

Beschreibung Component-Tester

Ein Gerät zum Bestimmen und Testen von elektronischen Bauteilen, mit vielen zusätzlichen Funktionen



Version 1.47m

Markus Reschke

© 2012

madires@theca-tabellaria.de



BJT-PNP
hFE=226
I_e=1.6mA
U_{be}=673mV+



BJT-PNP
hFE=226
I_e=1.6mA
PIN 1=C 2=B 3=E

Zusammengestellt
von bm-magic



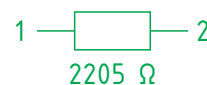
15. Juni 2022



U_f=670mV

25pF-17pF @0-5V

I_r=.80µA



N-E-MOS

V_t=3.1V

C_g=2.86nF

RDS=.2Ω



N-E-MOS

V_t=3.1V

C_g=2.86nF

3 → 2 U_f=598mV+



N-E-MOS

V_t=3.1V

C_g=2.86nF

PIN 1=G 2=D 3=S

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 Hinweise Lizenz	7
1.4	Unterschiede zur k-Firmware	8
1.5	Quellcode	8
1.6	Unterstützte Hardware	9
1.7	Unterstützte Hardware-Optionen	9
1.7.1	Bedienung	9
1.7.2	Verbesserungen	9
1.7.3	Zusätzliche Tests und Messungen	9
1.8	Firmware übersetzen	9
2	Hardware	10
2.1	Hardware-Optionen	10
2.2	Und natürlich die Software-Optionen	10
2.3	Verfügbare Sprachen	11
2.4	Konfiguration	11
2.5	Busse & Schnittstellen	11
2.5.1	I2C/SPI	11
2.5.2	Serielle TTL-Schnittstelle	12
2.5.3	One Wire	12
2.6	Anzeige-Module	12
2.6.1	Zeichensätze und Symbole	13
2.6.2	HD44780	13
2.6.3	ILI9163	14
2.6.4	ILI9341/ILI9342	14
2.6.5	ILI9481	15
2.6.6	ILI9486	15
2.6.7	ILI9488	15
2.6.8	PCD8544	16
2.6.9	PCF8814	16
2.6.10	SH1106	16
2.6.11	SSD1306	17
2.6.12	ST7036 (ungetestet)	17
2.6.13	ST7565R	18
2.6.14	ST7735	18
2.6.15	ST7920	18
2.6.16	STE2007/HX1230	19
2.6.17	VT100 Terminal	19
2.7	Tasten und Eingabeoptionen	19
2.7.1	Die Test-Taste	19

2.7.2	Drehencoder (Hardware-Option)	19
2.7.3	Mehr/Weniger-Tasten (Hardware-Option)	20
2.7.4	Touch-Screen (Hardware-Option)	20
2.7.5	Benutzerschnittstelle	21
2.7.6	Summer/Pieper (Hardware-Option)	21
2.7.7	Kommunikation mit PC	21
2.7.8	Serielle Ausgabe	22
2.7.9	Automatisierung	22
2.7.10	VT100-Ausgabe	22
3	Bedienung	23
3.0.1	Einschalten	23
3.0.2	Bauteilesuche	23
3.0.3	Batterieüberwachung	23
3.0.4	Ausschalten	24
3.1	Menü	24
3.1.1	PWM-Generator	24
3.1.2	Einfache PWM	24
3.1.3	Erweiterte PWM	25
3.1.4	Rechteck-Signalgenerator	25
3.1.5	Zenertest (Hardware-Option)	25
3.1.6	Logiktester (Hardware-Option)	25
3.1.7	Durchgangsprüfer (Hardware-Option)	26
3.1.8	ESR-Tool	26
3.1.9	Kondensatorleckstrom	26
3.1.10	R/C/L-Monitore	27
3.1.11	L/C-Meter (Hardware-Option)	27
3.1.12	Frequenzzähler (Hardware-Option)	28
3.1.13	Einfacher Zähler	28
3.1.14	Erweiterter Zähler	28
3.1.15	Klingeltester (Hardware-Option)	28
3.1.16	Ereigniszähler (Hardware-Option)	28
3.1.17	Drehencoder	29
3.1.18	Kontrast	29
3.1.19	Detektor/Decoder für IR-Fernbedienungen	29
3.1.20	IR-Fernbedienung	30
3.1.21	Opto-Koppler-Test	32
3.1.22	Modellbau-Servo-Test	32
3.1.23	OneWire-Scan	33
3.1.24	DS18B20-Temperatursensor	33
3.1.25	DHTxx-Sensoren	33
3.1.26	MAX6675/MAX31855	34
3.1.27	Selbsttest	34
3.1.28	Selbstabgleich	34
3.1.29	Speichern/Laden	35
3.1.30	Werte anzeigen	35
3.1.31	Zeichensatz/Symbole	35
3.1.32	Ausschalten	35
3.1.33	Exit	35
4	Messdetail	36
4.1	Widerstände	36
4.2	Kondensatoren	36
4.3	Induktivitäten	37
4.4	Bauteile entladen	37

4.5	ADC Oversampling	38
4.6	Ergebnisanzeige	38
4.7	Zusätzliche Hinweise	38
4.8	Hilfreiche Links	40
4.9	Firmware-Änderungen	40
5	Fernsteuerung	41
5.1	Fernsteuerungskommandos	41
5.1.1	ERR	41
5.1.2	OK	41
5.1.3	N/A	41
5.2	Basiskommandos	41
5.2.1	VER	41
5.2.2	OFF	41
5.3	Testkommandos	41
5.3.1	PROBE	41
5.3.2	COMP	41
5.3.3	MSG	41
5.3.4	QTY	41
5.3.5	NEXT	41
5.3.6	TYPE	41
5.3.7	HINT	42
5.3.8	MHINT	42
5.3.9	PIN	42
5.3.10	R	42
5.3.11	C	42
5.3.12	L	42
5.3.13	ESR	43
5.3.14	I_l	43
5.3.15	V_F	43
5.3.16	V_F2	43
5.3.17	C_D	43
5.3.18	I_R	43
5.3.19	R_BE	43
5.3.20	h_FE	43
5.3.21	h_FE_r	43
5.3.22	I_C	43
5.3.23	I_E	43
5.3.24	V_BE	43
5.3.25	I_CEO	43
5.3.26	V_th	43
5.3.27	C_GS	44
5.3.28	R_DS	44
5.3.29	V_GS_off	44
5.3.30	I_DSS	44
5.3.31	C_GE	44
5.3.32	V_GT	44
5.3.33	V_T	44
5.3.34	R_BB	44
5.3.35	V_Z	44
6	Code	45
6.1	Makefile	45
6.1.1	MCU-Typ	45
6.1.2	MCU-Taktfrequenz	45

6.1.3	Oszillator-Typ	45
6.1.4	Avrdude MCU-Typ	46
6.1.5	Avrdude ISP-Programmierer	46
6.2	config.h	47
6.2.1	Hardware Bedienung	47
6.2.2	Software Optionen	50
6.2.3	Umgehungslösungen für einige Tester	52
6.2.4	Umgehungslösungen für einige IDEs	52
6.2.5	Benutzerschnittstelle	53
6.2.6	Energieverwaltung	55
6.2.7	Messeinstellungen und Offsets	56
6.2.8	F & E - gedacht für Firmware-Entwickler	58
6.2.9	Busse	59
6.3	Config_328.h	60
6.3.1	LCD-Module	60
6.3.2	Port- und Pinbelegung	60
6.3.3	Busse	62
6.4	Config_644.h	63
6.4.1	LCD-Module	63
6.4.2	Port- und Pinbelegung	63
6.4.3	Busse	66
6.5	Config_1280.h	67
6.5.1	LCD-Module	67
6.5.2	Port- und Pinbelegung	67
6.5.3	Busse	70
7	Sammlung von Einstellungen	71
7.1	Arduino Nano, Uno oder Mega 2560	71
7.2	DIY Kit „AY-AT“ auch GM328A	71
7.3	BSide ESR02	73
7.4	Fish8840 TFT	74
7.5	GM328 !nicht GM328A!	74
7.6	Hiland M644	75
7.7	M12864 DIY Transistor Tester	76
7.8	MK-328	77
7.9	T3/T4	77
7.10	Multifunktionstester TC-1 und Familie (T7) mit ATmega324/644	79
7.11	Multifunktionstester T7 mit ATmega328	81
7.12	Arduino MEGA Anschluss	82
8	Programmieren des Component Testers	83
8.1	Konfigurieren des Component Testers	83
8.2	Programmierung des Mikrocontrollers	83
8.3	Betrieb System Linux	83
8.4	Benutzung unter Linux	84
8.5	Programm Pakete installieren	84
8.6	Download der Quellen	84
8.7	Benutzung der Schnittstellen	84
8.8	Gruppen Mitgliedschaft	85
8.9	Arbeitsumgebung	85
8.10	Firmware erzeugen und übertragen	86
9	Der Version Vorschritt	87
9.1	v1.47m 2022-07	87
9.2	v1.46m 2022-04	87

9.3	v1.45m	2021-12	88
9.4	v1.44m	2021-08	88
9.5	v1.43m	2021-03	89
9.6	v1.42m	2020-12	89
9.7	v1.41m	2020-09	89
9.8	v1.40m	2020-06	90
9.9	v1.39m	2020-03	91
9.10	v1.38m	2019-12	91
9.11	v1.37m	2019-09	92
9.12	v1.36m	2019-05	92
9.13	v1.35m	2019-02	92
9.14	v1.34m	2018-10	93
9.15	v1.33m	2018-05	93
9.16	v1.32m	2018-02	93
9.17	v1.31m	2017-12	94
9.18	v1.30m	2017-10	94
9.19	v1.29m	2017-07	94
9.20	v1.28m	2017-04	94
9.21	v1.27m	2017-02	94
9.22	v1.26m	2016-12	95
9.23	v1.25m	2016-09	95
9.24	v1.24m	2016-08	95
9.25	v1.23m	2016-07	95
9.26	v1.22m	2016-03	96
9.27	v1.21m	2016-01	96
9.28	v1.20m	2015-12	96
9.29	v1.19m	2015-11	96
9.30	v1.18m	2015-07	96
9.31	v1.17m	2015-02	97
9.32	v1.16m	2014-09	97
9.33	v1.15m	2014-09	97
9.34	v1.14m	2014-08	97
9.35	v1.13m	2014-07	97
9.36	v1.12m	2014-03	97
9.37	v1.11m	2014-03	97
9.38	v1.10m	2013-10	98
9.39	v1.09m	2013-07	98
9.40	v1.08m	2013-07	98
9.41	v1.07m	2013-06	98
9.42	v1.06m	2013-03	99
9.43	v1.05m	2012-11	99
9.44	v1.04m	2012-11	99
9.45	v1.03m	2012-11	99
9.46	v1.02m	2012-11	99
9.47	v1.01m	2012-10	99
9.48	v1.00m	2012-09	99
9.49	v0.99m	2012-09	100

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 Hinweise Lizenz

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

1.4. Unterschiede zur k-Firmware

Karl-Heinz hat eine richtig gute Dokumentation für den Tester geschrieben. [16]
Unbedingt lesen! Daher zähle ich nur die Hauptunterschiede zur k-Firmware auf:

- Benutzerschnittstelle
 - + Keine Panik! ;-)
 - + Touch Screen
 - + Automatisierung (Fernsteuerkommandos)
 - + zwei Sätze Abgleichwerte
- Adaptive Entladefunktion - Widerstandsmessung
 - + zusätzliche Methode für Widerstände $<10\ \Omega$ (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
 - + Erkennung von Schottky-Transistoren
- 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) Temperatur Sensor
- DHTxx Temperatur und Luftfeuchte Sensoren
- Klingeltester (LOPT/FBT-Tester)
- Logiktester
- Durchgangsprüfer
- MAX6675/MAX31855 Thermoelement-Konverter
- Ereigniszähler
- Strukturierter Quellcode
- einfache Rahmenwerke für Anzeigen und Datenbusse
- 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 ich 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:

[5] <https://github.com/madires/Transistortester-Warehouse>

[6] <https://github.com/kubi48/TransistorTester-source/tree/master/Markus>

[7] <https://github.com/Mikrocontroller-net/transistortester/tree/master/Software/Markus>

1.6. Unterstützte Hardware

Die Firmware läuft auf allen Testern, die zur Grundschialtung in Karl-Heinz's Dokumentation kompatibel sind und eine der folgenden MCUs nutzen:

- ATmega 328 config_328.h
- ATmega 324/644/1284 config_644.h
- ATmega 640/1280/2560 config_1280.h

Anpassungen an unterschiedliche Pin-Belegungen sind möglich. Als Anzeige kann man Text- oder Grafikdisplays (monochrom oder farbig) nutzen.

Für die unterstützten Controller bitte den Kapitel 2.6 „Anzeige-Module“ ab Seite 12 konsultieren.

1.7. Unterstützte Hardware-Optionen

1.7.1. Bedienung

- Drehimpulsgeber
- zusätzliche Tasten (mehr/weniger)
- Touch-Screen
- serielle Schnittstelle (TTL, RS232, USB-Seriell-Adapter)

1.7.2. Verbesserungen

- externe 2,5V Spannungsreferenz
- fester Abgleichskondensator
- Schutz-Relais zum Entladen von Kondensatoren

1.7.3. Zusätzliche Tests und Messungen

- Zener-Test / Messung einer externen Spannung <50V
- einfacher Frequenzzähler
- erweiterter Frequenzzähler
mit Vorteiler und Quarzoszillatoren für niedrige und hohe Frequenzen
- fester IR-RC-Empfänger
- LC-Meter
- Klingeltester (LOPT/FBT-Tester)
- Logiktester
- MAX6675/MAX31855 Thermoelement-Konverter

1.8. Firmware übersetzen

Als erstes solltest Du das **Makefile** siehe Kapitel 6.1 ab Seite 45 editieren und hier MCU-Modell, Frequenz, Oszillator-Typ und die Programmieradaptoreinstellungen anpassen.

- In der **config.h** siehe Kapitel 6.2 ab Seite 47 werden Bedienung und Menü Optionen eingestellt. Hier wähle Hardware- und Software-Optionen, die Sprache für die Bedienung, usw.

- Schließlich in der **config_<MCU>.h** die globale Konfiguration, wie Display und Zuordnung des Pins, die sich je nach verwendetem MCU unterscheiden. Details im Kapitel 6.5 ab Seite 67. Folgende config<MCU>.h Dateien gelten:

- ATmega 328 config_328.h
- ATmega 324/644/1284 config_644.h
- ATmega 640/1280/2560 config_1280.h

- Im Kapitel 7 „Clones“ ab Seite 71, findest Du Einstellungen zu verschiedenen Testerversionen bzw. Clonen. Wenn Dein Tester nicht dabei ist, schicke bitte die Einstellungen per EMail an den Autor [8] um anderen Benutzern damit zu helfen.

Alle Einstellungen und Werte sind in den Dateien selber erklärt.

Wie man die Firmware übersetzt siehst du dann im Abschnitt 8.10 auf Seite 86.

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
- LC-Meter
- 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
- Funktionen zum Beobachten von R/C/L Werten
- 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)
- OneWire-Scan
- Erkennung von UJTs
- Test für Kondensatorleckstrom
- DS18B20 Temperatursensor
- DHTxx Temperatur und Luftfeuchte Sensoren
- 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.

2.3. Verfügbare Sprachen

- Dänisch - von glenndk@mikrocontroller.net (benötigt kleine Änderungen im Font)
- Deutsch
- Englisch
- Italienisch - von Gino_09@EEVblog
- Polnisch - von Szpila
- Rumänisch - von Dumidan@EEVblog
- Russisch - von indman@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. (Standart: Punkt).

2.4. Konfiguration

Deine MCU wird in der Datei Makefile eingestellt. . .

Kapitel 6.1 ab Seite 45.

Deine gewünschte Optionen in der Datei config.h. . .

Kapitel 6.2 ab Seite 47.

Für die MCU-spezifischen Einstellungen, wie Pin-Zuordnungen und Display, stellst du, je nach MCU in der Datei config_MCU:

- ATmega 328 config_328.h
 - ATmega 324/644/1284 config_644.h
 - ATmega 640/1280/2560 config_1280.h
- nach der Vorlage . . .

Kapitel 6.5 ab Seite 67.

Nach dem editieren sollst Du die Firmware erzeugen nach Abschnitt 8.10 ab Seite 86.

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

2.5. Busse & Schnittstellen

2.5.1. 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 67 direkt konfiguriert werden. Alternativ kannst Du auch I2C bzw. SPI in config.h Seite 59 aktivieren und Ports & Pins in config-<MCU>.h festlegen (schaue nach I2C_PORT Seite 70 bzw. SPI_PORT) auf Seite 70.

Falls Du Bit-Bang-SPI plus Lese-Modus (SPI_RW) auswählst, bitte darauf achten auch SPI_PIN und SPI_MISO zu setzen. Beispiele dazu findest Du im Abschnitt zu SPI in config-<MCU>.h.

<Wenn mehrere ICs mit dem SPI-Bus verbunden werden, muss jeder IC über ein <eigenes /CS-Signal gesteuert werden. Nur im Fall eines einzigen IC am SPI-Bus <darf dessen /CS-Signal fest auf Masse gelegt werden.

2.5.2. 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 59 aktiviert siehe Abschnitt “Busses“, und Port & Pins sind in config-<MCU>.h definiert (schaue nach SERIAL_PORT) Seite 70.

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

2.5.3. One Wire Ein weiterer unterstützter Bus ist OneWire, welcher entweder die Test-Pins (ONEWIRE_PROBES) oder einen festen MCU-Pin (ONEWIRE_IO_PIN) Seite 59 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: DQ (Daten)

Probe #3: Vcc (Strom durch 680 Ω Widerstand begrenzt)

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

Funktionen, die genau einen Client am Bus erfordern, können optional den ROM-Code des Clients ausgeben (ONEWIRE_READ_ROM). Im Fall eines CRC-Fehlers oder wenn mehrere Clients am Bus hängen, ist die Ausgabe „-“. Ist der komplette ROM-Code Null, besteht ein Lesefehler. Ansonsten zeigt der erste Teil des ROM-Codes die Produktfamilie und der zweite Teil die Seriennummer.

2.6. Anzeige-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.14
- ILI9341/ILI9342	(grafisches Farb-Display 240x320 oder 320x240)	S.14
- ILI9481	(grafisches Farb-Display 320x480, teilweise ungetestet)	S.15
- ILI9486	(grafisches Farb-Display 320x480, teilweise getestet)	S.15
- ILI9488	(grafisches Farb-Display 320x480, teilweise getestet)	S.15
- PCD8544	(grafisches Display 84x48)	S.16
- PCF8814	(grafisches Display 96x65)	S.16
- SH1106	(grafisches Display 128x64)	S.16
- SSD1306	(grafisches Display 128x64)	S.17
- ST7036	(text Display, 3 Zeilen mit je 16 Zeichen, ungetestet)	S.17
- ST7565R	(grafisches Display 128x64)	S.18
- ST7735	(grafisches Farb-Display 128x160)	S.18
- ST7920	(grafisches Display bis zu 256x64)	S.18
- STE2007/HX1230	(grafisches Display 96x68)	S.20
- VT100 Terminal	(über serielle Schnittstelle)	S.19

Beachte die Versorgungsspannung und die Logikpegel des Anzeige-Moduls! Benutze Pegelumsetzer, sofern notwendig. Ein einfacher Pegelumsetzer mit Widerständen in Reihe unter Ausnutzung der internen Begrenzungsdioden des Anzeige-Controllers kann funktionieren, aber nur für langsame Busse, wie z.B. Bit-Bang SPI. Daher empfehle ich, Pegelumsetzer-ICs zu verwenden.

Falls Du IO-Pins einsparen möchtest, kannst Du bei fast allen Anzeige-Modulen /CS und /RES per Pullup/Pulldown-Widerstände fest verdrahten und die entsprechenden IO-Pins in der Konfiguration (config_<MCU>.h) ab Seite 67 auskommentieren, solange das Anzeige-Modul allein am Bus hängt.

Wenn das Display trotz korrekter Beschaltung nichts anzeigt, versuche den Kontrast zu ändern (config_<MCU>.h) siehe Seite 67.

Grafik-Anzeigen haben meistens Einstellmöglichkeiten für die Orientierung der Ausgabe, z.B. Bild um 90° rotieren, Bild horizontal spiegeln oder auch vertikal. Damit läßt sich das Bild an den jeweiligen Tester anpassen. Für grafische Farbanzeigen gibt es zusätzliche Einstellungen. Im normalen Farbmodus werden mehrere Farben verwendet, die mittels der Datei colors.h änderbar sind. Durch Auskommentieren von LCD_COLOR wird der Zwei-Farb-Modus aktiviert. Die Schriftfarbe ist dann COLOR_PEN und die Hintergrundfarbe COLOR_BACKGROUND. Sollten die RGB-Grundfarben Rot und Blau vertauscht sein, kannst Du per LCD_BGR die entsprechenden Farbkanäle tauschen. Bei manchen Anzeigen ist die Reihenfolge der RGB-Unterpixel verdreht und der Controller weiss nichts davon.

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

2.6.1. Zeichensätze und Symbole

Die Anzeigenmodulkonfiguration beinhaltet auch einen Zeichensatz und optional einen Symbolsatz. Passende Zeichensätze und Symbole sind jeweils unter den Einstellungen für jedes Anzeigenmodul in config_<MCU>.h gelistet. Für grafische Anzeigen kannst Du meistens aus mehreren Zeichensätzen wählen. Die Symbole werden über den Schalter SSW_SYMBOLS in config.h aktiviert.

Wenn Dir der Stil der alten Symbole besser gefällt, kannst Du diese weiterhin verwenden, indem Du die Symboleinstellung auf SYMBOLS_<size>_OLD_<format>.h änderst.

Zu Testzwecken kannst Du eine Menüfunktion zur Ausgabe aller Zeichen im Zeichensatz (SW_FONT_TEST) oder aller Bauteilsymbole (SW_SYMBOL_TEST) aktivieren.

2.6.2. HD44780 wird im 4-Bit-Modus betrieben.

Modul	config-<MCU>.h	328 75,6%	644 37,9%	1280 13,1%	Hinweis
DB4	LCD_DB4	PD0	PB4	PB0	
DB5	LCD_DB5	PD1	PB5	PB1	
DB6	LCD_DB6	PD2	PB6	PB2	
DB7	LCD_DB7	PD3	PB7	PB3	
RS	LCD_RS	PD4	PB2	PB4	
R/W	LCD_RW	Gnd	Gnd	Gnd	
E	LCD_EN1	PD5	PB3	PB5	

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	328 76,5%	644 38,2%	1280 9,8%	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.

2.6.3. ILI9163 wird mittels 4-Draht-SPI gesteuert.

Modul	config-<MCU>.h	328 87,4%	644 43,8%	1280 11,2%	Hinweis
/RESX	LCD_RES	PD4	PB2	PB4	optional
/CSK	LCD_CS	PD5	PB4	PB5	optional
D/CX	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI Clock
SDIO	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Tabelle 2.3. Pinbelegung für ILI9163.

Für eine korrekte Anzeige mußt Du evtl. mit den Einstellungen X/Y-Flip experimentieren. Bei Bedarf kannst Du auch die X-Richtung verschieben. 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.

2.6.4. ILI9341/ILI9342 wird mittels 4-Line-SPI oder 8-Bit Parallel gesteuert.

Modul	config-<MCU>.h	328 87,4%	644 51,7%	1280 13,2%	Hinweis
RES	LCD_RES	PD4	PB2	PB4	optional
CS	LCD_CS	PD5	PB4	PB5	optional
DC	LCD_DC	PD3	PB3	PB7	
SCK	LCD_SCK	PD2	PB7	PB1	SPI Clock
SDI	LCD_SDI	PD1	PB5	PB2	SPI MOSI
SDO	LCD_SDO	-	PB6	PB3	nur ILI9341*

* noch nicht benutzt

Tabelle 2.4. 4-Line-SPI Pinbelegung für ILI9341/ILI9342.

Modul	config-<MCU>.h	328	644	1280 13,1%	Hinweis
	LCD_PORT			PORTB	
/RES X	LCD_RES			PB4	optional
/CSX	LCD_CS			PB5	optional
D/CX	LCD_DC			PB7	
WRX	LCD_WR			PB0	
RDX	LCD_RD			PB6	
	LCD_PORT2			PORTL	
D0	LCD_DB0			PL0	LCD_PORT2 Pin #0
D1	LCD_DB1			PL1	LCD_PORT2 Pin #1
D2	LCD_DB2			PL2	LCD_PORT2 Pin #2
D3	LCD_DB3			PL3	LCD_PORT2 Pin #3
D4	LCD_DB4			PL4	LCD_PORT2 Pin #4
D5	LCD_DB5			PL5	LCD_PORT2 Pin #5
D6	LCD_DB6			PL6	LCD_PORT2 Pin #6
D7	LCD_DB7			PL7	LCD_PORT2 Pin #7

Tabelle 2.5. 8-Bit-Parallel Pinbelegung für ILI9341/ILI9342.

Für eine korrekte Anzeige mußt Du evtl. mit den Einstellungen X/Y-Flip und Rotate experimentieren. Und nicht vergessen, die X und Y-Auflösung korrekt zu setzen (ILI9341 ist 240x320 und ILI9342 ist 320x240). Manche Anzeigemodule haben den erweiterten Befehlssatz des ILI9341 deaktiviert (EXTC-Pin liegt auf Masse), und die Ausgabe ist dann oft nur schemenhaft. Das läßt sich per LCD_EXT_CMD_OFF korrigieren.

Aufgrund der hohen Pixelzahl ist die Ausgabe per SPI etwas langsam. Das Löschen der kompletten Anzeige dauert etwa 3 Sekunden bei Bit-Bang-SPI und 8MHz MCU-Takt. Besser Hardware-SPI oder den Parallel-Bus nutzen.

2.6.5. ILI9481 wird 8-Bit-Parallel-Bus, 16-Bit-Parallel-Bus oder 4-Line SPI gesteuert.

Modul	config-<MCU>.h	328	644 51,6%	1280 13,2%	Hinweis
	LCD_PORT			PORTB	
/RESX	LCD_RES		PC4	PB4	optional
/CSX	LCD_CS		PC3	PB5	optional
D/CX	LCD_DC		PC2	PB7	
WRX	LCD_WR		PC1	PB0	
RDX	LCD_RD		PC0	PB6	optional
	LCD_PORT2			PORTL	
D0	LCD_DB0		PB0	PL0	LCD_PORT2 Pin #0
D1	LCD_DB1		PB1	PL1	LCD_PORT2 Pin #1
D2	LCD_DB2		PB2	PL2	LCD_PORT2 Pin #2
D3	LCD_DB3		PB3	PL3	LCD_PORT2 Pin #3
D4	LCD_DB4		PB4	PL4	LCD_PORT2 Pin #4
D5	LCD_DB5		PB5	PL5	LCD_PORT2 Pin #5
D6	LCD_DB6		PB6	PL6	LCD_PORT2 Pin #6
D7	LCD_DB7		PB7	PL7	LCD_PORT2 Pin #7

Tabelle 2.6. Pinbelegung für ILI9481 8-Bit-Parallel.

Modul	config-<MCU>.h	328	644	1280	Hinweis
D8	LCD_DB8			PC0	LCD_PORT3 Pin #0
D9	LCD_DB9			PC1	LCD_PORT3 Pin #1
D10	LCD_DB10			PC2	LCD_PORT3 Pin #2
D11	LCD_DB11			PC3	LCD_PORT3 Pin #3
D12	LCD_DB12			PC4	LCD_PORT3 Pin #4
D13	LCD_DB13			PC5	LCD_PORT3 Pin #5
D14	LCD_DB14			PC6	LCD_PORT3 Pin #6
D15	LCD_DB15			PC7	LCD_PORT3 Pin #7

Tabelle 2.7. Zusätzliche Pinbelegung für ILI9481 16-Bit-Parallel.

Modul	config-<MCU>.h	328 87,4%	644 43,8%	1280 11,2%	Hinweis
RES	LCD_RES	PD4	PB2	PB4	optional
CS	LCD_CS	PD5	PB5	PB5	optional
D/C	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI Clock
SDA	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Tabelle 2.8. Nicht empfohlene 4-Draht-SPI Pinbelegung für ILI9481.

Wegen der hohen Auflösung der Anzeige und dem RGB666-Farbschema (3 Bytes pro Pixel) ist SPI recht langsam, selbst für Hardware-SPI und einen 16MHz-MCU-Takt. Daher rate ich von der Nutzung der SPI-Schnittstelle ab.

Üblicherweise muß die Anzeige für die korrekte Darstellung rotiert werden (LCD_ROTATE). Falls nötig, kannst Du die auch X und/oder Y umdrehen.

2.6.6. ILI9486 wird 8-Bit-Parallel-Bus, 16-Bit-Parallel-Bus oder 4-Line SPI gesteuert und er benutzt die gleiche Pin-Belegung wie der ILI9481.

2.6.7. ILI9488 wird 8-Bit-Parallel-Bus, 16-Bit-Parallel-Bus oder 4-Line SPI gesteuert und er benutzt die gleiche Pin-Belegung wie der ILI9481.

2.6.8. PCD8544 wird mittels SPI gesteuert.

Modul	config-<MCU>.h	328 82,5%	644 39,1%	1280 10,0%	Hinweis
/RES	LCD_RES	PD4	PB2	PB4	optional
/SCE	LCD_SCE	PD5	PB4	PB5	optional
D/C	LCD_DC	PD3	PB3	PB7	
SCLK	LCD_SCLK	PD2	PB7	PB1	SPI Clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Tabelle 2.9. 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.

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

Modul	config-<MCU>.h	328 82,6%	644 41,3%	1280 10,6%	Hinweis
/RES	LCD_RES	PD4	PB2	PB4	
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK	LCD_SCLK	PD2	PB7	PB1	SPI Clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Tabelle 2.10. 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.

2.6.10. SH1106 (teilweise ungetestet) 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	328 83,0%	644 41,6%	1280 10,7%	Hinweis
/RES	LCD_RESET	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
A0	LCD_A0	PD3	PB3	PB7	
SCL (D0)	LCD_SCL	PD2	PB7	PB1	SPI Clock
SI (D1)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Tabelle 2.11. Pinbelegung für SH1106 4-Line-SPI.

Modul	config-<MCU>.h	328 83,2%	644 41,6%	1280 10,7%	Hinweis
/RES	LCD_RESET	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
A0		Gnd			
SCL (D0)	LCD_SCL	PD2	PB7	PB1	SPI Clock
SI (D1)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Tabelle 2.12. Pinbelegung für SH1106 3-Line-SPI (nur Bit-Bang).

Modul	config-<MCU>.h	328 84,0%	644 41,9%	1280 10,7%	Hinweis
/RES	LCD_RESET	PD4	PB2	PB0	optional
/CS		Gnd			
SCL (D0)	I2C_SCL	PD1	PC0	PD0	
SDA (D1)	I2C_SDA	PD0	PC1	PD1	
SA0	Gnd (0x3c)				3.3V (0x3d)

Tabelle 2.13. Pinbelegung für SH1106 I2C.

Mit den X/Y-Flip Einstellungen kannst Du die Orientierung der Anzeige anpassen, wenn notwendig. Bei vielen SH1106-Display-Modulen muss der X-Offset auf 2 gesetzt werden.

2.6.11. 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	328 83,1%	644 41,6%	1280 10,7%	Hinweis
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
DC	LCD_DC	PD3	PB3	PB7	
SCLK (D0)	LCD_SCLK	PD2	PB7	PB1	SPI Clock
SDIN (D1)	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Tabelle 2.14. Pinbelegung für SSD1306 4-Line-SPI.

Modul	config-<MCU>.h	328 83,2%	644 41,6%	1280 10,7%	Hinweis
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK (D0)	LCD_SCLK	PD2	PB7	PB1	SPI Clock
SDIN (D1)	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Tabelle 2.15. Pinbelegung für SSD1306 3-Line-SPI (nur Bit-Bang).

Modul	config-<MCU>.h	328 84,1%	644 42,0%	1280 10,7%	Hinweis
/RES	LCD_RES	PD4	PB2	PB0	optional
/SCL (D0)	I2C_SCL	PD1	PC0	PD0	optional
SDA (D1&2)	I2C_SDA	PD1	PC1	PB0	
SA0 (D/C)	Gnd (0x3c)				3.3V (0x3d)

Tabelle 2.16. Pinbelegung für SSD1306 I2C.

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

2.6.12. ST7036 (ungetestet) wird per 4-bit-Parallel oder 4-Draht-SPI angesprochen.

Modul	config-<MCU>.h	328 76,2%	644 38,1%	1280 9,8%	Hinweis
DB4	LCD_DB4	PD0	PB4	PB0	
DB5	LCD_DB5	PD1	PB5	PB1	
DB6	LCD_DB6	PD2	PB6	PB2	
DB7	LCD_DB7	PD3	PB7	PB3	
RS	LCD_RS	PD4	PB2	PB4	
R/W	LCD_RW	Gnd	???	???	optional
E	LCD_EN	PD5	PB3	PB5	
XRESET		Vcc	???	???	optional

Tabelle 2.17. Pinbelegung für den 4-bit-Parallel-Schnittstelle von ST7036.

Modul	config-<MCU>.h	328 76,1%	644 38,1%	1280 9,8%	Hinweis
XRESET	LCD_RESET	PD4	PB2	PB4	optional
CSB	LCD_CS	PD5	PB4	PB5	optional
RS	LCD_RS	PD3	PB3	PB7	
SCL (DB6)	LCD_SCL	PD2	PB7	PB1	SPI Clock
SI (DB7)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Tabelle 2.18. 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.

2.6.13. ST7565R wird mittels 4/5-Line-SPI gesteuert.

Modul	config-<MCU>.h	328 83,2%	644 41,7%	1280 10,7%	Hinweis
/RES	LCD_RESET	PD0	PB2	PB4	optional
/CS1	LCD_CS	PD5	PB4	PB5	optional
A0	LCD_A0	PD1	PB3	PB7	
SCL (DB6)	LCD_SCL	PD2	PB7	PB1	SPI Clock
SI (DB7)	LCD_SI	PD3	PB5	PB2	SPI MOSI

Tabelle 2.19. 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.

2.6.14. ST7735 wird mittels 4-Draht-SPI gesteuert.

Modul	config-<MCU>.h	328 95,7%	644 47,9%	1280 12,2%	Hinweis
/RESX	LCD_RES	PD4	PB2	PB4	optional
/CSK	LCD_CS	PD5	PB4	PB5	optional
D/CX	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI Clock
SDIO	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Tabelle 2.20. 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.

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

Modul	config-<MCU>.h	328 83,5%	644 xx,x%	1280 xx,x%	Hinweis
/XRESET	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK	LCD_SCLK	PD2	PB7	PB1	SPI Clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Tabelle 2.21. Pinbelegung für SPI mit ST7920.

Modul	config-<MCU>.h	328 83,8%	644 41,8%	1280 10,7%	Hinweis
/XRESET	LCD_RESET	Vcc	Vcc	Vcc	
E	LCD_EN	PD5	PB3	PB5	
RS	LCD_RS	PD4	PB2	PB4	
RW	LCD_RW	Gnd	Gnd	Gnd	
D4	LCD_DB4	PD0	PB4	PB0	
D5	LCD_DB5	PD1	PB5	PB1	
D6	LCD_DB6	PD2	PB6	PB2	
D7	LCD_DB67	PD3	PB7	PB3	

Tabelle 2.22. Pinbelegung für den 4-bit-Parallel-Schnittstelle von 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.

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

Modul	config-<MCU>.h	328 82,5%	644 41,3%	1280 10,6%	Hinweis
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK	LCD_SCLK	PD2	PB7	PB1	
SDIN	LCD_SDIN	PD1	PB5	PB2	

Tabelle 2.23. Pinbelegung für SPI mit ST7920.

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

2.6.17. VT100 Terminal Der VT100-Treiber ersetzt ein LCD-Modul, und die komplette Ausgabe erfolgt über ein serielles 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.

2.7. 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.

2.7.1. 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. **Halbdruck**, 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).

2.7.2. 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 47

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.

2.7.3. Mehr/Weniger-Tasten (Hardware-Option) Wenn Dir Tasten lieber als ein Drehencoder sind, dann kannst Du alternativ auch zwei Tasten nutzen. Siehe Seite 47. 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.

2.7.4. 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, sie kann aber z.B. **nicht** die Zener-Test-Option mit Strom versorgen.

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 Hauptmenü starten. Die Prozedur ist recht einfach. Wenn Du ein Sternchen (gelbes * bei Farb-LCDs) siehst, drauf drücken. Danach löscht der Tester das Sternchen und zeigt die native x/y-Position an. Der erste Abgleichspunkt ist oben rechts, und der zweite unten links. Je nach Ergebnis wiederholt der Tester den Abgleich bis zu dreimal. Den Abgleich kannst Du aber jeder Zeit mit der Test-Taste abbrechen.

Wenn Du Probleme mit dem Abgleich, wie z.B. falsche x/y-Positionen oder einen Fehler nach der ersten Abgleichsrunde, hast, überprüfe bitte die Orientierung des Touch-Screen zum LCD-Modul. Der Treiber hat Schalter zum Verdrehen und Vertauschen der Orientierung. Oben links wird als Nullpunkt angenommen.

Hinweise zu den benötigten Einstellungen je nach x/y-Werten:

* Oben rechts wird als Nullpunkt angenommen.		
x	y	Einstellung
niedrig	niedrig	TOUCH_FLIP_X
niedrig	hoch	TOUCH_FLIP_X & TOUCH_FLIP_Y
hoch	niedrig	keine
hoch	hoch	TOUCH_FLIP_Y
Nach einem erfolgreichen Abgleich nicht vergessen, die Offset-Werte zu speichern. . . (Hauptmenü: Speichern).		

Tabelle 2.24. Einstellungen je nach x/y-Werten.

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, und config_1280.h, da der 328 zu wenig unbenutzte IO-Pins hat).

2.7.5. Benutzerschnittstelle Es gibt mehrere Optionen, die Display-Ausgabe für Anzeigen schicker oder verständlicher zu gestalten.

Für Anzeigen mit mehr als zwei Textzeilen:

- Anzeige von Tastenhinweisen statt Cursor, sofern verfügbar (UI_KEY_HINTS)

Für graphische Anzeigen:

- alle vorherigen Optionen
- schickes Pinout mit Symbolen für dreibeinige Halbleiter (SW_SYMBOLS), nicht für Anzeigen mit niedriger Auflösung
- rechte Testpins beim schicken Pinout unter/oberhalb vom Symbol anzeigen (UI_PINOUT_ALT)
- keine Ausgabe von Text-Pinout für dreibeinige Halbleiter (UI_NO_TEXTPINOUT, wenn SW_SYMBOLS aktiviert)
- Anzeige von Testpins in invertierter Darstellung (UI_PROBE_REVERSED)
- Anzeige eines Fragezeichensymbols bei fehlgeschlagenem Suchlauf (UI_QUESTION_MARK)
- Anzeige eines Zenerdiodensymbols bei allen Zener-Tests (UI_ZENER_DIODE)
- Anzeige eines Schwingquartzsymbols für LF/HF-Modi im erweiterten Frequenzzähler (UI_QUARTZ_CRYSTAL)

Für farbige Grafikanzeigen:

- alle vorherigen Optionen
- Farbcodierung für Testpins (UI_PROBE_COLORS)
- Titel mit eigener Farbe (UI_COLORED_TITLES)
- Cursor und Tastenhinweise mit eigener Farbe (UI_COLORED_CURSOR)
- Farbcode für Widerstände (SW_R_E24_5_CC, SW_R_E24_1_CC und SW_R_E96_CC)

2.7.6. Summer/Pieper (Hardware-Option) Falls Du auch ein akustisches Signal haben möchtest, kannst Du einen aktiven Summer (mit integriertem Oszillator) verwenden. Der Summer wird über einen festen Port-Pin kontrolliert (siehe BUZZER_CTRL in config_<MCU>.h) und kann direkt (< 20mA) bzw. über einen Transistor (NPN oder n-Kanal-MOSFET) angesteuert werden.

Im Augenblick wird der Summer nur vom Durchgangstest benutzt. Weitere Anwendungsfälle werden in der Zukunft folgen.




2.7.7. 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 59 in config.h).

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

Konvertierungstabelle:

	> <
	
	□
Ω	R
μ	u

Hinweise:

- 9600 8N1
- Newline ist <CR><LF>

2.7.8. 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 53 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.

2.7.9. 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 53 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“ 5 ab Seite 41.

2.7.10. VT100-Ausgabe Anstatt einer LCD-Anzeige kann die komplette Ausgabe über ein VT100-Terminal erfolgen (siehe VT100 im Abschnitt LCD-Module 2.6 auf Seite 19). 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 Abgleichwerten entdeckt (Problem mit dem EEPROM), zeigt er einen Prüfsummenfehler an und benutzt stattdessen die Firmware-Standardwerte.

Bei einem Tester mit einem manuellen Ein/Aus-Schalter statt dem Soft-Latch- Schalter der Referenzschaltung, bitte `POWER_SWITCH_MANUAL` in `config.c` aktivieren. In diesem Fall kann sich der Tester leider nicht selbst abschalten.

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 50 und `CYCLE_MAX` Seite 53 in `config.h` geändert werden.

Für den Auto-Hold-Modus gibt es eine optionale automatische Abschaltung (`POWER_OFF_TIMEOUT`) Seite 54, 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 55.

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. Ver-

polungsschutzdiode und PNP-Transistor zum Schalten der Stromversorgung.

3.0.4. Ausschalten Während das Ergebnis der letzten Bauteilsuche 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.

3.1. 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. Normalerweise springt der Tester nach dem Ausführen einer Funktion wieder zur Bauteilsuche zurück. Wenn du lieber im Menü bleiben möchtest, kannst du UI_MAINMENU_AUTOEXIT abschalten. Das Menü wird dann über ein explizites „Exit“ beendet.

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.

3.1.1. 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. Für die optionale Ausgabe der Pulsdauer zusätzlich PWM_SHOW_DURATION aktivieren.

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 Tastendruck 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 Tastendrücke 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.

Wenn Dein Tester nur den 10:1 Spannungsteiler ohne Boost-Konverter zum Messen einer externen Spannung hat, oder der Boost-Konverter ständig läuft, kannst Du den alternativen Modus (ZENER_UNSWITCHED) aktivieren, welcher die Spannung periodisch ohne Drücken der Test-Taste misst. Zeigt der Tester zwischen den Messungen unten rechts einen Cursor an, kannst Du den Zenertest über zweimaliges Drücken der Test-Taste beenden. Als weitere Option kannst Du den Zenertest auch während der normalen Bauteilesuche automatisch laufen lassen (HW_PROBE_ZENER). Wird kein Bauteil an den normalen Testpins gefunden, prüft der Tester die Spannung an den Zener-Testpins.

Für den Fall, dass Dein Tester einen Nicht-Standard-Spannungsteiler hat (nicht 10:1), aktiviere ZENER_DIVIDER_CUSTOM und setze die Widerstandswerte (ZENER_R1 und ZENER_R2).

Beschaltung für Zenerdiode:

Pin +: Kathode

Pin -: Anode

3.1.6. Logiktester (Hardware-Option) prüft den Zustand von Logiksignalen mittels einem festen ADC- Pin plus Spannungsteiler. Der Spannungsteiler sollte 4:1 sein, damit Spannungen bis 20V möglich sind (15V CMOS). Der ADC-Pin ist TP_LOGIC (in config_<MCU>.h) und der Spannungsteiler wird in config.h gesetzt (LOGIC_PROBE_R1 und LOGIC_PROBE_R2). Es dürfte keine schlechte Idee sei, den Eingangsschutz zu verbessern, z.B. mit einem Paar Schutzdioden. Nach dem Starten liest der Tester automatisch die Spannung, vergleicht die Spannung mit den Logikpegel-Schwellwerten und zeigt dann den Logikzustand plus die Spannung an:

0 für Low

1 für High

Z für undefiniert/HiZ

Zum Setzen der Logikfamilie bzw. Vcc/Vdd und den Logikpegel-Schwellwerten gibt es ein einfaches Menü. Mit der Testtaste wählst Du die gewünschte Einstellung aus (markiert mit einem Sternchen). Dann benutze den Drehencoder zum Ändern der Einstellung. Beim Ändern der Logikfamilie bzw. von Vcc/Vdd werden die Schwellwerte automatisch angepasst. Danach kannst Du die Schwellwerte bei Bedarf ändern. Im Fall einer unüblichen Vcc/Vdd-Spannung bitte die nächst höhere auswählen und dann die Schwellwerte entsprechend ändern. Die Standardwerte für Vcc/Vdd sind:

- TTL : 5V
- CMOS : 3.3V, 5V, 9V, 12V, 15V

Und wie gewohnt, zwei kurze Tastendrucke beenden den Logiktester.

3.1.7. Durchgangsprüfer (Hardware-Option) Diese Funktion prüft Schaltungen auf Durchgang und gibt Dir ein akustisches Signal über einen Summer/Pieper, während auch die Spannung zwischen den Testpins ausgegeben wird. Der Teststrom wird über Rl (680 Ω) auf etwa 7 mA begrenzt. Die Schwellwerte für den Summer sind:

- | | | |
|---------|----|---------------------------|
| < 100 | mV | kontinuierlicher Ton |
| 100-700 | mV | wiederholender kurzer Ton |
| > 700 | mV | kein Ton |

Der kurze Ton zeigt einen möglichen Halbleiterübergang an. Bei einer Unterbrechung oder einem sehr hohen Widerstand liegt die Spannung nahe 5 V.

Nach dem Starten vom Durchgangsprüfer zeigt der Tester die Beschaltung der Testpins für ein paar Sekunden an (kann mit Testtaste übersprungen werden). Und zwei kurze Tastendrucke beenden den Test.

Beschaltung der Testpins:

- | | |
|---------|---|
| Pin #1: | Vcc (680 Ω Widerstand zur Strombegrenzung) |
| Pin #3: | Masse |

3.1.8. ESR-Tool Das ESR-Tool 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 |

3.1.9. 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 (680 Ω) und schaltet auf Rh (470k Ω) 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 |

3.1.10. R/C/L-Monitore Die Monitorfunktionen messen ständig passive Bauteile an den Testpins #1 und #3. Nach dem Starten zeigt der Tester die Beschaltung der Testpins für ein paar Sekunden an, was durch die Testtaste übersprungen werden kann.

Zwischen den Messungen gibt es jeweils eine kurze Pause von einer oder zwei Sekunden, die durch einen Cursor unten rechts signalisiert wird. Während der Pause lässt sich der Monitor durch zwei kurze Tastendrucke (Testtaste) beenden.

Verfügbare Monitore:

- R-Monitor (Widerstand)
- C-Monitor (Kapazität plus optional ESR)
- L-Monitor (Induktivität)
- R/C/L-Monitor (R plus optional L, oder C plus optional ESR)
- R/L-Monitor (Widerstand plus optional Induktivität)

Für die C und L-Monitore gibt es Optionen zum Halten des letzten gültigen Messwerts (SW_MONITOR_HOLD_ESR, SW_MONITOR_HOLD_L).

Der letzte Wert wird in der dritten Textzeile angezeigt.

3.1.11. L/C-Meter (Hardware-Option) Das L/C-Meter basiert auf einer einfachen LC-Oszillatorschaltung, welche von mehreren preiswerten PIC-L/C-Metern benutzt wird. Das übliche Design ($82\mu\text{H}$ und 1nF) hat eine Grundfrequenz von etwa 595 kHz, und das Verbinden eines zusätzlichen Kondensators oder einer Induktivität verringert diese Frequenz.

Mit der Hilfe von einem Referenzkondensator mit bekanntem Wert, den gemessenen Frequenzen und etwas Mathematik kann der Wert des unbekannten Kondensators bzw. Induktivität berechnet werden.

Die PIC-L/C-Meter haben in der Regel einen Messbereich von 10nH bis 100mH, und 0.1pF bis 900nF. Sie scheinen eine Torzeit von 100ms für den Frequenz-zähler zu nutzen. Die m-Firmware hat dagegen eine automatische Bereichswahl mit Torzeiten von 100ms und 1000ms, um die Auflösung für kleine L/C-Werte zu verbessern. Somit starten die Messbereiche bei ca. 1nH und 10fF (0.01pF). Die maximal messbare Induktivität ist grob 150mH. Bzgl. der Kapazität bin ich bei der Schaltung auf ein Problem mit dem Ausgangssignal ab ca. 33nF gestoßen. Dort bekommen die Flanken kleine Sporne, wodurch der Frequenzzähler mehr Pulse sieht als wirklich da sind. Ein Benutzer erwähnte, dass es sich um ein bekanntes Problem der LM311-basierten Oszillatorschaltung handelt. Beim Testen mehrerer Modifikationen ergab sich keine nennenswerte Verbesserung. Daher scheint ein weiterer Komparator oder ein Logik-Gatter mit Schmitt-Trigger- Eingang die beste Lösung zu sein, um das Ausgangssignal des Oszillators zu bereinigen. Das CMOS Vierfach-NAND 4093 funktioniert dazu prima. Bei einem sauberen Ausgangssignal liegt das Maximum der Kapazität bei ca. 120nF (darüber wird der LC-Oszillator instabil). Der Tester begrenzt die Messungen auf einen unteren Wert von 10kHz, d.h. die theoretischen Maximalwerte sind 250mH bzw. 3,5fF, wenn der LC-Oszillator stabil lief.

Nach dem Starten vom L/C-Meter führt der Tester als Erstes einen Selbst- Abgleich durch, angezeigt durch eine "abgleichen..." Nachricht. Danach kann der Kondensator oder die Spule zum Messen verbunden werden. Ein kurzer Tastendruck schaltet zwischen C und L-Messung um (Standard: C-Messung). Die Frequenz des LC-Oszillators driftet mit der Zeit etwas (bis zu 100 Hz) und benötigt dann einen erneuten Abgleich.

Wenn Du einen größer werdenden Null- Wert oder ein "-" ohne angeklemmtem Bauteil siehst, bitte den Selbstabgleich über einen langen Tastendruck starten. Falls es ein Problem mit dem Abgleich gibt oder er durch einen Tastendruck abgebrochen wird, verlässt der Tester das L/C-Meter und gibt einen Fehler aus.

Zwei kurze Tastendrucke beenden das L/C-Meter.

Hinweise:

- Der Referenzkondensator sollte ein 1nF-Folientyp mit kleiner Toleranz sein. Du kannst aber auch einen üblichen Folienkondensator um die 1nF nehmen, ihn mit einem guten LCR-Meter messen, und LC_METER_C_REF entsprechend ändern.
- Wenn Dich die Frequenz des LC-Oszillators und ihr Driften interessiert, dann aktiviere LC_METER_SHOW_FREQ.

3.1.12. 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 [15] beschrieben.

3.1.13. 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 Tasten- drücke 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.

3.1.14. 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.

3.1.15. Klingeltester (Hardware-Option) (LOPT/FBT-Tester) prüft Spulen und Transformatoren auf einen Kurzschluß. Die Schaltung erzeugt einen Trigger-Puls und der Tester zählt dann einfach die Anzahl der Schwingungen, welche einen Hinweis auf den Q-Wert gibt. Die Schaltung kann über einen festen Pin (RING_TESTER_PIN) oder über die Testpins (RING_TESTER_PROBES) gesteuert werden. In beiden Fällen ist der T0-Pin des ATmega der Zählereingang (zählt bei fallender Flanke). Der T0-Pin kann auch parallel zum Ansteuern einer Anzeige verwendet werden.

Nach dem Starten des Klingeltesters wird die Belegung der Testpins angezeigt, sofern die Steuerung per Testpins aktiviert ist. Danach prüft der Tester automatisch Spulen/Trafos und zeigt die Anzahl der Schwingungen an. Wie üblich, beenden zwei kurze Tastendrucke den Test.

Interpretation der Schwingungsanzahl für die Schaltung mit der Darlington- Stufe, basierend auf dem Ringtester von Bob Parker:

Schwingungen	Q
0	Kurzschluß oder offen
1 - 3	niedriges Q (schlecht)
4 - 5	mittleres Q (unklar)
>= 6	hohes Q (gut)

Neben der oben erwähnten einfachen Schaltung kann du auch komplexere Varianten verwenden, solange diese mit ca. 5 V laufen, wenig Strom brauchen (< 20 mA) und Zählimpulse mit fallender Flanke generieren.

Beschaltung für die Kontrolle per Testpins:

Pin #1:	Vcc (5 V)
Pin #2:	Pulsausgang (mit 680Ω Widerstand zur Strombegrenzung)
Pin #3:	Masse
T0:	Zählereingang

3.1.16. 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 (Vorgabewert: 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** kannst Du mit (EVENT_COUNTER_TRIGGER_OUT) optional aktivieren, um ein anderes Gerät über die Testpins zu steuern.

Der Triggerausgang wird während des Zählens auf High gesetzt, was bedeutet: steigende Flanke beim Start und fallende Flanke bei Stop.

Beschaltung für Triggerausgang über die Testpins:

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

3.1.17. 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.

3.1.18. 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.

3.1.19. 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 50 μ s 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 (aka 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 μ PD1986C <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 (680 Ω 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.

3.1.20. IR-Fernbedienung Die IR-Fernbedienung sendet Fernbedienungscodes, 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 Fernbedienungscode, 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 I_f von 100mA nicht ausreichend ist.

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

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

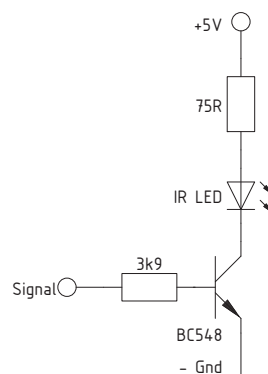


Abbildung 3.1. Beispiel 50mA IR Treiber mit (V_f 1.5V, I_f 100mA,)

Hinweis: Falls das Timing der Pulse/Pausen nicht passen sollte, bitte die alternative Warteschleifenmethode SW_IR_TX_ALTDELAY auf Seite 50 aktivieren.

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

3.1.21. 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.

3.1.22. 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

3.1.23. OneWire-Scan Der OneWire-Scan zeigt die ROM-Codes all angeschlossenen Busteilnehmer an. Zum Einrichten des OneWire-Busses siehe bitte den Abschnitt „Busse & Schnittstellen“ auf Seite 59. 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.

Bei jedem Tastendruck sucht der Tester nach dem nächsten Busteilnehmer und gibt dessen ROM-Code aus (in hexadezimal). Der erste Teil der Ausgabe ist der Familiencode und der zweite die Seriennummer. Der CRC-Wert wird weggelassen. Bei einem Familiencode $\geq 0x80$ (Bit 7 gesetzt) handelt es sich um einen kundenspezifischen Code, und die oberen (linken) drei Stellen der Seriennummer sind die Kunden-ID.

Der Tester informiert Dich, wenn er den letzten Busteilnehmer gefunden hat, bzw. bei CRC- oder Busfehlern. Im Fall des letzten Busteilnehmers oder eines Busfehlers kannst Du per Tastendruck einen komplett neuen Scandurchlauf starten. Anschluß siehe Seite 12.

Wie üblich beenden zwei kurze Tastendrucke die Funktion.

3.1.24. DS18B20-Temperatursensor Hiermit wird der OneWire-Temperatursensor DS18B20 ausgelesen. Zum Einrichten des OneWire-Busses Seite 59 siehe bitte den Abschnitt „Busse“. Bei Benutzung der Test-Pins informiert der Tester über die Beschaltung und wartet dann bis ein externer Pull-Up-Widerstand erkannt wurde. Mit einem Tastendruck lässt sich dies überspringen.

Nach dem 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. Und mit einem langen Tastendruck wählst Du den Auto-Modus (automatische Aktualisierung), welcher durch ein „*“ in der ersten Zeile signalisiert wird. Beschaltung für das DS18B20 Sensor:

Probe #1: Masse/Gnd

Probe #2: DQ (Daten)

Probe #3: VSS (680Ω Widerstand zur Strombegrenzung)

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

Zum Beenden zweimal kurz die Test-Taste drücken.

3.1.25. 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 $4,7k\Omega$ zwischen Data and Vdd wird benötigt! Manche Sensormodule haben bereits einen $10k\Omega$ Pull-Up-Widerstand integriert, welcher ebenfalls gut mit kürzeren Kabeln funktioniert.

Hinweis:

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

3.1.26. MAX6675/MAX31855 Thermoelement-Konverter

Die beiden MAX sind Thermoelement-Konverter mit Kaltstellenkompensation und werden über den SPI-Bus gesteuert. Der MAX6675 ist nur für Thermoelemente vom Typ K, während es den MAX31855 für mehrere Typen gibt. Der MAX31855 hat eine Betriebsspannung von 3.3V und benötigt daher Pegelwandler.

Die Bedienung für beide MAX ist gleich. Durch das Drücken der Testtaste wird eine Messung ausgelöst und die Temperatur angezeigt. Bei Fehlern kommt ein “-“ stattdessen. Für den automatischen Lesemodus (jede Sekunde) die Testtaste lange drücken (wird durch ein “*” nach dem MAX-Namen signalisiert). Ein zweiter langer Tastendruck beendet wieder den automatischen Lesemodus. Und zwei kurze Tastendrucke beenden die Funktion.

3.1.27. 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 R_L-Widerstände (Offset in mV)
- T3 Vergleich der R_H-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)

3.1.28. 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** automatisch im EEPROM gespeichert (siehe Speichern/Laden).

Der Spannungsoffset des Analogkomparators wird automatisch bei der Messung eines Kondensators bestimmt (bei der normalen Bauteilesuche), wenn der Kondensator einen Wert zwischen 100nF und $3,3\mu\text{F}$ 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 $3,3\mu\text{F}$ 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 ab-gleichenden 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, Abschnitt 6.2.7 ab Seite 56, auf spezifische Offsets je Test-Pin-Paar umschalten (CAP_MULTIOFFSET).

Das Gleiche ist für den Widerstandsoffset möglich (R_MULTIOFFSET).

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 $< 100\text{ pF}$
- Interner Widerstand vom IO-Pin im Low-Modus (RiL) $< 25\ \Omega$
- Interner Widerstand vom IO-Pin im High-Modus (RiH) $< 29\ \Omega$

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.

3.1.29. Speichern/Laden Nach dem Selbstabgleich kannst Du mit dieser Funktion die im EEPROM gespeicherten Abgleichswerte aktualisieren. 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. Wenn zwei zu wenig sind, kann ein drittes Profil aktiviert werden (UI_THREE_PROFILES).

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.

Als Option kannst Du das Lade-Menü automatisch nach dem Einschalten des Testers anzeigen lassen (UI_CHOOSE_PROFILE).

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

3.1.31. Zeichensatz/Symbole Die beiden Menüpunkte geben den kompletten Zeichensatzes bzw. die Bauteile- symbole zu Testzwecken aus. Zeilen/Blöcke starten mit der Adresse (hexadezimal) des ersten Zeichens/Symbols in jener Zeile/Block. Danach folgen entweder 8 Zeichen oder so viele Symbole wie in den Block passen.

3.1.32. Ausschalten Hiermit kannst Du den Tester abschalten, sofern die Funktion über SW_POWER_OFF auf Seite 54 aktiviert wurde.

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

4.1. 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 $10\ \Omega$ 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.

Wenn die optionale Prüfung auf E-Normwerte (SW_R_E*) auf Seite 52 aktiviert ist, nimmt der Tester den nächst kleineren und nächst größeren Normwert, und vergleicht diese mit dem gemessenen Widerstand unter Berücksichtigung der Bauteiltoleranz. Es gibt zwei Ausgabemodi. Im Text-Modus gibt der Tester die E-Serie plus Toleranz und passende Normwerte aus. Ein „-“ bedeutet, dass kein Normwert passt. Im Farbcode-Modus zeigt der Tester die E-Serie und den Farbcode des Normwertes inklusive Farbband für die Toleranz. Die Farben können je nach Displaymodul und Farbkombinationen unterschiedlich ausfallen. Falls eine Farbe danebenliegt, einfach den Farbwert (COLOR_CODE_*) in der Datei colors.h entsprechend ändern. Eine Internetsuche nach „RGB565 tool“ listet viele Online-Tools zum Auswählen der RGB565-Farbwerte.

4.2. 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.h) auf Seite 56, 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 4,7µF. 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 |

Die optionale Prüfung auf E-Normwerte gibt es ebenfalls für Kondensatoren (SW_C_E*) Seite 52, aber nur im Text-Modus, da es einfach zu viele unterschiedliche Farbcodes für Kondensatoren gibt.

4.3. 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.

Wenn Du eine Prüfung auf E-Normwerte möchtest, bitte die SW_L_E -Schalter in config.h siehe Seite 52 aktivieren (nur Text-Modus).

Hinweise: Bei unerwarteten Messwerten bitte die Messung wiederholen.

- Die Induktivitätsmessung wird nur durchgeführt, wenn der Widerstand der Spule kleiner als 2k Ω ist.

4.4. 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.

4.5. 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.

4.6. 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 V_f der Messung mit niedrigem Teststrom ($10\mu A$) 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 V_f an.

Leider unterstützt der Tester nicht diesen Messstrom.

Und für höhere Ströme liegt V_f 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 (SW_SYMBOLS), 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.

4.7. Zusätzliche Hinweise

Bipolartransistoren Ein Kleinbuchstabe hinter dem hFE-Wert zeigt den Typ der Testschaltung an, die zum Messen benutzt wurde:

- e: Emitterschaltung
- c: Kollektorschaltung

Wenn Du die Ausgabe vom hFE-Teststrom aktiviert hast (SW_HFE_CURRENT), dann zeigt der Tester I_C für die Emitterschaltung und I_E für die Kollektor-schaltung.

Beim Test auf Dioden wird V_f zum einen mit R_L (hoher Teststrom) und zum anderen mit R_h (niedriger Teststrom) gemessen. Die Ausgabefunktion für Bipolartransistoren interpoliert aus den beiden V_f -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.

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.

Bei einem Schottky-Transistor (Schottky-clamped BJT) wird die Schottky-Diode zwischen Basis und Kollektor incl. V_f angezeigt, sofern die Erkennung aktiviert ist (SW_SCHOTTKY_BJT) Siehe Seite 51. Es sei darauf hingewiesen, dass I_{CEO} bei einem Schottky-Transistor erhöht ist.

TRIACs 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 oder eine hohe Steuerspannung benötigen, können nicht erkannt werden, da der Tester max. ca. 7mA Strom und 5V Spannung zum Schalten hat. So kann z.B. ein DIAC mit einem V_{BO} von 20-200V nicht getestet werden. Das gleiche für Thyristoren und TRIACs mit hohem Triggerstrom.

* Abhilfen für manche Tester

Wenn Dein Tester eines der folgenden Probleme zeigt, kannst Du ein Workround versuchen:

- hFE viel zu hoch

Problem: Bei der Kollektorschaltung mit R_l als Basiswiderstand wird aus unbekanntem Grund die Basisspannung zu niedrig gemessen. Somit erscheint der Basisstrom ebenfalls niedriger und verursacht einen zu hohen hFE-Wert.

Betroffene Tester: Hiland M644

Abhilfe: Aktiviere NO_HFE_C_RL in config.h! Siehe Seite 52.

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 Stromversorgung gibt der Tester manchmal fälschlicherweise einen Elko um die $50\mu F$ aus, obwohl kein Bauteil angeschlossen ist.

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

- Problem mit der ESR-Messung bei low-ESR Elkos mit festem Elektrolyt.

4.8. Hilfreiche Links

[10] Deutsches Forum

<https://www.mikrocontroller.net/topic/248078>

Hauptsprache ist Deutsch, Englisch ist aber auch ok.

[11] Englisches Forum

[https://www.eevblog.com/forum/testgear/\\$20-lcr-esr-transistor-checker-project/](https://www.eevblog.com/forum/testgear/$20-lcr-esr-transistor-checker-project/)

Nur Englisch.

[12] Russisches Forum

<https://vrtp.ru/index.php?showtopic=16451>

[13] Infos zu verschiedenen Clonen, wie Bilder, Schaltpläne und Firmware

(von indman@EEVblog)

<https://yadi.sk/d/yW8xa5NJgUo5z>

[14] Anleitung plus div. Dateien für WinAVR (von indman@EEVblog)

https://drive.google.com/file/d/1-IJA8uTcsCA_6SYHEuMydjfS2vNgmwdH/view?usp=sharing

[15] Aktuelle k-Firmware

<https://github.com/kubi48/TransistorTester-source>

[16] Aktuelle Dokumentation für k-Firmware

<https://github.com/kubi48/TransistorTester-documentation>

4.9. Firmware-Änderungen

Findest du im Kapitel 9 ab Seite 87.

5.1. Fernsteuerungskommandos

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

5.1.1. ERR - unbekanntes Kommando

- Kommando im aktuellen Komponentenkontext nicht unterstützt
- Pufferüberlauf

5.1.2. OK - Kommando ausgeführt

(manche Kommandos benötigen etwas Zeit zum Ausführen)

5.1.3. N/A - Information/Wert nicht verfügbar

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

5.2. Basiskommandos

5.2.1. VER - gibt Version der Firmware zurück

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

5.2.2. OFF - schaltet den Tester aus

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

5.3. Testkommandos

5.3.1. 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“

5.3.2. COMP - gibt ID der Bauteilart zurück

- für IDs siehe COMP_* in common.h
- Beispielantwort für BJT: „30“

5.3.3. MSG - gibt Fehlermeldung zurück

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

5.3.4. QTY - gibt Anzahl der gefundenen Bauteile zurück

- Beispielantwort für BJT: „1“

5.3.5. NEXT - wählt zweites Bauteil aus

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

5.3.6. 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.“

5.3.7. HINT

- gibt Hinweise zu speziellen Merkmalen eines Bauteils 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)
 - D_CLAMP Schottky-Transistor (BJT)
benötigt aktivierte Schottky-Transistor-Erkennung
 - SYM Drain und Source symmetrisch (FET)
- Beispielantwort für BJT: „D_FB R_BE“
- Beispielantwort für FET (MOSFET): „D_FB“

5.3.8. MHINT

- gibt Hinweise zu Messungen zurück
- nur für BJT,
- verfügbare Merkmale:
 - h_FE_e h_FE-Messung mit Emitterschaltung (BJT)
 - h_FE_c h_FE-Messung mit Kollektorschaltung (BJT)
- Beispielantwort für BJT: „h_FE_e“

5.3.9. 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“

5.3.10. R - gibt Widerstandswert zurück

- nur für Widerstand (beinhaltet auch Induktivität)
- Beispielantwort: „122R“

5.3.11. C - gibt Kapazitätswert zurück

- nur für Kondensator
- example responses: „98nF“ „462uF“

5.3.12. L - gibt Induktivitätswert zurück

- nur für Widerstand (beinhaltet auch Induktivität)
- Beispielantwort: „115uH“

- 5.3.13. ESR** - gibt ESR-Wert zurück (Equivalent Series Resistance)
 - benötigt aktivierte ESR-Messung
 - nur für Kondensator
 - Beispielantwort: „0.21R“
- 5.3.14. I_l** - gibt I_leak zurück (Leckstromequivalent zur Selbstentladung)
 - nur für Kondensator
 - Beispielantwort: „3.25uA“
- 5.3.15. 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“
- 5.3.16. V_F2** - gibt V_F der Niedrigstrommessung zurück (forward voltage)
 - nur für Diode
 - Beispielantwort: „387mV“
- 5.3.17. C_D** - gibt C_D zurück (Kapazität der Diode)
 - nur für Diode
 - Beispielantwort: „8pF“
- 5.3.18. I_R** - gibt I_R zurück (reverse current)
 - nur für Diode
 - Beispielantwort: „4.89uA“
- 5.3.19. R_BE** - gibt R_BE zurück (Basis-Emitter-Widerstand)
 - nur für Diode und BJT
 - Beispielantworten: „38.2R“ „5171R“
- 5.3.20. h_FE** - gibt h_FE zurück (DC-Stromverstärkungsfaktor)
 - nur für BJT
 - Beispielantwort: „234“
- 5.3.21. 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“
- 5.3.22. I_C** - gibt den Teststrom I_C der hFE-Messung zurück
 - benötigt aktivierte Ausgabe des Teststroms der hFE-Messung
 - für hFE-Messung mit Emitterschaltung
 - nur für BJT
 - Beispielantwort: „3245uA“
- 5.3.23. I_E** - gibt den Teststrom I_E der hFE-Messung zurück
 - benötigt aktivierte Ausgabe des Teststroms der hFE-Messung
 - für hFE-Messung mit Kollektorschaltung
 - nur für BJT
 - Beispielantwort: „3245uA“
- 5.3.24. V_BE** - gibt V_BE zurück (Basis-Emitter-Spannung)
 - nur für BJT
 - Beispielantwort: „657mV“
- 5.3.25. I_CEO** - gibt I_CEO zurück (Kollektor-Emitter-Strom, offene Basis)
 - nur für BJT
 - Beispielantwort: „460.0uA“
- 5.3.26. V_th** - gibt V_th zurück (threshold voltage)
 - nur für FET (MOSFET) und IGBT
 - Beispielantwort: „2959mV“

- 5.3.27. C_GS** - gibt C_GS zurück (Gate-Source-Kapazität)
 - nur für FET (MOSFET)
 - Beispielantwort: „3200pF“
- 5.3.28. R_DS** - gibt R_DS_on zurück (Drain-Source-Widerstand durchgeschaltet)
 - nur für FET (MOSFET)
 - Beispielantwort: „1.20R“
- 5.3.29. V_GS_off** - gibt V_GS(off) zurück (Cutoff Spannung)
 - nur für Verarmungstyp-FET
 - Beispielantwort: 3072mV"
- 5.3.30. I_DSS** - gibt I_DSS zurück (Drain-Source-Strom, kurzgeschlossenes Gate)
 - nur für Verarmungstyp-FET
 - Beispielantwort: „6430μA“
- 5.3.31. C_GE** - gibt C_GE zurück (Gate-Emitter-Kapazität)
 - nur für IGBT
 - Beispielantwort: „724pF“
- 5.3.32. V_GT** - gibt V_GT zurück (Gate-Trigger-Spannung)
 - nur für Thyristor und TRIAC
 - Beispielantwort: „865mV“
- 5.3.33. V_T** - gibt V_T zurück (Offset-Spannung)
 - nur für PUT
 - Beispielantwort: „699mV“
- 5.3.34. R_BB** - gibt R_BB zurück (interbase resistance)
 - benötigt aktivierte UJT-Erkennung
 - nur für UJT
 Beispielantwort: „4758R“
- 5.3.35. V_Z** - gibt V_Z zurück (Zener- oder externe Spannung)
 - benötigt aktvierten Zener-Test während normaler Bauteilesuche
 - nur für Zenerdiode oder externe Spannung (Hardware-Option)
 - Beispielantwort: „6750mV“

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.

6.1. 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).

6.1.1. MCU-Typ

```
# avr-gcc: MCU model
# - ATmega 328/328P : atmega328
# - ATmega 328PB    : atmega328pb
# - ATmega 324P/324PA : atmega324p
# - ATmega 640      : atmega640
# - ATmega 644/644P/644PA : atmega644
# - ATmega 1280     : atmega1280
# - ATmega 1284/1284P : atmega1284
# - ATmega 2560     : atmega2560
MCU = atmega328
```

Listing 6.1. Worgewählt ist atmega328

6.1.2. MCU-Taktfrequenz

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

Listing 6.2. Worgewählt ist 8MHz

6.1.3. Oszillator-Typ

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

Listing 6.3. Worgewählt ist Crystal

6.1.4. Avrdude MCU-Typ

```
# avrdude: part number of MCU
# - ATmega 328 : m328
# - ATmega 328P : m328p
# - ATmega 328PB : m328pb
# - ATmega 324P : m324p
# - ATmega 324PA : m324pa
# - ATmega 640 : m640
# - ATmega 644 : m644
# - ATmega 644P : m644p
# - ATmega 644PA : m644p
# - ATmega 1280 : m1280
# - ATmega 1284 : m1284
# - ATmega 1284P : m1284p
# - ATmega 2560 : m2560
PARTNO = m328p
```

Listing 6.4. Worgewählt ist m328p

6.1.5. Avrdude ISP-Programmierer Bei dem Programmierer sind notwendig:
Name; BitClock und Port.

```
# Arduino as ISP
#PROGRAMMER = stk500v1
#PORT = /dev/ttyACM0
#OPTIONS = -b 19200

# Bus Pirate
#PROGRAMMER = buspirate
#PORT = /dev/bus_pirate
#OPTIONS = -B 10.0

# Diamex ALL-AVR/AVR-Prog
PROGRAMMER = avrispmkII
PORT = usb
OPTIONS = -B 1.0

# Pololu USB AVR Programmer
#PROGRAMMER = stk500v2
#PORT = /dev/ttyACM0
#OPTIONS = -B 1.0

# USBasp
#PROGRAMMER = usbasp
#PORT = usb
#OPTIONS = -B 20

# USBtinyISP
#PROGRAMMER = usbtiny
#PORT = usb
#OPTIONS = -B 5.0

# Arduino Uno bootloader via serial/USB
#PROGRAMMER = arduino
#PORT = /dev/ttyACM0
#OPTIONS = -D -b 115200

# Arduino Mega2560 bootloader via serial/USB
#PROGRAMMER = wiring
#PORT = /dev/ttyACM0
#OPTIONS = -D -b 115200
```

Listing 6.5. Worgewählt ist Diamex

Falls Dein Programmierer nicht aufgeführt ist, bitte die passenden Werte eintragen.
Weitere Informationen findest Du im Avrdude-Handbuch oder der Online-Dokumentation [9].

6.2. config.h

Diese Datei dient zum Einstellen von Bedienung und Funktionen. Da es eine normale C-Header-Datei ist, werden die bekannten Kommentarrregeln 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.

6.2.1. 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 Deaktivieren auskommentieren

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 $\geq 5,5$ -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 in der config_<MCU>.h
- `//#define HW_ZENER` - zum Aktivieren einkommentieren

Nicht-Standard-Spannungsteiler für Zener-Prüfung

- Standard-Spannungsteiler ist 10:1
 - ZENER_R1: oberer Widerstand in Ω
 - ZENER_R2: unterer Widerstand in Ω
- ```
//#define ZENER_DIVIDER_CUSTOM - zum Aktivieren einkommentieren
#define ZENER_R1 180000 -Bei Bedarf Wert ändern
#define ZENER_R2 20000 -Bei Bedarf Wert ändern
```

### alternativer Modus für Zener-Check: Aufwärtswandler nicht schalten

- wenn der DC-DC-Aufwärtswandler die ganze Zeit läuft
  - bei Messung einer externen Spannung (Schaltung ohne Hochsetzsteller)
- ```
//#define ZENER_UNSWITCHED              - zum Aktivieren einkommentieren
```

Zener-Prüfung bei normaler Abtastung

- erfordert, dass der Hochsetzsteller die ganze Zeit läuft (ZENER_UNSWITCHED)
- Die Min-/Max-Spannungen sind für die Erkennung einer gültigen Zener-Spannung gedacht.

Die Min.-Spannung sollte höher als das Grundrauschen sein, während die Max. Spannung niedriger sein sollte als die Ausgangsspannung des Hochsetzstellers.

- ```
//#define HW_PROBE_ZENER - zum Aktivieren einkommentieren
#define ZENER_VOLTAGE_MIN 1000 /* min. voltage in mV */ -Bei Bedarf ändern
#define ZENER_VOLTAGE_MAX 30000 /* max. voltage in mV */ -Bei Bedarf ändern
```

### Fester Signalausgang

- falls der OC1B-Pin der MCU als dedizierter Signalausgang verdrahtet ist, anstatt den R1-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 /
```

Klingeltester (LOPT/FBT-Tester)

- verwendet T0 direkt als Zählereingang
 - zum Aktivieren einkommentieren
 - wähle den Impulsausgang: entweder dedizierter Pin oder Sonden
 - siehe RINGTESTER_PORT in config-<MCU>.h für dedizierten Pin
- ```
//#define HW_RING_TESTER
#define RING_TESTER_PIN /* dedicated pin */
//#define RING_TESTER_PROBES /* probes */
```



### 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_OUT` - 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

### L/C Meter

- Benutzt T0 direkt als Frequenzeingang
- siehe LC\_CTRL\_PORT in config-<MCU>.h für Port-Pins
- `//#define HW_LC_METER` - zum Aktivieren einkommentieren

### L/C Meter Wert des Referenzkondensators C\_p (in 0,1 pF)

- sollte etwa 1000pF betragen
- `#define LC_METER_C_REF 10000` - zum Deaktivieren auskommentieren

### L/C Meter zeigt auch die Frequenz des LC-Oszillators an

- hilft, der Frequenzdrift des Oszillators zu erkennen
- erfordert Display mit mehr als zwei Textzeilen
- `//#define LC_METER_SHOW_FREQ` - zum Aktivieren einkommentieren

### Relais für Parallelkondensator (Sampling-ADC)

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

### Logiktester

- siehe TP\_LOGIC in config-<MCU>.h für dedizierte Port-Pin
- verwendet Spannungsteiler (Standard: 4:1, R1=10k, R2=3.3k, bis zu 20V)
- LOGIC\_PROBE\_R1: oberer Widerstand in  $\Omega$
- LOGIC\_PROBE\_R2: unterer Widerstand in  $\Omega$
- erfordert zusätzliche Tasten (z.B. Drehgeber) und ein Display mit mehr als 4 Zeilen
- zum Aktivieren und Einstellen der Widerstandswerte einkommentieren
- `#define HW_LOGIC_PROBE`
- `//#define LOGIC_PROBE_R1 10000`
- `#define LOGIC_PROBE_R2 3300`

### Durchgangsprüfer

- siehe BUZZER\_CTRL in config-<MCU>.h für Port-Pin
- `//#define HW_BUZZER` - zum Aktivieren einkommentieren

### **MAX6675** Thermoelement-Konverter

- siehe MAX6675\_CS in config\_<MCU>.h für dedizierte Port-Pin
- erfordert SPI-Bus und SPI-Leseunterstützung
- //#define HW\_MAX6675 - zum Aktivieren einkommentieren

### **MAX31855** thermocouple converter \* - siehe MAX31855\_CS in config\_<MCU>.h für dedizierten Portpin

- erfordert SPI-Bus und SPI-Leseunterstützung
- //#define HW\_MAX31855 - zum Aktivieren einkommentieren

## **6.2.2. Software Optionen**

### **PWM-Generator mit einfacher Bedienung**

- Ausgang über OC1B
- #define SW\_PWM\_SIMPLE - zum Deaktivieren auskommentieren

### **PWM-Generator mit erweiterter Bedienung**

- Ausgang über OC1B
- 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**

- 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

### **ESR Tool** ESR-Messung im Schaltkreis

- erfordert die Aktivierung von SW\_ESR oder SW\_OLD\_ESR
- //#define SW\_ESR\_TOOL - 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**

- Ausgang über OC1B
- 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

### **Schottky Transistor (Schottky-geklemmter BJT) prüfen**

#define SW\_SCHOTTKY\_BJT - zum Deaktivieren auskommentieren

### **Servo-Test**

- Ausgang über OC1B
- erfordert zusätzliche Tasten und Anzeige mit mehr als 2 Textzeilen

//#define SW\_SERVO - zum Aktivieren einkommentieren

### **Tempertur messen mit dem DS18B20**

- aktiviere auch ONEWIRE \_PROBES oder ONEWIRE \_IO \_PIN (siehe 'Busse')

//#define SW\_DS18B20 - zum Aktivieren einkommentieren

### **OneWire-ROM-Code lesen** und anzeigen.

- Option für OneWire-bezogene Werkzeuge
- erfordert eine Anzeige mit mehr als zwei Zeilen

//#define SW\_ONEWIRE\_READ\_ROM - zum Aktivieren einkommentieren

### **OneWire-Scan** zeigt die ROM-Codes all angeschlossenen Busteilnehmer an.

- erfordert eine Anzeige mit mehr als zwei Zeilen
- aktiviere auch ONEWIRE \_PROBES oder ONEWIRE \_IO \_PIN (siehe 'Busse')

//#define SW\_ONEWIRE\_SCAN - 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 auskommentieren

### **Anzeige I<sub>C</sub>/I<sub>E</sub>** Prüfstrom für hFE-Messung

- I<sub>C</sub> für gemeinsamen Emitterkreis
- I<sub>E</sub> für gemeinsamen Kollektorkreis

//#define SW\_HFE\_CURRENT - zum Aktivieren einkommentieren

### **R/C/L Monitore**

- Überwachung passiver Komponenten, die an die Sonden #1 und #3 angeschlossen sind
- Monitore für L erfordern die Aktivierung von SW\_INDUCTOR
- für ESR muss entweder SW\_ESR oder SW\_OLD\_ESR aktiviert werden

//#define SW\_MONITOR\_R /\* nur R \*/

//#define SW\_MONITOR\_C /\* nur C plus ESR \*/

//#define SW\_MONITOR\_L /\* nur L \*/

//#define SW\_MONITOR\_RCL /\* R plus L, oder C plus ESR \*/

//#define SW\_MONITOR\_RL /\* R plus L \*/ - uncomm.(ein||mehrere)

### **C/L-Monitore:** (Automatisches Halten)

- erfordert Anzeige mit mehr als zwei Textzeilen
- zum Aktivieren auskommentieren (eine oder mehrere)

```
// #define SW_MONITOR_HOLD_ESR /* Auto-Hold ESR (C-Monitor) */
// #define SW_MONITOR_HOLD_L /* Auto-Hold L (L-Monitor) */
```

### **DHT11, DHT22 und kompatible Feuchtigkeits- und Temperatursensoren**

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

### **Widerstand auf Übereinstimmung mit dem Normwert der E-Reihe prüfen**

- erfordert eine Anzeige mit mehr als 2 Textzeilen
- Farbcode-Modus erfordert eine Farbgrafikanzeige

```
// #define SW_R_E24_5_T E24 5% tolerance, text
// #define SW_R_E24_5_CC E24 5% tolerance, color-code
// #define SW_R_E24_1_T E24 1% tolerance, text
// #define SW_R_E24_1_CC E24 1% tolerance, color-code
// #define SW_R_E96_T E96 1% tolerance, text
// #define SW_R_E96_CC E96 1% tolerance, color-code
// #define SW_R_E96_EIA96 E96 1% tolerance, EIA-96-code- uncomm.(ein||mehrere)
```

### **Kondensator auf Übereinstimmung mit dem Normwert der E-Reihe prüfen**

- erfordert eine Anzeige mit mehr als 2 Textzeilen

```
// #define SW_C_E6_T E6 20% tolerance, text
// #define SW_C_E12_T E12 10% tolerance, text - uncomm.(ein||mehrere)
```

### **Induktivität auf Übereinstimmung mit dem Normwert der E-Reihe prüfen**

- erfordert eine Anzeige mit mehr als 2 Textzeilen

```
// #define SW_L_E6_T E6 20% tolerance, text
// #define SW_L_E12_T E12 10% tolerance, text - uncomm.(ein||mehrere)
```

### **Durchgangsprüfung**

- erfordert Summer (HW\_BUZZER)

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

## **6.2.3. Umgehungslösungen für einige Tester**

### **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
```

## **6.2.4. Umgehungslösungen für einige IDEs**

### **Startzyklen des Oszillators** (nach dem Aufwecken aus dem Stromsparmodus):

- 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)

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

Listing 6.6. Wert ändern falls er NICHT zum Tester passt!

### 6.2.5. Benutzerschnittstelle

**Sprachwahl** Standard ist ISO 8859 -1.

Vorwahl \_2 entspricht ISO 8859 -2.

- Russisch ist immer Windows -1251.

```
#define UI_ENGLISH
//#define UI_CZECH
//#define UI_CZECH_2
//#define UI_DANISH
//#define UI_GERMAN
//#define UI_ITALIAN
//#define UI_POLISH
//#define UI_POLISH_2
//#define UI_ROMANIAN
//#define UI_RUSSIAN
//#define UI_RUSSIAN_2
//#define UI_SPANISH
```

Listing 6.7. Wunschsprache auswählen, hier ist Englisch vorgewählt.

**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 einkommentieren

**Hexadezimale Werte in Großbuchstaben** statt Kleinbuchstaben anzeigen

//#define UI\_HEX\_UPPERCASE

- zum Aktivieren einkommentieren

**Standard-Betriebsmodus Auto-Hold** - anstelle des kontinuierlichen Modus

#define UI\_AUTOHOLD

- zum Deaktivieren auskommentieren

**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

**Einstellungsprofil** - nach dem Einschalten auszuwählen.

- Für Tester mit zusätzlichen, fest eingebauten, Messkabeln.

//#define UI\_CHOOSE\_PROFILE

- 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\_SERIAL\_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

### **Pinbelegung mit Bauteilsymbolen** für 3-Pin-Halbleiter

- erfordert Grafikanzeige und Symbol-Bitmaps (`config_<MCU>.h`)

`#define SW_SYMBOLS`

- zum Deaktivieren auskommentieren

### **Alternative Pinbelegung** Anzeige der rechten Tsetpinnummern über/unter dem Symbol

- erfordert, dass Komponentensymbole (`SW_SYMBOLS`) aktiviert sind

`// #define UI_PINOUT_ALT`

- zum Aktivieren einkommentieren

### **Anzeige Fragezeichensymbol** bei fehlgeschlagenem Testlauf

- erfordert, dass Komponentensymbole (`SW_SYMBOLS`) aktiviert sind

`// #define UI_QUESTION_MARK`

- zum Aktivieren einkommentieren

### **Anzeige Zener-Dioden-Symbol** bei der Zener-Prüfung

- erfordert, dass Komponentensymbole (`SW_SYMBOLS`) aktiviert sind

`// #define UI_ZENER_DIODE`

- zum Aktivieren einkommentieren

### **Erweiterter Frequenzzähler:** Anzeige des Quarzsymbols für LF/HF-Modi

- erfordert, dass Komponentensymbole (`SW_SYMBOLS`) aktiviert sind

`// #define UI_QUARTZ_CRYSTAL`

- zum Aktivieren einkommentieren

### **Abschalten der textbasierten Pinbelegung** für 3-Pin-Halbleiter

- erfordert die Aktivierung von Bauteilsymbolen (`SW_SYMBOLS`)

`// #define UI_NO_TEXTPINOUT`

- zum Aktivieren einkommentieren

### **Deaktivieren der textbasierten Pinbelegung** der inneren Diode für MOSFETs

`// #define UI_NO_BODYDIODE_TEXTPINOUT`

- zum Aktivieren einkommentieren

### **Anzeige von Testpin-IDs mit umgekehrten Farben**

- erfordert Schriftart mit zusätzlichen Zeichen

- temporäre Option `UI_PROBE_REVERSED_X` für umgekehrtes 'x', falls möglich  
da nicht alle Schriftarten dieses Zeichen enthalten (Schriftart prüfen!)

`// #define UI_PROBE_REVERSED`

- zum Aktivieren einkommentieren

`// #define UI_PROBE_REVERSED_X`

- zum Aktivieren einkommentieren

### **Farbcodierung für Testpins**

- erfordert farbige Grafikanzeige

- `colors.h` bearbeiten, um die richtigen Sondenfarben auszuwählen

(`COLOR_PROBE_1`, `COLOR_PROBE_2` and `COLOR_PROBE_3`)

`#define UI_PROBE_COLORS`

- zum Deaktivieren auskommentieren

### **Farbiges Titel**

- erfordert farbige Grafikanzeige

- `colors.h` bearbeiten, um die bevorzugte Farbe auszuwählen (`COLOR_TITLE`)

`// #define UI_COLORED_TITLES`

- zum Aktivieren einkommentieren

### **Buntes Cursor- und Tasten-Hinweise**

- erfordert farbige Grafikanzeige

- `colors.h` bearbeiten, um die bevorzugte Farbe (`COLOR_CURSOR`) auszuwählen

`// #define UI_COLORED_CURSOR`

- zum Aktivieren einkommentieren

### Hauptmenü automatisch verlassen

- nach Ausführung einer Funktion

```
//#define UI_MAINMENU_AUTOEXIT
```

 - zum Aktivieren einkommentieren

### Menüpunkt Ausschalten

```
//#define SW_POWER_OFF
```

 - zum Aktivieren einkommentieren

### Display Font Test

```
definiere SW_FONT_TEST
```

 - zum Deaktivieren auskommentieren

### Anzeige von Komponentensymbolen zu Testzwecken

- erfordert, dass Symbole aktiviert sind (SW\_SYMBOLS) im Kapitel 6.2.5 auf Seite 54

```
//#define SW_SYMBOL_TEST
```

 - zum Aktivieren einkommentieren

### Runden der Werten für DS18B20

- DS18B20 (0.1 °C/F)

```
//#define UI_ROUND_DS18B20
```

 - zum Aktivieren einkommentieren

### Speicherung von Firmware-Daten (Texte, Tabellen usw.)

- Selbsteinstellungsdaten werden immer in EEPROM gespeichert
- Schriften und Symbole werden immer in Flash gespeichert

```
#define DATA_EEPROM /* store data in EEPROM */
//#define DATA_FLASH /* store data in Flash */ - einen unkommentierten
```

## 6.2.6. Energieverwaltung

### Typ des Leistungsschalters

- weich einrastender Netzschalter (Standard)

wie in der Referenzschaltung des Testers

Tester ist in der Lage, sich selbst auszuschalten

- manueller Netzschalter

Tester ist nicht in der Lage, sich selbst auszuschalten

```
//#define POWER_SWITCH_SOFT
```

```
#define POWER_SWITCH_MANUAL
```

 -ausgewählt

### 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  $\Omega$
- Standardwerte sind: R1=10k, R2=3,3k

```
#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
```

## 6.2.7. 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 (Vcc) 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 350 $\Omega$

```
#define RH_OFFSET ... 350 - ggf. Wert anpassen
```

### **Widerstand der Testpins/-kabel** (in 0,01 $\Omega$ ) - Standardoffset für Leiterbahnen und Testkabel

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

```
#define R_ZERO ... 20 - ggf. Wert anpassen
```



**Widerstandsoffset für einzelne Testkabel-Paare** , falls sie sich sehr unterscheiden

- wird im Selbstabgleich aktualisiert

// #define R\_MULTIOFFSET

- ggf. Wert anpassen

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

- wird im 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

**Test-Kabelpaar spezifische Kapazität** anstelle von Durchschnittswert für alle Test Kabel-Paare

- wird im Selbstabgleich aktualisiert

// #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  $\mu$ F

CAP\_FACTOR\_MID        für Kondensatoren 4,7 - 47  $\mu$ F

CAP\_FACTOR\_LARGE      für Kondensatoren > 47  $\mu$ F

#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

**100nF AREF-Puffer-Kondensator**

- von einigen MCU-Karten verwendet

- wird die Messzeit erhöhen

- Empfehlung: durch 1nF-Kondensator ersetzen

#define ADC\_LARGE\_BUFFER\_CAP

- zum Deaktivieren auskommentieren

## 6.2.8. F & E - gedacht für Firmware-Entwickler

**Lesefunktionen für das Anzeigemodul** aktivieren

- Display-Treiber und Schnittstelleneinstellungen müssen dies unterstützen

// #define LCD\_READ

- zum Aktivieren einkommentieren

**ID des Anzeige-Controllers lesen**

- ID wird auf dem Willkommensbildschirm angezeigt (nach Firmware-Version)

- erfordert Display-Lesefunktionen (LCD\_READ)

- empfohlen: serielle Ausgabe (UI\_SERIAL)

// #define SW\_DISPLAY\_ID

- zum Aktivieren einkommentieren

**Liest Register des Display-Controllers** und gibt sie über TTL seriell aus.

- erfordert Display-Lesefunktionen (LCD\_READ) und serielle Ausgabe (UI\_SERIAL)

// #define SW\_DISPLAY\_REG

- zum Aktivieren einkommentieren

## 6.2.9. 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 einkommentieren
- 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 einkommentieren

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

### 6.3. Config\_328.h

enthält hardwarenahe Einstellungen für Anzeigen, Tasten und so weiter. 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.

#### 6.3.1. LCD-Module

Die Displays und ihre Einstellungen werden im Kapitel 2.6. ab Seite 12 behandelt. Als Beispiel dient ST7565R-Modul in dem #if 0 und #endif mit “//“ auskommentiert wurden. **Modul ST7565R** als Beispiel

```
//#if 0
#define LCD_ST7565R /* display controller ST7565R */
```

Listing 6.8. Durch // #if 0 und #define LCD\_ST7565R ... richtiges Modul ausgewählt

```
//#endif
```

Listing 6.9. Das Ende der Schleife ist auch auskommentiert

#### 6.3.2. 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 TP1 PC0 /* test pin / probe #1 */
#define TP2 PC1 /* test pin / probe #2 */
#define TP3 PC2 /* test pin / probe #3 */
```

Listing 6.10. Pins nicht ändern

##### Feste HW Optionen

- Mögliche Optionen.

```
#define TP_ZENER PC3 /* test pin for Zener check (10:1 voltage divider) */
#define TP_REF PC4 /* test pin for 2.5V reference and relay */
#define TP_BAT PC5 /* test pin for battery (4:1 voltage divider) */
#define TP_LOGIC PC3 /* test pin for Logic Probe (4:1 voltage divider) */
```

Listing 6.11. Pins nicht ändern

##### Testwiderstände

- Für die PWM- / Rechteckwellenausgabe über Testpin 2 muss R\_RL\_2 PB2/OC1B sein.
- Diesen Port nicht mit POWER\_CTRL oder TEST\_BUTTON teilen!

```
#define R_RL_1 PB0 /* Rl (680R) for test pin #1 */
#define R_RH_1 PB1 /* Rh (470k) for test pin #1 */
#define R_RL_2 PB2 /* Rl (680R) for test pin #2 */
#define R_RH_2 PB3 /* Rh (470k) for test pin #2 */
#define R_RL_3 PB4 /* Rl (680R) for test pin #3 */
#define R_RH_3 PB5 /* Rh (470k) for test pin #3 */
```

Listing 6.12. Pins nicht ändern

##### dedizierte Signalausgabe über OC1B - bitte nicht ändern!

```
#define SIGNAL_OUT PB2 /* MCU's OC1B pin */
```

Listing 6.13. nicht ändern!

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

```
#define POWER_CTRL PD6 /* control pin (1: on / 0: off) */
```

Listing 6.14. nicht ändern!

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

```
#define TEST_BUTTON PD7 /* test/start push button (low active) */
```

Listing 6.15. ggf. anpassen

### Drehenkoder

```
#define ENCODER_A PD3 /* rotary encoder A signal */
#define ENCODER_B PD2 /* rotary encoder B signal */
```

Listing 6.16. ggf. anpassen

### Tasten Auf/Ab

```
#define KEY_INC PD2 /* increase push button (low active) */
#define KEY_DEC PD3 /* decrease push button (low active) */
```

Listing 6.17. ggf. anpassen

### Frequenzzähler

- Einfache und erweiterte Version
- Eingang muss Pin PD4/T0 sein

```
#define COUNTER_IN PD4 /* signal input T0 */
```

Listing 6.18. nicht ändern!

### ring tester

- Eingang muss Pin PD4/T0 sein (benutze COUNTER\_IN)

```
#define RINGTESTER_OUT PD5 /* pulse output */
```

Listing 6.19. ggf. anpassen

### IR-Detektor/Decoder

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

```
#define IR_DATA PD5 /* data signal */
```

Listing 6.20. ggf. anpassen

### Summer

```
#define BUZZER_CTRL PD5 /* control pin (low: off / high: on) */
```

Listing 6.21. ggf. anpassen

### MAX6675

```
#define MAX6675_CS PD5 /* port pin used for /CS */
```

Listing 6.22. ggf. anpassen

### MAX31855

```
#define MAX31855_CS PD5 /* port pin used for /CS */
```

Listing 6.23. ggf. anpassen

### 6.3.3. Busse

**SPI** - SCK PB5, MOSI PB3, MISO PB4 and /SS PB2

- Bit-Bang-SPI könnte bereits im Anzeigenbereich eingestellt sein

```
/* for bit-bang and hardware SPI */
#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 PB5 /* pin for SCK */
#define SPI_MOSI PB3 /* pin for MOSI */
#define SPI_MISO PB4 /* pin for MISO */
#define SPI_SS PB2 /* pin for /SS */
#endif
```

Listing 6.24. ggf. anpassen

**I2C** - Hardware-I2C (TWI) verwendet SDA PC4 und SCL PC5

- Bit-Bang-I2C könnte bereits im Anzeigebereich eingestellt sein

```
/* for bit-bang I2C */
#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 PC4 /* pin for SDA */
#define I2C_SCL PC5 /* pin for SCL */
#endif
```

Listing 6.25. ggf. anpassen

#### serielle TTL-Schnittstelle

- hardware USART0 verwendet Rx PD0 & Tx PD1

```
/* for hardware RS232 */
#define SERIAL_USART 0 /* use USART0 */
/* for bit-bang RS232 */
#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 16 /* PCINT# for Rx pin */
```

Listing 6.26. ggf. anpassen

**OneWire** - dedizierter I/O-Pin

```
#define ONEWIRE_PORT PORTD /* port data register */
#define ONEWIRE_DDR DDRD /* port data direction register */
#define ONEWIRE_PIN PIND /* port input pins register */
#define ONEWIRE_DQ PD5 /* DQ (data line) */
```

Listing 6.27. ggf. anpassen

## 6.4. Config\_644.h

enthält hardwarenahe Einstellungen für Anzeigen, Tasten und so weiter. 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.

### 6.4.1. LCD-Module

Die Displays und ihre Einstellungen werden im Kapitel 2.6. ab Seite 12 behandelt. Als Beispiel dient ILI9341-Modul in dem #if 0 und #endif mit “//“ auskommentiert wurden.

#### Modul ILI9341 als Beispiel

```
//#if 0
#define LCD_ILI9341 /* display controller ILI9341/ILI9342 */
```

Listing 6.28. Durch // #if 0 und #define ILI9341 ... richtiges Modul ausgewählt

```
//#endif
```

Listing 6.29. Das Ende der Schleife ist auch auskommentiert

### 6.4.2. 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 TP1 PA0 /* test pin / probe #1 */
#define TP2 PA1 /* test pin / probe #2 */
#define TP3 PA2 /* test pin / probe #3 */
```

Listing 6.30. Pins nicht ändern

#### Feste HW Optionen

```
#define TP_ZENER PA3 /* test pin for for Zener check (10:1 voltage divider) */
#define TP_REF PA4 /* test pin for 2.5V reference and relay */
#define TP_BAT PA5 /* test pin for battery (4:1 voltage divider) */
#define TP_CAP PA7 /* test pin for self-adjustment cap */
#define TP_LOGIC PA6 /* test pin for Logic Probe (4:1 voltage divider) */
```

Listing 6.31. 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 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 6.32. ggf. anpassen

#### Feste HW Optionen

dedizierte Signalausgabe über OC1B - bitte nicht ändern!

```
#define SIGNAL_OUT PD4 /* MCU's OC1B pin */
```

Listing 6.33. nicht ändern!

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

```
#define POWER_CTRL PC6 /* control pin (1: on / 0: off) */
```

Listing 6.34. ggf. anpassen

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

```
#define TEST_BUTTON PC7 /* test/start push button (low active) */
```

Listing 6.35. ggf. anpassen

**Drehenkoder**

```
#define ENCODER_A PC3 /* rotary encoder A signal */
#define ENCODER_B PC4 /* rotary encoder B signal */
```

Listing 6.36. ggf. anpassen

**Tasten auf/ab**

```
#define KEY_INC PC4 /* increase push button (low active) */
#define KEY_DEC PC3 /* decrease push button (low active) */
```

Listing 6.37. ggf. anpassen

**Frequenzzähler**

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

```
#define COUNTER_IN PB0 /* signal input T0 */
```

Listing 6.38. nicht ändern!

**Steuerung für erweiterten Frequenzzähler**

```
#define COUNTER_CTRL_DIV PC0 /* prescaler (low 1:1, high x:1) */
#define COUNTER_CTRL_CH0 PC1 /* channel addr #0 */
#define COUNTER_CTRL_CH1 PC2 /* channel addr #1 */
```

Listing 6.39. ggf. anpassen

**Klingeltester**

- Eingang muss Pin PD7 / T0 sein (benutze COUNTER\_IN)

```
#define RINGTESTER_OUT PC0 /* pulse output */
```

Listing 6.40. ggf. anpassen

**L/C meter**

- Frequenz Eingang, welches an Pin PB0/T0 angeschlossen ist

```
#define LC_CTRL_CP PC0 /* reference cap (low: on / high: off) */
#define LC_CTRL_LC PC1 /* L/C selection (low: C / high: L) */
```

Listing 6.41. nicht ändern!

**IR-Detektor/Decoder**

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

```
#define IR_DATA PC2 /* data signal */
```

Listing 6.42. 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_RH PC5 /* Rh (470k) for fixed cap */
```

Listing 6.43. ggf. anpassen

## Summer

```
#define BUZZER_CTRL PC2 /* control pin (low: off / high: on) */
```

Listing 6.44. ggf.anpassen

## MAX6675

```
#define MAX6675_CS PC2 /* port pin used for /CS */
```

Listing 6.45. ggf.anpassen

## MAX31855

```
#define MAX31855_CS PC2 /* port pin used for /CS */
```

Listing 6.46. ggf.anpassen

## Relais für Parallelkondensator (Sampling-ADC)

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

```
#define CAP_RELAY_CTRL PC2 /* control pin */
```

Listing 6.47. ggf. anpassen



### 6.4.3. Busse

**SPI** - SCK PB7, MOSI PB5, MISO PB6 and /SS PB4

- Bit-Bang-SPI könnte bereits im Anzeigenbereich eingestellt sein

```
/* for bit-bang and hardware SPI */
#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 */
#define SPI_SS PB4 /* pin for /SS */
#endif
```

Listing 6.48. ggf. anpassen

**I2C** - Hardware-I2C (TWI) verwendet SDA PC1 and SCL PC0

- Bit-Bang-I2C könnte bereits im Anzeigenbereich eingestellt sein

```
/* for bit-bang I2C */
#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 6.49. ggf. anpassen

### serielle TTL-Schnittstelle

- hardware USART verwendet

USART0: Rx PD0 und Tx PD1

USART2: Rx PD2 und Tx 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 6.50. 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 6.51. ggf. anpassen

## 6.5. Config\_1280.h

enthält hardwarenahe Einstellungen für Anzeigen, Tasten und so weiter. 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.

### 6.5.1. LCD-Module

Die Displays und ihre Einstellungen werden im Kapitel 2.6. ab Seite 12 behandelt. Als Beispiel dient ILI9486 in dem #if 0 und #endif mit “//“ auskommentiert wurden.

**Module ILI9486** als Beispiel

```
//#if 0
//#define LCD_ILI9481 /* display controller ILI9481 */
#define LCD_ILI9486 /* display controller ILI9486 */
//#define LCD_ILI9488 /* display controller ILI9488 */
```

Listing 6.52. Durch // #if 0 und #define ILI9486 ... richtiges Modul ausgewählt

```
//#endif
```

Listing 6.53. Das Ende der Schleife ist auch auskommentiert

### 6.5.2. 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 TP1 PF0 /* test pin / probe #1 */
#define TP2 PF1 /* test pin / probe #2 */
#define TP3 PF2 /* test pin / probe #3 */
```

Listing 6.54. Pins nicht ändern

#### Feste HW Optionen

```
#define TP_ZENER PF3 /* test pin for Zener check (10:1 voltage divider) */
#define TP_REF PF4 /* test pin for 2.5V reference and relay */
#define TP_BAT PF5 /* test pin for battery (4:1 voltage divider) */
#define TP_CAP PF7 /* test pin for self-adjustment cap */
#define TP_LOGIC PF6 /* test pin for Logic Probe (4:1 voltage divider) */
```

Listing 6.55. nicht ändern!

#### Testwiderstände

- Für die PWM- / Rechteckwellenausgabe über Testpin 2 muss R\_RL\_2 PB6/OC1B sein.
- Diesen Port nicht mit POWER\_CTRL oder TEST\_BUTTON teilen!

```
#define R_RL_1 PK0 /* Rl (680R) for test pin #1 */
#define R_RH_1 PK1 /* Rh (470k) for test pin #1 */
#define R_RL_2 PK2 /* Rl (680R) for test pin #2 */
#define R_RH_2 PK3 /* Rh (470k) for test pin #2 */
#define R_RL_3 PK4 /* Rl (680R) for test pin #3 */
#define R_RH_3 PK5 /* Rh (470k) for test pin #3 */
```

Listing 6.56. ggf. anpassen

#### dedizierte Signalausgabe über OC1B - bitte nicht ändern!

```
#define SIGNAL_OUT PB6 /* MCU's OC1B pin */
```

Listing 6.57. nicht ändern!

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

```
#define POWER_CTRL PA6 /* control pin (1: on / 0: off) */
```

Listing 6.58. ggf. anpassen

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

```
#define TEST_BUTTON PA7 /* test/start push button (low active) */
```

Listing 6.59. ggf. anpassen

**Drehenkoder**

```
#define ENCODER_A PA3 /* rotary encoder A signal */
#define ENCODER_B PA1 /* rotary encoder B signal */
```

Listing 6.60. ggf. anpassen

**Tasten auf/ab**

```
#define KEY_INC PA3 /* increase push button (low active) */
#define KEY_DEC PA1 /* decrease push button (low active) */
```

Listing 6.61. ggf. anpassen

**Frequenzzähler**

- Einfache und erweiterte Version
- Eingang muss Pin PD7 / T0 sein

```
#define COUNTER_IN PD7 /* signal input T0 */
```

Listing 6.62. nicht ändern!

**Steuerung für erweiterten Frequenzzähler**

```
#define COUNTER_CTRL_DIV PD4 /* prescaler (low 1:1, high x:1) */
#define COUNTER_CTRL_CH0 PD5 /* channel addr #0 */
#define COUNTER_CTRL_CH1 PD6 /* channel addr #1 */
```

Listing 6.63. ggf. anpassen

**Klingeltester**

- Eingang muss Pin PD7 / T0 sein (benutze COUNTER\_IN)

```
#define RINGTESTER_OUT PD4 /* pulse output */
```

Listing 6.64. ggf. anpassen

**L/C meter**

- Frequenz Eingang, welches an Pin PB0/T0 angeschlossen ist

```
#define LC_CTRL_CP PC4 /* reference cap (low: on / high: off) */
#define LC_CTRL_LC PC5 /* L/C selection (low: C / high: L) */
```

Listing 6.65. nicht ändern!

**IR-Detektor/Decoder**

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

```
#define IR_DATA PA0 /* data signal */
```

Listing 6.66. 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_RH PA5 /* Rh (470k) for fixed cap */
```

Listing 6.67. ggf. anpassen

## Summer

```
#define BUZZER_CTRL PA2 /* control pin (low: off / high: on) */
```

Listing 6.68. ggf.anpassen

## MAX6675

```
#define MAX6675_CS PA2 /* port pin used for /CS */
```

Listing 6.69. ggf.anpassen

## MAX31855

```
#define MAX31855_CS PA2 /* port pin used for /CS */
```

Listing 6.70. ggf.anpassen

## Relais für Parallelkondensator (Sampling-ADC)

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

```
#define CAP_RELAY_CTRL PA2 /* control pin */
```

Listing 6.71. ggf. anpassen

### 6.5.3. Busse

#### **SPI** - SCK PB1, MOSI PB2, MISO PB3 and /SS PB0

- Bit-Bang-SPI könnte bereits im Anzeigenbereich eingestellt sein

```
/* for bit-bang and hardware SPI */
#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 PB1 /* pin for SCK */
#define SPI_MOSI PB2 /* pin for MOSI */
#define SPI_MISO PB3 /* pin for MISO */
#define SPI_SS PB0 /* pin for /SS */
#endif
```

Listing 6.72. ggf. anpassen

#### **I2C** - Hardware-I2C (TWI) verwendet SDA PD1 and SCL PD0

- Bit-Bang-I2C könnte bereits im Anzeigebereich eingestellt sein

```
/* for bit-bang I2C */
#ifndef I2C_PORT
#define I2C_PORT PORTD /* port data register */
#define I2C_DDR DDRD /* port data direction register */
#define I2C_PIN PIND /* port input pins register */
#define I2C_SDA PD1 /* pin for SDA */
#define I2C_SCL PD0 /* pin for SCL */
#endif
```

Listing 6.73. ggf. anpassen

#### **serielle TTL-Schnittstelle**

- hardware USART verwendet

USART0: Rx PE0 und Tx PE1

USART2: Rx PH0 und Tx PH1

USART1: Rx PD2 und Tx PD3

USART3: Rx PJ0 und Tx PJ1

```
/* for hardware TTL serial */
#define SERIAL_USART 0 /* use USART0 */
/* for bit-bang TTL serial */
#define SERIAL_PORT PORTE /* port data register */
#define SERIAL_DDR DDRE /* port data direction register */
#define SERIAL_PIN PINE /* port input pins register */
#define SERIAL_TX PE1 /* pin for Tx (transmit) */
#define SERIAL_RX PE0 /* pin for Rx (receive) */
#define SERIAL_PCINT 8 /* PCINT# for Rx pin */
```

Listing 6.74. ggf. anpassen

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

```
#define ONEWIRE_PORT PORTA /* port data register */
#define ONEWIRE_DDR DDRA /* port data direction register */
#define ONEWIRE_PIN PINA /* port input pins register */
#define ONEWIRE_DQ PA4 /* DQ (data line) */
```

Listing 6.75. ggf. anpassen

---

## Kapitel 7      Sammlung von Einstellungen

---

für verschiedene Testermodelle.

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.

### 7.1. Arduino Nano, Uno oder Mega 2560

- Nano/Uno: ATmega 328, 16MHz clock      Mega 2560: ATmega 2560, 16MHz clock
- Lade das Arduino Schema und Pinout-Diagramm herunter, um die Zuordnung zwischen Arduino und seine Pins Belegung zu sehen. Siehe als Beispiel MEGA auf Seite 82.
- Der integrierte USB2serial-Adapter macht die serielle Kommunikation einfach (aktiviere SERIAL\_HARDWARE und die entsprechenden Optionen).
- Benutze **NICHT** die TXD/RXD-Pins nicht für etwas anderes.
- Wenn du den D13-Pin des Arduino verwenden möchtest, entferne die LED-Schaltung, die sonst stören würde.
- Hier sind nur die wesentlichen Einstellungen aufgeführt

```
#define HW_FIXED_SIGNAL_OUTPUT
```

Listing 7.1. Hardware Optionen

```
#define UI_AUTOHOLD
```

Listing 7.2. Benutzerschnittstelle

```
#define POWER_SWITCH_MANUAL
#define BAT_NONE
```

Listing 7.3. Leistungsregelung

```
#define ADC_LARGE_BUFFER_CAP /* 100nF cap at AREF */
```

Listing 7.4. Messeinstellungen und Offsets

### 7.2. DIY Kit „AY-AT“ auch GM328A

- ATmega328
- Farb-LCD-Modul ST7735 (Bit-Bang-SPI)
- Drehencoder (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
- Einstellungen für das semikompatible Display ST7735, bereitgestellt von b0hoon4@gmail.com

```
#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 7.5. HW optionen

```

#define ENCODER_A PD3 /* rotary encoder A signal */
#define ENCODER_B PD1 /* rotary encoder B signal */

```

Listing 7.6. Drehkoder

Der Eingang für den Frequenzzähler ist PD4 (T0)

```

#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_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_FLIP_X /* enable horizontal flip */
// #define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90Grad) */
// #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 7.7. LCD Modul

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

```

#define LCD_DOTS_X 160 /* number of horizontal dots */
#define LCD_DOTS_Y 128 /* number of vertical dots */
#define LCD_LATE_ON /* turn on LCD after clearing it */
#define FONT_8X8_HF /* 8x8 font */
#define SYMBOLS_30X32_HF /* 30x32 symbols */

```

Listing 7.8. semikompatible Display ST7735

Offsets der Induktivitätskompensation für 20MHz-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 7.9. Offsets der Induktivitätskompensation für 20MHz-Modell

### 7.3. BSide ESR02

- (DTU-1701) - ATmega 328, 8MHz clock
- ST7565 display (bit-bang SPI)
- Externe 2,5-V-Spannungsreferenz (TL431)
- bereitgestellt von indman@EEVBlog

```
#define HW_REF25
```

Listing 7.10. Hardware Optionen

```
#define BAT_DIVIDER
#define BAT_EXT_UNMONITORED
#define BAT_R1 47000
#define BAT_R2 47000
#define BAT_OFFSET 420
```

Listing 7.11. Pinbelegung zur Leistungsregelung

```
#define LCD_ST7565R /* display controller 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_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_X /* enable horizontal flip */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 15 /* 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 7.12. LCD Modul



## 7.4. Fish8840 TFT

- ATmega328, 8 MHz Takt; ST7565 Farbdisplay (Bit-Bang-SPI)
- Externe 2,5-V-Spannungsreferenz (TL431)
- Einstellungen von indman@EEVBlog/bdk100@vrtp.ru

```
#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_OFFSET_X /* enable x offset of 4 dots */
#define LCD_OFFSET_Y /* enable y offset of 2 dots */
#define LCD_FLIP_X /* enable horizontal flip */
//#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90Grad) */
#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 7.13. LCD Modul

## 7.5. GM328 !nicht GM328A!

welcher oben als 7.2 „AY-AT“ beschrieben ist.

- 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 7.14. LCD Modul

## 7.6. 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; (aber keine Eingangspufferstufe für direkte Frequenzmessung)
- Fester Kondensator für Selbstabgleich; (bei Problemen ersetze MLCC durch 220nF-Folienelko)
- 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
```

Listing 7.15. Hardware Optionen

```
#define NO_HFE_C_RL /* if hFE values too high */
```

Listing 7.16. Workarounds! **!Aktiviere NO HFE C RL!**

```
#define LCD_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_OFFSET_X /* enable x offset of 4 dots */
//#define LCD_FLIP_X /* enable horizontal flip */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_START_Y 0 /* start line (0-63) */
#define LCD_CONTRAST 3 /* 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 7.17. LCD Modul

```
#define POWER_CTRL PB1 /* controls power (1: on / 0: off) */
```

Listing 7.18. Pinbelegung zur Leistungsregelung

```
#define ENCODER_A PB5 /* rotary encoder A signal */
#define ENCODER_B PB7 /* rotary encoder B signal */
```

Listing 7.19. Pinbelegung Drehkoder

```
#define KEY_INC PB5 /* increase push button (low active) */
#define KEY_DEC PB7 /* decrease push button (low active) */
```

Listing 7.20. Pinbelegung für auf/ab Richtung

```
#define ADJUST_RH PC6 /* Rh (470k) for fixed cap */
```

Listing 7.21. Pinbelegung für Festkondensator zur Selbsteinstellung

## 7.7. M12864 DIY Transistor Tester

- ATmega328, 8 MHz Takt
- 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 7.22. 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 7.23. 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 PD3 /* rotary encoder A signal */
#define ENCODER_B PD1 /* rotary encoder B signal */
```

Listing 7.24. Drehkoder

```
#define KEY_PORT PORTD /* port data register */
#define KEY_DDR DDRD /* port data direction register */
#define KEY_PIN PIND /* port input pins register */
#define KEY_INC PD3 /* increase push button (low active) */
#define KEY_DEC PD1 /* decrease push button (low active) */
```

Listing 7.25. Pinbelegung für auf/ab Richtung

## 7.8. MK-328

- ATmega328, 8 MHz Takt; ST7565 Farbdisplay (Bit-Bang-SPI)
- Externe 2,5-V-Spannungsreferenz (TL431)
- Einstellungen von brunosso@EEVblog und bestätigt von Peeps@EEVblog

```
#define HW_REF25
```

Listing 7.26. 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 (optional) */
#define LCD_CS PD5 /* port pin used for /CS1 (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_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 */
```

## 7.9. T3/T4

Listing 7.27. LCD-Modul

- 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 */
```

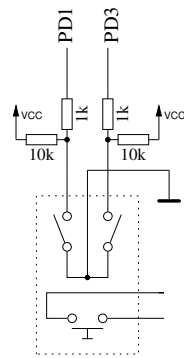
Listing 7.28. LCD Modul

Es gibt T4 Varianten, die andere Display Pinbelegung haben:

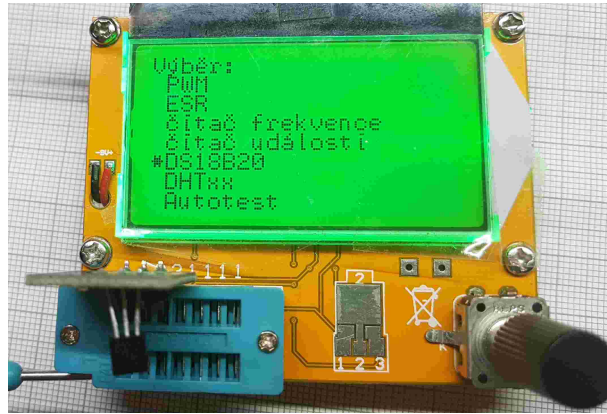
```
#define LCD_RESET PD0 /* port pin used for /RES */
#define LCD_A0 PD1 /* port pin used for A0 */
#define LCD_SCL PD2 /* port pin used for SCL */
#define LCD_SI PD3 /* port pin used for SI (LCD's data input) */
#define LCD_CS PD5 /* port pin used for /CS1 (optional) */
```

Listing 7.29. andere mögliche Display Einstellungen

Und umgebaute Variante mit Drehkoder, nach Karl-Heinz Kübbeler [4]:



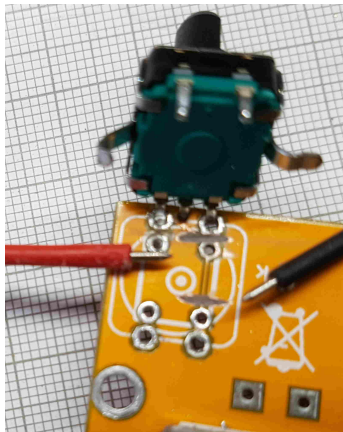
(a) Schema aus khk Buch



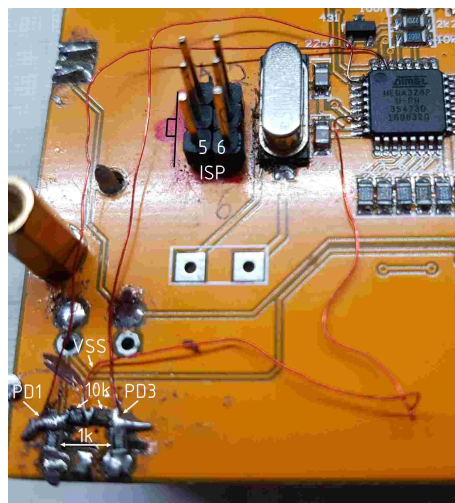
(b) Umgebautes T4 mit 6x8 ISO8859\_2 CZ Font.

Abbildung 7.1. T4 mit Drehkoder Bedienung.

Umbau Bilder:



(a) Leiterbahnunterbrechungen  
Vorderseite



(b) und auf der Rückseite

Abbildung 7.2. T4 Umbau. Drehkoder und ISP

Funktionierende Beispiel Konfiguration:

```
#define HW_ENCODER
#define ENCODER_PULSES 4
#define ENCODER_STEPS 24
#define HW_REF25
#define UREF_25 2495
#define HW_FREQ_COUNTER_BASIC
#define HW_EVENT_COUNTER
#define EVENT_COUNTER_TRIGGER_OUT
#define SW_PWM_SIMPLE
#define SW_INDUCTOR
#define SW_ESR
#define SW_DS18B20
#define SW_REVERSE_HFE
#define SW_DHTXX
#define UI_COMMA
#define UI_AUTOHOLD
#define UI_KEY_HINTS
#define POWER_OFF_TIMEOUT 30
#define SW_POWER_OFF
#define UI_ROUND_DS18B20
#define ONEWIRE_PROBES
```

Listing 7.30. Verwendete Einstellungen in config.h



## 7.10. Multifunktionstester TC-1 und Familie (T7) mit ATmega324/644

(sehr schlechte Pinbelegung); 16-MHz-Takt; ST7735 Farbdisplay (Bit-Bang-SPI)

- Externe 2,5-V-Spannungsreferenz (TL431)
- festes IR-Empfangsmodul; Boostkonverter für Zenertest  
(läuft die ganze Zeit, nicht standardmäßiger Spannungsteiler 100k/12k)
- Fester Kondensator für Selbstabgleich  
(bei Problemen MLCC durch 220nF-Filmkap. ersetzen)
- Stromversorgung durch Li-Ion-Akku 3.7V
- Testexemplare von jellytot@EEVblog und joystik@EEVblog
- erste Informationen von indman@EEVblog

### Hinweise:

- Die Steuer-MCU U4 (STC15L104W) muss durch eine einfache Schaltung mit zwei Transistoren ersetzt werden, (TC1-Mod, siehe Source-Repository für TC1-Mod.kicad.tgz, 5 $\mu$ A Standby Strom insgesamt) oder mit einer modifizierten Firmware neu programmiert werden (siehe <https://github.com/atar-axis/tc1-u4>). Der Bezeichner kann auch U3 (T7 Plus) oder U5 für neuere PCBs sein.

Die neueste PCB-Revision hat auch einen Footprint für eine alternative MCU (Details unbekannt). Die Hardware-Modifikation sollte trotzdem funktionieren.

- Setzen Sie das Byte für die erweiterte Sicherung auf 0xfd (Brown-out-Erkennung).
- Wenn D2 (Gleichrichterdioden für Zener-Testspannung) heiß wird, ersetzen Sie sie durch eine Schottky-Diode, die für 80 V Sperrspannung oder mehr ausgelegt ist, z. B. SS18.
- Ersetzen Sie C11 und C12 (Filterkappen für die Zener-Prüfspannung) durch eine 10 oder 22 $\mu$ F Elektrolytkondensator mit niedrigem ESR-Wert, der für 100 V oder mehr ausgelegt ist, wegen des MLCC DC Vorspannungs-Kapazitäts-Derating-Problem.
- Je nach verwendetem LCD-Modul müssen Sie möglicherweise LCD\_FLIP\_X anstelle von LCD\_FLIP\_Y.
- Der TC-1 kann kein Signal (PWM/Rechteckwelle/etc.) an Tastkopf 2 ausgeben. Verwenden sie PD4 (OC1B) als dedizierten Signalausgang (fügen Sie einen Widerstand zur Strombegrenzung hinzu) und aktivieren sie HW\_FIXED\_SIGNAL\_OUTPUT in config.h.
- Wenn Sie einen Drehgeber oder Tasten zum Ein- und Ausschalten hinzufügen möchten, verwenden sie bitte PB5 (D/C des Displays) und PB6 (SDA des Displays).
- Sie können auch den Frequenzzähler erhalten, indem Sie PB0 (T0) als Eingang verwenden und eine einfache Eingangsstufe hinzufügen.
- PD0 Lötbrücke (offen: pull-up, kurzgeschlossen: Gnd, unbenutzt durch m-firmware)

```
#define HW_REF25
#define HW_ZENER
#define ZENER_DIVIDER_CUSTOM
#define ZENER_R1 100000
#define ZENER_R2 12000
#define ZENER_UNSWITCHED
#define HW_IR_RECEIVER
#define HW_ADJUST_CAP
```

Listing 7.31. Hardware Optionen

```
#define HW_PROBE_ZENER
#define ZENER_VOLTAGE_MIN 1000 /* min. voltage in mV */
#define ZENER_VOLTAGE_MAX 40000 /* max. voltage in mV */
```

Listing 7.32. Verschiedene Einstellungen

```
#define IR_DATA PD3 /* data signal */
```

Listing 7.33. Pinbelegung für festen IR-Detektor / Decoder

```
#define ADJUST_RH PC6 /* Rh (470k) for fixed cap */
```

Listing 7.34. Pinbelegung für Festkondensator zur Selbsteinstellung

```

#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 7.35. Anschlussbelegung für Prüfspitzen

```

#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 7.36. Pinbelegung für Sondenwiderstände

```

#define POWER_CTRL PD2 /* controls power (1: on / 0: off) */

```

Listing 7.37. Pinbelegung zur Leistungsregelung

```

#define TEST_BUTTON PD1 /* test/start push button (low active) */

```

Listing 7.38. Pinbelegung für Testknopf

```

#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_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 LCD_FLIP_X /* enable horizontal flip */
#define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90Grad) */
#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 7.39. LCD Modul

## 7.11. Multifunktionstester T7 mit ATmega328

- ATmega328, 16MHz Takt
  - ST7735 Farbdisplay (Hardware SPI)  
/RESET-Leitung auf separatem MCU-Port
  - fest eingebautes IR-Empfängermodul
  - Hochsetzsteller für Zener-Prüfung  
(läuft die ganze Zeit, ca. 26V, nicht standardisierter Spannungsteiler 100k/12k)
  - Anschluss für serielle TTL-Schnittstelle (J5, 5V)
  - Einstellungen bereitgestellt von lhlad@EEVblog (siehe [https://gitlab.com/a11059/t7\\_328\\_m](https://gitlab.com/a11059/t7_328_m))
- Hinweise:**
- Dieses Modell kann keine Signale ausgeben (PWM/Squarewave/etc.).
  - Der Zweck der zusätzlichen MCU (STC15L104W) ist unbekannt.

```
#define HW_ZENER
#define ZENER_DIVIDER_CUSTOM
#define ZENER_R1 100000
#define ZENER_R2 12000
#define ZENER_UNSWITCHED
#define HW_IR_RECEIVER
```

Listing 7.40. Hardware Optionen

```
#define HW_PROBE_ZENER
#define ZENER_VOLTAGE_MIN 500
#define ZENER_VOLTAGE_MAX 25500
```

Listing 7.41. Verschiedene Einstellungen

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

Listing 7.42. Leistungsregelung

```
#define TP_ZENER PC3 /* test pin with 10:1 voltage divider */
#define TP_BAT PC4 /* test pin with 4:1 voltage divider */
```

Listing 7.43. Anschlussbelegung für Prüfspitzen

```
#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 7.44. Pinbelegung für Sondenwiderstände

```
#define POWER_CTRL PB1 /* control pin (1: on / 0: off) */
```

Listing 7.45. Pinbelegung zur Leistungsregelung

```
#define TEST_BUTTON PB4 /* test/start push button (low active) */
```

Listing 7.46. Pinbelegung für Testknopf

```
#define IR_DATA PB0 /* data signal */
```

Listing 7.47. Pinbelegung für festen IR-Detektor/Decoder



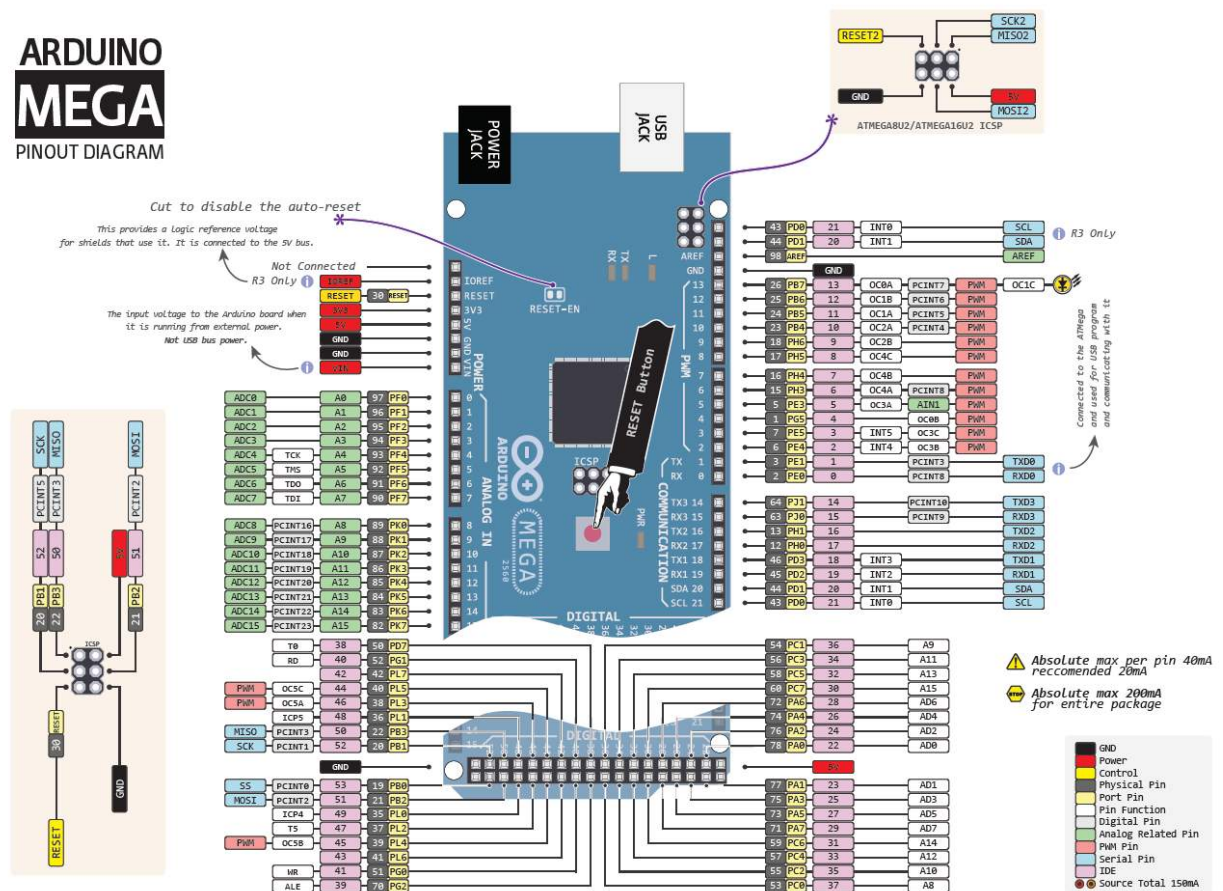
```

#define LCD_ST7735 /* display controller ST7735 */
#define LCD_GRAPHIC /* graphic display */
#define LCD_COLOR /* color display */
#define LCD_SPI /* SPI interface */
#define LCD_PORT PORTB /* port data register */
#define LCD_DDR DDRB /* port data direction register */
#define LCD_RES PC5 /* port pin used for /RESX (optional) */
#define LCD_RES_PORT PORTC /* separate port for /RESX (optional) */
#define LCD_RES_DDR DDRC /* separate port for /RESX (optional) */
// #define LCD_CS PB? /* port pin used for /CSX (optional) */
#define LCD_DC PB2 /* port pin used for D/CX */
#define LCD_SCL PB5 /* port pin used for SCL */
#define LCD_SDA PB3 /* 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_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 LCD_FLIP_X /* enable horizontal flip */
// #define LCD_FLIP_Y /* enable vertical flip */
#define LCD_ROTATE /* switch X and Y (rotate by 90Grad) */
// #define LCD_BGR /* reverse red and blue color channels */
#define LCD_LATE_ON /* turn on LCD after clearing it */
#define FONT_10X16_HF /* 10x16 font */
#define SYMBOLS_30X32_HF /* 30x32 symbols */
#define SPI_HARDWARE /* hardware SPI */

```

Listing 7.48. LCD Modul

## 7.12. Arduino MEGA Anschluss



---

## ***Kapitel 8 Programmieren des Component Testers***

---

Um die Verzweiflungen 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.**

### **8.1. Konfigurieren des Component Testers**

Dazu bitte Kapitel 1.8 auf Seite 9 lesen.

### **8.2. 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.

### **8.3. 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.

## 8.4. Benutzung unter Linux

auf neu installierten OS.

Für diejenigen, die nicht gerne schreiben, bietet Linux eine einfachere Möglichkeit.

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

Danach die Maus zum Namen des Dokumentes führen, hier linke Maustaste drücken und das Dokument bis zum linken Rand des Bildschirms ziehen, bis ein möglicher Rahmen angezeigt wird. Nun wird die Maus losgelassen.

Die Anleitung nimmt nun die linke Hälfte des Bildschirms ein.

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

## 8.5. Programm Pakete installieren

Nun brauchen wir einen Internetzugang.

Um den Tester zu programmieren, müssen zuerst Programmpakete installiert werden:

'binutils-avr', 'avrdude', 'avr-libc' und 'gcc-avr'. Weiter die Versionsverwaltung 'git'.

Jetzt in diesem Dokument zu dieser Seite, bis zu diesem Text navigieren:

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

den Text mit gedrückter linker Maustaste markieren, die Maus auf den Cursor des rechten Befehlsfensters 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.

Damit werden nun alle Software Pakete durch 'apt' heruntergeladen und 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!

Ob die Versionsverwaltung 'git' erfolgreich installiert wurde, kann man mit dem Kommando:

```
git version
```

überprüfen. Das Programm sollte mit der Ausgabe seiner Versionsnummer antworten.

## 8.6. Download der Quellen

und der Dokumentation aus dem Git-Archiv wird erreicht mit einer Anweisung:

```
git clone https://github.com/Mikrocontroller-net/transistortester
```

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

Die Kontrolle des Vorhandenseins. Terminal Fenster öffnen,

```
ls
```

und mit **Enter** oder **↵** bestätigen.

Um neue Updates heruntergeladen, reicht es in der Zukunft:

```
cd ~/tra
```

gefolgt von **Tab** und **↵** eingeben, und nun in diesem Verzeichnis

```
git pull
```

eingzugeben gefolgt von **↵**.

**Hinweis:** Bei Problemen oder Fehlermeldungen ist es am einfachsten, den vorhandenen Ordner 'transistortester' löschen oder umbenennen und dann mit der oben benannten Anweisung neu downloaden.

## 8.7. 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 bekannte Programmer nutzen zu können, ist 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}=="1ff8", 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'.

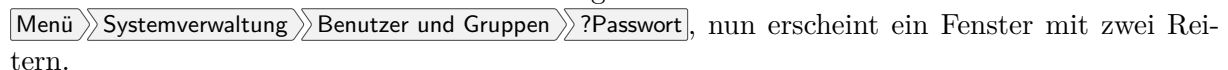
## 8.8. Gruppen Mitgliedschaft

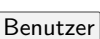
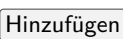
Deswegen sollte die eigene Benutzerkennung sowohl Mitglied der Gruppe 'plugdev' als auch der Gruppe 'dialout' sein. Das Kommando:

```
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  auf sein Name klickt, sieht man rechts sein Profil und die Gruppen Zugehörigkeiten. Mit dem Button  ist nun Möglich neue Gruppen hinzuzufügen.

## 8.9. Arbeitsumgebung


Vorbereitung.

Zu erst in der Taskleiste mit grünem Ordnersymbol (Nemo) zum /transistortester/Software/-Markus/ navigieren, mit der rechten Maustaste auf ComponentTester-1.(höchste Nr.)m.tgz klicken, in der Auswahl <hier entpacken> der Ordner Dekomprimieren und Nemo schließen. 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 und Kommando: 'ls' 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**  zu drücken: !

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

Das Verzeichnis des benötigten Models mit der Maus markieren.

Nun den blinkender Cursor mit Hilfe von  zur letztem Zeichen des Text 'MyT' positionieren und diese Zeichen löschen. Nach dem das letzte Zeichen gelöscht wurde, die  Taste der Maus drücken. Erst jetzt  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:

```
 +  + , cd  My  
```

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  
```

**Hier ist am wichtigsten, den VORHANDENEN USB-Programmer anmelden.**  
Siehe dazu im Kapitel 6.1.5, auf der Seite 46, Thema PROGRAMMER.

## 8.10. Firmware erzeugen und übertragen

Nach dem Editieren vom Makefile, config.h und config-<MCU>.h bitte „make“ ausführen oder Dein IDE, um die Firmware zu übersetzen. Als Ergebnis werden zwei Dateien erzeugt:

- ComponentTester.hex Firmware im Intel-Hex-Format
- ComponentTester.eep EEPROM-Daten im Intel-Hex-Format

Die Firmware wird in das Flash geschrieben und die EEPROM-Daten in das EEPROM.

Die Daten enthalten zwei Sätze an Standardabgleichswerten, Texte und Tabellen.

Wenn Du bei der Aktualisierung der Firmware die alten Abgleichswerte im EEPROM beibehalten möchtest, kann Du mit dem Schalter DATA\_FLASH in config.h die Texte und Tabellen in die Firmware verschieben.

In diesem Fall muss dann **nur** die Firmware in das Flash geschrieben werden; das EEPROM bleibt unverändert.

Das Makefile bietet folgende Targets:

|              |                                                 |
|--------------|-------------------------------------------------|
| make clean   | alle Objektdaten löschen                        |
| make         | zum Übersetzen des Programms                    |
| make fuses   | Fuse-Bits setzen (via avrdude)                  |
| make upload  | Firmware und EEPROM-Daten brennen (via avrdude) |
| make prog_fw | nur Firmware brennen (via avrdude)              |
| make prog_ee | nur EEPROM-Daten brennen (via avrdude)          |

Nun bleibt nur die Freude über das erreichter Erfolg.

**9.1. v1.47m 2022-07**

- Option zum Anzeigen eines Schwingquartzsymbols für LF/HF-Modi im erweiterten Frequenzzähler (UI\_QUARTZ\_CRYSTAL, Vorschlag von indman@EEVblog).
- Neue Zeichensätze font\_8x12t\_iso8859-2\_hf.h (FONT\_8X12T\_ISO8859\_2\_HF) und font\_8x16\_iso8859-2\_hf.h (FONT\_8X16\_ISO8859\_2\_HF, thanks to b0hoon@EEVblog).
- Korrektur von fehlerhafter Referenz für '\ ' in font\_8x8\_iso8859-2\_h.h, vermurkstem Diodensymbol A-C und 'W' in font\_6x8\_iso8859-2\_hf.h (Dank an b0hoon@EEVblog).
- Option zum Anzeigen eines Zenerdiodensymbols bei allen Zener-Tests (UI\_ZENER\_DIODE, Vorschlag von indman@EEVblog).
- Neuer Symbolsatz 32x39, ideal für Zeichensätze mit Höhe von 26 Pixel (SYMBOLS\_32X39\_HF, Dank an indman@EEVblog).
- Optionen zum Halten von ESR und Induktivität für C und L-Monitor (SW\_MONITOR\_HOLD\_ESR, SW\_MONITOR\_HOLD\_L, Vorschlag von theHWcav@EEVblog).
- Zeichenübersetzung für inverse Testpinnummern in Serial\_Char() ergänzt.
- ST7735-Treiber um Option für /RESX auf separatem Port erweitert (LCD\_RES\_PORT, LCD\_RES\_DDR, nach Idee von lhlad@EEVblog).
- Option zum Anzeigen eines Fragezeichensymbols bei fehlgeschlagenem Suchlauf (UI\_QUESTION\_MARK, Vorschlag von indman@EEVblog).
- Symbole für Fragezeichen, Zenerdiode und Quarz in allen Symbol-Sätzen ergänzt (Dank an indman@EEVblog).
- Zusätzliche Positionierungsmöglichkeiten für Bauteilsymbole (PIN\_CENTER, PIN\_ALT\_CENTER, Vorschlag von indman@EEVblog).
- Kleines Diodensymbol vor Vf für MOSFETs und IGBTs anzeigen (Vorschlag von indman@EEVblog).
- Zahlen 1-3 und x-Zeichen in invertierter Darstellung plus Grad-Zeichen in font\_8x16alt\_win1251\_hf.h ergänzt; sowie 1-3 und x in invertierter Darstellung in font\_8x8alt\_win1251\_vf.h aktualisiert. Das gleiche auch für font\_16x26\_win1251\_hf.h (Dank an indman@EEVblog).
- Option zum Abschalten der Ausgabe vom text-basierten Pinout der Body-Diode von MOSFETs (UI\_NO\_BODYDIODE\_TEXTPINOUT, Vorschlag von indman@EEVblog).
- Symbole 32x39 (SYMBOLS\_32X39\_HF, Dank an indman@EEVblog).
- Alternative Symbole 24x24 (SYMBOLS\_24X24\_ALT1\_H, SYMBOLS\_24X24\_ALT2\_H, SYMBOLS\_24X24\_ALT1\_HF, SYMBOLS\_24X24\_ALT2\_HF, SYMBOLS\_24X24\_ALT1\_VFP, SYMBOLS\_24X24\_ALT2\_VFP, SYMBOLS\_24X24\_ALT1\_VP\_F und SYMBOLS\_24X24\_ALT12\_VP\_F), 30x32 (SYMBOLS\_30X32\_ALT1\_HF und SYMBOLS\_30X32\_ALT2\_HF) und 32x32 (SYMBOLS\_32X32\_ALT1\_HF und SYMBOLS\_32X32\_ALT2\_HF, Dank an indman@EEVblog).
- Option für Bauteilsymbole: rechte Testpinnummern ober/unterhalb vom Symbol ausgeben (UI\_PINOUT\_ALT, Vorschlag von indman@EEVblog).
- In FontTest() Problem mit fehlenden Leerzeichen für manche Anzeigentreiber beseitigt.
- Testausgabe von Bauteilsymbolen (SW\_SYMBOL\_TEST, Vorschlag von indman@EEVblog).
- Russische Texte aktualisiert (Dank an indman@EEVblog).
- Fehlendes ')' -Zeichen in font\_10x16\_hf.h ergänzt und auch '\*' wieder hergestellt (gemeldet von pepe10000@EEVblog).

**9.2. v1.46m 2022-04**

- Option zum Abschalten des Text-Pinouts für 3-Pin-Halbleiter, wenn die Bauteilsymbole aktiviert sind (UI\_NO\_TEXTPINOUT, Vorschlag von bffargo@EEVblog).

- Pin-Bezeichner für MOSFET-Diode, IGBT-Freilaufdiode, BJT-Freilaufdiode, Basis-Emitter-Widerstand und Basis-Kollektor-Diode vom Schottky-Transistor sind nun auch farbkodiert.
  - Ausgabe der Symbolpinbelegung für symmetrische JFET verbessert (zeigt nun 'x' für Drain und Source).
  - Programmlogik und Ausgabe der Pin-Bezeichner für 3-Pin-Halbleiter überarbeitet.
  - Rumänische Texte aktualisiert (Dank an Dumidan@EEVblog).
  - Zeichensatz font\_8x12t\_win1251\_vfp.h aktualisiert (Dank an indman@EEVblog).
  - Bauteilsymbole überarbeitet. Kreise entfernt und Pin-Bezeichnung ergänzt ( Vorschlag von indman@EEVblog).
- Alte Symbole können weiterhin benutzt werden (setze SYMBOLS\_<size>\_OLD\_<format>.h in config\_<MCU>.h).
- Anzeige von Testpins in invertierter Darstellung (UI\_PROBE\_REVERSED, Vorschlag von Feliciano@EEVblog).
  - Zahlen 1-3 in invertierter Darstellung in mehreren Zeichensätzen ergänzt ( Dank an Feliciano@EEVblog).
  - Russische Texte aktualisiert (Dank an indman@EEVblog).
  - Spanische Texte aktualisiert (Dank an pepe10000@EEVblog).
  - Rumänische Texte aktualisiert (Dank an Dumidan@EEVblog).
  - Polnische Texte #2 aktualisiert (Dank an Jacon@EEVblog).

### 9.3. v1.45m 2021-12

- Neue Hardware-Optionen: Thermoelemente-Konverter MAX6675 und MAX31855 HW\_MAX6675, HW\_MAX31855, (Vorschlag von ricktendo@EEVblog).
- Optional drittes Profil für Abgleichwerte UI\_THREE\_PROFILES (Vorschlag von indman@EEVblog).
- Neue Funktion: Durchgangsprüfer SW\_CONTINUITY\_CHECK, (Vorschlag von indman@EEVblog).
- Neue Hardware-Option: akustisches Signal, d.h. aktiver Summer/Pieper HW\_BUZZER, (Vorschlag von indman@EEVblog).
- Neue Hardware-Option: Logiktester HW\_LOGIC\_PROBE, (Vorschlag von Dumidan@EEVblog).
- A-C Diodensymbol, Å und Ü in font\_12x16\_hf.h und font\_12x16\_iso8859-2\_hf.h korrigiert (gemeldet von Feliciano@EEVblog).
- Neue Hardware-Option: Ringtester (LOPT/FBT-Tester, HW\_RING\_TESTER).
- Option für Hauptmenü: entweder im Menü bleiben oder automatisch Menü verlassen (UI\_MAINMENU\_AUTOEXIT).
- Zeichensatz font\_8x8\_hf.h aktualisiert (Dank an Feliciano@EEVblog).

### 9.4. v1.44m 2021-08

- Ausgabe von EIA-96-Codes für Widerstands-E96-Normwerte SW\_R\_E96\_EIA96, (Vorschlag von Chris02@EEVblog).
- Unterstützung von 4-Draht-SPI in Treibern für ILI9481 und ILI9486 ergänzt.
- Timeout-Problem bei ST7920-Treiber beseitigt (gemeldet von ralleiner@mikrocontroller.net).
- Änderung der Entladefunktion: Bei einem Entladeproblem wird nun die unbelastete Spannung des DUTs gemessen und von der Entlade-Fehlermeldung angezeigt, z.B. die unbelastete Spannung einer Batterie.
- Verdrehtes Diodensymbol für zweite Diode bei Ausgabe von anti-parallelen Dioden korrigiert (gemeldet von horo@EEVblog).
- Option für low-aktives Backlight im HD44780-Treiber für PCF8574-I2C-Adapter ergänzt LCD\_BACKLIGHT\_LOW, (Vorschlag von Andreas Deimel).
- Fehlerhafte Symbole für p-Kanal-MOSFETs in 24x24-Symbol-Bitmaps korrigiert (gemeldet von Peeps@EEVblog).

### 9.5. v1.43m 2021-03

- R&D-Anzeigentreiber zur Identifizierung von Controllern.
  - Steuersignal LCD\_RD ist nun optional für ILI9341, ILI9481, ILI9486 und ILI9488.
  - Modifizierter Treiber für ST7735-teil-kompatible Anzeigen (Vorschlag von b0hoon4@gmail.com).
  - Option zum Ausführen des Zenertests während der normalen Bauteilesuche ( HW\_PROBE\_ZENER, Vorschlag von indman@EEVblog). Benötigt ZENER\_UNSWITCHED.
- Und entsprechendes Fernsteuerkommando ergänzt (V\_Z).
- Option für anwenderspezifischen Spannungsteiler für den Zenertest (ZENER\_DIVIDER\_CUSTOM, ZENER\_R1 und ZENER\_R2).
  - Option für niedrige Auflösung im Zenertest entfernt, und auch den Konfigurationsschalter ZENER\_HIGH\_RES.
  - Russische Texte aktualisiert (Dank an indman@EEVblog).
  - Optionale Anzeige der Pulsdauer für PWM-Generatoren (PWM\_SHOW\_DURATION, Vorschlag von Gast „hamburger“ @mikrocontroller.net).
  - Source-Archiv beinhaltet nun auch das Verzeichnis „dep“, um ein paar IDEs glücklich zu machen (Vorschlag von DAIRVINE@EEVblog).
  - Konfigurationsschalter SW\_PROBE\_COLORS zu UI\_PROBE\_COLORS umbenannt.
  - Überschriften können andersfarbig ausgegeben werden (UI\_COLORED\_TITLES).
- Ebenfalls Cursor und Bedienungshinweise (UI\_COLORED\_CURSOR).
- Rumänische Texte aktualisiert (Dank an Dumidan@EEVblog).
  - Alternative polnische Texte aktualisiert (Dank an Jacon@EEVblog).
  - Spanische Texte aktualisiert (Dank an pepe10000@EEVblog).

### 9.6. v1.42m 2020-12

- Aktualisierte 6x8-Zeichensätze (Dank an Bohu).
  - Neuer 6x8-Zeichensatz nach ISO8859-2 (FONT\_6X8\_ISO8859\_2\_HF, Dank an Bohu).
  - Neue Hardware-Option: LC-Meter (HW\_LC\_METER). Basiert auf einer einfachen Schaltung mit einem LC-Oszillator, ähnlich den preiswerten LC-Meter- Bausätzen.
  - Display\_Value() unterstützt nun auch Fempto (f).
  - Wenn CYCLE\_MAX auf 255 gesetzt ist, wird die Abschaltautomatik deaktiviert und der Tester läuft solange, bis er manuell abgeschaltet wird.
  - Problem mit Anzeige seltsamer Frequenzen im erweiterten Frequenzzähler in einer bestimmten Konstellation beseitigt (gemeldet von Szybkijanek@EEVblog).
- Ebenfalls Anzeige bei fehlendem Signal oder zu niedriger Frequenz auf „0Hz“ auf „-“ geändert.
- Der einfache Frequenzzähler zeigt nun ein „-“ statt „0Hz“ an, wenn kein Signal anliegt oder die Frequenz zu niedrig ist.
  - Option zur Ausgabe des Teststroms I\_C bzw. I\_E der hFE-Messung ( SW\_HFE\_CURRENT, Vorschlag von Obelix2007@EEVblog),
- und entsprechende Fernsteuerkommandos ergänzt (I\_C und I\_E).
- Rumänische Texte (Dank an Dumidan@EEVblog).
  - Konfigurationsschalter für ESR-Tool (SW\_ESR\_TOOL, Vorschlag von indman@EEVblog).
  - Textausgabe von E-Norm-werten gibt nun auch Einheit aus (Vorschlag von indman@EEVblog).
  - Alternative polnische Texte aktualisiert (Dank an Jacon@EEVblog).
  - Letzte Textzeile auf dem Display wurde einfach überschrieben, wenn UI\_KEY\_HINTS aktiviert ist (gemeldet von Obelix2007@EEVblog).
- Funktion zum warten auf Benutzer vorm Löschen der letzten Zeile eingebaut.
- Spanische Texte aktualisiert (Dank an pepe10000@EEVblog).
  - Fehler in „#define“ für E96-Norm-Werte korrigiert (gemeldet von Obelix2007@EEVblog).
  - Treiber für Display mit ILI9488 8/16-Bit Parallel und 4-Draht-SPI, (Dank an Bohu für ein Testexemplar).

### 9.7. v1.41m 2020-09

- Neues Fernsteuerkommando „MHINT“ für Hinweise zu Messungen (momentan nur Typ der



Testschaltung für h\_FE).

- Auto-Modus (automatische Aktualisierung) für DS18B20 eingebaut. Vorschlag von (Obelix2007@EEVblog).
- Zusätzliche Endladefunktion zum Beseitigen eines ESR-Problems bei manchen Testerclones (gemeldet von indman@EEVblog). ESR-Wert war viel zu hoch für Elko  $\geq 470\mu\text{F}$  an Testpins #1 und #2, im ESR-Tool, C-Monitor und RCL-Monitor. Und für Elko  $\geq 4700\mu\text{F}$  in allen Modi. Betroffene Tester sind Hiland M644 und TC-1-Varianten.
- Grad-Zeichen in font\_16x26\_hf.h und font\_16x26\_iso8859-2\_hf.h aktualisiert (Änderungen von Bohu).
- Treiber für OLED-Controller SH1106 (3-Wire SPI, 4-Wire SPI und I2C; Dank an Old-Papa für ein Test-Display).
- Fehlende Initialisierung des /RES-Signals für I2C im SSD1306-Treiber ergänzt.
- Ausgabe des Typs der Testschaltung für hFE.
- Ausgabe von Hexadezimalwerten auf Kleinbuchstaben geändert, und Schalter zum Aktivieren von Grossbuchstaben ergänzt (UI\_HEX\_UPPERCASE).
- Option für OneWire-Funktionen zum zusätzlichen Auslesen und Anzeigen des ROM-Codes (ONEWIRE\_READ\_ROM, Vorschlag von indman@EEVblog).
- DQ und Vcc-Testpins für OneWire-Bus passend zur Pinbelegung von Sensoren vertauscht (Vorschlag von indman@EEVblog).
- Erkennung von Schottky-Transistoren (SW\_SCHOTTKY\_BJT).
- Verbesserung der Erkennung von FET-Verarmungstypen mittels Filter für Schottky-Transistoren.
- Textzeilenverwaltung zum Löschen der Anzeige optimiert.
- Spezieller Display-Treiber zum Identifizieren von Display-Controllern.
- R/C/L-Monitor (SW\_MONITOR\_RCL, Vorschlag von indman@EEVblog).
- Problem mit wechselnden Induktivitätswerten im L und R/L-Monitor gelöst ( gemeldet von indman@EEVblog).
- Messung von V\_GS(off) für Verarmungstyp-FETs (Vorschlag von joshto@EEVblog). Fernsteuerkommandos um den Befehl "V\_GS\_off" erweitert.
- ILI9341-Treiber um Unterstützung von 8-Bit-Parallel-Bus erweitert. Außerdem Konfigurationsschalter für ILI9341 mit deaktiviertem erweiterten Befehlssatz ergänzt (LCD\_EXT\_CMD\_OFF, Dank an Bohu für ein Testexemplar).
- Kyrillischer 16x26-Zeichensatz (FONT\_16X26\_WIN1251\_HF, Dank an Yuriy\_K@VRTP.RU).
- Problem mit fehlendem  $\mu$  (micro) in mehreren ISO8859-2-Zeichensätzen behoben (gemeldet von indman@EEVblog und Obelix2007@EEVblog).
- Zahlendreher bei ATmega 2560 im Makefile korrigiert (gemeldet von Bohu).
- Farbiger Batteriestatus für Farbanzeigen (Vorschlag von indman@EEVblog).
- Alternativen 8x16-Win1251-Zeichensatz durch aktualisierte Version von indman@EEVblog ersetzt.
- Problem mit der Gültigkeitsprüfung von kleinen Widerständen in CheckResistor() beseitigt (gemeldet von indman@EEVblog).
- SmallResistor(), MeasureInductance() und GetGateThreshold() für die Unterstützung von ADC\_LARGE\_BUFFER\_CAP aktualisiert.
- Fehlende Aktualisierung der ADC-Referenzquelle in MeasureInductance() und GetGateThreshold() ergänzt.
- Option zur Auswahl des Abgleichprofils nach dem Einschalten UI\_CHOOSE\_PROFILE, (Vorschlag von Bohu).
- Eigene Monitorfunktion für Widerstände und Induktivitäten  
SW\_MONITOR\_R, SW\_MONITOR\_L, (Vorschlag von indman@EEVblog).
- Spanische Texte aktualisiert (Dank an pepe10000@EEVblog).

## 9.8. v1.40m 2020-06

- Treiber für Displays mit ILI9481 oder ILI9486.
- Logikfehler für Farbanzeigen bei abgeschalteter Farb-Funktion beseitigt.
- Ergänzung von Konfigurationsschalter für Farbanzeigen zum Vertauschen der Farbkanäle für

Rot und Blau (LCD-\_BGR).

- R/L-Monitor verbessert, um Streuung der Messwerte zu reduzieren. Hilft nicht in allen Fällen. (gemeldet von indman@EEVblog).
  - Problem mit “#ifdef” für Funktionen Display\_HexByte() and Display\_HexDigit() beseitigt (gemeldet von AlcidePiR2@EEVBlog).
  - Unterstützung von ATmega 640/1280/2560.
  - Die Erkennung der Drehrichtung in ReadEncoder() war umgedreht. Auf korrekte Richtung geändert und Einstellungen in config\_<MCU>.h und Clones aktualisiert.
  - Initialisierungsproblem bei Hardware-SPI beseitigt.
  - Alternativer Modus für Zenertest, wenn Boost-Konverter ständig läuft oder gar nicht vorhanden ist (ZENER\_UNSWITCHED, (Vorschlag von indman@EEVblog).
  - Konfigurationsschalter für 100nF AREF-Pufferkondensator anstatt 1nF (ADC\_LARGE\_BUFFER\_CAP). Wird für bestimmte MCU-Platinen benötigt.
  - Alternative kyrillische Fonts 8x16 (FONT\_8X16ALT\_WIN1251\_HF) und 8x8 (FONT\_8X8ALT\_WIN1251\_VF, Dank an indman@EEVblog).
  - Russische Texte aktualisiert (Dank an indman@EEVblog).
  - Option zum Nutzen eines einfachen Ein/Aus-Schalters anstatt des standard- mäßigen Soft-Latch-Schalters (POWER\_SWITCH\_MANUAL).
  - Erkennung von zwei kurzen Berührungen der Mitte in ReadTouchScreen() ergänzt.
  - Algorithmusfehler bzgl. TOUCH\_FLIP\_X/TOUCH\_FLIP\_Y in Touch\_CharPos() beseitigt (gemeldet von Bohu).
  - Mehrere ISO8859-2-Zeichensätze durch aktualisierte Versionen von Bohu ersetzt.
  - Option zum Prüfen von Widerständen auf E-Normwerte (SW\_R\_E\*).
- Auch für Kondensatoren (SW\_C\_E\*) und Induktivitäten (SW\_L\_E\*).

### 9.9. v1.39m 2020-03

- Polnische Texte nach ISO 8859-2 (Dank an Jacon).
- Problemchen mit Preprocessor-Macro in SPI.c und Syntaxfehler in ADS7843.c behoben (gemeldet von Bohu).
- Option zum Speichern von Firmwaredaten im Flash anstatt EEPROM DATA\_FLASH, (Vorschlag von Vitaliy).
- Kyrillische Zeichensätze zu “win1251” umbenannt and tschechische Zeichensätze durch ISO8859-2-Zeichensätze ersetzt (Dank an Bohu).
- Funktion zur Ausgabe des Zeichensatzes zu Testzwecken (SW\_FONT\_TEST).
- OneWire-Scan zur Ausgabe der ROM-Codes der Busteilnehmer (SW\_ONEWIRE\_SCAN).
- Option für spezifischen Widerstandsoffset je Testpin-Paar (R\_MULTIOFFSET, Vorschlag von Vitaliy).

### 9.10. 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 (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).

### 9.11. 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).

### 9.12. 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.

### 9.13. 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  $\Omega$  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.

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

- Leckstromtest für Kondensatoren.
- Standardwert für RH\_OFFSET auf 350  $\Omega$  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.

### 9.15. 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.

### 9.16. 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.

### 9.17. 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.

### 9.18. 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.

### 9.19. 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.

### 9.20. 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).

### 9.21. 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 Pulsweite. 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.

### 9.22. 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.

### 9.23. 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 Tester Versionen bzw. Clone.
- Kleines Problem mit den 24x24-VP Symbolen in config.h beseitigt. Hinweis von lordstein@EEVblog und hapless@EEVblog.

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

- Messung vom Selbstendladungsleckstrom von Kondensatoren größer  $4,7\mu F$ .
  - 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. v1.23m 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 (Unijunction Transistor) eingebaut.
- Dank an edavid@EEVblog für das Zusenden von ein paar UJT zum Testen.
- Kleinere Optimierungen für ILI9341 und 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.

### 9.26. 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.

### 9.27. 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.

### 9.28. 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.

### 9.29. 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.

### 9.30. 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 Abgleich-werte 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“.

### 9.31. v1.17m 2015-02

- Verbesserung von CheckDiode(). Gemessenes Vcc wird bei der Widerstands- prüfung beachtet. Außerdem Erkennungsproblem von Widerständen um die 2 kΩ bei optionalem DC-DC-Konverter (HW\_ZENER) gelöst.
- Fehlerhafte Kommentare korrigiert.
- Integer-Datentypen aufgeräumt.

### 9.32. 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.

### 9.33. v1.15m 2014-09

- Erweiterung von TestKey() um die Erkennung der dynamischen Drehgeschwindig- keit 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).

### 9.34. v1.14m 2014-08

- Benutzerschnittstelle für Drehencoder angepasst.
  - Compiler-Warnung bzgl. R\_Pin2 in ShowDiode() beseitigt (Dank an Milan Petko).
  - Widerstände zwischen 1,5 kΩ und 3 kΩ 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).

### 9.35. 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).

### 9.36. 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.

### 9.37. 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<sub>BE</sub> 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.
- V<sub>th</sub> ist nun negativ für P-Kanal FETs.
- Messung von V<sub>GT</sub> für Thyristoren und Triacs.
- Wegen wachsender Firmwaregröße gibt es den PWM-Generator nur noch für den ATmega328.

### 9.38. 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.

### 9.39. 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.

### 9.40. 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.

### 9.41. v1.07m 2013-06

- Diodenausgabe optimiert und Anzeige von V<sub>f</sub> 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-Überlauferfehler 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.

#### **9.42. 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 Tastendrucke 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).
- Endladefunktion 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.

#### **9.43. 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.

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

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

#### **9.45. 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 :-)

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

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

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

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

#### **9.48. v1.00m 2012-09**

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

#### 9.49. v0.99m 2012-09

- Die erste veröffentlichte Version basierend auf Karl-Heinz.

---

## *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/madires/Transistortester-Warehouse>  
*Komplette Software Sammlung*
- [6] <https://github.com/kubi48/TransistorTester-source/tree/master/Markus>  
*Komplette Software Sammlung*
- [7] <https://github.com/Mikrocontroller-net/transistortester/tree/master/Software/Markus>  
*Komplette Software Sammlung*
- [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*,
- [10] <https://www.mikrocontroller.net/topic/248078>  
*Hauptsprache ist Deutsch, Englisch ist aber auch ok.*
- [11] [https://www.eevblog.com/forum/testgear/\\$20-lcr-esr-transistor-checker-project/](https://www.eevblog.com/forum/testgear/$20-lcr-esr-transistor-checker-project/)  
*Nur Englisch.*
- [12] <https://vrtp.ru/index.php?showtopic=16451> - Russisches Forum  
*Infos zu verschiedenen Clonen, wie Bilder, Schaltpläne und Firmware*
- [13] <https://disk.yandex.ru/d/yW8xa5NJgUo5z> - Russische Seite (von indman@EEVblog)  
*Infos zu verschiedenen Clonen, wie Bilder, Schaltpläne und Firmware*
- [14] [https://drive.google.com/file/d/1-IJA8uTcsCA\\_6SYHEuMydjfS2vNgmwdH/edit](https://drive.google.com/file/d/1-IJA8uTcsCA_6SYHEuMydjfS2vNgmwdH/edit)  
*Anleitung plus div. Dateien für WinAVR (von indman@EEVblog)*
- [15] <https://github.com/kubi48/TransistorTester-source>  
*Aktuelle k-Firmware*
- [16] <https://github.com/kubi48/TransistorTester-documentation>  
*Aktuelle Dokumentation für k-Firmware*