

# Leitfaden zum GM328A einem Instrument zum identifizieren und messen von elektronischen Bauteilen und elektrischen Einheiten.

Version 1.13k  
aber auch  
1.52m

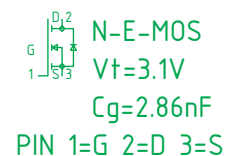
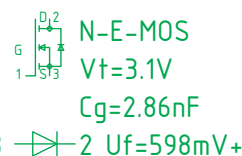
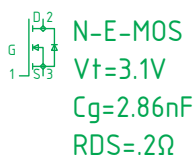
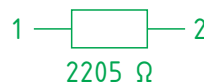
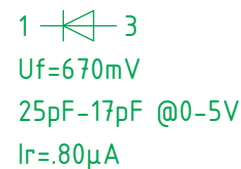
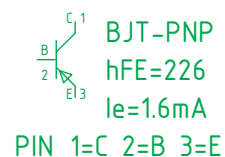
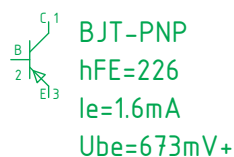
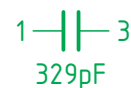
Karl-Heinz Kübbeler  
kh\_kuebbeler@web.de

&

Markus Reschke  
madires@theca-tabellaria.de

Zusammengestellt  
von bm-magic

12. Mai 2024



13.05.2019/MOR

---

## *Inhaltsverzeichnis*

---

<b>1</b>	<b>Grundlagen</b>	<b>6</b>
1.1	Einleitung . . . . .	6
1.1.1	HINWEIS! . . . . .	6
1.2	Sicherheitshinweise . . . . .	6
1.3	Lizenz . . . . .	6
1.3.1	Zusätzliche Hinweise zur Lizenz . . . . .	6
1.4	Versions Unterschiede . . . . .	6
1.4.1	Spezifikation . . . . .	6
<b>2</b>	<b>Hardware</b>	<b>7</b>
2.1	Beschreibung . . . . .	7
2.2	Die Bedienung . . . . .	8
2.2.1	Taster . . . . .	8
2.2.2	Drehencoder . . . . .	8
<b>3</b>	<b>Eigenschaften</b>	<b>9</b>
3.0.1	Auswahlmenü . . . . .	11
3.1	Wichtige Bemerkungen . . . . .	11
3.2	Problemfälle . . . . .	11
<b>4</b>	<b>Funktionsmenü in der k-Version</b>	<b>12</b>
4.1	Optionale Menüfunktionen . . . . .	12
4.2	Selbsttest und Kalibration . . . . .	15
<b>5</b>	<b>Funktionsmenü in der m-Version</b>	<b>17</b>
5.0.1	Einschalten . . . . .	17
5.0.2	Bauteilesuche . . . . .	17
5.0.3	Batterieüberwachung . . . . .	17
5.0.4	Ausschalten . . . . .	18
5.1	Menü . . . . .	18
5.1.1	PWM-Generator . . . . .	18
5.1.2	Einfache PWM . . . . .	18
5.1.3	Erweiterte PWM . . . . .	19
5.1.4	Rechteck-Signalgenerator . . . . .	19
5.1.5	Voltmeter und Zenertest . . . . .	19
5.1.6	Logiktester (Hardware-Option) . . . . .	20
5.1.7	Durchgangsprüfer (Hardware-Option) . . . . .	20
5.1.8	ESR-Tool . . . . .	20
5.1.9	Kondensatorleckstrom . . . . .	21
5.1.10	R/C/L-Monitore . . . . .	21
5.1.11	L/C-Meter (Hardware-Option) . . . . .	21
5.1.12	Frequenzzähler (Hardware-Option) . . . . .	22
5.1.13	Einfacher Zähler . . . . .	22

5.1.14	Erweiterter Zähler . . . . .	22
5.1.15	Klingeltester (Hardware-Option) . . . . .	22
5.1.16	Ereigniszähler (Hardware-Option) . . . . .	23
5.1.17	Drehencoder . . . . .	23
5.1.18	Kontrast . . . . .	24
5.1.19	Detektor/Decoder für IR-Fernbedienungen . . . . .	24
5.1.20	IR-Fernbedienung . . . . .	25
5.1.21	Opto-Koppler-Test . . . . .	26
5.1.22	Fotodioden-Test . . . . .	26
5.1.23	Dioden/LED-Schnelltest . . . . .	27
5.1.24	Modellbau-Servo-Test . . . . .	27
5.1.25	OneWire-Scan . . . . .	27
5.1.26	DS18B20/DS18S20 OneWire Temperatursensoren . . . . .	28
5.1.27	DHTxx-Temperatur & Luftfeuchte-Sensoren . . . . .	28
5.1.28	MAX6675/MAX31855 . . . . .	29
5.1.29	BH1750 Umgebungslichtsensor . . . . .	29
5.1.30	Licht . . . . .	29
5.1.31	Selbsttest . . . . .	29
5.1.32	Selbstabgleich . . . . .	30
5.1.33	Speichern/Laden . . . . .	31
5.1.34	Werte anzeigen . . . . .	31
5.1.35	Zeichensatz/Symbole . . . . .	31
5.1.36	Ausschalten . . . . .	31
5.1.37	Exit . . . . .	31
5.2	Die Mögliche Optionen in der config.h . . . . .	32
5.2.1	Hardware Bedienung . . . . .	32
5.2.2	Software Optionen . . . . .	36
5.2.3	Umgehungslösungen für einige Tester . . . . .	38
5.2.4	Umgehungslösungen für einige IDEs . . . . .	39
5.2.5	Benutzerschnittstelle . . . . .	39
5.2.6	Energieverwaltung . . . . .	42
5.2.7	Messeinstellungen und Offsets . . . . .	43
5.2.8	F & E - gedacht für Firmware-Entwickler . . . . .	44
5.2.9	Busse . . . . .	44
<b>6</b>	<b>Programm Code</b>	<b>45</b>
6.1	In der k-Version . . . . .	45
6.2	In der m-Version . . . . .	45
6.3	Makefile . . . . .	45
6.3.1	MCU-Typ . . . . .	45
6.3.2	MCU-Taktfrequenz . . . . .	45
6.3.3	Oszillator-Typ . . . . .	45
6.3.4	Avrdude MCU-Typ . . . . .	46
6.3.5	Avrdude ISP-Programmierer . . . . .	46
6.4	config.h . . . . .	47
6.4.1	Für diesen Tester muss geändert werden: . . . . .	47
6.4.2	Nun folgt die Qual der Wahl . . . . .	47
6.5	Config_328.h . . . . .	50
6.5.1	Notwendige Änderungen . . . . .	50
6.5.2	Information . . . . .	51
<b>7</b>	<b>Programmieren des Testers</b>	<b>52</b>
7.1	Konfigurieren des Component Testers . . . . .	52
7.2	Programmierung des Mikrocontrollers . . . . .	52
7.3	Betrieb System Linux . . . . .	52

7.4	Benutzung unter Linux . . . . .	53
7.5	Programm Pakete installieren . . . . .	53
7.6	Download der Quellen . . . . .	53
7.7	Benutzung der Schnittstellen . . . . .	53
7.8	Gruppen Mitgliedschaft . . . . .	54
7.9	Arbeitsumgebung . . . . .	54
7.10	Firmware erzeugen und übertragen . . . . .	55
7.11	Arbeitsumgebung mit der k-Version . . . . .	56
7.12	Nötige Hardware zum Programmieren . . . . .	56
7.12.1	Programmer . . . . .	56
<b>8</b>	<b>Technische Daten</b>	<b>57</b>
8.1	Hilfe . . . . .	57
8.2	Und für Entspannung . . . . .	57
8.3	Schema GM 328 A . . . . .	58

## Vorwort

**Grundsätzliches** Jeder Bastler kennt das folgende Problem: Man baut einen Transistor aus oder man nimmt einen aus einer Bastelkiste. Wenn man die Typenbezeichnung erkennen kann und man bereits ein Datenblatt hat oder eins bekommen kann, ist alles in Ordnung. Aber wenn man keine Datenblätter findet, hat man keine Idee, was das für ein Bauteil sein kann. Mit den konventionellen Messmethoden ist es schwierig und zeitaufwändig den Typ des Bauteils und dessen Parameter herauszufinden. Es könnte ein NPN, PNP, N- oder P-Kanal-MOSFET usw. sein. Es war die Idee von Markus F., diese Arbeit von einem AVR-Mikrocontroller erledigen zu lassen.

**vorhergehende Sätze** sind abgeschrieben aus der Transistor Tester Einleitung von Karl-Heinz Kübbeler. Eigentlich stammt ein großer Teil dieses Dokuments aus Auszügen seiner Arbeit, ...**bei dem ich mich hiermit bedanken möchte** ...

**Mein Kontakt mit diesem Tester** war rein zufällig. Als ich billige Teile für mein neues Projekt suchte, fand ich ihn bei chinesischen Lieferanten zu einem Preis, der unter dem hiesigen Preis für das Display lag. Nachdem ich den Tester bekam und ich den Batterieanschluss verlötet hatte, war ich überhaupt nicht überrascht, als das Display dunkel blieb. Der Verkäufer, den ich kontaktiert habe, hat mir eine ...-Adresse gesendet:

<https://www.youtube.com/watch?v=0bfxyy1K3po>, und als ich sah, wie der Prüfer den Encoder drückte, schämte ich mich ... und war gleichzeitig

**erstaunt** was diese kleine Platinchen alles kann. Beim ersten Einschalten wurde ich zu einer Kalibrierung geführt, die so organisiert ist, dass kein Fehler gemacht werden kann. Weiter wurde mir auch Folgendes mitgeteilt: ... [svn://mikrocontroller.net/transistortester](https://mikrocontroller.net/transistortester) wo ich die komplette Dokumentation bekommen habe, in der ich gelesen habe, dass es verschiedene Bedienungssprachen gibt.

**In meinem jugendlichen Leichtsinn** habe ich beschlossen, ihm deutsch beizubringen. An dieser Stelle sollte ich hinzufügen, dass ich beim Kauf des Testers am Anfang 2018 73 Jahre alt war.

- Eigentlich begann es, als ich mit 72 beschloss, dass es Zeit ist, Programmieren zu lernen. Vor über 30 Jahren habe ich (zu dieser Zeit in Eile) einen Ereignis Zähler gebaut, der mir immer noch einen guten Service bietet. Wie es aussieht, lebt Notbehelf am längsten ... dennoch, habe ich entschieden, als mein erstes Projekt diesen Zähler in Software zu entwerfen. Ich habe einen AVR-Kurs mit einem Kit gekauft und abgearbeitet, aber wie die meisten Kurse endete es mit LEDs einzuschalten und mit denen eine Ampel für Fußgänger zu machen. Bei der ersten Problemen, wandte ich mich an den Autor (khk), der mich geduldig (innerhalb von ca. drei Monaten) in ca. 50 Mails soweit gebracht hatte, dass der Tester sogar Tschechisch sprechen kann. Aus Dankbarkeit versprach ich, seine Dokumentation ins Tschechische zu übersetzen. In weiteren E-Mails stellte mir khk LaTeX vor, in dem diese Dokumentation geschrieben ist.

**Die tschechische Übersetzung** ... ..., stellte ich mir vor, wie eine Puppe einen Krieg. Schon nach den ersten fünf Sätzen wurde mir klar, dass vor 50 Jahren als in nach Deutschland kam, keine Informatik gab und dass ich, alle Fachbegriffe NUR in Deutsch oder Englisch kann. Der Versuch, mit Google zu übersetzen, verlief sehr schlecht. Ich brauchte also für 130 Seiten fast ein Jahr. Als ich fertig war, hatte khk aus persönlichen Gründen keine Zeit, deshalb hat er meine Arbeit noch nicht veröffentlicht ...

**Software für den Tester** entwickelt von Anfang an ein weiterer Entwickler, Markus Reschke. Seine Software ist sehr interessant, aber die Konfiguration ist nicht so gut beschrieben. (Es gibt "nur-\*.txt und das noch meistens in Englisch).

- Um sie besser zu verstehen, habe ich sie nach \*.pdf konvertiert. Der Autor hat die Konvertierung auf seiner [3] Website veröffentlicht.

**...auch diesem Entwickler möchte ich hiermit danken...**

## 1.1. Einleitung

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

Beiden Entwickler haben ihre Werke gut dokumentiert.

In folgendem werden Auszüge aus ihren Dokumenten verwendet.

Bitte, lies bestimmt die Originale.

**1.1.1. HINWEIS!** Leider kann der Originalzustand der chinesischen Software-Variante nicht gesichert werden, da die Sicherheits-Bits des ATmega328 gesetzt sind. Es gibt also keinen Weg zurück zur Originalversion der Software.

Vorschlag: Da dieses Model eine DIP Version des ATmega328P auf Sockel verwendet, kannst Du das Original aufbewahren und, je nach Einsatz, verschiedene Konfigurationen auf neue ATmega328P brennen. ;-)

## 1.2. Sicherheitshinweise

Der Tester ist kein Multimeter!

Es ist ein einfacher Tester für Bauteile, der verschiedene Parameter 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 beide Entwickler den ursprünglichen Autor bisher nicht erreichen können.

Um das Problem des Fehlens einer vernünftigen Open-Source-Lizenz zu lösen, wurde 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 zur Lizenz

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

## 1.4. Versions Unterschiede

Während die Firmware von Karl-Heinz die offizielle Version ist und auch ältere ATmega MCUs unterstützt, ist die Markus Version zum Ausprobieren und Testen neuer Ideen gedacht. Außerdem ist sie auf ATmegas mit mindestens 32kB Flash beschränkt.

**1.4.1. Spezifikation** Beide Versionen sind für den Einsatz in verschiedenen Testern mit unterschiedlicher Hardware konzipiert. Einige Optionen können in diesem Tester nicht verwendet werden, zumindest nicht ohne Hardwaremodifikation.

Andererseits bietet die Software so viele Möglichkeiten, dass sie die Kapazität des ATmega-Speichers übersteigt.

Es ist also meistens nicht möglich, gleichzeitig alle Optionen zu testen.

## 2.1. Beschreibung

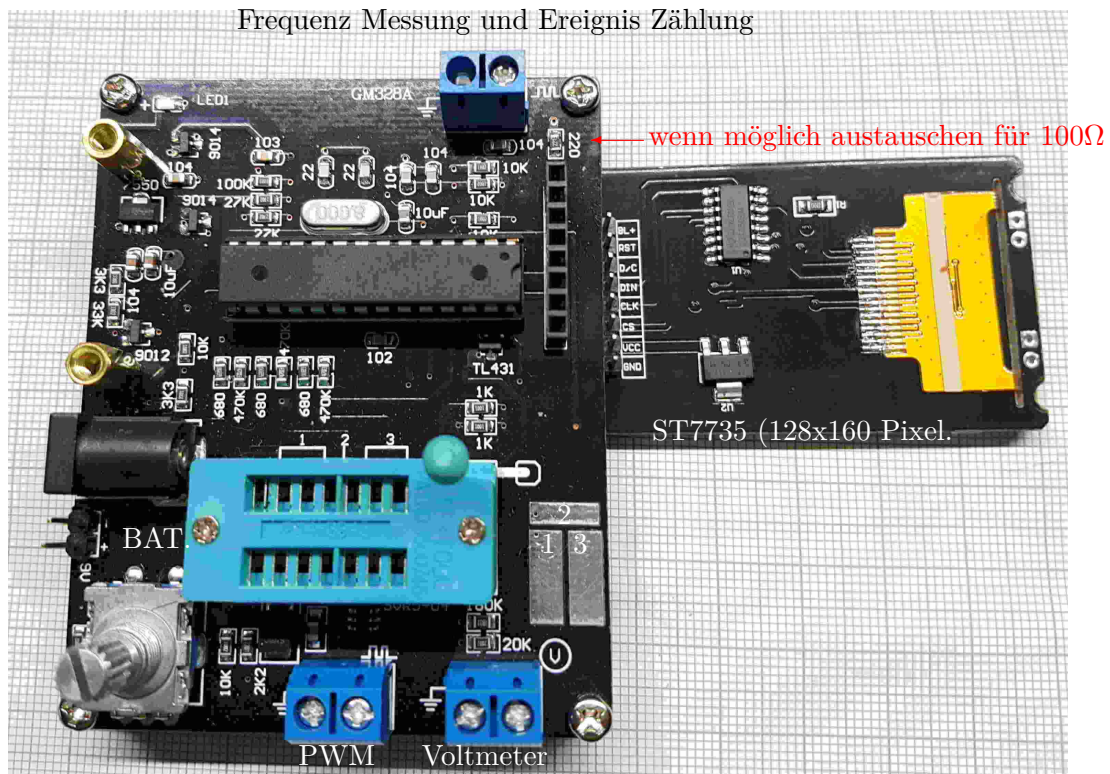


Abbildung 2.1. Ansicht mit abgenommenem LCD Display

- Wie schon aus dem erkennbar verwendet dieser Tester ATmega 328 P mit DIP Sockel, was externes Programmieren erlaubt, was auch nötig ist, weil es kein ISP Anschluss gibt. Die Platine ist mit einem 8MHz Quarz bestückt, wobei die Nachrüstung eines Trimmers (genauere Frequenz) möglich ist.
- Das Gerät verwendet ein Farbdisplay mit ST7735 Controller (128x160 Pixel). Wie es auf dem Bild 2.1 rechts oben gut sichtbar ist, verwendet Display einen CD4050 (IC1) Puffer für die Anpassung der Signalpegel und einen 3.3V Spannungsregler (IC2) für die Spannungsversorgung.
- Austausch des Widerstandes für die Hintergrundbeleuchtung kann die Lesbarkeit verbessern.
- Externe 2,5V Referenz ist durch TL431 realisiert.
- Auf der Platine ist ein Netzteil Anschluss und vorbereitete Lötungen für 9V Batterie.
- Zum bedienen ist ein Drehcodear mit Taster vorhanden.
- Die Testpins sind über 14 poligen Textool Anschluss zugänglich, für SMD Bauteile ist hier eine Testunterlage.
- Wie man auf dem Schema sieht 8.3 auf Seite 58 sind die Testpins durch Dioden Array IC SRV05-4 (IC2) teilweise geschützt, denn ein absoluter Schutz ist nicht möglich.
- Tester bietet über ein Klemmenpaar (X2) einen Frequenzausgang. Dieser geht aber lediglich parallel zu TP\_2.
- Ein weiterer Anschluss (X3) dient zur messen von positiven DC Gleichstannung bir max 50V. Dieser Eingang bietet keinen Schutz!
- Und eine dritte Doppelklemme (X4) bietet, ohne Schutz, einen Frequenzeingang.

## 2.2. Die Bedienung

ist realisiert durch einen Impulsdrehgeber mit Drucktaster und ist relativ einfach.

Trotzdem sind einige Hinweise erforderlich. In jedem Fall kannst du Bauteile mit drei Anschlüssen mit den drei Testports in beliebiger Reihenfolge verbinden. Bei zweipoligen Bauteilen kannst du die beiden Anschlüsse mit beliebigen Testports verbinden. Normalerweise spielt die Polarität keine Rolle, auch Elektrolytkondensatoren können beliebig angeschlossen werden. Die Messung der Kapazität wird aber so durchgeführt, dass der Minuspol am Testport mit der kleineren Nummer liegt. Da die Messspannung aber zwischen 0,3V und maximal 1,3V liegt, spielt auch hier die Polarität keine wichtige Rolle. Wenn das Bauteil angeschlossen ist, sollte es während der Messung nicht berührt werden. Lege es auf einen isolierenden Untergrund ab, wenn es nicht in einem Sockel steckt. Berühre auch nicht die Isolation der Messkabel, das Messergebnis kann beeinflusst werden.

Nun sollte der Starttaster gedrückt werden.

**Was hier geschieht, hängt von der Softwarekonfiguration ab.**

Damit ein Vergleich möglich ist wurden hier gleiche Konfigurationen und Zeiten benutzt.

**2.2.1. Taster** schaltet den **Tester ein** und dient zur **Bedienung**. Im automatischem Modus wartet Tester 30 Sec auf ein Bauteil ... bevor ... für die Batterie Schonung, abschaltet. Falls ist, in dieser Zeit, ein Bauteil eingelegt ist, benehmen sich die Versionen **unterschiedlich**. **In der m-Version** schaltet Tester konsequent nach 30 Sec ab aber du kannst durch längerer Druck früher beenden.

- Eine neue Messung erreichst du mit einem kurzem Druck oder durch drehen des Enkoders nach rechts.
- Und das Auswahlmenü durch Doppeldruck oder durch drehen nach links.

Tester unterscheidet zwischen:

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

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

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

Für einige Funktionen wird der Cursor nicht angezeigt, da die erwartete Eingabe klar sein sollte.

**In der k-Version** schaltet der Tester **nicht aus**, sondern wartet auf den nächsten Bauteil, denn dann automatisch testet.

Zum ausschalten ist ein Tastendruck oder schnelles drehen, in beliebigen Richtung, nötig.





- Auswahlmenü erreichst nach dem einschalten **ohne** eines eingelegten Bauteils durch ein langes Druck ( $> 0.5s$ ) oder durch ein schnelles drehen des Drehencoders.



**2.2.2. Drehencoder** verleiht dem Tester zusätzliche Funktionalität, die kontextabhängig ist. Manche Funktionen erlauben über die Drehgeschwindigkeit größere Änderungen oder Sprünge von Werten. Der Lese-Algorithmus berücksichtigt die Anzahl der Gray-Code-Pulse pro Schritt und auch die Anzahl der Schritte für eine volle  $360^\circ$  Umdrehung. Die Erkennung der Drehgeschwindigkeit misst die Zeit von zwei Schritten.

Also solltest Du den Encoder mindestens um zwei Schritte für mittlere Geschwindigkeit drehen. Für höhere Geschwindigkeiten sind es drei Schritte.

Ein einzelner Schritt resultiert immer in der niedrigsten Geschwindigkeit. Details kannst du direkt in der Dokumentation des Autoren erfahren. Siehe [3].



1. Automatische Erkennung von NPN und PNP bipolaren Transistoren, N- und P-Channel MOSFETs, JFETs, Dioden, Doppeldioden, N- und P-IGBTs, Thyristoren und Triacs. Für Thyristoren und Triacs müssen die Zünd- und Halteströme für die richtige Erkennung erreicht werden können. Bei IGBTs muß die Gate Schwellwertspannung unter 5V liegen.
2. Darstellung der Pin-Belegung der erkannten Bauteile.
3. Messung des Stromverstärkungsfaktors und der Basis-Emitter-Schwellspannung für bipolare Transistoren.
4. Darlington-Transistoren können durch die höhere Schwellspannung und durch den hohen Stromverstärkungsfaktor erkannt werden.
5. Automatische Erkennung einer Schutzdiode bei bipolaren Transistoren und bei MOSFETs.
6. Messung der Schwellwert-Spannung, der Gate-Kapazität und des  $R_{DSon}$  bei einer Gate-spannung von knapp 5V von MOSFETs.
7. Bis zu zwei Widerstände werden gemessen und mit den  Symbolen und den Widerstands-Werten mit bis zu vier Dezimalstellen in der richtigen Dimension angezeigt. Alle Symbole werden eingerahmt mit den gefundenen Testpin Nummern des Testers (1-3). Deshalb können auch Potentiometer gemessen werden. Wenn der Schleifer eines Potentiometers auf eine Endposition gestellt ist, kann der Tester nicht mehr zwischen mittlerem Anschluss und Endanschluss unterscheiden.
8. Die Auflösung der Widerstandsmessung ist jetzt bis zu 0,01Ω, Werte von bis zu 50MΩ werden erkannt.
9. Ein Kondensator kann erkannt und gemessen werden. Der wird mit dem Symbol  und dem Kapazitätswert mit bis zu vier Dezimalstellen in der richtigen Dimension angezeigt. Der Wert kann zwischen 25pF (bei 8MHz Takt, 50pF bei 1MHz Takt) bis 100mF liegen. Die Auflösung kann bis zu 1pF (bei 8MHz Takt) betragen.
10. Bei Kondensatoren mit einer Kapazität über 20nF wird zusätzlich der äquivalente Serienwiderstand (ESR) des Kondensators mit einer Auflösung von 0,01Ω gemessen und mit zwei Dezimalstellen angezeigt. Diese Fähigkeit steht nur zur Verfügung, wenn der ATmega mindestens 16K Flashspeicher besitzt.
11. Für Kondensatoren mit einem Kapazitätswert über 5000pF kann der Spannungsverlust Vloss nach einem Ladepuls bestimmt werden. Der Spannungsverlust gibt einen Hinweis auf die Güte des Kondensators.
12. Bis zu zwei Dioden werden mit dem Symbol  oder dem Symbol  in der richtigen Reihenfolge angezeigt. Zusätzlich werden die Schwellspannungen angezeigt.
13. Eine LED wird als Diode erkannt, die Schwellspannung ist viel höher als bei einer normalen Diode. Doppeldioden werden als zwei Dioden erkannt.
14. Zener-Dioden können erkannt werden, wenn die Zener-Spannung unter 4,5V ist. Sie werden als zwei Dioden angezeigt, man kann das Bauelement nur mit den Spannungen erkennen. Die äußeren Testpin-Nummern, welche die Dioden Symbole umgeben, sind in diesem Fall identisch. Man kann die wirkliche Anode der Diode nur durch diejenige Diode herausfinden, deren Schwellwert-Spannung nahe bei 700mV liegt!
15. Wenn mehr als 3 Dioden erkannt werden, wird die gefundene Anzahl der Dioden zusammen mit der Fehlermeldung angezeigt. Das kann nur passieren, wenn Dioden an alle drei Test-Pins angeschlossen sind und wenigstens eine Zener-Diode ist. In diesem Fall sollte man nur zwei Test-Pins anschließen und die Messung erneut starten, eine Diode nach der anderen.
16. Der Kapazitätswert einer einzelnen Diode in Sperr-Richtung wird automatisch ermittelt. Bipolare Transistoren können auch untersucht werden, wenn nur die Basis und entweder

- Kollektor oder Emitter angeschlossen wird.
17. Die Anschlüsse einer Gleichrichter-Brücke können mit nur einer Messung herausgefunden werden.
  18. Kondensatoren mit Kapazitätswerten von unter  $25pF$  werden normalerweise nicht erkannt, aber sie können zusammen mit einer parallel geschalteten Diode oder mit einem parallel geschalteten Kondensator mit wenigstens  $25pF$  gemessen werden. In diesem Fall muss der Kapazitätswert des parallel geschalteten Bauteils vom Messergebnis abgezogen werden. Bei Prozessoren mit mindestens 32K Flash Speicher wechselt der Tester mit einem Kondensator  $> 25pF$  an TP1 und TP3 in eine Kondensator-Meßfunktion, die auch Kapazitäten ab  $1pF$  direkt mißt.
  19. Bei Widerständen unter  $2100\Omega$  wird auch eine Induktivitätsmessung durchgeführt. Dabei wird zusätzlich zum Widerstands-Symbol  ein Induktivitäts Symbol  angezeigt. Der Anzeigebereich ist etwa  $0,01mH$  bis über  $20H$ , die Genauigkeit ist allerdings nicht hoch. Das Ergebnis wird nur bei einem Einzelwiderstand zusammen mit dem Widerstandswert angezeigt.
  20. Die Messzeit beträgt ungefähr zwei Sekunden, nur Kapazitätsmessungen und Induktivitätsmessungen können länger dauern.
  21. Die Software kann für Messserien mit vorgebbbarer Wiederhol-Zahl konfiguriert werden, bevor die automatische Abschaltung ausschaltet.
  22. Eingebaute Selbsttest-Funktion inklusive einem optionalen  $50Hz$  Frequenz-Generator um die Genauigkeit der Taktfrequenz und der Verzögerungszeiten zu überprüfen.
  23. Wählbare Möglichkeit den Nullabgleich für die Kondensatormessung und die Innenwiderstände für die Portausgänge beim Selbsttest automatisch zu bestimmen. Ein externer Kondensator mit einer Kapazität zwischen  $100nF$  und  $20\mu F$  an Pin 1 und Pin 3 ist notwendig, um die Offset-Spannung des analogen Komparators zu kompensieren. Dies kann den Messfehler bei Kapazitätsmessungen bis zu  $40\mu F$  reduzieren. Mit dem gleichen Kondensator wird eine Korrekturspannung zum Einstellen der richtigen Verstärkung für die ADC Messung mit der internen  $1,1V$  Referenzspannung berechnet.
  24. Anzeige des Kollektor-Emitter-Reststroms  $I_{CE0}$  mit stromloser Basis ( $1\mu A$  Auflösung) und des Kollektor-Emitter Reststroms  $I_{CES}$  mit der Basis auf Emitter-Potential gehalten. Diese Werte werden nur angezeigt, wenn sie nicht Null sind (besonders für Germanium-Transistoren).
  25. Der Tester wechselt vom Multifunktionstest zu einem Modus als Widerstands-Messgerät, wenn bei der automatischen Bauteile-Erkennung nur ein Widerstand an Test Pin 1 (TP1) und Test Pin 3 (TP3) erkannt wird. Wenn in der Makefile auch die Induktivitätsmessung beim Widerstandsmessgerät mit der Option `RMETER_WITH_L` gewünscht wurde, werden bei der Widerstandsmessung auch Induktivitäten gemessen. Der Betriebsmodus wird durch **[R]** oder **[RL]** auf der rechten Seite von Zeile 1 des Displays angezeigt. Genau so wechselt der Tester zu einem Kapazitätsmessgerät, wenn bei der Bauteile-Untersuchung ein Kondensator an TP1 und TP3 erkannt wurde. Dieser Betriebsmodus wird durch **[C]** auf der rechten Seite der Zeile 1 angezeigt. In dieser Betriebsart können Kondensatoren ab  $1pF$  gemessen werden. Lediglich für den automatischen Start der Funktion braucht man einen Kondensator mit mehr als  $25pF$ . Beide Sonderfunktionen können durch einen Tastendruck beendet werden. Der Tester fährt dann mit der normalen Messfunktion fort.
  26. Es kann eine Dialogfunktion gewählt werden, die weitere Einsatzmöglichkeiten zugänglich machen kann. Natürlich kann über den Dialog auch zu der Transistortester-Funktion zurückgekehrt werden.
  27. Mit Dialogfunktion kann am PD4-Port eine Frequenzmessung vorgenommen werden. Die Auflösung beträgt bei Eingangsfrequenzen über  $33kHz$  ein Hertz. Bei niedrigeren Frequenzen kann die Auflösung bis zu  $0,001mHz$  betragen. Lesen Sie bitte das Unterkapitel 2.2.4 in der Dokumentation [2], wie ein Frequenzsignal angeschlossen werden muss.
  28. Mit Dialogfunktion und ohne serielle Ausgabe kann eine externe Spannung bis  $50V$  über einen 10:1 Spannungsteiler am PC3 Pin gemessen werden.
  29. Mit Dialogfunktion kann eine Frequenzausgabe auf dem TP2-Pin (PB2-Port des ATmega) erfolgen. Derzeit können von Frequenzen von  $1Hz$  bis  $2MHz$  eingestellt werden.

30. Mit Dialogfunktion kann eine feste Frequenz mit einstellbarer Pulsweite auf dem TP2-Pin (PB2-Port des ATmega) ausgegeben werden. Die Breite kann mit kurzem Tastendruck um 1% und mit längerem Tastendruck um 10% erhöht werden.
31. Mit der Dialogfunktion kann eine spezielle Kondensatormessung mit ESR-Messung gestartet werden. Die Funktion wird bei der Auswahl C+ESR@TP1:TP3 genannt. Kapazitäten ab etwa  $2\mu F$  bis zu  $50mF$  können meistens wegen der geringen Messspannung von etwa  $300mV$  im eingebauten Zustand gemessen werden.
32. Es kann der ADC mit der Sampling-Methode so genutzt werden, daß Kondensatoren unter  $100pF$  mit einer Auflösung von  $0.01pF$  gemessen werden können. Mit der gleichen Methode können auch Spulen unter  $2mH$  mit deutlich besserer Auflösung über die Resonanzfrequenz mit einem parallelgeschalteten Kondensator bekannter Größe bestimmt werden.
33. Die Batterieüberwachung kann nach deinem Wünschen eingestellt werden. Jeder Zyklus beginnt mit der Anzeige der Batteriespannung und des Status (ok, schwach, leer). Fällt die Batteriespannung unter die Schwellenspannung, schaltet sich das Prüfgerät ab. Die Batterie wird auch während des Betriebs überprüft.

**3.0.1. Auswahlmenü** bietet weitere Möglichkeiten des Testers. Diese Funktionen sind, je nach der benutzten Version unterschiedlich. Manche Funktionen mit dem gleichen oder ähnlichen Namen, unterscheiden sich im Umfang und Bedienung. Genaue Beschreibung findest du direkt in der jeweiligen Einleitung. Siehe [3].

### 3.1. Wichtige Bemerkungen

Stelle immer sicher, dass **Kondensatoren** vor dem Anschluss an den Tester **entladen** sind! Der Tester könnte sonst beschädigt werden bevor er eingeschaltet ist. Es gibt nur wenig Schutzfunktion der ATmega-Anschlüsse. Besondere Vorsicht ist auch geboten, wenn versucht wird, Bauelemente in einer Schaltung zu messen. Das Gerät sollte in jedem Fall vorher von der Stromversorgung getrennt sein und man sollte sicher sein, dass **keine Restspannung** im Gerät vorhanden ist. Beim Messen kleiner Widerstandswerte muss man besonders auf die Übergangswiderstände achten. Es spielt die Qualität und der Zustand von Steckverbindern eine große Rolle, genau so wie die Widerstandswerte von Messkabeln. Dasselbe gilt auch für die Messung des ESR-Wertes von Kondensatoren. Bei schlechten Anschlusskabeln mit Krokodilklemmen wird so aus einem ESR von  $0,02\Omega$  leicht ein Wert von  $0,61\Omega$ . Wenn möglich sollte man Kabel mit Testklemmen an die drei Testports parallel zu vorhandenen Sockeln fest anschließen (anlöten). Dann braucht der Tester für kleine Kapazitäten nicht jedesmal neu kalibriert werden, wenn mit oder ohne eingesteckte Testkabel gemessen wird. Für die Kalibration des Nullwiderstandes macht es aber im allgemeinen einen Unterschied, ob die Testpins direkt am Sockel oder am Ende der Kabel mit den den Testklemmen verbunden wird. Nur im letzteren Fall ist der Widerstand von Kabel und Klemmen mit kalibriert. Im Zweifelsfall kann man die Kalibration mit dem Kurzschluß am Testsockel durchführen und danach den Widerstand der kurzgeschlossenen Klemmen mit dem Tester messen.

### 3.2. Problemfälle

Bei den Messergebnissen sollten Sie immer im Gedächtnis behalten, dass die Schaltung des Transistortesters für Kleinsignal-Bauelemente ausgelegt ist. Normalerweise beträgt der maximale Messstrom etwa  $6mA$ . Leistungshalbleiter machen oft wegen hoher Restströme Probleme bei der Erkennung oder beim Messen der Sperrschicht-Kapazität. Bei Thyristoren und Triacs werden oft die Zündströme oder die Halteströme nicht erreicht. Deswegen kann es vorkommen, dass ein Thyristor als NPN-Transistor oder Diode erkannt wird. Ebenso ist es möglich, dass ein Thyristor oder Triac gar nicht erkannt wird.

Probleme bei der Erkennung machen auch Halbleiter mit integrierten Widerständen. So wird die Basis-Emitter-Diode eines BU508D-Transistors wegen eines parallel geschalteten internen  $42\Omega$  Widerstandes nicht erkannt. Folglich kann auch die Transistorfunktion nicht geprüft werden. Probleme bei der Erkennung machen oft auch Darlington-Transistoren höherer Leistung. Hier sind auch oft Basis-Emitter-Widerstände verbaut, welche die Erkennung wegen der hier verwendeten kleinen Messströme erschweren.

Normalerweise wird beim Start des Testers für eine Sekunde die Batteriespannung angezeigt. Wenn die Spannung eine Grenze unterschreitet, wird eine Warnung hinter der Batteriespannung ausgegeben. Wenn Sie eine aufladbare 9V-Batterie benutzen, sollten Sie den Akku möglichst bald austauschen oder nachladen. In der zweiten Zeile die gemessene Betriebsspannung mit „VCC=x.xxV“ angezeigt.

**Einzelmessung** Wenn der Tester für Einzelmessung konfiguriert ist (POWER\_OFF-Option), schaltet der Tester nach einer Anzeigezeit von 28 Sekunden (konfigurierbar) wieder automatisch aus, um die Batterie zu schonen. Während der Anzeigezeit kann aber auch vorzeitig eine neue Messung gestartet werden. Nach der Abschaltung kann natürlich auch wieder eine neue Messung gestartet werden, entweder mit dem gleichen Bauteil oder mit einem anderen Bauteil.

**Dauermessung** Einen Sonderfall stellt die Konfiguration ohne die automatische Abschaltfunktion dar. Hierfür wird die POWER\_OFF-Option in der Makefile nicht gesetzt. Diese Konfiguration wird normalerweise nur ohne die Transistoren für die Abschaltung benutzt. Es wird stattdessen ein externer Ein-/Aus-Schalter benötigt. Hierbei wiederholt der Tester die Messungen solange, bis ausgeschaltet wird.

**Serienmessung** In diesem Konfigurationsfall wird der Tester nicht nach einer Messung sondern erst nach einer konfigurierbaren Zahl von Messungen abgeschaltet. Hierfür wird der POWER\_OFF-Option in der Makefile eine Wiederholzahl (z.B. 5) zugewiesen. Im Standardfall wird der Tester nach fünf Messungen ohne erkanntes Bauteil abgeschaltet. Wird ein angeschlossenes Bauteil erkannt, wird erst bei der doppelten Anzahl, also zehn Messungen abgeschaltet. Eine einzige Messung mit nicht erkanntem Bauteil setzt die Zählung für erkannte Bauteile auf Null zurück. Ebenso setzt eine einzige Messung mit erkanntem Bauteil die Zählung für die nicht erkannten Bauteile auf Null zurück. Dies hat zur Folge, dass auch ohne Betätigung des Starttasters immer weiter gemessen werden kann, wenn Bauteile regelmäßig gewechselt werden. Ein Bauteilwechsel führt in der Regel durch die zwischenzeitlich leeren Klemmen zu einer Messung ohne erkanntes Bauteil.

Eine Besonderheit gibt es in diesem Betriebsmodus für die Anzeigezeit. Wenn beim Einschalten der Starttaster nur kurz gedrückt wurde, beträgt die Anzeigezeit der Messergebnisse nur 5 Sekunden. Wenn der Starttaster bis zum Erscheinen der ersten Meldung festgehalten wurde, beträgt die Anzeigezeit wie bei der Einzelmessung 28 Sekunden. Ein vorzeitiger neuer Messbeginn ist aber während der Anzeigezeit durch erneutes Drücken des Starttasters möglich.

### 4.1. Optionale Menüfunktionen

Wenn die Menüfunktion eingeschaltet ist, startet der Tester nach einem längeren Tastendruck ( $> 0.5s$ ) ein Auswahlmenü für zusätzliche Funktionen. Die wählbaren Funktionen erscheinen in Zeile 4 als gekennzeichnete Funktion. Dabei wird die vorige und nächste Funktion in Zeile 3 und 5 angezeigt. Nach einer längeren Wartezeit ohne jegliche Bedienung kehrt das Programm zu der normalen Transistortester-Funktion zurück. Durch kurzen Tastendruck kann zur nächsten Auswahl gewechselt werden. Mit einem längeren Tastendruck startet die angezeigte Zusatzfunktion. Nach Anzeige der letzten Funktion „Schalte aus“ wird wieder die erste Funktion angezeigt.

Die Menü-Auswahl kann auch mit einem schnellen Drehen des Encoders während der Anzeige einer vorausgegangenen Messung aufgerufen werden. Die Menüfunktionen können mit einem langsamen Drehen des Encoders in beliebiger Richtung ausgewählt werden. Die ausgewählte Menüfunktion kann aber nur mit einem längeren Tastendruck gestartet werden. Innerhalb einer Menüfunktion können Parameter durch eine langsame Drehung des Encoders ausgewählt wer-



den. Eine schnelle Drehung des Encoders kehrt zur Menü-Auswahl zurück.


**Frequenz** Die Zusatzfunktion „Frequenz“ (Frequenzmessung) benutzt als Eingang den PD4-Pin des ATmega, der auch an das LCD angeschlossen ist. Es wird immer zunächst die Frequenz gemessen, bei Frequenzen unter  $25\text{kHz}$  wird auch die mittlere Periode des Eingangssignals bestimmt und daraus die Frequenz mit einer Auflösung von bis zu  $0,001\text{Hz}$  berechnet. Bei gesetzter POWER\_OFF-Option in der Makefile wird die Dauer der Frequenzmessung auf 8 Minuten beschränkt. Die Frequenzmessung wird durch Tastendruck beendet und in das Auswahlmenü zurückgekehrt.

**f-Generator** Bei der Zusatzfunktion „f-Generator“ (Frequenz-Generator) können die Frequenzen zwischen 1Hz und 2MHz gewählt werden. Die Einstellung der Frequenz kann jeweils nur für die höchste dargestellte Stelle (Ziffernposition) verändert werden. Für die Stellen 1Hz bis 10kHz sind jeweils die Ziffern 0-9 wählbar. Bei der 100kHz Stelle ist 0-20 wählbar. In Spalte 1 der Frequenzzeile wird durch ein > oder < Symbol angezeigt, ob durch einen längeren Tastendruck ( $> 0,8\text{s}$ ) zur höheren oder niedrigen Stelle geschaltet wird. Zur niedrigeren Stelle (<) kann nur geschaltet werden, wenn die augenblickliche Stelle auf 0 gestellt wurde und wenn nicht die Stelle mit 1Hz Schritten gewählt wurde. Bei der gewählten 100kHz Stelle ist das > Symbol durch ein R Zeichen ersetzt. Der längere Tastendruck bewirkt dann eine Rücksetzung der Frequenz auf den Startwert 1Hz. Bei gesetzter POWER\_OFF-Option in der Makefile muss für den Frequenzwechsel die Taste länger gedrückt werden, da durch einen kurzen Tastendruck ( $< 0,2\text{s}$ ) nur die Zeitüberwachung von 4 Minuten zurückgesetzt wird. Die abgelaufene Zeit wird in Zeile 1 durch einen Punkt für jede abgelaufene 30 Sekunden angezeigt. Durch einen regelmäßigen kurzen Tastendruck kann die vorzeitige Abschaltung der Frequenzerzeugung verhindert werden. Ein längerer Tastendruck ( $> 2\text{s}$ ) kehrt wieder zur Auswahl der Funktionen zurück.

**10-bit PWM** Bei der Zusatzfunktion „10-bit PWM“ (Pulsweitenmodulation) wird eine feste Frequenz mit einstellbarer Pulsweite an Pin TP2 erzeugt. Mit einem kurzen Tastendruck ( $< 0,5\text{s}$ ) wird die Pulsweite um 1% erhöht, mit einem längeren Tastendruck um 10%. Bei Überschreiten von 99% werden 100% vom erhöhten Wert abgezogen. Bei gesetzter POWER\_OFF-Option in der Makefile wird die Frequenzerzeugung nach 8 Minuten ohne Bedienung beendet. Durch sehr langen Tastendruck ( $> 1,3\text{s}$ ) kann die Frequenzerzeugung auch beendet werden.

**C+ESR@TP1:3** Bei der Zusatzfunktion „C+ESR@TP1:3“ wird eine separate Kondensatormessung mit ESR-Messung an TP1 und TP3 gestartet. Messbar sind Kondensatoren mit mehr als  $2\mu\text{F}$  bis zu  $50\text{mF}$ . Wegen der geringen Messspannung von etwa 300mV sollte in vielen Fällen die Messung in der Schaltung ohne vorherigen Ausbau möglich sein. Bei gesetzter POWER\_OFF-Option in der Makefile ist die Anzahl der Messungen auf 250 beschränkt, kann aber sofort wieder gestartet werden. Die Mess-Serie kann durch einen längeren Tastendruck beendet werden.

**Widerstandsmeßgerät** Mit dem 1—— 3 Symbol wird der Tester in ein Ohmmeter an TP1 und TP3 verwandelt. Diese Betriebsart wird durch ein [R] in der rechten Ecke der ersten Displayzeile angezeigt. Weil bei dieser Betriebsart die ESR-Meßfunktion nicht benutzt wird, beträgt auch für Widerstände unter  $10\Omega$  die Auflösung nur  $0,1\Omega$ . Wenn die Ohmmeter Funktion mit der Induktivitätsmessung konfiguriert wurde, erscheint hier ein 1—— 3 Symbol. Dann beinhaltet die Ohmmeter Funktion die Messung von Induktivitäten für Widerstände unter  $2100\Omega$ . In der rechten Ecke der ersten Zeile des Displays wird dann ein [RL] angezeigt. Für Widerstände unter  $10\Omega$  wird dann auch die ESR-Meßmethode benutzt, wenn keine Induktivität festgestellt wurde. Damit erhöht sich die Auflösung für Widerstände unter  $10\Omega$  auf  $0,01\Omega$ . In dieser Betriebsart werden die Meßwerte fortlaufend ermittelt. Mit einem Tastendruck beendet der Tester diese Betriebsart und kehrt wieder zum Menü zurück. Die gleiche Betriebsart wird auch automatisch gestartet, wenn zwischen TP1 und TP3 ein einzelner Widerstand angeschlossen wurde und der Start-Taster gedrückt wurde. In diesen Fall kehrt der Tester mit einem Tastendruck wieder zu der normalen Teseterfunktion zurück.

**Kondensatormeßgerät** Mit dem 1—— 3 Symbol wird der Tester in ein reines Kondensator-

Meßgerät an TP1 und TP3 verwandelt. Die Betriebsart wird durch ein [C] in der rechten Ecke der ersten Zeile des Displays angezeigt. Bei dieser Betriebsart können Kondensatoren ab  $1pF$  bis zu  $100mF$  gemessen werden. In dieser Betriebsart werden die Meßwerte fortlaufend ermittelt. Mit einem Tastendruck wird dieser Sonderbetrieb beendet und der Tester kehrt wieder zum Menü zurück. Auch hier wird wie bei der Widerstands-Meßfunktion in den Betriebs-Modus automatisch gewechselt, wenn zwischen TP1 und TP3 ein Kondensator mit der normalen Testerfunktion gemessen wurde. Der Tester kehrt nach dem automatischen Start wieder zu der normalen Testerfunktion zurück, wenn der Taster gedrückt wird.

**Impulsdrehgeber** Mit der Zusatzfunktion „Impulsdrehgeber“ kann ein Drehgeber untersucht werden. Die drei Kontakte des Impulsdrehgebers müssen vor dem Start der Zusatzfunktion beliebig an die drei Testpins des Transistortesters angeschlossen werden. Nach dem Start der Funktion muss der Drehknopf nicht zu schnell gedreht werden. Wenn der Test erfolgreich abgeschlossen ist, wird die Pinbelegung der Kontakte symbolisch in Zeile 2 dargestellt. Dabei wird der gemeinsame Anschluss herausgefunden und für etwa zwei Sekunden angezeigt, ob an den Raststellung beide Kontakte offen ('o') oder beide Kontakte geschlossen ('C') sind. Ein Impulsdrehgeber mit offenen Kontakten an den Raststellungen wird so mit der Zeile 2 „1-/-2-/-3 o“ zwei Sekunden lang angezeigt. Natürlich wird die richtige Pinnummer des gemeinsamen Kontaktes in der Mitte anstelle der '2' angezeigt. Wenn auch die geschlossene Schalterstellung an den Raststellungen vorkommt, wird außerdem in Zeile 2 „1—2—3 C“ zwei Sekunden lang angezeigt. Mir ist kein Impulsdrehgeber bekannt, der immer nur geschlossene Kontakte an jeder Raststellung hat. Die Stellungen der Kontakte zwischen den Raststellungen werden nur kurz ( $< 0,5s$ ) ohne die Kennbuchstaben 'o' oder 'C' in Zeile 2 angezeigt. Wenn der Impulsdrehgeber für die Bedienung des Testers eingesetzt werden soll, muß die Makefile Option `WITH_ROTARY_SWITCH=2` für Drehgeber mit nur den offenen Kontakten ('o') und die Option `WITH_ROTARY_SWITCH=1` für Drehgeber mit offenen ('o') und geschlossenen ('C') Kontakten an den Raststellungen.

**C( $\mu F$ )-Korrektur** Mit dieser Menüfunktion kann ein Korrekturwert für die Messung grösserer Kapazitätswerte geändert werden. Die Funktion ist die gleiche, die auch mit der Makefile Option `C_H_KORR` voreingestellt werden kann. Werte über Null reduzieren den Ausgabewert der Kapazität um diesen Prozent Wert, Werte unter Null würden den Ausgabewert anwachsen lassen. Ein kurzer Tastendruck verringert den Korrekturwert um 0.1%, ein längerer Tastendruck vergrößert den Korrekturwert um 0.1%. Ein sehr langer Tastendruck speichert den Wert. Es ist eine Eigenschaft des Meßverfahrens, daß Kondensatoren mit geringer Güte wie Elektrolyt-Kondensatoren mit zu großer Kapazität gemessen werden. Erkennbar ist die Güte über den Parameter Vloss. Gute Kondensatoren haben kein Vloss oder nur 0.1%. Zum Abgleich dieses Parameters sollten also nur Kondensatoren mit über  $50\mu F$  hoher Güte verwendet werden. Übrigens halte ich die Bestimmung des exakten Kapazitätswertes von Elektrolyt-Kondensatoren für überflüssig, da die Kapazität sowohl von der Temperatur als auch der DC-Spannung abhängt.

**Selbsttest** Mit der Zusatzfunktion „Selbsttest“ wird ein vollständiger Selbsttest mit Kalibration durchgeführt. Dabei werden sowohl die Testfunktionen T1 bis T7 (wenn nicht verhindert mit der Option `NO_TEST_T1_T7`) als auch die Kalibration mit dem externen Kondensator jedes Mal durchgeführt.

**Spannung** Die Zusatzfunktion „Spannung“ (Spannungsmessung) ist nur möglich, wenn die serielle Ausgabe deaktiviert wurde. Da am Port PC3 ein 10:1-Spannungsteiler vorgesehen ist, können Spannungen bis 50V gemessen werden. Bei gesetzter `POWER_OFF`-Option in der Makefile und ohne Bedienung wird die Messung nach 4 Minuten beendet. Die Messung kann aber durch einen besonders langen Tastendruck ( $> 4$  Sekunden) vorher beendet werden.

**FrontColor** dient dazu, die Schriftfarbe anpassen zu können. Du kannst eine der drei Farben rot, grün und blau mit einem längeren Tastendruck wählen. Die Intensität der gewählten Farbe, die mit einem  $>$  Zeichen in Spalte 1 gekennzeichnet wird, kann durch Drehen des Impulsdrehgebers verändert werden.

**BackColor** ist, wie Menü zu vor zuständig für personalisieren der Hintergrundfarbe.

**Zeige Daten** Die Funktion „Zeige Daten“ zeigt neben der Software-Versionsnummer die Daten des Abgleichs an. Dies sind die Nullwiderstände  $R_0$  von Pin 1:3, 2:3 und 1:2. Außerdem werden die Ausgangswiderstände der Ports zur 5V-Seite ( $R_{iHi}$ ) und zur 0V Seite ( $R_{iLo}$ ) angezeigt. Die Nullkapazitätswerte ( $C_0$ ) werden ebenfalls in allen Pinkombinationen angezeigt (1:3, 2:3, 1:2 und 3:1, 3:2 2:1). Danach werden auch die Spannungskorrekturen für die Komparatorspannung ( $REF\_C$ ) und für die Referenzspannung ( $REF\_R$ ) angezeigt. Bei Graphikdisplays werden danach noch die verwendeten Symbole für die Bauteile und der Fontsatz angezeigt. Jede Seite wird 15 Sekunden angezeigt. Es kann aber auch durch Tastendruck oder einer Rechtsdrehung des Impulsdrehgebers zur nächsten Seite geblättert werden. Mit einer Linksdrehung des Impulsdrehgebers kann die Ausgabe wiederholt werden oder zur vorigen Seite zurückgeblättert werden.

**Schalte aus** Mit der Zusatzfunktion „Schalte aus“ kann der Transistortester abgeschaltet werden.

**Transistor** Natürlich kann mit der Funktion „Transistor“ (Transistortester) wieder zu der normalen Transistortester-Funktion zurückgekehrt werden.

Bei gesetzter `POWER_OFF`-Option in der Makefile sind alle Zusatzfunktionen zeitbeschränkt, damit die Batterie nicht verbraucht wird.

## 4.2. Selbsttest und Kalibration

Wenn die Software mit der Selbsttestfunktion konfiguriert ist, kann der Selbsttest durch einen Kurzschluss aller drei Testports und Drücken der Starttaste eingeleitet werden. Für den Start des Selbsttests muss die Starttaste innerhalb von 2 Sekunden noch einmal gedrückt werden, sonst wird mit einer normalen Messung fortgefahren. Beim Selbsttest werden die im Kapitel 5 des Transistortesters [2] beschriebenen Tests ausgeführt. Wenn der Tester mit einer Menüfunktion (Option `WITH_MENU`) konfiguriert ist, wird der vollständige Selbsttest mit den Tests T1 bis T7 nur bei dem „Selbsttest“ ausgeführt, der als Menüfunktion ausgewählt werden kann. Außerdem wird beim Aufruf über die Menüfunktion der Abgleich mit dem externen Kondensator durchgeführt, sonst nur für den ersten Aufruf. Damit kann der durch die kurzgeschlossenen Testports automatisch gestartete Abgleich schneller durchgeführt werden. Die viermalige Testwiederholung bei T1 bis T7 kann vermieden werden, wenn der Starttaster gedrückt gehalten wird. So kann man uninteressante Tests schnell beenden und sich durch Loslassen des Starttasters interessante Tests viermal wiederholen lassen. Der Test 4 läuft nur automatisch weiter, wenn die Verbindung zwischen den Testports gelöst wird.

Wenn die Funktion `AUTO_CAL` in der Makefile gewählt ist, wird beim Selbsttest eine Kalibration der Nullwertes für die Kondensatormessung durchgeführt. Für die Kalibration des Nullwertes für die Kondensatormessung ist wichtig, dass die Verbindung zwischen den Testpins (Kurzschluss) während des Tests 4 wieder gelöst wird! Sie sollten während der Kalibration (nach dem Test 6) weder die Testports noch angeschlossene Kabel berühren. Die Ausrüstung sollte aber die gleiche sein, die später zum Messen verwendet wird. Anderenfalls wird der Nullwert der Kondensatormessung nicht richtig bestimmt. Die Kalibration des Innenwiderstandes der Port-Ausgänge wird mit dieser Option vor jeder Messung durchgeführt.

Bei der Kalibration werden zwei Sonderschritte gemacht, wenn die `samplingADC` Funktion in der Makefile gewählt wurde (`WITH_SamplingADC = 1`). Nach der normalen Bestimmung der Nullwerte der Kapazitätsmessung werden dann auch die Nullwerte mit der Sampling Methode (`C0samp`) bestimmt. Als letzter Teil der Kalibration wird der Anschluß eines Testkondensators für die Spulenmessung an Pin 1 und Pin 3 angefordert mit der Meldung 1—||— 3 10-30nF(L). Der Kapazitätswert sollte hierbei zwischen  $10nF$  und  $30nF$  liegen, um eine meßbare Resonanzfrequenz bei der späteren Parallelschaltung mit einer Spule ( $< 2mH$ ) zu erreichen. Für Spulen mit mehr als  $2mH$  Induktivität sollte die normale Testfunktion ohne die Parallelschaltung eines Kondensators ausreichen. Ein Parallelschalten des Kondensators sollte hier nicht mehr zu einer Verbesserung des Meßergebnisses führen.

Nach der Bestimmung der Nullkapazitäten ist der Anschluss eines Kondensators mit einer beliebiger Kapazität zwischen  $100nF$  und  $20\mu F$  an Pin 1 und Pin 3 erforderlich. Dazu wird

in Zeile 1 ein  $1\text{---}3 > 100\text{nF}$  angezeigt. Sie sollten den Kondensator erst nach der Ausgabe der C0 Werte oder nach der Ausgabe dieser Aufforderung anschließen. Mit diesem Kondensator wird die Offset-Spannung des analogen Komparators kompensiert, um genauere Kapazitätswerte ermitteln zu können. Die Verstärkung für ADC-Messungen mit der internen Referenz-Spannung wird ebenfalls mit diesem Kondensator abgeglichen, um bessere Widerstands-Messergebnisse mit der AUTOSCALE\_ADC-Option zu erreichen. Wenn die Menüfunktion beim Tester ausgewählt wurde (Option WITH\_MENU) und der Selbsttest nicht als Menüfunktion gestartet wurde, wird der Abgleich mit dem externen Kondensator nur bei der ersten Kalibration durchgeführt. Die Kalibration mit dem externen Kondensator kann nur wiederholt werden, wenn der Selbsttest als Menüfunktion ausgewählt wird.

Der Nullwert für die ESR-Messung wird als Option ESR\_ZERO in der Makefile vorbesetzt. Mit jedem Selbsttest wird der ESR Nullwert für alle drei Pinkombinationen neu bestimmt. Das Verfahren der ESR-Messung wird auch für Widerstände mit Werten unter  $10\Omega$  benutzt um hier eine Auflösung von  $0,01\Omega$  zu erreichen.



**5.0.1. 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.h` aktivieren. In diesem Fall kann sich der Tester leider nicht selbst abschalten.

**5.0.2. 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 36 und `CYCLE_MAX` Seite 39 in `config.h` geändert werden.

Für den Auto-Hold-Modus gibt es eine optionale automatische Abschaltung (`POWER_OFF_TIMEOUT`) Seite 47, welche nur während der Bauteilesuche, Ergebnisausgabe und im Hauptmenü aktiv ist.

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

Ist die Summer/Pieper-Option vorhanden, kannst Du einen kurzen Bestätigungston für das Ende der Bauteilesuche aktivieren (`UI_PROBING_DONE_BEEP`). Daneben gibt es auch eine Option zum temporären Umschalten in den Auto-Hold-Modus nachdem ein Bauteil erkannt wurde (im kontinuierlichen Modus, `UI_AUTOHOLD_FOUND`).

Dies hilft beim Lesen der Ergebnisausgabe.

**5.0.3. 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 47.

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 Stromversorgung wird der Batteriestatus dann auf „ext“ (für extern) gesetzt. Die Schwellwerte für eine schwache und leere Batterie werden über BAT\_WEAK und BAT\_LOW gesetzt, während BAT\_OFFSET den Spannungsverlust durch die Schaltung definiert, z.B. Verpolungsschutzdiode und PNP-Transistor zum Schalten der Stromversorgung.

**5.0.4. 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.

Du kannst die Anzeige eines kleinen Batteriesymbols für den Batteriestatus anstatt der textbasierten Variante aktivieren (UI\_BATTERY). Auch gibt es die Möglichkeit, den Batteriestatus in der letzten Zeile nach der Ergebnisausgabe anzuzeigen (UI\_BATTERY\_LASTLINE).

## 5.1. 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.

**5.1.1. 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

**5.1.2. 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.

**5.1.3. 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.

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

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

Ein langer Tastendruck stellt die Frequenz zurück auf 1kHz und zwei kurze Tastendrucke beenden den Signalgenerator.

Beschaltung bei Signalausgabe über die Testpins:

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

Pin #1 und #3: Masse

**Hinweis:** Drehencoder oder andere Eingabeoption notwendig!

**5.1.5. Voltmeter und Zenertest** Weil der GM328A keinen DC-DC-Konverters besitzt, können Z-Dioden mit max. 5V Durchlass Spannung gemessen werden. Der Anschluss erfolgt über zusätzliche Testpins.

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.

Der Boost-Konverter kann auch über einen festen I/O-Pin geschaltet werden, um die Batterielaufzeit zu erhöhen (ZENER\_SWITCHED).

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.

Diese Option steht nur zu Verfügung, sofern entweder ZENER\_UNSWITCHED oder ZENER\_SWITCHED aktiviert ist.

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

**5.1.6. 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 Standard- werte für Vcc/Vdd sind:

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

Und wie gewohnt, zwei kurze Tastendrücke beenden den Logiktester.

**5.1.7. 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 R1 (680  $\Omega$ ) auf etwa 7 mA begrenzt. Die Schwellwerte für den Summer sind:

Spannung:	aktiver Summer:	passiver Summer:
< 100 mV	kontinuierlicher Ton	wiederholender Piep mit hoher Frequenz
100-700 mV	wiederholender kurzer Ton	wiederholender Piep mit niedriger Frequenz
> 700 mV	kein Ton	

Der kurze Ton zeigt einen möglichen Halbleiterübergang an. Bei einer Unter- -brechung 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 kuze Tastendrücke beenden den Test.

Beschaltung der Testpins:

- Pin #1: Vcc (680 $\Omega$  Widerstand zur Strombegrenzung)
- Pin #3: Masse

**5.1.8. 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

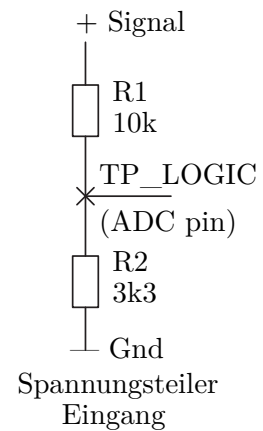


Abbildung 5.1

**5.1.9. 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 (680Ω) und schaltet auf Rh (470kΩ) um, sobald der Strom einen bestimmten Grenzwert unterschreitet.

Jeder Testzyklus beginnt mit der Anzeige der Belegung der Testpins. Nach dem Verbinden des Kondensators startet ein Druck der Testtaste das Laden (oder Rechtsdrehung bei einem Drehencoder).

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

Sobald der Entladegrenzwert erreicht ist, startet der Tester einen neuen Testzyklus. Zum Verlassen des Tests zweimal kurz die Testtaste drücken.

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

Beschaltung für Kondensator:

Pin #1:	Plus
Pin #3:	Minus

**5.1.10. R/C/L-Monitore** Die Minitorfunktionen 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 Tastendrücke (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.

**Hinweis:**

- Die Kapazitätswerte für Elkos können etwas niedriger sein als in der normalen Bauteilesuche (verursacht durch die wiederholten Messungen mit einem Gleichstromsignal).

**5.1.11. 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 (82μH 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 Frequenzzähler zu nutzen. Die mFirmware 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 liefe.

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`.

**5.1.12. 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 [2] beschrieben.

**5.1.13. 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.

**5.1.14. 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 (gepuffert 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.

**5.1.15. 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)
$\geq 6$	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 $680\Omega$ Widerstand zur Strombegrenzung)
Pin #3:	Masse
T0:	Zählereingang

**5.1.16. 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 ( $680\Omega$ Widerstand zur Strombegrenzung)
Pin #3:	Masse

**5.1.17. 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.

**5.1.18. 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.

**5.1.19. 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 (SW\_IR\_RECEIVER).

Die zweite Variante ist ein festes Modul, welches mit einem bestimmten MCU-Pin verbunden ist (HW\_IR\_RECEIVER).

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  $50\mu\text{s}$  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  $\mu$ 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.

Ist die Summer/Pieper-Option vorhanden, kannst Du einen kurzen Bestätigungston für gültige Daten-Pakete aktivieren (SW\_IR\_RX\_BEEP).

- IR-Empfängermodul an Testpins bitte erst im IR-Fernbedienungsdetektor anschließen!

Beschaltung für das TSOP-Modul (Standard-Pinout SW\_IR\_RX\_PINOUT\_G\_V\_D):

- |         |  |
|---------|--|
| Pin #1: | Masse/Gnd  |
| Pin #2: | Vs (680 $\Omega$ Widerstand zur Strombegrenzung) |
| Pin #3: | Data/Out   |



Die Firmware kann auch auf ein alternatives Pinout umgestellt werden, sofern es notwendig sein sollte. Dies ist insbesondere für Tester mit Nullkraftsockel nützlich:

- alternative Beschaltung SW\_IR\_RX\_PINOUT\_D\_G\_V  
Pin #1: Data/Out  
Pin #2: Gnd  
Pin #3: Vs (680Ω Widerstand zur Strombegrenzung)
- alternative Beschaltung SW\_IR\_RX\_PINOUT\_D\_V\_G  
Pin #1: Data/Out  
Pin #2: Vs (680Ω Widerstand zur Strombegrenzung)  
Pin #3: Gnd

**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.

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

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

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

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

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

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

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

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

Du kannst diese aber nach Belieben ändern.

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

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

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

Unterstützte Protokolle und ihre Datenfelder:

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

Optionale Protokolle (SW\_IR\_RX\_EXTRA):

- Thomson <Gerät:4> <Funktion:7>

Die Datenfelder sind durch Leerzeichen getrennt und ihre Syntax ist:

<Feldname>:<Anzahl Bits>

Beschaltung bei Signalausgabe über die Testpins:

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

Pin #1 und #3: Masse

**Hinweis:** 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 5.2 zeigt einen Treiber, welcher die IR-LED ( $V_f$  1.5V,  $I_f$  100mA) mit 50mA schaltet.

**Hinweis:**

Falls das Timing der Pulse/Pausen nicht passen sollte, bitte die alternative Warteschleifenmethode `SW_IR_TX_ALTDELAY` auf Seite 36 aktivieren. Dies ist notwendig, wenn der C-Compiler die Standardwarteschleife trotz Anweisung, den Inline-Assembler-Code beizubehalten, optimiert.

**5.1.21. 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.

**5.1.22. Fotodioden-Test** Mit dieser Testfunktion kannst Du den Strom einer Fotodiode beobachten. Als erstes wird für ein paar Sekunden die Testpinbelegung angezeigt, welche mit einem Tastendruck vorzeitig beendet werden kann. Dann springt der Tester zum Beobachten des Stroms  $I_P$  mit der Diode im Sperrrichtungsbetrieb, gekennzeichnet durch „rev“. Ein kurzer Tastendruck schaltet den Betriebsmodus auf Vorwärtsrichtung (Photoelement) um, gekennzeichnet durch „no“. Ein weiterer Tastendruck schaltet den Modus wieder zurück. Und wie Du bereits ahnst, zwei kurze Tastendrucke beenden die Testfunktion.

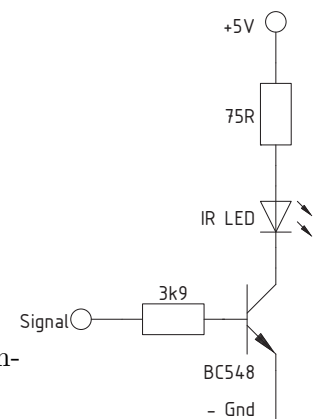


Abbildung 5.2.  
IR-Treiber

Zum Testen der Fotodiode diese z.B. temporär mit der Hand abdunkeln oder mit einer geeigneten Lichtquelle aus unterschiedlichen Entfernungen bestrahlen. Weniger Licht resultiert in weniger Strom, und umgekehrt. Evtl. fängt der Tester etwas EM-Störungen auf, die zu einem kleinen Strom führen können, speziell im Sperrrichtungsbetrieb. Das sieht man sehr gut, wenn gar keine Fotodiode angeklemt ist. Es sei auch darauf hingewiesen, dass Photodioden im Sperrrichtungsbetrieb einen Dunkelstrom haben.

**Warnung:** Keine Solarzellen testen!

Beschaltung von Test-Pins:

Pin #1: Anode

Pin #3: Kathode

**5.1.23. Dioden/LED-Schnelltest** Der Schnelltest schaut ständig nach einer Diode oder LED an den Testpins #1 und #3. Er ist dazu gedacht, Dioden/LEDs schnell zu prüfen und deren Polarität festzustellen. LEDs blinken während dem Testen.

Nach dem Starten zeigt der Tester die Beschaltung der Testpins für ein paar Sekunden an, was durch die Testtaste übersprungen werden kann. Wenn eine Diode oder LED gefunden wurde, wird deren Pinbelegung und  $V_f$  angezeigt. Die Anode ist immer auf der linken Seite, und die Kathode auf der rechten. Im Falle von zwei anti-parallelen Dioden/LEDs wird auch die zweite gezeigt. Du kennst schon die Übung: zum Beenden zweimal kurz die Testtaste drücken.

**5.1.24. 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 Pulsbreite 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 Frequenzwahl (durch ein Sternchen markiert).

In der Frequenzwahl 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 5.1. Pinbelegungen für typische 3-Pin-Servo-Stecker

**5.1.25. 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 44. 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 Scann Durchlauf starten. Anschluß siehe Seite 44.

Wie üblich beenden zwei kurze Tastendrucke die Funktion.

#### 5.1.26. DS18B20/DS18S20 OneWire Temperatursensoren werden ausgelesen.

Zum Einrichten des OneWire-Busses Seite 44 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 den Verbinden des DS18B20/DS18S20 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. Mit einem langen Tastendruck wählst Du den Auto-Modus (automatische Aktualisierung), welcher durch ein „\*“ in der ersten Zeile signalisiert wird.

Ein weiterer langer Tastendruck schaltet wieder zurück zum manuellen Modus.

Beschaltung für das DS18B20 Sensor:

Probe #1: Masse/Gnd

Probe #2: DQ (Daten)

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

**Hinweis:** Ein externer Pull-Up-Widerstand von 4,7kΩ zwischen #2 DQ und #3 Vcc benötigt!

Tips: - Option für DS18B20: Runden auf 0,1 °C/F (UI\_ROUND\_DS18B20)

- Option für DS18S20: hohe Auflösung (DS18S20\_HIGHRES)

- Die DS18B20-Funktion kann auch den Sensor DS1822 lesen.

Zum Beenden zweimal kurz die Test-Taste drücken.

#### 5.1.27. DHTxx-Temperatur & Luftfeuchte-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 4,7kΩ zwischen Data and Vdd wird benötigt! Manche Sensormodule haben bereits einen 10kΩ Pull-Up-Widerstand integriert, welcher ebenfalls gut mit kürzeren Kabeln funktioniert.

**Hinweis:** Wegen des Strombedarfs des Sensor kann der 680 Ω Testwiderstand nicht zur Strombegrenzung genutzt werden.

Also Vorsicht, ein Kurzschluss kann die MCU beschädigen.

#### 5.1.28. 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.

##### Hinweise:

- Passe MAX31855\_CS in config\_<MCU>.h an!
- Auch benötigst Du die SPI-Lesefunktion (SPI\_RW).

**5.1.29. BH1750 Umgebungslichtsensor** ist ein günstiger Umgebungslichtsensor mit I2C-Bus und 3,3V Betriebsspannung. Mit einem 3,3V Spannungsregler und Pull-Up-Widerständen für SCL und SDA (auf 3,3V) kann er meistens direkt an einem 5V ATmega betrieben werden, sofern keine anderen I2C-ICs am Bus Probleme machen. Ansonsten sind Pegelwandler nötig.

Zum Messen einmal kurz die Testtaste drücken. Alternativ kann mit einem langen Tastendruck in den automatischen Modus gewechselt werden (ein “\*” erscheint in der Titelzeile); und auch wieder zurück. Im automatischen Modus wird jede Sekunde gemessen. Bei einem Fehler zeigt der Tester ein “-“ statt der Lichtstärke. Und wie üblich, zwei kurze Tastendrucke beenden die Funktion.

##### Hinweise:

- BH1750-Module haben meistens einen 3,3V LDO und 4,7kΩ Pull-Up-Widerstände ( auf 3,3V) für SCL und SDA.
- Ein “Error“ direkt nach dem Funktionsaufruf deutet auf ein Sensorproblem hin, z.B. keine I2C-Kommunikation.

**5.1.30. Licht** Dies ist ein allgemeiner Schaltausgang zum Steuern von einem zusätzlichen Schaltungsteil, wie z.B. einem LED-Licht. Der Tester schaltet einfach den Ausgang zwischen Low und High um, jedes Mal bei Aufruf. Lasten kleiner 20mA können direkt betrieben werden.

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

Bei Problemen kannst Du das Warten mit einem Tastendruck abbrechen.

Der Selbsttest führt jeden Test 5-mal aus.

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

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

Die Testschritte sind:

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

##### Hinweise:

- Wenn die T2-Werte sich zwischen den Testpin-Paaren mehr als 5 mV unterscheiden, könnte das an zu ungleichen Rl-Testwiderständen liegen.
- Das gleiche gilt für die T3-Werte und Rh-Testwiderstände.
- Lange Testkabel haben einen negativen Einfluss und verschlechtern die Werte in T2, T3, T5 und T6.
- Seltsame Werte sind meistens ein Zeichen für defekte MCU-Pins, Kurzschlüsse, Schmutz oder größere Kontaktprobleme.

**5.1.32. 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 3,3µF hat.

Außerdem wird gleichzeitig der Offset der internen Spannungsreferenz gemessen. Bevor der Selbstabgleich ausgeführt wird, solltest Du einen Folienkondensator mit einer Kapazität zwischen 100nF und 3,3µF min. 3-mal messen, damit die oben erwähnten Offsets bestimmt werden können. Typischerweise liefert die erste Messung einen zu niedrigen Wert, die zweite einen zu hohen und erst die dritte einen korrekten Wert.

Das wird durch die sich selbst abgleichenden Offsets verursacht. Mit einem festen Kondensator zum Selbstabgleich wird der automatische Abgleich in der Kapazitätsmessung durch eine eigene Funktion ersetzt, welche während des Selbstabgleichs ausgeführt wird. Somit brauchst Du nicht mehr vorher einen Folienkondensator zu messen. Falls der Kapazitätsoffset zwischen den Test-Pin-Paaren zu sehr variiert, kannst Du in config.h, Abschnitt 5.2.7 ab Seite 43, 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 mΩ )
- 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 Ω (zwei in Reihe)
- Kapazität Testpin/Kabel < 100 pF
- Interner Widerstand vom IO-Pin im Low-Modus (RiL) < 25 Ω
- Interner Widerstand vom IO-Pin im High-Modus (RiH) < 29 Ω

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

- Falls der Kapazitätsoffset (A6) Null ist, dürfte es an einem Entladeproblem liegen, (CAP\_DISCHARGED erhöhen).

### **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.



**5.1.33. 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).

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

**5.1.35. Zeichensatz/Symbole** Die beiden Menüpunkte geben den kompletten Zeichensatz 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.

**5.1.36. Ausschalten** Hiermit kannst Du den Tester abschalten, sofern die Funktion über SW\_POWER\_OFF auf Seite 40 aktiviert wurde.

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

## 5.2. Die Mögliche Optionen in der config.h

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

### 5.2.1. Hardware Bedienung

#### Drehkoder zur Bedienung

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

#### Anzahl der Grey-Code-Impulse pro Schritt oder Rastung

- Ein Drehkoderimpuls 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 Rastunken oder Schritte

- wird zur Erfassung der Drehgeschwindigkeit des Drehkoders 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 Drehkoder an

#### Mehr/Weniger-Tasten zur Bedienung

- Alternative zu Drehkoder
- siehe KEY \_PORT für Port-Pins (config- <MCU> .h)
- //#define HW\_INCDEC\_KEYS - zum Aktivieren ein kommentieren

#### 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 ein kommentieren

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

- siehe Datenblatt der Spannungsreferenz
- oder nehme >= 5,5-stelliges DMM, um die Spannung zu messen
- #define UREF\_25 ... 2495 / -Bei Bedarf Wert ändern

#### Schutzrelais zum Entladen von Kondensatoren

- Standard-Pin: PC4 (ATmega 328)
- Low-Signal: Testpins kurzschließen
- High-Signal über externe Referenz: Kurzschluss beseitigen
- //#define HW\_DISCHARGE\_RELAY - zum Aktivieren ein kommentieren

#### 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 (für alternative Kontrollmodi siehe unten)
- Siehe TP\_ZENER für den Port-Pin in der config\_<MCU>.h
- //#define HW\_ZENER - zum Aktivieren ein kommentieren



### Nicht-Standard-Spannungsteiler für Zener-Prüfung

- Standard-Spannungsteiler ist 10:1
- ZENER\_R1: oberer Widerstand in  $\Omega$
- ZENER\_R2: unterer Widerstand in  $\Omega$
- // #define ZENER\_DIVIDER\_CUSTOM - zum Aktivieren ein kommentieren
- #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 oder wenn nur Messung einer externen Spannung (Schaltung ohne Hochsetzsteller)
- // #define ZENER\_UNSWITCHED - zum Aktivieren ein kommentieren

### alternativer Modus für Zener-Check: Wandler über dedizierten MCU-Pin schalten

- Aufwärtswandler wird über einen dedizierten I/O-Pin gesteuert
- siehe BOOST\_PORT in config\_<MCU>.h für Port-Pin
- zwei Ansteuerungsmethoden:
  - ZENER\_BOOST\_HIGH high activ / aktiviert wenn high
  - ZENER\_BOOST\_LOW low activ / aktiviert, wenn low
- Kommentar entfernen, um eine Ansteuerungsmethode zu aktivieren und zu wählen
- // #define ZENER\_SWITCHED
- // #define ZENER\_BOOST\_HIGH /\* hoch aktiv \*/
- #define ZENER\_BOOST\_LOW /\* niedrig aktiv \*/

### 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 ein kommentieren
- #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 ein kommentieren

### 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 ein kommentieren

### 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 ein kommentieren
- #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 ein kommentieren
- 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 ein kommentieren

### **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 ein kommentieren

### **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
- für einen Bestätigungston aktivieren SW\_IR\_RX\_BEEP
- // #define HW\_IR\_RECEIVER - zum Aktivieren ein kommentieren

### **Fester Kondensator für Selbstabgleich**

- Siehe TP\_CAP und ADJUST\_PORT für Port-Pins (config- <MCU> .h)
- // #define HW\_ADJUST\_CAP - zum Aktivieren ein kommentieren

### **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 ein kommentieren

### **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 ein kommentieren

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

- // #define HW\_CAP\_RELAY - zum Aktivieren ein kommentieren

## 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 ein kommentieren
- ```
#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
- BUZZER\_ACTIVE: aktiver Summer mit integriertem Oscillator  
BUZZER\_PASSIVE: pasivner Summer
- ```
// #define HW_BUZZER - zum Aktivieren ein kommentieren
#define BUZZER_ACTIVE /* active buzzer */ - wähle dein Typ
// #define BUZZER_PASSIVE /* passive buzzer */
```

## 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 ein kommentieren
```

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

- erfordert SPI-Bus und SPI-Leseunterstützung
- ```
// #define HW_MAX31855 - zum Aktivieren ein kommentieren
```

## LED licht Dies ist ein allgemeiner Schaltausgang bis 20 mA

- siehe FLASHLIGHT\_CTRL v config\_<MCU>.h für dein port pin
- ```
// #define HW_FLASHLIGHT - zum Aktivieren ein kommentieren
```

## BH1750VFI Umgebungslichtsensor

- erfordert I2C-Bus und I2C-Leseunterstützung
  - Kommentar entfernen, um zu aktivieren und die richtige I2C-Adresse auszuwählen
- ```
// #define HW_BH1750
#define BH1750_I2C_ADDR 0x23 /* I2C Adresse 0x23 (ADDR low) */
// #define BH1750_I2C_ADDR 0x5c /* I2C Adress 0x5c (ADDR high) */
```

## 5.2.2. 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 ein kommentieren

### PWM-Generator: zeigt auch die Impulsdauer an

- die Dauer basiert auf der Auflösung des Timers
- `#define PWM_SHOW_DURATION` - zum Deaktivieren auskommentieren

### 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 ein kommentieren

### ESR Tool ESR-Messung im Schaltkreis

- erfordert die Aktivierung von `SW_ESR` oder `SW_OLD_ESR`
- `//#define SW_ESR_TOOL` - zum Aktivieren ein kommentieren

### Drehencoder-Prüfung

- `//#define SW_ENCODER` - zum Aktivieren ein kommentieren

### 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

### Pinbelegung für IR-Empfängermodul

- besonders nützlich für Prüfgeräte mit ZIF-Buchse
- `#define SW_IR_RX_PINOUT_G_V_D` /\* 1-Gnd 2-Vcc 3-Data - default
- `//#define SW_IR_RX_PINOUT_D_G_V` /\* 1-Data 2-Gnd 3-Vcc \*/
- `//#define SW_IR_RX_PINOUT_D_V_G` /\* 1-Data 2-Vcc 3-Gnd \*/

### Strombegrenzungswiderstand für IR-Empfängermodul

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

### Bestätigungston für gültiges Datenframe/-paket

- erfordert Summer (`HW_BUZZER`)
- `//#define SW_IR_RX_BEEP` - zum Aktivieren ein kommentieren

### Zusätzliche Protokolle für den IR-Detektor/Dekoder

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

### 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 ein kommentieren

### **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 ein kommentieren

### **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 ein kommentieren

### **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 ein kommentieren

### **Temperatur messen mit dem DS18B20**

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

//#define SW\_DS18B20 - zum Aktivieren ein kommentieren

### **Temperatur messen mit dem DS18S20** OneWire Temperatur Sensor

- DS18S20\_HIGHRES: Einschaltung hohe Empfindlichkeit (0,01 °C)  
standart Empfindlichkeit ist 0,5°C

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

//#define SW\_DS18S20 - zum Aktivieren ein kommentieren

//#define DS18S20\_HIGHRES /\* high resolution (0.01°C) \*/ - ein kommentieren

### **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 ein kommentieren

### **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 ein kommentieren

### **Kondenstorleckstromtest**

- erfordert eine Anzeige mit mehr als zwei Zeilen

//#define SW\_CAP\_LEAKAGE - zum Aktivieren ein kommentieren

### **Anzeige der umgekehrten hFE für BJTs**

- hFE für Kollektor und Emitter vertauscht

#define SW\_REVERSE\_HFE - zum Deaktivieren auskommentieren

### **Anzeige I\_C/I\_E** Prüfstrom für hFE-Messung

- I\_C für gemeinsamen Emittterkreis
- I\_E für gemeinsamen Kollektorkreis

//#define SW\_HFE\_CURRENT - zum Aktivieren ein kommentieren

### **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 ein kommentieren
```

#### **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 ein kommentieren
```

#### **Zeigt zusätzliche Informationen zu einem möglichen Potentiometer/Trimpot an**

- zeigt die Summe beider Widerstände und das Verhältnis des ersten Widerstands in % an
- ```
// #define SW_R_TRIMMER - zum Aktivieren ein kommentieren
```

#### **Zeigt den Selbstentladung-Spannungsverlust (in %) eines Kondensators > 50nF**

```
// #define SW_C_VLOSS - zum Aktivieren ein kommentieren
```

#### **Fotodioden-Test**

```
// #define SW_PHOTODIODE - zum Aktivieren ein kommentieren
```

#### **Dioden/LED-Schnellprüfung**

- erfordert ein Display mit mehr als 2 Textzeilen
- ```
// #define SW_DIODE_LED - zum Aktivieren ein kommentieren
```

### **5.2.3. 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 ein kommentieren
```

## Alternative Stromversorgungssteuerung für Klone mit SCT15L104W Management MCU.

- Problem:  
Tester schaltet sich nach dem ersten Prüfzyklus plötzlich aus
- Betroffene Tester:  
T7-H, vermutlich auch andere Modelle der TC-1-Familie
- `//#define PASSIVE_POWER_CTRL` - zum Aktivieren ein kommentieren

### 5.2.4. Umungungslösungen 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)

```
798 #ifndef OSC_STARTUP
799     #define OSC_STARTUP 16384
800 #endif
```

Listing 5.1. Wert ändern falls er NICHT zum Tester passt!

### 5.2.5. Benutzerschnittstelle

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

- Vorwahl `_2` entspricht ISO 8859 -2. - Russisch ist immer Windows -1251.

```
826 // #define UI_ENGLISH
827 // #define UI_BRAZILIAN
828 // #define UI_CZECH
829 // #define UI_CZECH_2
830 // #define UI_DANISH
831 // #define UI_FRENCH
832 #define UI_GERMAN
833 // #define UI_ITALIAN
834 // #define UI_POLISH
835 // #define UI_POLISH_2
836 // #define UI_ROMANIAN
837 // #define UI_RUSSIAN
838 // #define UI_RUSSIAN_2
839 // #define UI_SPANISH
```

Listing 5.2. Wunschsprache auswählen,

**Komma anstelle eines Punktes** , um einen Dezimalbruch anzugeben.

- `// #define UI_COMMA` - zum Aktivieren ein kommentieren

**Temperaturen in Fahrenheit** anstelle von Celsius anzeigen.

- `// #define UI_FAHRENHEIT` - zum Aktivieren ein kommentieren

**Hexadezimale Werte in Großbuchstaben** statt Kleinbuchstaben anzeigen

- `// #define UI_HEX_UPPERCASE` - zum Aktivieren ein kommentieren

**Standard-Betriebsmodus Auto-Hold**

- anstelle des Dauermodus
- `#define UI_AUTOHOLD` - zum Deaktivieren auskommentieren

**Wechselt vorübergehend in den Auto-Hold-Modus**, wenn ein Bauteil erkannt wird.

- nur im kontinuierlichen Modus
- `// #define UI_AUTOHOLD_FOUND` - zum Aktivieren ein kommentieren

**Menüaufruf durch Kurzschluss** aller drei Testpins.

- altes Standardverhalten
- `// #define UI_SHORT_CIRCUIT_MENU` - zum Aktivieren ein kommentieren

**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 ein kommentieren

**Einstellungsprofil** - nach dem Einschalten auszuwählen.

- Für Tester mit zusätzlichen, fest eingebauten, Messkabeln.
- ```
//#define UI_CHOOSE_PROFILE
```
- zum Aktivieren ein kommentieren

**ein drittes Profil für Anpassungswerte**

- ```
//#define UI_THREE_PROFILES
```
- zum Aktivieren ein kommentieren

**Ausgabe über serielle TTL-Schnittstelle**

- aktiviere ebenfalls SERIAL\_BITBANG oder SERIAL\_HARDWARE (siehe 'Busse')
- ```
//#define UI_SERIAL_COPY
```
- zum Aktivieren ein kommentieren

**Bedienung des Testers über serielle TTL-Schnittstelle**

- aktiviere ebenfalls SERIAL\_BITBANG oder SERIAL\_HARDWARE und SERIAL\_RW
- ```
//#define UI_SERIAL_COMMANDS
```
- zum Aktivieren ein kommentieren

**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 ein kommentieren

**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 Testpinnummern über/unter dem Symbol

- erfordert, dass Komponentensymbole (SW\_SYMBOLS) aktiviert sind
- ```
//#define UI_PINOUT_ALT
```
- zum Aktivieren ein kommentieren

**Anzeige Fragezeichensymbol** bei fehlgeschlagenem Testlauf

- erfordert, dass Komponentensymbole (SW\_SYMBOLS) aktiviert sind
- ```
//#define UI_QUESTION_MARK
```
- zum Aktivieren ein kommentieren

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

- erfordert, dass Komponentensymbole (SW\_SYMBOLS) aktiviert sind
- ```
//#define UI_ZENER_DIODE
```
- zum Aktivieren ein kommentieren

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

- erfordert, dass Komponentensymbole (SW\_SYMBOLS) aktiviert sind
- ```
//#define UI_QUARTZ_CRYSTAL
```
- zum Aktivieren ein kommentieren

**DS18B20/DS18S20/DHTXX:** Anzeige des Sensorsymbols

- erfordert die Aktivierung von Komponentensymbolen (SW\_SYMBOLS)
- ```
//#define UI_ONEWIRE
```
- zum Aktivieren ein kommentieren (noch nicht unterstützt)

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

- erfordert die Aktivierung von Bauteilsymbolen (SW\_SYMBOLS)
- ```
//#define UI_NO_TEXTPINOUT
```
- zum Aktivieren ein kommentieren

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

- ```
//#define UI_NO_BODYDIODE_TEXTPINOUT
```
- zum Aktivieren ein kommentieren



**Batteriestatus:** Anzeigesymbol

- erfordert Schriftart mit zusätzlichen Zeichen (Schriftart prüfen!)
- nicht verfügbar für HD44780 und ST7036 basierte Displays
- kann nicht mit LCD\_VT100 verwendet werden

```
// #define UI_BATTERY - zum Aktivieren ein kommentieren
```

**Batteriestatus:** Anzeige in der letzten Zeile nach der Anzeige des Messergebnisses

```
// #define UI_BATTERY_LASTLINE - zum Aktivieren ein kommentieren
```

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

- erfordert Schriftart mit zusätzlichen Zeichen
- nicht verfügbar für HD44780 und ST7036 basierte Displays

```
// #define UI_PROBE_REVERSED - zum Aktivieren ein kommentieren
```

**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 ein kommentieren
```

**Buntes Cursor- und Tasten-Hinweise**

- erfordert farbige Grafikanzeige
- colors.h bearbeiten, um die bevorzugte Farbe (COLOR\_CURSOR) auszuwählen

```
// #define UI_COLORED_CURSOR - zum Aktivieren ein kommentieren
```

**farbige Werte**

- nur der Wert, nicht die Einheit
- erfordert farbige Grafikanzeige
- colors.h bearbeiten, um bevorzugte Farbe (COLOR\_VALUE) auszuwählen

```
// #define UI_COLORED_VALUES - zum Aktivieren ein kommentieren
```

**Menüs:** seitenweises Blättern im Menü statt positionsweises Blättern

- beschleunigt die Menübedienung bei Grafikdisplays, besonders bei hochauflösenden Farbdisplays

```
// #define UI_MENU_PAGEMODE - zum Aktivieren ein kommentieren
```

**Hauptmenü automatisch verlassen**

- nach Ausführung einer Funktion

```
// #define UI_MAINMENU_AUTOEXIT - zum Aktivieren ein kommentieren
```

**Menüpunkt Ausschalten**

```
// #define SW_POWER_OFF - zum Aktivieren ein kommentieren
```

**Hauptmenü: Anzeige der Schriftart** für Testzwecke

- Standard-Ausgabeformat:  
Indexnummer (hex) und 8 Zeichen (einschließlich nicht verfügbarer Zeichen)
- gepacktes Ausgabeformat:  
kein Index, nur verfügbare Zeichen, komplette Textzeile

```
// #define SW_FONT_TEST - zum Aktivieren ein kommentieren
```

```
// #define FONT_PACKED /* gepacktes Ausgabeformat */
```

**Anzeige von Komponentensymbolen** zu Testzwecken

- erfordert, dass Symbole aktiviert sind (SW\_SYMBOLS) im Kapitel 5.2.5 auf Seite 40

```
// #define SW_SYMBOL_TEST - zum Aktivieren ein kommentieren
```

**Runden der Werten für DS18B20**

- DS18B20 (0.1 °C/F)

```
// #define UI_ROUND_DS18B20 - zum Aktivieren ein kommentieren
```

### Infos und einige andere Texte mittig ausrichten

- erfordert Anzeige mit mehr als 3 Textzeilen
- `//#define UI_CENTER_ALIGN` - zum Aktivieren ein kommentieren

### Bestätigungssignal, wenn die Sondierung abgeschlossen ist

- erfordert Summer (HW\_BUZZER)
- `//#define UI_PROBING_DONE_BEEP` - zum Aktivieren ein kommentieren

### 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

### 5.2.6. 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\_DIRECT 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 verhindert das Abschalten, wenn die gemessene Spannung unter 0,9V liegt (verursacht durch den Leckstrom der Diode).
- `//#define BAT_EXT_UNMONITORED` - zum Aktivieren ein kommentieren

#### 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

### 5.2.7. 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  $V_{ref} = 5V$  ( $V_{cc}$ ) und 1,07 mV für  $V_{ref} = 1,1 V$  (Bandgap).

- Wird zur gemessenen Spannung der Bandgap-Referenz addiert.

```
#define UREF_OFFSET ... 0
```

- ggf. Wert anpassen

**Genaue Werte** der Testwiderstände

- Standardwert für Rl ist 680  $\Omega$

- Der Standardwert für Rh beträgt 470 k $\Omega$

/ Rl in  $\Omega$  /

```
#define R_LOW ... 680
```

- ggf. Wert anpassen

/ Rh in  $\Omega$  /

```
#define R_HIGH ... 470000
```

- ggf. Wert anpassen

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

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

- Der Standardoffset beträgt 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 ein kommentieren

**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

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

#### Lesefunktionen für das Anzeigemodul aktivieren

- Display-Treiber und Schnittstelleneinstellungen müssen dies unterstützen
- ```
//#define LCD_READ
```
- zum Aktivieren ein kommentieren

#### 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 ein kommentieren

#### 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 ein kommentieren

### 5.2.9. 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 ein kommentieren
  - 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

#### 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
```
- SPI-Leseunterstützung aktivieren
- 
- ```
//#define SPI_SLOWDOWN
```
- Verlangsamung von Bit-Bang-SPI

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

Wie schon erwähnt, kann die Firmware an verschiedene Tester und verschiedene Konfigurationen konfiguriert werden.

### 6.1. In der k-Version

konfigurierst du über die Makefile. Da der Entwickler für dieses Tester in einem Ordner Vor-konfiguriert hatte, brauchst du in der Praxis lediglich dein Programmer und Anzeigesprache einstellen. Die Zeile 204 habe ich nur aktiviert, weil der Programmschritt sowieso abläuft. Lediglich die Ausgabe wird unterdrückt:

|                  |                            |
|------------------|----------------------------|
| Software         | Original KHK Version 1.13k |
| Untenordner Name | mega328_color_kit          |
| Benutzt FLASH    | 98 %                       |
| Benutzt EEPROM   | 87.8 %                     |
| Zeile:           | Änderung im Makefile:      |
| 70               | UI_LANGUAGE = LANG_GERMAN  |
| 204              | CFLAGS += -DFREQUENCY_50HZ |
| 389              | PROGRAMMER=usbasp          |
| 390              | BitClock=20                |
| 391              | PORT=usb                   |

Tabelle 6.1. Benutzte k-Software und Modifikation im Makefile

Außerdem wurden mit Erfolg getestet : POLOLU a USBtiny ISP

### 6.2. In der m-Version

sieht die Situation ganz anders aus. Der Entwickler verlässt sich vollständig auf das technische Wissen seiner Unterstützer und ihr Sinn zum Experimentieren.

Als Hilfe dient die Datei Clones.txt, in der die verschiedenen Klone kurz beschrieben werden. Dieser Tester wird dort beispielsweise unter dem Namen AY AT Clone behandelt. (Wer ahnt es schon?) Dies war einer der Gründe, diesen Leitfaden zu verfassen.

Die Einstellungen sind hier verteilt über die Dateien [Makefile](#), [config.h](#) und [config\\_328.h](#).

### 6.3. Makefile

Im Makefile werden Einstellungen durch das Setzen von bestimmten Variablen durchführt. 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).

#### 6.3.1. MCU-Typ

```
25 MCU = atmega328
```

Listing 6.1. Vorgewählt ist atmega328

#### 6.3.2. MCU-Taktfrequenz

```
32 FREQ = 8
```

Listing 6.2. Vorgewählt ist 8MHz

#### 6.3.3. Oszillator-Typ

```
38 OSCILLATOR = Crystal
```

Listing 6.3. Vorgewählt ist Crystal

### 6.3.4. Avrdude MCU-Typ

```
72 PARTNO = m328p
```

Listing 6.4. Vorgewählt ist m328p

### 6.3.5. Avrdude ISP-Programmierer

Bei dem Programmierer sind notwendig:

- Name
- BitClock
- Port

```
76 # Arduino as ISP
77 #PROGRAMMER = stk500v1
78 #PORT = /dev/ttyACM0
79 #OPTIONS = -b 19200
80
81 # Bus Pirate
82 #PROGRAMMER = buspirate
83 #PORT = /dev/bus_pirate
84 #OPTIONS = -B 10.0
85
86 # Diamex ALL-AVR/AVR-Prog
87 PROGRAMMER = avrispmkII
88 PORT = usb
89 OPTIONS = -B 1.0
90
91 # Pololu USB AVR Programmer
92 #PROGRAMMER = stk500v2
93 #PORT = /dev/ttyACM0
94 #OPTIONS = -B 1.0
95
96 # USBasp
97 #PROGRAMMER = usbasp
98 #PORT = usb
99 #OPTIONS = -B 20
100
101 # USBtinyISP
102 #PROGRAMMER = usbtiny
103 #PORT = usb
104 #OPTIONS = -B 5.0
105
106 # Arduino Uno bootloader via serial/USB
107 #PROGRAMMER = arduino
```

Listing 6.5. Vorgewählt ist Diamex

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

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

Weitere Informationen findest Du im im Teil 7.7,

Avrdude-Handbuch oder der Online-Dokumentation [5].

## 6.4. config.h

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

### 6.4.1. Für diesen Tester muss geändert werden:

```
39 #define HW_ENCODER
```

Listing 6.6. Coder Einstellung

```
61 #define ENCODER_STEPS 20
```

Listing 6.7. Anzahl Rastunken

```
83 #define HW_REF25
```

Listing 6.8. Externe Referenz

```
117 #define HW_ZENER
```

Listing 6.9. Aktivierung von Voltmeter

```
140 #define ZENER_UNSWITCHED
```

Listing 6.10. Periodische Messung

```
167 #define HW_PROBE_ZENER
```

Listing 6.11. automatische Umschaltung zum Zusatzklemmen

```
191 #define HW_FREQ_COUNTER_BASIC
```

Listing 6.12. Aktivierung von Frequenzzähler

```
207 // #define FREQ_COUNTER_PRESCALER 16 /* 16:1 */
```

Listing 6.13. Deaktivierung Prescaler der nicht da ist

**6.4.2. Nun folgt die Qual der Wahl** der Unmenge von möglichen Optionen der config.h, wählen zu müssen, weil der Speicher Grenzen setzt. Glücklicher weise, kann man mehrere Sätze erstellen, um sie, je nach Bedarf, wechseln zu können.

```
235 #define HW_EVENT_COUNTER
```

Listing 6.14. Aktivierung von Ereigniszähler

```
245 #define EVENT_COUNTER_TRIGGER_OUT
```

Listing 6.15. Aktivierung von Triggerausgang für Ereigniszählers

```
388 // #define SW_PWM_SIMPLE
```

Listing 6.16. Deaktivierung einfachen PWM

```
398 #define SW_PWM_PLUS
```

Listing 6.17. Aktivierung erweiterten PWM

```
407 #define PWM_SHOW_DURATION
```

Listing 6.18. zeigt beim PWM auch die Impulsdauer an )

```
425 // #define SW_ESR
```

Listing 6.19. Deaktivierung IR Empfänger (kein Modul vorhanden)

453 **#define** SW\_SQUAREWAVE

Listing 6.20. Aktivierung von Rechtecksignalgenerator)

463 *//#define SW\_IR\_RECEIVER*

Listing 6.21. Deaktivierung IR Empfänger (kein Modul vorhanden)

541 *//#define SW\_OPTO\_COUPLER*

Listing 6.22. Deaktivierung von Opto Kopplern

549 *//#define SW\_UJT*

Listing 6.23. Deaktivierung SW\_UJT

557 *//#define SW\_SCHOTTKY\_BJT*

Listing 6.24. Deaktivierung Schotky UJT

567 *//#define SW\_SERVO*

Listing 6.25. Aktivierung von Servos testen

577 **#define** SW\_DS18B20

Listing 6.26. Aktivierung von Temperatur Messung mit DS18B20

1168 **#define** UI\_ROUND\_DS18B20

Listing 6.27. Erhöhung der Messgenauigkeit auf 0,1°C

589 **#define** DS18S20\_HIGHRES */\* high resolution (0.01 C) \*/*

Listing 6.28. Aktivierung von Temperatur Messung mit DS18S20 auf 0,01°C

670 **#define** SW\_DHTXX

Listing 6.29. Temperatur und Feuchte Messung mit DHT11 und DHT22 Sensoren

724 **#define** SW\_R\_TRIMMER

Listing 6.30. zeigt die Summe beider Widerstände und das Verhältnis in % an

740 **#define** SW\_PHOTODIODE

Listing 6.31. Aktivierung Messung von Foto Dioden

832 *//#define UI\_GERMAN*

Listing 6.32. Deaktivierung von englisch und Aktivierung von deutsch

847 **#define** UI\_COMMA

Listing 6.33. Benutzung von Strich statt Punkt

872 **#define** UI\_AUTOHOLD

Listing 6.34. Aktivierung von Einzelmessung

890 **#define** UI\_SHORT\_CIRCUIT\_MENU

Listing 6.35. Menüaufruf durch Kurzschluss aller drei Testpins

901 **#define** UI\_KEY\_HINTS

Listing 6.36. Aktivierung Hilfe für Menü



```
867 #define POWER_OFF_TIMEOUT 30
```

Listing 6.37. Automatische Abschaltung nach 30 Sec.

```
995 #define UI_QUESTION_MARK
```

Listing 6.38. Bei nicht erfolgreichem Messen Fragezeichen anzeigen

```
1013 #define UI_QUARTZ_CRYSTAL
```

Listing 6.39. Anzeige des Quarz Symbols für LF/HF-Modi

```
1050 #define UI_BATTERY
```

Listing 6.40. Batterie Ikone anzeigen

```
1058 #define UI_BATTERY_LASTLINE
```

Listing 6.41. Batterie Zustand nach dem Messen anzeigen

```
1079 //#define UI_PROBE_COLORS
```

Listing 6.42. Deaktivierung von Farbkennzeichnung der Testpins

```
1099 #define UI_COLORED_CURSOR
```

Listing 6.43. Buntes Cursor- und Tasten-Hinweise

```
1110 #define UI_COLORED_VALUES
```

Listing 6.44. farbige Werte anzeigen

```
1136 #define SW_POWER_OFF
```

Listing 6.45. Aktivierung der Vorwahl Ausschalten

```
1177 #define UI_CENTER_ALIGN
```

Listing 6.46. Texte mittig ausrichten

```
1196 //#define DATA_EEPROM /* store data in EEPROM */
```

```
1197 #define DATA_FLASH /* store data in Flash */
```

Listing 6.47. Aktivierung von speichern der Daten im Flash

```
1563 #define ONEWIRE_PROBES /* via probes */
```

Listing 6.48. Aktivierung von OneWire Probes

## 6.5. Config\_328.h

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

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

Listing 6.49. Aktives LCD ST756R Modul deaktiviert

```
672 #endif
```

Listing 6.50. Aktives LCD ST756R Modul deaktiviert

```
681 // #if 0
682 #define LCD_ST7735          /* display controller ST7735 */
```

Listing 6.51. LCD\_ST7735 aktiviert

```
689 #define LCD_RES    PD0      /* port pin used for /RESX (optional) */
690 #define LCD_CS     PD5      /* port pin used for /CSX (optional) */
691 #define LCD_DC     PD1      /* port pin used for D/CX */
692 #define LCD_SCL    PD2      /* port pin used for SCL */
693 #define LCD_SDA    PD3      /* port pin used for SDA */
```

Listing 6.52. Pin Zuordnung anpassen

```
699 #define LCD_FLIP_X      /* enable horizontal flip */
```

Listing 6.53. horizontales drehen ein kommentieren

```
703 // #define LCD_LATE_ON    /* turn on LCD after clearing it */
```

Listing 6.54. Für Beginn mit einem leerem Display Kommentar deaktivieren

```
753 // #define FONT_8x16_ALT_HF      /* 8x16 alternative font */
754 // #define FONT_10X16_HF         /* 10x16 font */
755 // #define FONT_6X8_IS08859_2_HF /* 6x8 Central European font */
756 // #define FONT_8X8_IS08859_2_HF /* 8x8 Central European font */
757 #define FONT_8X12T_IS08859_2_HF /* thin 8x12 Central European font */
758 // #define FONT_8X16_IS08859_2_HF /* 8x16 Central European font */
759 // #define FONT_10X16_IS08859_2_HF /* 10x16 Central European font */
```

Listing 6.55. Wähle eine Schriftart. Du kannst auch eine andere hinzufügen die sich im Bitmaps-Verzeichnis befindet. Alle Fonds, die HF am Ende haben, sind möglich.

```
714 #define SYMBOLS_24X24_HF      /* 24x24 symbols */
715 // #define SYMBOLS_30X32_HF    /* 30x32 symbols */
716 // #define SYMBOLS_30X32_ALT1_HF /* 30x32 alternative symbols #1 */
717 // #define SYMBOLS_30X32_ALT2_HF /* 30x32 alternative symbols #2 */
718 // #define SYMBOLS_32X32_HF    /* 32x32 symbols */
719 // #define SYMBOLS_32X32_ALT1_HF /* 32x32 alternative symbols #1 */
```

Listing 6.56. Hier kannst Du aus Symbole hinzufügen

In der Zeile 714 wurde eine Symbolreihe der Größe 24x24 Pixel aus dem Verzeichnis hinzugefügt.

**Hinweis:** Je größer die Zeichen sind, desto mehr Speicherplatz wird verbraucht und desto weniger Platz bleibt für Optionen.

```
726 // #endif
```

Listing 6.57. Kommentieren am Ende nicht vergessen

```

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

```

Listing 6.58. Einstellung der A Spur des Drehkoderns

```

963 #define KEY_INC PD1 /* increase push button (low active) */
964 #define KEY_DEC PD3 /* decrease push button (low active) */

```

Listing 6.59. Einstellung der A Spur des Drehkoderns

```

993 #define IR_PORT PORTC /* port data register */
994 #define IR_DDR DDRC /* port data direction register */
995 #define IR_PIN PINC /* port input pins register */
996 #define IR_DATA PC6 /* data signal */

```

Listing 6.60. Einstellung für IR Detektor/Decoder

**6.5.2. Information** Diese Einstellung wurde so gewählt, damit ein Vergleich der Versionen möglich war. Zum aktivieren von anderen Funktionen in der m-Version, musst du, für dich unwichtige, Optionen in der v config.h deaktivieren und andere Optionen dafür aktivieren.

|            |                 |
|------------|-----------------|
| Software   | m-version 1.52m |
| Clone Name | AY AT Clone     |
| Program    | 98,2 %          |
| Data       | 12,1 %          |
| EEPROM     | 2,1 %           |

Tabelle 6.2. Data für aktivierte m-Software für GM328A

**Bemerkung:** falls zu viele Optionen, **zu viel Platz** im ATmega 328, fördern, zum Beispiel bei dem Versuch in der config\_328 bei dem LCD Modul auf Seite 47 Zeile 754 font 10x16\_HF zu aktivieren,...erscheint ...nach dem ... **Aufruf** **make**...

```

/usr/lib/gcc/avr/5.4.0/../../../../avr/bin/ld: ComponentTester section `.text' will not fit in region `text'
/usr/lib/gcc/avr/5.4.0/../../../../avr/bin/ld: region `text' overflowed by 2234 bytes
collect2: Fehler: ld gab 1 als Ende-Status zurück
make: *** [Makefile:198: ComponentTester] Fehler 1
bohu@fuji-w:/media/bohu/Programme/Elektronik/TTester/GM328A/GM328a_52m$

```

Abbildung 6.1. hier wird die Gier, durch den Compiler bestraft ;-)

nach der Änderung auf font 8x12T\_ISO8859\_2\_HF sieht es schon viel besser aus:

```

AVR Memory Usage
-----
Device: atmega328

Program:  32166 bytes (98.2% Full)
(.text + .data + .bootloader)

Data:      248 bytes (12.1% Full)
(.data + .bss + .noinit)

EEPROM:    22 bytes (2.1% Full)
(.eeprom)

bohu@fuji-w:/media/bohu/Programme/Elektronik/TTester/GM328A/GM328a_52m$

```

Abbildung 6.2. nun reicht es : **make upload**

a ...die Operation war erfolgreich. Ctester ist jetzt persönlich und „unbezahlbar“.

**Vergiss nicht den Selbstset, Selbstabgleich und da nach... die Ergebnisse im Speicher abzulegen!** nach dem Kapitel 5.1.31.

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 [2] 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 [2] übernommen.

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

### 7.1. Konfigurieren des Component Testers

Dazu bitte Kapitel 6 ab der Seite 45 lesen.

### 7.2. Programmierung des Mikrocontrollers

Die Programmierung des Testers wird über die Datei Makefile gesteuert.

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

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

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

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

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

Beide Dateien müssen in den richtigen Speicher geladen werden.

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

Wenn Sie das Makefile zusammen mit dem Programm avrdude [5] benutzen, brauchen Sie keine genaue Kenntnis über die Einzelheiten der fuses.

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

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

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

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

### 7.3. Betrieb System Linux

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

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

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

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

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

## 7.4. 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.

## 7.5. 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.

## 7.6. Download der Quellen

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

```
git clone https://github.com/kubi48/TransistorTester-source
```

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

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-source' löschen oder umzubenennen und dann mit dem oben benannten Anweisung neu downloaden.

## 7.7. 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 Programmierer 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}=="1fffb", MODE="0666"
```

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

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

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

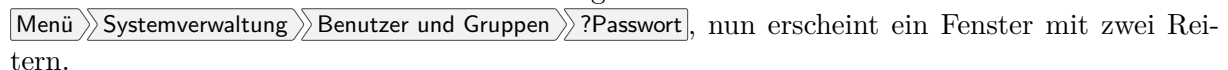
## 7.8. Gruppen Mitgliedschaft

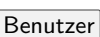

Deswegen sollte die eigene Benutzerkennung sowohl Mitglied der Gruppe 'plugdev' als auch der Gruppe 'dialout' sein. Das Kommando:

```
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.

## 7.9. Arbeitsumgebung

Vorbereitung.

Zu erst in der Taskleiste mit Ordnersymbol (Nemo) zum /TransistorTester-source/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-source/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:

```
ln -s ~/TransistorTester-source/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 6.3.5, auf der Seite 46, Thema PROGRAMMER.

## 7.10. 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)          |

**Hinweise** zu Compiler/Linker-Optimierungen:

Im Makefile gibt es mehrere Zeilen von CFLAGS und LDFLAGS mit Compiler und Linker-Optionen. Daneben gibt es auskommentierte Zeilen mit zusätzlichen Optimierungsoptionen, welche die Größe der Firmware reduzieren können, aber nicht von allen Compilern unterstützt werden.

Hier ist Experimentieren angesagt. ;)

**Hinweise** zu speziellen Einstellungen im Makefile:

- In einer Linux/Unix-Umgebung kann OPTIMIZE\_VECTORS aktiviert werden, um die Interruptvektortabelle zu optimieren und damit die Firmwaregröße zu reduzieren.



## 7.11. Arbeitsumgebung mit der k-Version

Hier bietet sich an, ein neues Textdokument zu erstellen und hinein den richtigen Pfad einfach von dem Terminal Fenster zu kopieren. Zu dem kannst du in den deine oft benutzte Befehle einschreiben, damit du sie bei der Hand hast. Der Vorgang ist folgend:

Drücke gleichzeitig [Strg] + [Alt] + [t] und kopiere dahin folgendes Befehl:

```
xed r-Version.txt [Enter]
```

und ein neues Dokument ist erstellt und geöffnet. Hinein kopiere die nächsten Zeilen:

```
cd TransistorTester-source/trunk/mega328_color_kit/  
xed Makefile  
make clean  
make  
make fuses-crystal  
make upload
```

- Und deine weitere Bemerkungen. Nun musst du es nur auf der Arbeitsfläche speichern.

Nun bleibt nur die Freude über das erreichter Erfolg.

## 7.12. Nötige Hardware zum Programmieren

Für die Anfänger die noch nichts besitzen gelten die folgende Informationen.

**7.12.1. Programmer** braucht unten Linux kein Treiber. Du musst ihm lediglich, wie im Teil 7.7 und in der Makefile 6.3.5 korrekt einstellen.

Ein großer Vorteil von 'avrdude' ist, das es mit preiswerten Programmer zufrieden ist, der du schon unten 2 Euro bekommst.

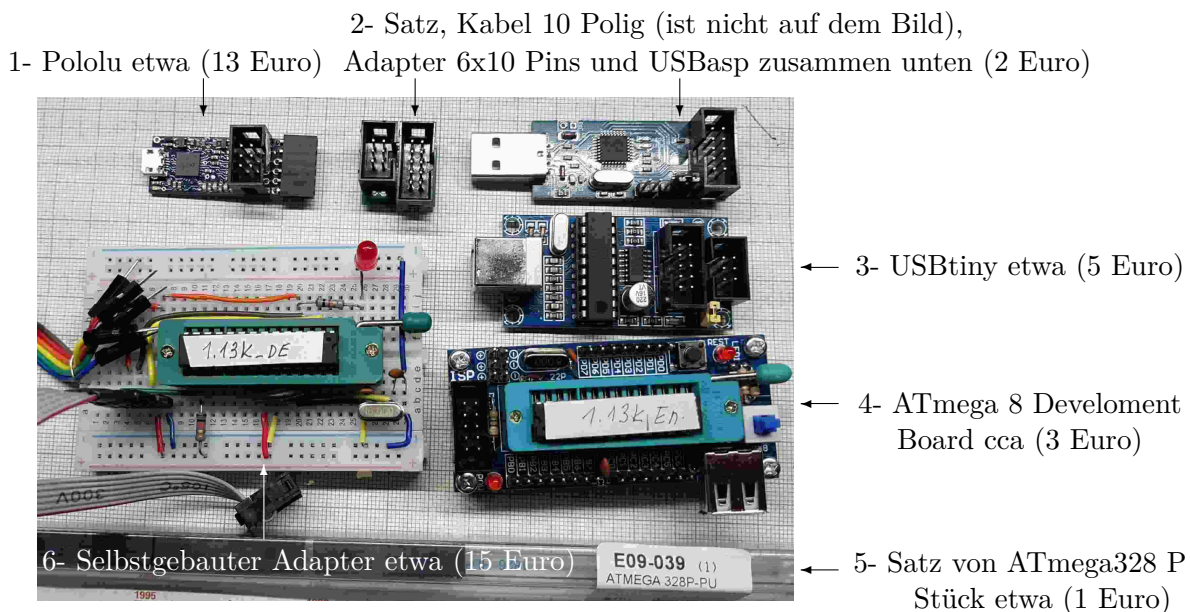


Abbildung 7.1. Hardware

Wie du aus dem Bild 7.1 entnehmen kannst, brauchst du nur

- (2)- Programmer
- (4)- kleine Entwicklungsplatine
- (5)- und für Sicherheit Ersatz ATmega 328 P.

Das alles, bekommst du, zusammen für unten 10 Euro.

Natürlich, kannst auch deine Umgebung, wie unten (6) selbst bauen, falls du aber die darin benutzten Teile noch kaufen musst, lohnt es sich nicht.



|                          |                                                 |
|--------------------------|-------------------------------------------------|
| Model                    | GM 328 A                                        |
| Größe                    | 78 x 63 x 28 mm                                 |
| Bauart                   | SMD                                             |
| AVR                      | ATmega 328P                                     |
| Quarz                    | 8 MHz                                           |
| Anzeige                  | ST7735 (128 x 160 Pixel)                        |
| IDE möglich              | nein                                            |
| Bedienung                | Impulsdrehgeber mit Taster                      |
| Spannungsversorgung      | 9V Netzteil oder 9V E-Block                     |
| Verbrauch Betrieb        |                                                 |
| Verbrauch Standby        | 20 nA                                           |
| Messspannung             | 5V                                              |
| Messstrom                | 6 mA                                            |
| Bestimmung und Messung   | Transistoren, MOSFET, JFET, P-IGBT, Dioden,     |
| Bestimmung und Messung   | Zenerdioden bis 5V, Thyristoren und Triacs      |
| Bestimmung und Messung   | Widerstände, Kondensatoren, Spulen, Quarzen     |
| Frequenz Messung         | 1 Hz - 2MHz                                     |
| Frequenz Generierung     | 1 Hz - 2MHz                                     |
| Generierung von Impulsen | Festfrequenz von 7,8 kHz; Impulsbreite 1% - 99% |
| Spannungsmessung         | 0V - 50 V DC                                    |
| Bereich Widerstände      | 0,01 -                                          |
| Bereich Kondensatoren    | 1pF - 100mF                                     |
| Bereich Spulen           | 0,01mH -                                        |

Tabelle 8.1. Technische Daten

### 8.1. Hilfe

kannst du im deutschen Forum versuchen [6].

Änglisches Forum erreichst du auf [7]

und fallst du slowakisch beherrscht auf [8].

### 8.2. Und für Entspannung

oder für die Beschäftigung deines Nachwuchses: [12].

Folgendes Schema zeigt eine Bauteilverbindung **ohne Garantie** auf Vollständigkeit. Das Schema besitzt keinen Maßstab. Eingebaute Bauteile wurden mit diesem Tester bestimmt. Bei den Kondensatoren war die Bauteilbestimmung nicht möglich.

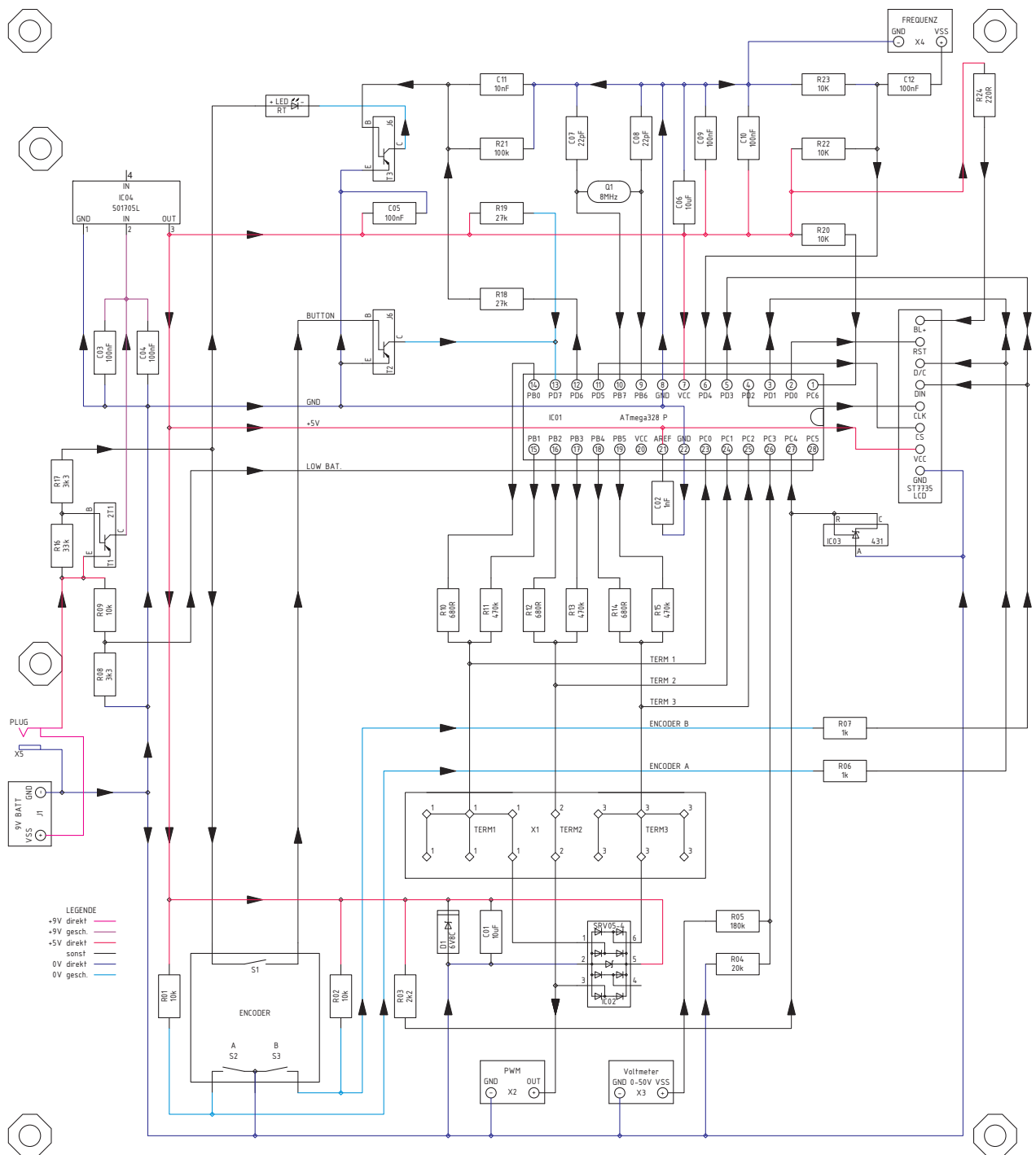


Abbildung 8.1. Schema Bauteile Verbindungen

Wie es sichtbar ist, wäre sehr einfach möglich, den zweipoligen PWM Stecker (X2) für ein drei-poligen auszutauschen um Festanschluss von Testkabeln zu realisieren.



---

## *Literaturverzeichnis*

---

- [1] <http://www.mikrocontroller.net/articles/AVR-Transistortester>  
*Online Dokumentation des Transistortesters, Online Article, 2009-2011*
- [2] <https://github.com/kubi48/TransistorTester-documentation>  
*Aktuelle Anleitung zum Transistor Tester*
- [3] <https://github.com/madires/Transistortester-Warehouse>  
*Komplette Software Sammlung*
- [4] <https://github.com/kubi48/TransistorTester-source>  
*Komplette Software Sammlung*
- [5] <http://www.mikrocontroller.net/articles/AVRDUDE>  
*Online Dokumentation avrdude IDE*
- [6] <https://www.mikrocontroller.net/topic/248078>  
*Hauptsprache ist Deutsch, Englisch ist auch ok.*
- [7] [https://www.eevblog.com/forum/testgear/\(dolar-zeichen\)  
20-lcr-esr-transistor-checker-project/](https://www.eevblog.com/forum/testgear/(dolar-zeichen)20-lcr-esr-transistor-checker-project/)  
*Nur Englisch.*
- [8] <https://svetelektro.com/> *Všetko zo sveta elektroniky*  
*Ein Slowakisches Elektronik Portal seit 2006*
- [9] [https://www.ebay.de/itm/usbasp-avrisp-usbisp-Programmer-usb-10Pin\  
-Convert-to-6P-Adapter-Board-STK50-AHS/302923364644](https://www.ebay.de/itm/usbasp-avrisp-usbisp-Programmer-usb-10Pin-Convert-to-6P-Adapter-Board-STK50-AHS/302923364644)  
*Programmer.*
- [10] [https://https://www.ebay.de/itm/335327569081?itmmeta=  
01HXF2CAJ87N4E5YAMYEMGXK8B&hash=item](https://https://www.ebay.de/itm/335327569081?itmmeta=01HXF2CAJ87N4E5YAMYEMGXK8B&hash=item)  
*ATmega 8 Entwicklung Umgebung*
- [11] [www.aliexpress.com/item/4000069589587.html](http://www.aliexpress.com/item/4000069589587.html)  
*Hiland644 Tester*
- [12] <https://dragaosemchama.com/en/2017/01/rex/>  
*Spiel Tetris für den Tester und weitere Infos*