrommessung zurück (forward voltage)  
 - nur für Diode  
 - Beispielantwort: „387mV“
- 8.3.16. C\_D** - gibt C\_D zurück (Kapazität der Diode)  
 - nur für Diode  
 - Beispielantwort: „8pF“
- 8.3.17. I\_R** - gibt I\_R zurück (reverse current)  
 - nur für Diode  
 - Beispielantwort: „4.89uA“
- 8.3.18. R\_BE** - gibt R\_BE zurück (Basis-Emitter-Widerstand)  
 - nur für Diode und BJT  
 - Beispielantworten: „38.2R“ „5171R“
- 8.3.19. h\_FE** - gibt h\_FE zurück (DC-Stromverstärkungsfaktor)  
 - nur für BJT  
 - Beispielantwort: „234“
- 8.3.20. h\_FE\_r** - gibt den umgekehrten h\_FE zurück (Kollektor und Emitter verdreht)  
 - nur für BJT  
 - Beispielantwort: „23“ - gibt h\_FE zurück (DC-Stromverstärkungsfaktor)  
 - nur für BJT  
 - Beispielantwort: „234“
- 8.3.21. V\_BE** - gibt V\_BE zurück (Basis-Emitter-Spannung)  
 - nur für BJT  
 - Beispielantwort: „657mV“
- 8.3.22. I\_CEO** - gibt I\_CEO zurück (Kollektor-Emitter-Strom, offene Basis)  
 - nur für BJT  
 - Beispielantwort: „460.0uA“
- 8.3.23. V\_th** - gibt V\_th zurück (threshold voltage)  
 - nur für FET (MOSFET) und IGBT  
 - Beispielantwort: „2959mV“
- 8.3.24. C\_GS** - gibt C\_GS zurück (Gate-Source-Kapazität)  
 - nur für FET (MOSFET)  
 - Beispielantwort: „3200pF“
- 8.3.25. R\_DS** - gibt R\_DS\_on zurück (Drain-Source-Widerstand durchgeschaltet)  
 - nur für FET (MOSFET)  
 - Beispielantwort: „1.20R“
- 8.3.26. I\_DSS** - gibt I\_DSS zurück (Drain-Source-Strom, kurzgeschlossenes Gate)  
 - nur für Verarmungstyp-FET  
 - Beispielantwort: „6430μA“
- 8.3.27. C\_GE** - gibt C\_GE zurück (Gate-Emitter-Kapazität)  
 - nur für IGBT  
 - Beispielantwort: „724pF“
- 8.3.28. V\_GT** - gibt V\_GT zurück (Gate-Trigger-Spannung)  
 - nur für Thyristor und TRIAC  
 - Beispielantwort: „865mV“
- 8.3.29. V\_T** - gibt V\_T zurück (Offset-Spannung)  
 - nur für PUT  
 - Beispielantwort: „699mV“

**8.3.30. R\_BB** - gibt R\_BB zurück (interbase resistance)  
- requires UJT detection to be enabled  
- nur für UJT  
Beispielantwort: „4758R“

- Die erste veröffentlichte Version basierend auf Karl-Heinz' 0.99k ist als **v0.99m 2012-09** erschienen und im selbem Monat noch die Version **v1.00m 2012-09** in der folgende Änderungen implementiert wurden:

- Einfaches Menü zur Auswahl von Selbsttest,
- Selbstabgleich,
- Speichern der Abgleichwerte im EEPROM
- Anzeige der Abgleichswerte.
- hFE von 16 auf 32 Bit geändert (keine 65k-Begrenzung mehr).

### **9.1. v1.01m 2012-10**

- Prüfsumme für gespeicherte Abgleichwerte plus Überprüfung.
- Messfunktion für kleine Widerstände (Auflösung: 0,01 Ohm).
- Selbstabgleich um Nulloffset für den Widerstand der Messkabel erweitert.
- CheckResistor() führt zusätzliche Messung für kleine Widerstände (<10 Ohm) durch.
- Funktion zum Vergleich von skalierten Werten eingefügt.
- Mehrere Funktionen an die variable Skalierung von Werten angepasst.

### **9.2. v1.02m 2012-11**

- Obere Grenze für den Widerstand der Messleitungen von 1,00 Ohm im Selbstabgleich eingebaut.
- Die Funktionen für den Selbsttest und -Abgleich führen einen Kurzschluss test durch und geben Rückmeldung.
- Das Hauptmenü gibt Rückmeldung über Erfolg/Fehler der ausgewählten Aktion.

### **9.3. v1.03m 2012-11**

- Erkennungsproblem von Leistungsdioden gelöst. Dioden mit hohem Leckstrom wurden als Widerstand erkannt.
  - Compiler-Warnungen bzgl. nicht initialisierter Variablen beseitigt.
- Vergrößert die Firmware um 44 Bytes :-)

### **9.4. v1.04m 2012-11**

- Einfache Logik in die Dioden-Ausgabe einbaut, damit bei anti-parallelen Dioden die Kapazitätsmessung entfällt.

### **9.5. v1.05m 2012-11**

- LargeCap\_table[] und SmallCap\_table[] vom EEPROM in das Flash verschoben, um den EEPROM-Bedarf zu reduzieren. Die Firmware mit deutschen Texten brauchte mehr als die 512 Bytes vom ATmega168.

### **9.6. v1.06m 2013-03**

- Mehrere kleine Verbesserungen und etwas aufgeräumt.
- TestKey() so erweitert, dass der Benutzer über die erwartete Eingabe informiert werden kann.
- TestKey()-Funktion bzgl. kurzer Tastendrücke verbessert.
- PWM-Generator zur Erzeugung von pulsweitenmodulierten Signalen mit unterschiedlichen Frequenzen und frei wählbarem Tastverhältnis eingebaut.
- Implementation einer Schlaffunktion, um den Stromverbrauch des Testers zu reduzieren. Der durchschnittliche Stromverbrauch wird damit auf etwas die Hälfte verringert (Hintergrundbeleuchtung ausgenommen).
- Entladefunktion verbessert. Wenn das Entladen fehl schlägt, werden die betroffenen Pins und die Restspannung ausgegeben. Das sollte helfen, einen zu niedrigen Wert für CAP\_DISCHARGED

zu entdecken.

- Möglichkeit zum Setzen von Fehlertypen eingebaut.

### **9.7. v1.07m 2013-06**

- Diodenausgabe optimiert und Anzeige von Vf für niedrige Ströme eingebaut.
- Diodenerkennung verbessert. Kondensatoren und Widerstände werden deutlich besser ausgeschlossen. Die Kondensatorerkennung wird bei erkannter Diode übersprungen, um den Suchvorgang zu verkürzen.
- Array-Überlauffehler in CheckResistor() beseitigt.
- Anzeigelogik für Cursor verbessert, um das Vorhandensein weiterer Infos bzw. die erneute Bauteilsuche anzuzeigen.
- Bedienung vom PWM-Generator verbessert, um das Beenden aus Versehen zu vermeiden (braucht nun zwei kurze Tastendrucke).
- Generische Menüfunktion eingebaut und alle Menüs darauf umgestellt ( geändertes Layout!).
- TestKey() produziert nun einen schönen, blinkenden Cursor.

### **9.8. v1.08m 2013-07**

- Da SmallResistor() bei bestimmten Induktivitäten keinen korrekten Gleich- Stromwiderstandswert liefern kann, wurde CheckResistor() um eine Erkennung der Problemfälle ergänzt, um die Messwerte der Standardmessung beizubehalten.
- Induktivitätsmessung eingebaut (nur für ATmega328/P)
- Kleinere Verbesserungen bei der Anzeige von Dioden und Bipolartransistoren.
- Leckstrommessung eingebaut.
- Problem bei Germanium-Transistoren mit hohem Leckstrom gelöst. Wurden als P-Kanal JFET erkannt.
- Ein paar Funktionen umbenannt und Kommentare ergänzt bzw. umformuliert.

### **9.9. v1.09m 2013-07**

- Erkennung von IGBTs eingebaut.
- Zusätzliche Überprüfung von MOSFETs eingebaut.
- Die hFE-Messung für Bipolartransistoren berücksichtigt den Leckstrom bei der Messung in Emitterschaltung.
- Bei MOSFETs wird die Richtung der integrierten Diode angezeigt.
- Problem mit verdrehten Drain und Source-Pins bei Anreicherungs-MOSFETs gelöst.
- Lösung für Probleme mancher IDEs mit dem Makefile. Wichtige Werte bzw. Einstellungen können auch in config.h gesetzt werden.

### **9.10. v1.10m 2013-10**

- Unterstützung für externe 2,5V Spannungsreferenz eingebaut (Hardware-Option).
- Unterstützung für Schutz-Relais (Kondensatorentladung) eingebaut ( Hardware-Option).
- Auf-Wiedersehen-Text beim Ausschalten zu Willkommen-Text beim Einschalten geändert, um die Erkennung einer zu niedrigen Versorgungsspannung zu erleichtern und den Spannungseinbruch durch einen DC-DC-Konverter beim Einschalten abzumildern.
- Test von Zenerdioden implementiert (Hardware-Option).
- Das Hauptmenü hat eine Beenden-Option, um das Menü bei Aufruf aus Versehen zu verlassen.
- Unterstützung von 16MHz MCU-Takt.

### **9.11. v1.11m 2014-03**

- Pin-Erkennung von Triacs verbessert (G und MT1). Die Ausgabe zeigt MT1 und MT2.
  - Dedizierte Ausgabefunktion für Pinbelegung von Halbleitern. Ausgabe auf das Format „123=“ zur besseren Lesbarkeit umgestellt.
  - Mehrere Ausgabefunktionen optimiert.
  - Test von Bipolartransistoren verbessert, um Transistoren mit Schutzdiode auf dem gleichen Substrat zu erkennen (erzeugt einen parasitären zweiten Transistor). Die Transistorausgabe kennzeichnet diesen Spezialfall mit einem "+" hinter der Typenangabe.
  - Diodenausgabe um Anzeige von möglichem Bipolartransistor mit Schutzdiode und Basis-Emitter-Widerstand erweitert. Dieser wird als Doppeldiode erkannt.
- Die Ausgabe des Basis-Emitter-Widerstands signalisiert diesen Spezialfall.



- Ausgabe von Bipolartransistoren um Anzeige von Basis-Emitter-Widerstand ergänzt. Wenn ein Basis-Emitter-Widerstand gefunden wurde, wird die Ausgabe von hFE und V\_BE übersprungen, da beide Werte nicht stimmen können,
- Erkennung der integrierten Diode von Verarmungs-FETs im Diodentest verbessert.
- Erkennungsproblem von Drain und Source bei Verarmungs-FETs beseitigt.
- Erkennung von symmetrischem Drain und Source bei Verarmungs-FETs.
- Vth ist nun negativ für P-Kanal FETs.
- Messung von V\_GT für Thyristoren und Triacs.
- Wegen wachsender Firmwaregröße gibt es den PWM-Generator nur noch für den ATmega328.

#### **9.12. v1.12m 2014-03**

- Umlautproblem bei deutschen Texten gelöst (Dank an Andreas Hoebel).
- ESR-Messung für Kondensatoren >0,18μF.
- LCD-Modul-Ausgabe optimiert, um ein paar Bytes Flash einzusparen.

#### **9.13. v1.13m 2014-07**

- Tschechische Texte (Dank an Kapa).
- Direkte ESR-Messung und PWM-Generator geben benutzte Testpins aus.
- Handhabung von Precompiler-Anweisungen für Optionen optimiert.
- Unterstützung von Drehencodern für die Bedienung (Hardware-Option).

#### **9.14. v1.14m 2014-08**

- Benutzerschnittstelle für Drehencoder angepasst.
- Compiler-Warnung bzgl. R\_Pin2 in ShowDiode() beseitigt (Dank an Milan Petko).
- Widerstände zwischen 1,5k und 3k-Ohm wurden als Doppeldioden erkannt. Toleranzen der Widerstandserkennung in CheckDiode() angepasst (Danke an nessatse).
- ShortCircuit() so modifiziert, dass man das Erzeugen eines gewollten Kurzschlusses bei Problemen abbrechen kann.
- Frequenzzähler eingebaut (Hardware-Option).

#### **9.15. v1.15m 2014-09**

- Erweiterung von TestKey() um die Erkennung der dynamischen Drehgeschwindigkeit des optionalen Drehencoders.
- Rechteck-Signalgenerator mit variabler Frequenz implementiert.
- MeasureInductance() zur Rückgabe der Zeit in ns geändert und die Berechnung in MeasureInductor() angepasst (Dank an Vlastimil Valouch).

#### **9.16. v1.16m 2014-09**

- Test für Drehencoder.
- Ein paar Kleinigkeiten in MeasureInductance() verbessert, um die Genauigkeit zu erhöhen.
- ShowAdjust() um die Anzeige der Absolutwerte von Vcc und der internen Spannungsreferenz ergänzt (Vorschlag von Vlastimil Valouch).
- Mehrere kleine Verbesserungen.

#### **9.17. v1.17m 2015-02**

- Verbesserung von CheckDiode(). Gemessenes Vcc wird bei der Widerstandsprüfung beachtet.

Außerdem Erkennungsproblem von Widerständen um die 2k bei optionalem DC-DC-Konverter (HW\_ZENER) gelöst.

- Fehlerhafte Kommentare korrigiert.
- Integer-Datentypen aufgeräumt.

#### **9.18. v1.18m 2015-07**

- MenuTool() so verbessert, dass nur eine geänderte Liste aktualisiert wird. Ansonsten wird nur der Auswahlindikator aktualisiert.
- Fehler im Variablenmanagement in config.h beseitigt.
- Möglichkeit zum Zurücksetzen auf Firmware-Standardwerte beim Einschalten.
- Funktionen zum Speichern/Lesen der Abgleichwerte optimiert.
- Treiber für ST7565R Grafikmodule.
- Einfache Umgebung zum Einbinden von weiteren LCD-Controllern entworfen.

Generische Displayfunktionen nach display.c verschoben.

Jeder Controller erhält eine eigene Source- und Headerdatei.

Der alte Treiber für HD44780 wurde an die neue Umgebung angepasst.

- Benutzerschnittstelle für den flexiblen Umgang mit mehrzeiligen LCD-Modulen umgebaut.

- Sourceabhängigkeiten zu ATmega168 entfernt (zu klein ;).

- Bedienungslogik in MenuTool() optimiert.

- Neue Firmware-Edition gestartet, welche auch grafische LCD-Module unterstützt.

Diese Version heißt "Trendy Edition". Die alte Firmwareversion nennt sich nun "Classic Edition".

### **9.19. v1.19m 2015-11**

- Grafisches Pinout für 3-Pin Halbleiter. Zeigt Symbol plus Testpins.

- Farbunterstützung eingebaut.

- Direkte Ausgabe der Diodenanzahl in ShowDiode(), wenn mehr als 3 Dioden gefunden wurden (nicht länger per Show\_Fail()). Hinweis von hapless@EEVblog

- LCD\_ClearLine() in allen LCD-Modul-Treibern so erweitert, dass auch nur der Zeilenrest gelöscht werden kann, um das Löschen speziell für Grafik-LCDs zu beschleunigen.

Die Idee ist, zuerst den Text auszugeben und dann den Zeilenrest zu löschen, anstatt zuerst die ganze Zeile zu löschen und dann den Text auszugeben.

- Treiber für ILI9341/ILI9342 basierte LCD-Module geschrieben.

Dank an Overtuner@EEVblog-Forum für zwei LCD-Module zum Testen.

- Problem mit dem  $\mu$ /micro Zeichen in den Zeichensatzdateien gelöst.

- Fehler bei Zeichen größer 8x8 in LCD\_Char() für den ST7565R beseitigt.

- Tschechische Texte aktualisiert (Dank an Kapa).

- Kleinen Fehler in MenuTool() beim Sprung vom letzten zum ersten Punkt beseitigt.

### **9.20. v1.20m 2015-12**

- Funktion zum Erkennen und Decodieren von IR-Fernsteuerungen implementiert.

Benötigt ein TSOP IR-Empfängermodul.

- MainMenu() geändert, um RAM-Nutzung zu verkleinern.

### **9.21. v1.21m 2016-01**

- Lizenziert unter der EUPL V.1.1

- Laden und Speichern von Abgleichs-werten optimiert und Unterstützung von zwei Abgleichprofilen eingebaut.

- IR-Detektor um RC-6 erweitert. Tasten-Problem bei vorzeitigem Entfernen des IR-Empfängermoduls beseitigt.

Konfigurationsschalter zum Abschalten des Vs-Widerstands zur Strombegrenzung für 5V IR-Empfängermodule.

### **9.22. v1.22m 2016-03**

- Test für Opto-Koppler mit Ausgabe von V\_f der LED, CTR und t\_on bzw. t\_off Zeiten (Typen mit Transistorausgang).

Dank an all\_repair@EEVblog für Opto-Koppler zum Testen.

### **9.23. v1.23m 2016-07**

- Unterstützung von PCD8544 und ST7735 kompatiblen LCD-Modulen.

Dank an hansibull@EEVblog für ein PCD8544-Display.

- wait.s für 20MHz MCU-Takt ergänzt.

- MeasureESR() unterstützt nun auch andere ADC-Taktraten als 125kHz.

- Erkennung von PUTs (Programmable Unijunction Transistor) und UJT's (Unijunction Transistor) eingebaut.

Dank an edavid@EEVblog für das Zusenden von ein paar UJT's zum Testen.

- Kleinere Optimierungen für ILI9341 and ST7565R.

- Erneut Problem mit Zeichen größer 8x8 für den ST7565R beseitigt.

- Der /RES Port-Pin für ILI9341 wurde ignoriert. Fehler beseitigt und auch eine falsche Verzögerung für Hardwareresets korrigiert.

- Unterstützung von individuellen Datenleitungen für HD44780 basierte LCD-Module.

- Benutzer-definierbarer Spannungsteiler für Batteriespannung.

- Ausgabe von If für Opto-Koppler ergänzt.
- Testpins vom ESR-Tool auf 1-3 geändert, um mit der k-firmware kompatibel zu sein.
- MCU-spezifische globale Einstellungen wurden in jeweils eigene Header- Dateien verschoben. Mehrere kleine Anpassungen, um auch ATmega664/1284 zu unterstützen.
- Tschechische Texte aktualisiert. Dank an Kapa.

#### **9.24. v1.24m 2016-08**

- Messung vom Selbstendladungsleckstrom von Kondensatoren größer  $4,7\mu\text{F}$ .
  - Typenerkennung von Bipolar-Transistoren mit Diode auf dem gleichen Substrat.
  - Messung von Leckstrom für Ströme bis in den nA-Bereich erweitert.
- Für Dioden und Bipolar-Transistoren werden Leckströme über 50nA angezeigt.
- Die Anzeige von Freilaufdioden bei Transistoren prüft nun auf eine korrekte Diode (Pins und Polarität).
  - Fehler in der Anzeige von Freilaufdioden bei Bipolar-Transistoren beseitigt.
  - Funktion zum Suchen einer bestimmten Diode geschrieben und mehrere andere Funktionen entsprechend angepasst.
  - Erkennung von Dioden verbessert, um auch Germanium-Dioden mit sehr niedrigem  $V_f$  bei kleinen Strömen zu finden.
  - Problem mit LCD\_ClearLine(0) für ILI9341 und ST7735 gelöst.
  - Verbesserung der Erkennung von Verarmungs-FETs. Germanium-Transistoren mit hohem Leckstrom werden ausgefiltert.

Auch werden nun FETs mit niedrigem  $I_{DSS}$  erkannt. Messung von  $I_{DSS}$ .

#### **9.25. v1.25m 2016-09**

- Jede Menge Änderungen zur Unterstützung von ATmega 324/644/1284.
- Management der Test-Widerstände auf variable Port-Pins umgestellt.
- Software-Option für Farbkodierung der Testpins.
- Farbmanagement zentralisiert.
- Datei mit Auflistung der Einstellungen für verschiedene Testerversionen bzw. Clone.
- Kleines Problem mit den 24x24-VP Symbolen in config.h beseitigt. Hinweis von lordstein@EEVblog und hapless@EEVblog.

#### **9.26. v1.26m 2016-12**

- Kompensation für Induktivitätsmessung eingebaut (benötigt weitere Arbeit).
- Anpassung von FrequencyCounter() zur Unterstützung von ATmega 324/644/1284.
- Problem in der Logik der Induktivitätsmessung beseitigt. Hinweis von indman@EEVblog.
- Fehler in der Handhabung der Spannungsreferenzen für ATmega 324/644/1284 gelöst.
- Erkennung der Drehgeschwindigkeit von Drehencodern verbessert, um unterschiedliche Werte für Pulse/Schritt bzw. Pulse/Rastung besser zu handhaben.
- Alle Treiber für SPI-basierte LCD-Module um Hardware SPI erweitert.

#### **9.27. v1.27m 2017-02**

- GetLeakageCurrent() um Hochstrommessung für CLDs erweitert. Dank an texaspyro@EEVblog für ein paar Testdioden.
- Fehler in MilliSleep() beseitigt.
- Problem mit großer Induktivität in Diodenerkennung beseitigt.
- Kompensation für Induktivitätsmessung im mH-Bereich.
- Unterstützung für PCF8574 basierte LCD-Adapter in Treiber für HD44780.
- Treiber für bit-bang und Hardware I2C.
- Fehler in der Handhabung der variablen Pinbelegung für HD44780 basierte LCD-Module beseitigt.
- Farbiges Pinout für mehrere Menüfunktionen.
- Prüffunktion für Modellbau-Servos.
- Alternativer PWM-Generator mit variabler Frequenz und Puls-weite. Benötigt Drehencoder und größeres Display.
- Ausgabe von  $R_{DS}$  für MOSFETs und  $V_f$  der intrinsischen Diode.
- Unterstützung für festes IR-Empfängermodul in IR-Detektor/Dekoder.
- Edition im Namen entfernt, da die Classic Edition inzwischen veraltet ist.

### 9.28. v1.28m 2017-04

- Mehr/Weniger-Tasten als Alternative zum Drehencoder (HW\_INCDEC\_KEYS).
- Zurücksetzen auf Standardfrequenz im Rechteck-Generator ergänzt.
- Weitere Verbesserungen der Erkennung der Drehschwindigkeit von Dreh-encodern (ENCODER\_STEPS). Änderungen in den Funktionen, welche die Drehgeschwindigkeit nutzen.
- Zurücksetzen auf Standardwerte im alternativen PWM-Generator ergänzt.
- Russischer Text von indman@EEVblog (nur 8x16 Zeichensatz horizontal ausgerichtet)
- Unterstützung von festem Folienkondensator für den Selbstabgleich von Spannungsoffsets.
- Potentiellen Fehler in der Handhabung vom V\_ref Offset in SmallCap() beseitigt.
- Konfigurationsoption für LCD-Module mit ST7735, um mit gelöschter Anzeige zu starten (keine zufälligen Punkte).

### 9.29. v1.29m 2017-07

- Unterstützung von Touch-Screens und Treiber für ADS7843 kompatible Controller.
- Fehler in Kontrast-Einstellung für PCD8544 korrigiert.
- Dummen Fehler in CheckSum() beseitigt.
- Treiber für ST7920 basierte LCD-Module mit 64x128 Pixel.
- SmallResistor() optimiert und Erkennungslogik in CheckResistor() verbessert, damit sehr kleine Widerstände in Verbindung mit Kontaktwiderständen der Prüfkabel besser erkannt werden.
- Steuerlogik und Schwellwert für Darlingston-Transistoren in Get\_hFE\_C() geändert, um Problem mit manchen NPN-Typen zu beseitigen.
- Zentraler SPI-Treiber. Treiber und Konfiguration von LCD-Modulen entsprechend angepasst.
- Italienischer Text von Gino\_09@EEVblog.
- Unterstützung von HD44780 mit kyrillischem Zeichensatz durch hapless@EEVblog.

### 9.30. v1.30m 2017-10

- Option für Komma statt Punkt für Dezimalstellen.
- Unterstützung für den erweiterten Frequenzzähler mit Eingangspuffer, LF und HF-Quarzoszillator.
- Kleinere Verbesserungen beim einfachen Frequenzzähler.
- Problem mit Torzeit in Frequenzzähler für Frequenzen unter 10kHz bei MCU-Takt von 20MHz beseitigt.
- ESR-Messung für kürzere Lade-Pulse modifiziert, d.h. der ESR kann nun für Kondensatoren ab 10nF gemessen werden. Wer die alte Messmethode bevorzugt, kann diese alternativ aktivieren.
- Fehler in der Kurzschlusserkennung der Testpins beseitigt.
- LCD-Treiber für ST7920 um 180° rotierte Ausgabe erweitert.

### 9.31. v1.31m 2017-12

- IR-Fernbedienung (Sender).
- Unterstützung für feste Signalausgabe über OC1B, wenn OC1B nicht für den Testwiderstand von Testpin #2 benutzt wird.
- Einstellungen für Batterie-Überwachung geändert, um auch andere Optionen zur Stromversorgung zu unterstützen.
- Treiber für SSD1306 basierte OLED-Module.
- Farbunterstützung für Auswahl von Menüpunkten oder Parametern.
- Treiber für ILI9163 basierte LCD-Module.
- Problemchen in Rechteckgenerator beseitigt.
- LCD-Treiber für PCD8544 um 180° rotierte Ausgabe erweitert.
- Editierfehler in Servo\_Check() berichtigt.

### 9.32. v1.32m 2018-02

- Ausgabe von gefundenen Bauteilen zusätzlich über serielle Schnittstelle.
- Treiber für serielle TTL-Schnittstelle (Hardware & Bit-Bang).
- Aktualisierung von var\_russian.h (Dank an indman@EEVblog).
- Unterstützung von X&Y-Offsets im ST7735-Treiber.
- Einstellungen der Batterieüberwachung geändert.

Schalter zum Abschalten der Batterieüberwachung und für nicht überwachte externe Stromversorgung ergänzt.

- Konfigurationsschalter zur Auswahl des alternativen Betriebsmodus beim Starten (UI\_AUTOHOLD)
  - Filter für Germanium-Transistoren mit hohem Leckstrom in der Funktion zur Erkennung von Verarmungstyp-FETs verbessert.
  - Graphisches Pinout im Treiber für PCD8544 ergänzt.
- Fehler in der Funktion LCD\_CharPos() für rotierte Ausgabe im PCD8544-Treiber beseitigt.
- Funktionen für graphisches Pinout verbessert und teilweise nach display.c verschoben.
- Separate Ausgabe vom Pinout falls notwendig.
- Indikator bei Nutzung einer externen Spannungsreferenz (Werte Anzeigen).
  - IR-Decoder verbessert und optionale Protokolle eingebaut.
  - Zusätzliche Protokolle für IR-Fernbedienung.

### 9.33. v1.33m 2018-05

- Orientierung von TRIAC-Symbol in symbols\_32x32\_hf.h korrigiert.
- Fernsteuerkommandos zur Automatisierung (über die serielle TTL-Schnittstelle).
- Der X & Y-Offset für den ST7735-Treiber kann nun verändert werden.
- Aufrufen des Menüs per Kurzschluss der Testpins ist nun eine Option (UI\_SHORT\_CIRCUIT\_MENU).
- Problem mit Entlade-Relais in Verbindung mit Drehencoder beseitigt.
- Konfigurationsschalter zum Ausschalten der MCU-Schlafmodi ergänzt.
- Datenempfang für serielle TTL-Schnittstelle (Bit-Bang & Hardware USART).
- Fehler in serieller Textausgabe beseitigt, und serielle Ausgabe für Resultate des Opto-Koppler-Tests eingebaut.
- Dänischer Text (von glennndk@mikrocontroller.net).
- Einstellungen für Korrekturfaktoren für Kondensatoren.

### 9.34. v1.34m 2018-10

- Leckstromtest für Kondensatoren.
- Standardwert für RH\_OFFSET auf 350 Ohm geändert.
- Problem mit fehlendem Menüeintrag für festes IR-Empfängermodul beseitigt.
- Polnischer Text (Dank an Szpila).
- Displaytreiber für Ausgabe auf VT100-Terminal.
- Unterstützung von Temperatursensor DS18B20.
- Treiber für OneWire-Bus.

### 9.35. v1.35m 2019-02

- Bei dem Kapazitäts-Offset kann man statt dem bisherigen Durchschnittswert für alle Test-Pins nun auch Test-Pin spezifische Offsets nutzen (CAP\_MULTIOFFSET\_).
- Pin Definition für ST7920 im 4-Bit Parallel-Modus in config\_644.h korrigiert (gemeldet von jakeisprobably@EEVblog).
- Unterstützung von 3-Line SPI in SSD1306-Treiber eingebaut.
- Der SPI-Treiber kann nun auch 9-Bit-Worte senden (nur Bitbang).
- Problem mit steigender Abweichung bei Widerständen zwischen 7k5 und 19k5 Ohm in CheckResistor() gelöst (gemeldet von Vitaliy).
- Alternative Verzögerungsschleife in IR\_Send\_Pulse() eingebaut, welche per SW\_IR\_TX\_ALTDelay aktiviert wird (Dank an Vitaliy).
- Der Konfigurationsschalter für zusätzliche IR-Protokolle SW\_IR\_EXTRA wurde durch SW\_IR\_RX\_EXTRA für den Empfänger/Decoder und SW\_IR\_TX\_EXTRA für den IR-Sender ersetzt.
- Problem mit fehlendem Newline für Fernsteuerkommandos in Display\_NextLine() beseitigt.
- Ausgabe für SIRC in IR\_Decode() geändert, um näher am Protokoll zu sein (Vorschlag von Vitaliy).
- Fehler in IR\_Send\_Code() für SIRC-20 beseitigt (gemeldet von Vitaliy).
- Aktualisierung von var\_russian.h (Dank an indman@EEVblog).
- Automatische Abschaltung für Auto-Hold-Modus (POWER\_OFF\_TIMEOUT).

- Pin-Konfiguration für Test-Taste und Stromversorgungskontrolle getrennt (CONTROL\_PORT -> POWER\_PORT und BUTTON\_PORT).
- Mehrere kleine Verbesserungen.

### 9.36. v1.36m 2019-05

- Optionalen 6x8 Zeichensatz im ST7565R-Treiber ergänzt.
- Optionaler Menüpunkt zum Abschalten des Testers (SW\_POWER\_OFF).
- TestKey() und Zener\_Tool() um Batterieüberwachung ergänzt.
- Erkennung von zwei kurzen Betätigungen der Testtaste in TestKey() eingebaut, und doppelte Funktionalität in mehreren Funktionen entfernt, um Größe der Firmware zu verkleinern.
- Treiber für Displays mit ST7036 (4-bit parallel & 4-wire SPI, ungetestet).
- Eigene Funktionen für Stromversorgung und Batterieüberwachung zur besseren Integration mit anderen Funktionen.
- Treiber für Displays mit PCF8814 (3-line SPI; Dank an Mahmoud Laouar fürs Testen).
- Treiber für Displays mit STE2007/HX1230 (3-line SPI).
- Fehler in Funktion LCD\_Clear im PCD8544-Treiber beseitigt.
- Fehlender kyrillischer Font im ST7565R-Treiber eingetragen (gemeldet von Andrey@EEVblog).
- Aktualisierung von font\_8x16\_cyrillic\_vfp.h (Dank an Andrey@EEVblog).
- Problem mit falschem Zeichen in font\_HD44780\_cyr.h gelöst.

### 9.37. v1.37m 2019-09

- Fehler in DS18B20\_Tool(), wenn ONEWIRE\_IO\_PIN aktiviert ist, beseitigt (gemeldet von bm-magic).
- Problem bei der Anzeige der Watchdog-Fehlermeldung auf Farbanzeigen gelöst.
- Neue Funktion: Ereigniszähler (HW\_EVENT\_COUNTER, Vorschlag von bm-magic).
- Der einfache Frequenzzähler benutzt nun TestKey() zur Benutzereingabe. Zum Beenden zweimal kurz Taste drücken (war vorher ein Tastendruck).
- Option zur Anzeige des umgekehrten hFE-Wertes von Transistoren (SW\_REVERSE\_hFE, Vorschlag von towe96@EEVblog). Ebenfalls Fernsteuerkommandos um den Befehl „h\_FE\_r“ erweitert.
- Bitclock-Einstellung (BITCLOCK) für avrdude in Makefile (Vorschlag von bm-magic).
- Problem bei der TRIAC-Erkennung im Fall von einem zu hohen I\_GT in Q3 oder zu hohem I\_H beseitigt (I\_GT-Problem gemeldet von petroid).
- Texte Tester\_str, PWM\_str, Hertz\_str and CTR\_str in sprachspezifisch Header-Dateien verschoben (Vorschlag von indman@EEVblog).
- Ausgabe von Frequenzwerten (Hertz) auf festen String geändert (vorher „H“ als Einheit für DisplayValue() plus zusätzliches „z“).
- Option zum Anzeigen von Bedienungshilfen (UI\_KEY\_HINTS). Momentan nur „Menu/Test“ (Vorschlag von carrascoso@EEVblog).
- Polnische Texte aktualisiert (C szpila@EEVblog).
- Russische Texte (Dank an indman@EEVblog).
- Spanische Texte (Dank an pepe10000@EEVblog).

### 9.38. v1.38m 2019-12

- Optionales Runden der Temperatur für DS18B20 (UI\_ROUND\_DS18B20, Vorschlag von Obelix2007@EEVblog)
- Unterstützung von DHT11, DHT22 und kompatiblen Sensoren (SW\_DHTXX, Dank an indman@EEVblog und Obelix2007@EEVblog fürs Testen).
- Zwei dünne kyrillische Zeichensätze ergänzt (Dank an Andrey@EEVblog).
- Ausgabe von Bipolartransistoren so geändert, dass nun V\_BE und hFE auch im Fall eines B-E-Widerstands angezeigt werden. Ebenfalls Fernsteuerkommandos entsprechend angepasst.
- Tschechische Texte plus mehrere Zeichensätze mit tschechischen Zeichen (Dank an Bohu).
- Funktionen zum Beobachten von R/C/L (SW\_MONITOR\_RL und SW\_MONITOR\_C, Vorschlag von indman@EEVblog).
- Trigger-Ausgang für Ereigniszähler (Vorschlag von Bohu).
- Tschechische Texte aktualisiert (Dank an Bohu).
- Die hFE-Messung mit Common-Collector-Schaltung und Rl als Basiswiderstand kann deaktiviert werden.

viert werden (NO\_HFE\_C\_RL), damit bestimmte Tester keine überhöhten Ergebnisse liefern (gemeldet von Obelix2007@EEVblog).

- Option zur Ausgabe der Zener-Spannung in hoher Auflösung (ZENER\_HIGH\_RES, Vorschlag von Andbro@EEVblog).
- OneWire\_Probes() verbessert, um Fehl-Erkennung zu minimieren.
- Russische Texte aktualisiert (Dank an indman@EEVblog).
- Spanische Texte aktualisiert (Dank an pepe10000@EEVblog).

---

## *Literaturverzeichnis*

---

- [1] <http://www.mikrocontroller.net/articles/AVR-Transistortester>  
*Online Dokumentation des Transistortester*, Online Article, 2009-2011
- [2] Markus Frejek <http://www.mikrocontroller.net/topic/131804>  
*Forum von Markus Frejek*, 2009.
- [3] [http://www.mikrocontroller.net/articles/AVR\\_Transistortester](http://www.mikrocontroller.net/articles/AVR_Transistortester)  
*Kurze Eigenschaftenbeschreibung des TransistorTester von Karl-Heinz K.*, Online Article, 2012
- [4] <http://www.mikrocontroller.net/topic/248078>  
*Thread von Karl-Heinz K.*, Thread und Software Versionen, 2012
- [5] <https://github.com/svn2github/transistortester/blob/master/Doku/trunk/pdftex/german/ttester.pdf>  
*Aktuelle Transistor Tester Beschreibung*
- [6] <https://www.mikrocontroller.net/svnbrowser/transistortester/Software/Markus>  
*Komplette Software Sammlung* 2012-2019
- [7] <https://github.com/madires/Transistortester-Warehouse>  
*Komplette Software Sammlung* 2012-2019
- [8] [madires@theca-tabellaria.de](mailto:madires@theca-tabellaria.de)  
*Autor Mail*
- [9] <http://www.mikrocontroller.net/articles/AVRDUDE>  
*Online Dokumentation des avrdude programmer interface*,  
Online Dokument, 2004-2011
- [10] <https://www.mikrocontroller.net/topic/248078>  
*Hauptsprache ist Deutsch, Englisch ist aber auch ok.*
- [11] [https://www.eevblog.com/forum/testgear/\(Dollarzeichen\)20-lcr-esr-transistor-checker-project/](https://www.eevblog.com/forum/testgear/(Dollarzeichen)20-lcr-esr-transistor-checker-project/)  
*Nur Englisch.*