Ergänzung zur Bedienung des AVR-Transistortester

K.-H. Kübbeler

Gilt nur für Testversion 0.92k

Einleitung:

Basierend auf der Software von Markus Frejek habe ich angefangen die Software zu modifizieren. Der Grund war zu Beginn eigentlich nur ein Problem, das ich mit dem Programmieren des EEprom hatte. Da sich der Flashspeicher aber problemlos programmieren ließ, dachte ich mir, das ich am schnellsten und günstigsten zu einer lauffähigen Version komme, wenn ich die Texte und Parameter einfach aus dem EEprom in den Flashspeicher verlagere. **Ab Version 0.92k kann in der Makefile (USE_EEPROM) gewählt werden, ob EEprom benutzt wird.** Aber ich hatte mir in diesem Zusammenhang die Software genauer angesehen und hatte dabei zunächst eine Idee, die ich dann auch testweise umgesetzt habe:

Für Ausgaben auf dem Display wird die Spannung in mV gebraucht und nicht die Schritte des AD-Wandlers. Das muss in der Originalsoftware jedes Mal umgerechnet werden. Die original AD-Wandler Routine (ReadADC) liest den Wandler zwanzig Mal, addiert die Ergebnisse und dividiert danach durch 20. Damit ist die Auflösung wieder 5000mV/1023 . Wenn ich nun den AD-Wandler statt 20 Mal 22 Mal lesen lasse, das Ergebnis nochmal verdopple und danach durch 9 dividiere ist mein Maximalwert 22*2*1023/9=5001 was prima zu der gewünschten mV Anzeige passt.

Mit dieser Idee fing dann richtige Arbeit an. Jetzt mußte ich alle Spannungsabfragen im Programm auf die neue Auflösung anpassen. So lernte ich das Programm von Markus Frejek immer besser kennen. Meine aktuelle Version von ReadADC liest 45 Mal, wobei der jeweils erste Messwert vergessen wird , die nächsten 44 Messungen werden addiert und durch 9 geteilt. Da ich anfangs auch noch den Ehrgeiz hatte, den Code der abgespeckten Version in 4K Programmspeicher zu bekommen, habe ich sicher auch Änderungen vorgenommen die bei richtiger Betrachtung nicht nötig waren. So habe ich beispielsweise die Warteschleifen durch Aufrufe einer selbst entwickelten Assembler-Warteschleifenroutine ersetzt, die selbst nur 66 Byte lang ist, jeder Aufruf kostet aber nur 1 Wort Befehl (2 Byte) und es wird ein Bereich von 10µs bis 5s im Raster
1 2 3 4 5 10 …. zur Verfügung gestellt. Die Routine beinhaltet ab 100ms schon den WatchDog Reset.

Zwischenwerte wie z.B. 8ms Wartezeit kosten 2 Aufrufe. Mir ist keine Technik bekannt, die bei vielen Warteaufrufen wirtschaftlicher wäre, wobei meine Technik Taktgenau arbeitet, wenn die unterste Routine (10µs) das tut. Lediglich durch den zusätzlichen WatchDog Reset ist die 100ms um 1µs bei 1MHz Taktrate zu lang. Die Aufrufe der Routinen benötigen keine Register mit Ausnahme des Stack-Pointers für die Verwaltung der Rückkehradressen im RAM. Das sind in der derzeitigen Version maximal 28 Byte. So konnte ich wertvollen Speicherplatz einsparen. Das Programm für die 8MHz Variante wird dadurch kürzer als die 1MHz Variante, da ich dann auch in den LCD-Routinen mit dieser Wartetechnik arbeite (es stehen dann auch Aufrufe für 1µs,2µs,3µs,4µs und 5µs zur Verfügung).

Neue Features und Änderungen:

1. Die Widerstandsmessung wurde so erweitert, das auch Potis oder Trimmer angeschlossen werden können. Die Widerstandsbelegung wird in Zeile 1 des LCD-Displays dann im Format 1-□-2-□-3 angezeigt, wobei in der zweiten Zeile dann der Widerstand zwischen Pin 1 und Pin 2 und dahinter der zwischen Pin 2 und 3 angezeigt. Wenn man zu faul ist den Gesamtwiderstandswert auszurechnen, kann man den mittleren Pin entfernen und die Messung neu starten. Wenn der Schleifer des Potis an einem Ende steht, kann der Transistortester den mittleren von dem Endpin nicht mehr unterscheiden!

2. Kondensatoren werden in symbolischer Darstellung angezeigt im Format Pinnummer-Kondensatorsymbol-Pinnummer.

3. Die Messung von Kondensatoren wird durch eine Ladezeitmessung ermittelt. In der Originalsoftware ist das eine Programmschleife, die den jeweiligen Input-Pin abfragt und mitzählt. Nachteil des Softwarezählers ist, dass die Schleifenzeit (Anzahl der Befehle für einen Durchlauf) selbstverständlich in die Berechnung einfließt. Ich habe diese Lösung für kleine Kondensatoren (bis ungefähr 50µF) durch eine technische Möglichkeit des AVR's ersetzt, das ein Zähler, der mit der vollen Taktfrequenz (also 1MHz bzw. auch 8MHz) laufen kann, seinen Zählerstand bei einem äußeren Ereignis zwischenspeichern kann. Dieses äußere Ereignis kann auch der Komparator-Ausgang des AVR's sein. So entlade ich den Kondensator, starte den Zähler bei 0 und starte die Ladung des Kondensators mit dem 470kΩ Widerstand. Jetzt frage ich in der Schleife ab, ob entweder ein Zählerübertrag oder ein Zwischenspeicher Ereignis eingetreten ist. Die Übertrags-Ereignisse zähle ich brav mit und beim Eintreten des Zwischenspeicher-Ereignisses stoppe ich den Zähler und prüfe, ob ich noch einen Übertrag zusätzlich machen muss, denn leider lässt sich der Zähler nicht automatisch durch das Zwischenspeicher-Ereignis stoppen. Von der so ermittelten Gesamtzeit ziehe ich bei der Messung von kleinen Kondensatoren eine experimentell bestimmte Konstante ab. Inwieweit diese Konstante von Exemplar zu Exemplar angepasst werden muss, kann ich nicht sagen. Größere Kondensatoren über 50µF werden in einem Vortest gemessen, indem mit bis zu 500 einzelnen 10ms Ladepulsen über den 680Ω Widerstand solange geladen wird, bis die Ladespannung im stromlosen Zustand mehr als 300mV beträgt. Die Kapazität wird dann aus der Zahl der Ladepulse und der erreichten Ladespannung aus einer Tabelle ermittelt. Die Tabelle enthält die Umrechnungsfaktoren von Ladezeit in nF im 50mV Abstand und wird interpoliert.
   Die niedrige Ladespannung führt aber zu einer deutlichenVerkürzung der Messzeit, die zweimal wirkt, einmal beim Laden und nochmal beim Entladen. Außerdem stören parallel geschaltete Dioden meist nicht mehr, da die Flussspannung nicht erreicht wird.

4. Die Messung von Kondensatoren wird nur noch in 3 Kombinationen gemessen. Pin1 (-)und Pin 3 (+), Pin 1 und Pin 2, sowie Pin 2 und Pin 3. Auf die Messungen mit umgekehrter Polarität wird verzichtet.

5. Leider litt die Messgenauigkeit der Kapazitätsmessung unter den Exemplar-Streuungen der AVR's, wie ich aus der bisher einzigen Rückmeldung wusste. Als Grund vermute ich die große Exemplar-Streuung der internen 1,3V Referenz, die ich bei der Kapazitätsmessung als Vergleichswert benutze. Deshalb besitzt die aktuelle Software intern eine Tabelle im 50mV Abstand, in der die theoretische Abhängigkeit der Ladezeit von der Vergleichs-Spannung abgelegt ist. Jetzt wird vor der Messung die Referenzspannung im Verhältnis zu der 5V

Betriebsspannung gemessen und der entsprechende Umrechnungsfaktor für die Kapazitätsmessung linear interpoliert. Diese Funktion kostet natürlich wertvollen Programmspeicherplatz. Ob damit die Exemplar-Streuungen kompensiert werden können muss noch getestet werden. Alternativ wäre auch eine Lösung denkbar, bei der die Spannung nur einmal vorher gemessen wird und dann der berechnete Faktor einmal in der individuellen Software vom Compiler eingetragen werden. Das würde aber bedeuten, dass jeder Nutzer seinen Tester kalibrieren muss, wenn er genauere Messergebnisse wünscht. Außerdem ist mir nicht bekannt, wie sich die Referenzspannung mit der Betriebstemperatur ändern. Jetzt wird bei jedem Einschalten neu gemessen, wenn die Funktion in der Makefile gewählt wird (WITH_AUTO_REF).

6. Die Entladefunktion von Kondensatoren wurde insofern verändert, das der Kondensator bei Unterschreitung von 1300mV über die Ausgabe Ports der ADC Pinne kurzgeschlossen wird. Das halte ich für vertretbar, da der Innenwiderstand von jedem Port Ausgang etwa 20Ω beträgt. Auch die im Datenblatt in Figure 149 (Seite 258) gezeigten Kurven gehen bis 2V. Garantieren kann ich das natürlich nicht. Ich habe mit meinem Atmega8L die Funktion mit 15mF Kondensatoren ausprobiert, ohne das ich einen Schaden bemerkt hätte. Der Strom sollte nach Rechnung auch unter den spezifizierten 40mA bleiben. Schaden kann der AVR natürlich nehmen, wenn HV Kondensatoren vor der Messung nicht entladen wurden.

7. Das Format der Pin-Angabe bei Dioden (A: K:) habe ich durch eine Symboldarstellung ersetzt, die sprachunabhängig jedem Bastler bekannt sein sollte. Leider lässt das Display keine vernünftige Zehnerdioden oder Doppeldiodendarstellung zu, weshalb die Darstellung mit Einzeldioden in Flussrichtung dargestellt wird. Auf die Pin-Angaben achten! Wenn bei zwei Dioden die äußeren Pinnummern gleich sind, kann es sich um Zehnerdioden oder Doppel-LEDs oder ähnliches handeln. Hierbei auf die Flussspannungen achten.

8. Die Diodenmessung von Einzeldioden habe ich mit einer Kapazitätsmessung in Sperrrichtung ergänzt. Damit kann man vielleicht die Eignung für bestimmte Zwecke abschätzen. Hier habe ich Werte von wenigen pF bis zu 5nF (Basis Emitter Strecke eines BU508A) gemessen. Eine Parallelschaltung aus Kondensator und Diode wird bis etwa 330µF und normaler Flussspannung der Diode noch erkannt. Bei größeren Kapazitätswerten wird nur der Kondensator erkannt. Selbstverständlich kann man die Messung auch bei den Transistor-Dioden durchführen, wenn man nur 2 Klemmen also die Basis und entweder Kollektor oder Emitter anschließt.

9. Die Darstellung der Pin-Nummern bei bipolaren Transistoren habe ich aus programm-ökonomischen Gründen durch eine EBC=123 Darstellung ersetzt.

10. Die Transistormessung wurde durch eine Messung in Kollektorschaltung (Emitterfolger) ergänzt. Bei Leistungstransistoren wird so ein vernünftigerer hFE bestimmt. Bei dieser Messmethode besteht keine Gefahr der Sättigung der Basis, auch wenn die Basis mit dem 680Ω Widerstand gespeist wird. Bei Darlington-Transistoren ist der Spannungsabfall bei einem 680Ω Basiswiderstand zu klein, so dass ab Version 9.2k gegebenenfalls auch mit dem 470KΩ Basiswiderstand gemessen wird. Die übliche Messmethode mit Emitterschaltung wird ebenfalls durchgeführt und der jeweils höhere Verstärkungsfaktor dargestellt. Wegen der sehr hohen Verstärkungsfaktoren (>20000) der Darlington-Transistoren wurde die Darstellung auf B= anstelle hFE= geändert (spart Platz bei der Darstellung).

11. Die Unterschiede in den Referenzspannungen der Atmega8, Atmega88, Atmega168 und Atmega328 werden jetzt nach Datenblatt berücksichtigt. Es besteht aber nach wie vor die Möglichkeit, per Schalter in der Makefile „WITH_AUTO_REF“ das Programm so zu kompilieren, das anstelle des Modell abhängigen Faktors für die Umrechnung der Kondensator Ladezeit in Kapazität, die aktuell im Programm gelesene Referenzspannung zu benutzen (Tabelle wurde für Atmega88 Linie verlängert). Aber wenn die Kurven im Handbuch nicht nur für ein Messexemplar gelten, dürfte die unter Punkt 4 beschriebene Korrektur unter normalen Betriebsbedingungen gar nicht nötig sein.

12. Die Mess-Routine für große Kapazitätsmessungen wurde so erweitert, das bis etwa 100mF gemessen werden könnte. Mangels Testobjekt konnte ich das noch nicht testen. Goldcaps haben sich für Tests als ungeeignet erwiesen.

13. Die Makefile wurde für den AVR-Betrieb mit 8MHz Quarz ergänzt. Die entspechenden Fuses werden durch einen „make fuses-crystal“ Aufruf eingestellt. Für den 1MHz Betrieb der ATmega88 Linie wird dann der Clock-Vorteiler programmiert (/8). Der ATmega8 müsste für den 1MHz Betrieb auch mit einem 1MHz Quarz bestückt werden, wenn man Quarz-Betrieb möchte.

14. Ab Version 0.92k kann das Programm in der Makefile angepasst werden. Die Auswahl betrifft Sprache, Kapazitätsmessung, Widerstandsmessung, UART-Ausgabe, Ausgabe der Batteriespannung zu Beginn der Messung, Selbsttestfunktion, Benutzung des EEPROM sowie der Taktfrequenz (1MHz, 8MHz), des Zielprozessors (atmega8, atmega48, atmega88, atmega168 und atmega328) und des Programmers (avrisp2), sofern man avrdude als Programmer-Interface benutzt (Aufruf make upload). Das Programm avrdude prüft vor einer Programmierung die Kennung des angeschlossenen Prozessors. Bei falscher Wahl bricht avrdude die Aktion mit einer Fehlermeldung ab. Es kann zwischen einer Version mit internem RC-Taktgenerator (make fuses) und zwischen einer Quarz-Oszillator-Version (make fuses-crystal) gewechselt werden. Voraussetzung ist für die Quarz-Version ein bestückter 8MHz Quarz (Pin 9+10 des AVRs). Beim ATmega8 muss für die 1MHz Variante ein 1MHz Quarz bestückt werden, bei den anderen Prozessoren (ATmega88, ATmega168 ..) wird für die 1MHz Variante ein Clock-Vorteiler benutzt (8MHz : 8). Linux-Nutzer sollten also in 4 Schritten zu einer Testversion kommen:

    - Editieren der Makefile, Anpassung an Zielprozessor und Programmer.
    - „make“ aufrufen.
    - ISP-Stecker verbinden und „make upload“ aufrufen.
    - Gegebenenfalls „make fuses“ oder make „fuses-crystal“ aufrufen und ISP-Stecker entfernen.

15. Ab Version 0.92k sind die ADC-Leseroutinen so ergänzt, das bei der Division durch 9 gerundet wird (+4 vor der Division).

Optimierungen bezüglich der Messgenauigkeit sind nach meiner Meinung noch nicht abgeschlossen. Hier wäre ein wenig mehr Informationsrückfluss nützlich.
Ich hoffe, dass ich keinen wichtigen Punkt vergessen habe, und komme damit zum letzten Punkt, dem ich ein eigenes Kapitel widmen möchte.

Selbsttest-Funktion

Ab meiner Software 0.9k habe ich eine Selbsttestfunktion eingebaut. Die Bedienung dazu ist sehr einfach. Die meisten werden nach meiner Meinung Prüfschnüre mit Krokodil-Klemmen zum Anschluss von Prüflingen benutzen. Um den Tester in die Selbsttestfunktion zu bringen, werden einfach alle drei Prüfklemmen auf ein Stück blanken Draht (z.B. eine Seite eines Widerstandes) geklemmt und dann der Startknopf gedrückt. Schon startet das Selbsttest-Programm. Nach Ablauf des Programms macht der Transistortester mit dem normalen Messprogramm weiter, endet also in der Regel mit „unbekanntes Bauteil“. Die traurige Seite dieser Selbsttestfunktion ist, dass auch bei der 8K Version des AVRmega der Speicher zu Ende geht. Die Länge des Programms beträgt jetzt mit der Selbsttestfunktion schon 8000 Byte in der Atmega8 Version. Wenn ich das Programm (0.9k) für einen Atmega88 kompiliere, liegt es mit 8122 Byte schon hart am Limit . Ausser der Benutzung des Eeproms (USE_EEPROM) sehe ich kaum Luft zum sparen. Ich habe ja schon einige Routinen wie ReadADC und Hilfsfunktionen in Assembler programmiert.
Beim Ablauf der Selbsttest-Schritte wird in der 1. Zeile immer ein T mit fortlaufender Nummer angezeigt, jeder einzelne Test wird 8 Mal wiederholt und dann zum nächsten Schritt oder Ende weitergegangen. Bei den Tests werden nur Messergebnisse dargestellt. Es werden keine Fehleranalysen durchgeführt. Interpretieren muss man die Messergebnisse selbst. Noch ein wichtiger Hinweis für diejenigen, die auf dem Transistortester einen ISP-Stecker verbaut haben. Generell darf bei den Messungen kein Programmiergerät angeschlossen sein!

Doch nun zu den einzelnen **Test**s:

1. Messung der 1.3V Referenzspannung (Bandgap Reference). Dargestellt wird in Zeile 1 der Text Ref= und die gemessene Spannung in mV, in der zweiten Zeile werden die daraus berechneten Faktoren für die Kapazitätsmessungen dargestellt.

2. Vergleich der 680Ω Widerstände.
   Dargestellt wird in Zeile 1 der kryptische Text „L1+2- .3- 2+.“. Das soll Folgendes bedeuten: Das L steht für Low also die 680Ω Widerstände. 1+ steht dafür das der Widerstand von Pin1 nach + (VCC) geschaltet wird. Das nachfolgende 2- steht für die Schaltung des Pin2 Widerstandes nach -. Das Ergebnis der Spannungsmessung steht in der 2.Zeile an der ersten Stelle. Es folgt eine „ .3-“ was heißen soll dass beim Zweiten Messwert der Pin1 Widerstand beibehalten wird, aber anstelle des Pin2 Widerstandes jetzt der Pin3 Widerstand nach Masse geschaltet wird. Die letzte Messung wird entsprechend mit Pin2 Widerstand nach + und dem Pin3 Widerstand nach Masse gemessen. Bitte beachten Sie, dass die Auflösung des ADC nur knapp 5mV beträgt, also nicht allzu kritisch sein.
   Theoretisch müsste unter Berücksichtigung der Pin-Widerstände 5001 / (18+680+680+22) * (18+680) = 2493 für alle Kombinationen erscheinen.

3. Vergleich der 470kΩ Widerstände.
   In Zeile 1 wird dargestellt „H1+2- .3- 2+“. Hier wird also die gleiche Messung wie unter dem zweiten Test beschrieben mit den Hochohmigen Widerständen wiederholt.
   Theoretisch wieder 5001 / (18 + 470000 + 470000 + 22) * (18 + 470000) = 2500

4. Hier wird gar nicht gemessen, es erfolgt nur der Hinweis, das es jetzt Zeit ist die zusammengeschalteten Mess-Klemmen zu trennen (Text: isolate Probe).
   Die Werte in der 2. Zeile sind ohne Bedeutung, es wird nur nicht gelöscht.

5. Es wird geprüft, ob ohne angeschlossenes Bauteil mit dem hochohmigen Widerstand nach – (Masse) geschaltet werden kann. Anzeige in Zeile 1 ist RH- .
   Hier sollte für alle drei Pins eine 0 in Zeile 2 erscheinen.

6. Es wird geprüft, ob ohne angeschlossenes Bauteil mit dem hochohmigen Widerstand nach + geschaltet werden kann. Anzeige in Zeile1 ist RH+ .
   Hier sind für alle drei Pins 5001 der Idealwert. Abweichungen vom Idealwert bei Test 5 und Test 6 deuten auf einen Fehler oder ein Isolationsproblem hin (Lötmittelreste oder ähnlich).

7. Innenwiderstandmessung der Pin-Anschlüsse Richtung Masse.
   Text in der 1. LCD-Zeile ist Ri_Lo = (mV). In der zweiten Zeile erscheinen 3 Spannungen. Die AVR Ausgänge sind natürlich keine idealen Schalter, Die Anschlüsse haben einen Innenwiderstand, der hier versucht wird, mit dem Stromfluß des 680Ω Widerstand gegen + zu bestimmen. Aber allzu große Hoffnung, dass man hiermit die Messgenauigkeit durch kalibrieren weiter steigern kann muss ich dämpfen, es werden nur die 3 Pinne (PC1-3) der ADC-Eingänge (PC0 bis PC2) untersucht. Für einen vollständige Abgleich der individuellen Parameter wäre auch die Messung der 680Ω Ausgänge erforderlich (Pin PB0,PB2 und PB4), was nicht ohne Eingriff möglich ist. Um auf die Ohm-Werte zu kommen müssen die Spannungswerte noch durch ca. 7 geteilt werden.

8. Innenwiderstandsmessung der ADC Pin-Anschlüsse Richtung VCC (+).
   Hier wird der Stromfluss mit den 680Ω gegen Masse erzeugt.
   Test in der 1. LCD-Zeile ist Ri_Hi= (mV). In der zweiten Zeile erscheinen 3 Spannungen, die bis zum Idealwert 5001 fehlen. Also die gleiche Messung wie bei Test 7 zur anderen Seite.

9. 50Hz Rechteck-Generator auf Pin 2, Pin 1 ist GND.
   Dieser Test läuft 8 Mal jeweils 10 Sekunden.
   Dient der Überprüfung der Zeit der Warte-Aufrufe (wait10ms()) mit Oszilloskop oder Frequenzzähler. Diese Überprüfung macht nur im Quarzoszillatorbetrieb Sinn, da das Ergebnis sonst durch die Genauigkeit der internen RC-Oszillators verfälscht wird.
   Eine genaue Taktfrequenz ist bei der Kapazitätsmessung von Kondensatoren wichtig.

Zum Schluss wird noch der Text „Auto Test End“ in Zeile 1 und die Versionsnummer der Software in Zeile 2 ausgegeben und das Programm fährt mit einer normalen Bauelemente-Messung fort.

Anhang A

Messergebnisse Selbsttest

Software Version 0.92k

Mikrocontroller		1. Messwert	2. Messwert	3. Messwert
Mega8 @ 8MHz Kennung 1E 93 07 WITH_AUTO_REF	Test 1 Referenzspannung Sollwert 1298mV	Bandgap Ref **1237**		
		RHfakt. 750	RLfakt. 489	
	Test 2 Vergleich 680Ω Idealwert: 2493	RL1+ RL2- **2488**	RL1+ RL3- **2488**	RL2+ RL3- **2484**
	Test 3 Vergleich 470kΩ Idealwert: 2500	RH1+ RH2- **2493**	RH1+ RH3- **2493**	RH2+ RH3- **2493**
	Probe's trennen			
	Test 5 470kΩ (Isolation) Idealwert: 0	RH1- **0**	RH2- **0**	RH3- **0**
	Test 6 470kΩ (Isolation) Idealwert: 5001	RH1+ **4995**	RH2+ **4995**	RH3+ **4995**
	Test 7 Pinwiderstand Low	TP1- RL1+ **132**	TP2- RL2+ **132**	TP3- RL3+ **137**
	Test 8 Pinwiderstand High (VCC - Spannung)=	TP2+ RL1- **151**	TP2- RL2+ **151**	TP3- TP3+ **151**
Mega88 @ 1MHz Kennung 1E 93 0A	Test 1 Referenzspannung Sollwert 1102mV	Bandgap Ref **1086**mV		
		RHfakt. 567	RLfakt. 621	
	Test 2 Vergleich 680Ω	RL1+ RL2- **2493**	RL1+ RL3- **2493**	RL2+ RL3- **2493**
	Test 3 Vergleich 470kΩ	RH1+ RH2- **2498**	RH1+ RH3- **2502**	RH2+ RH3- **2502**
	Probe's trennen			
	Test 5 470kΩ (Isolation)	RH1- **0**	RH2- **0**	RH3- **0**
	Test 6 470kΩ (Isolation)	RH1+ **5001**	RH2+ **5001**	RH3+ **5001**
	Test 7 Pinwiderstand Low	TP1- RL1+ **132**	TP2- RL2+ **132**	TP3- RL3+ **132**
	Test 8 Pinwiderstand High (VCC - Spannung)=	TP2+ RL1- **156**	TP2- RL2+ **156**	TP3- TP3+ **156**

Anhang B

Messergebnisse Widerstandsmessung

Software Version 0.92k

Widerstand	Messwert Mega8 @ 1MHz, Originalsoftware Kennung 1E 93 07		Messwert Mega8 @ 8MHz, Kennung 1E 93 07		Messwert Mega88 @ 1MHz Kennung 1E 93 0A	
60Ω 0,1%	**61Ω**	1,7%	**62Ω**	3,3%	**60Ω**	0%
120Ω 0,1%	**120Ω**	0%	**122Ω**	1,6%	**120Ω**	0%
240Ω 0,1%	**240Ω**	0%	**240Ω**	0%	**239Ω**	0,4%
340Ω 0,1%	**339Ω**	0,3%	**339Ω**	0,3%	**340Ω**	0%
680Ω 0,1%	**677Ω**	0,4%	**677Ω**	0,4%	**679Ω**	0,1%
1360Ω 0,1%	**1357kΩ**	0,2%	**1352Ω**	0,6%	**1361Ω**	0,1%
3,90kΩ 0,1%	**3865Ω**	0.9%	**3859Ω**	1,1%	**3896Ω**	0,1%
7,80kΩ 0,1%	**7676Ω**	1,6%	**7652Ω**	1,9%	**7776Ω**	0,3%
11,0kΩ 0,1%	**10902Ω**	0,9%	**10,70kΩ**	2,7%	**10.91kΩ**	0,8%
22,0kΩ 0,1%	**21,1kΩ**	1,6%	**21,1kΩ**	4,1%	**21,1kΩ**	4,1%
44,0kΩ 0,1%	**43,1kΩ**	0.9%	**43,0kΩ**	2,3%	**43,0kΩ**	2,3%
50kΩ 0,1%	**49,2kΩ**	4,1%	**49,2kΩ**	1,6%	**49,2kΩ**	1,6%
100kΩ 0,1%	**99,0kΩ**	2,0%	**98,9kΩ**	1,1%	**98,9kΩ**	1,1%
200kΩ 0,1%	**196,8kΩ**	2,6%	**197,7kΩ**	1,2%	**198,6kΩ**	0,7%
270kΩ 0,1%	**266,3kΩ**	1,4%	**267,5kΩ**	0,9%	**268,6kΩ**	0,5%
470kΩ 0,1%	**77,46µF ?**		**467,1kΩ**	0,6%	**469,0kΩ**	0,2%
940kΩ 0,1%	**919,6kΩ**	2,2%	**935,7kΩ**	0,5%	**940,0kΩ**	0%
1,00MΩ 0,1%	**973.8kΩ**	2,6%	**990,8kΩ**	0,9%	**996,2kΩ**	0,4%
2,00MΩ 0,1%	**1922,0kΩ**	3,9%	**1,975MΩ**	1,3%	**1,996MΩ**	0,2%
10MΩ 1%	**8433,8kΩ**	15,7%	**9,574MΩ**	4%	**10,11MΩ**	1%
Zugabe: 50MΩ 1%	**----**		**43,87MΩ**	12%	**52,94MΩ**	6%

Anhang C

Messergebnisse Kapazitätsmessung

Software Version 0.92k

Konden-sator	Messwert Multimeter PeakTech 3315	Messwert Mega8 @1MHz Kenn. 1E 93 07 Originalsoftware		Messwert Mega8 @8MHz Kenn. 1E 93 07 WITH_AUTO_REF		Messwert Mega88 @1MHz Kenn. 1E 93 0A		Messwert Mega88 @8MHz Kenn. 1E 93 0A	
56pF	58pF	----		58pF	0%	(42pF mit Diode)	28%	58pF	0%
110pF	114pF	----		114pF	0%	111pF	3%	117pF	3%
220pF	225pF	0,27nF	20%	226pF	0%	222pF	0%	228pF	0%
1000pF	1034pF	1,18nF	14%	988pF	4%	966pF	7%	981pF	5%
3,3nF	3,47nF	3,90nF	12%	3287pF	5%	3223pF	7%	3230pF	7%
7500pF	7,23nF	8,43nF	16%	7241pF	0%	7079pF	3%	7092pF	2%
10nF	10,45nF	11,93nF	14%	10,13nF	3%	9935pF	5%	9922pF	5%
33nF	33,2nF	37,27nF	12%	31,97F	4%	31.26nF	6%	31,36nF	6%
100nF	97nF	112,36nF	16%	97,02nF	0%	94.98nF	3%	94,93nF	3%
330nF	333nF	385,41nF	14%	333,2nF	0%	325,9nF	2%	325,7nF	2%
1µF	955nF	1,10µF	15%	954nF	0%	934nF	2%	934nF	2%
2,2µF	2,2µF	2,54µF	15%	2198nF	0%	2150nF	2%	2149nF	2%
22µF	21,94µF	25,61µF	16%	21,92µF	0%	21,41µF	3%	21,41µF	3%
47µF	47,5µF	55,29µF	16%	47,30µF	0%	46,70µF	1%	46,9µF	1%
100µF	97,5µF	112,94µF	15%	91,30µF	6%	91,7µF	6%	91,3µF	6%
220µF	229µF	272,69µF	19%	219,2µF	5%	219,2µF	5%	219,2µF	5%
1000µF	1071µF	1371,95µF	28%	1076µF	0%	1063µF	1%	1066µF	1%
2200µF	2,231mF	3308,58µF	48%	2302µF	3%	2288µF	3%	2309µF	3%
4700µF	4,75mF	6103,00µF	28%	5042µF	6%	5042µF	6%	4982µF	5%
14,2mF	14,4mF	5Ω ??		15,13mF	5%	15,13mF	5%	15,31mF	6%

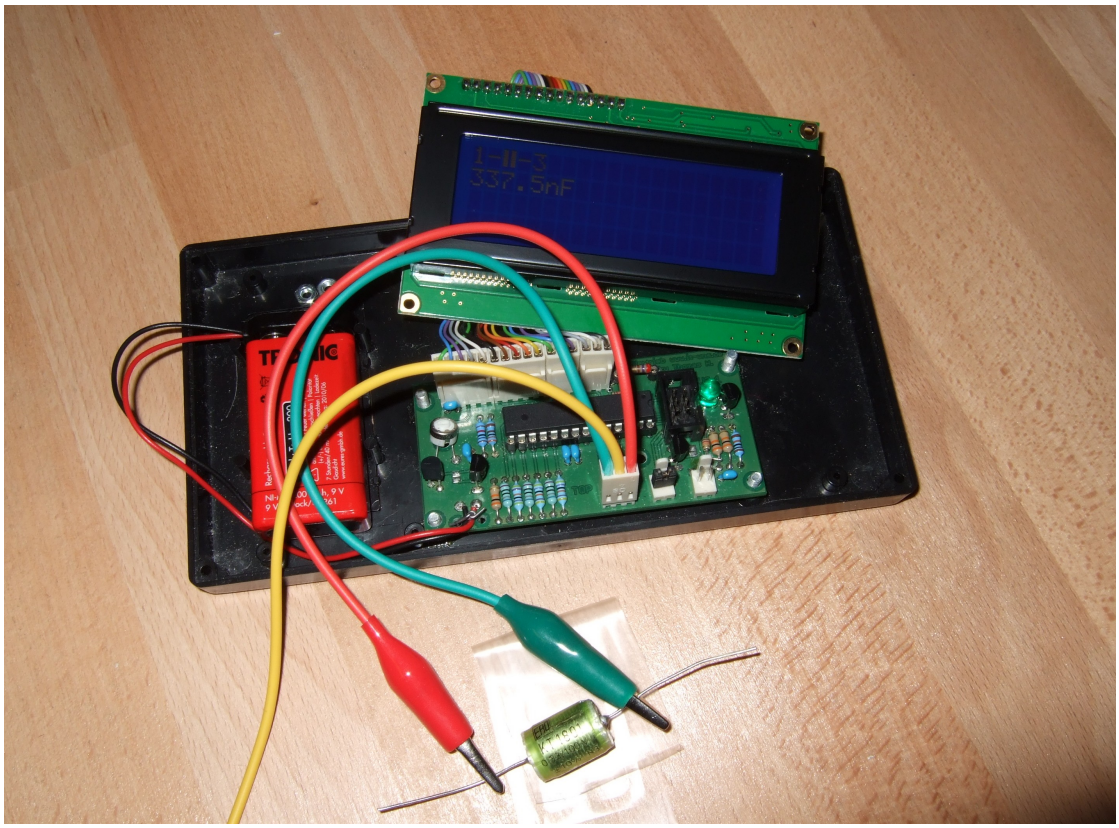
Anhang D

Messergebnisse Halbleiter

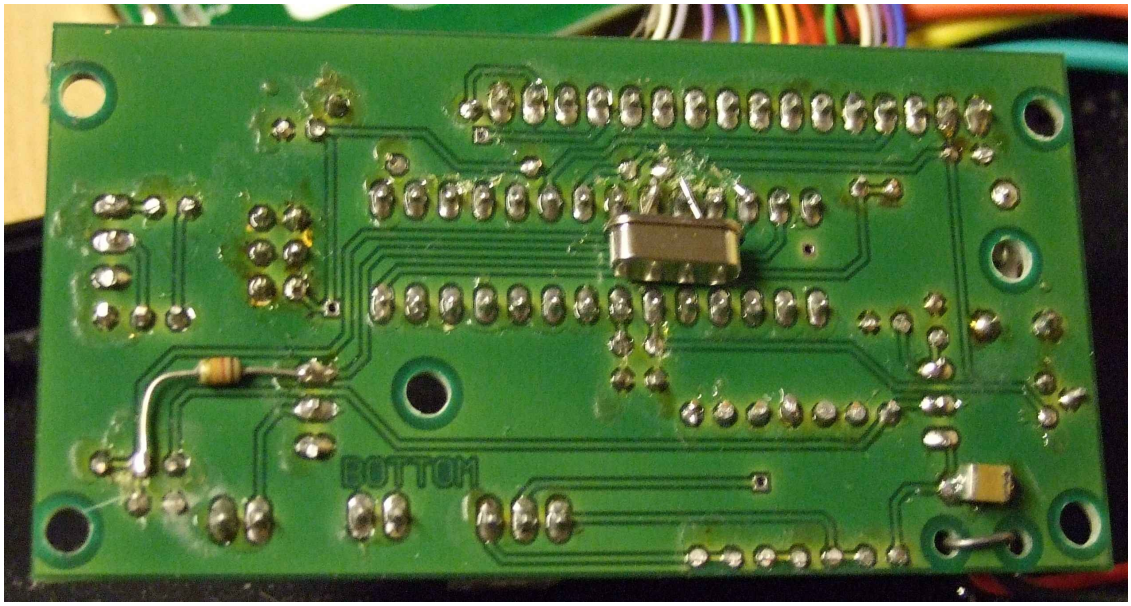
Software Version 0.92k

Halbleiter	Messwert Mega8 Originalsoftware Kennung 1E 93 07	Messwert Mega8 @8MHz Kennung 1E 93 07 WITH_AUTO_REF	Messwert Mega88 @8MHz Kennung 1E 93 0A
1N4148	Diode, 711mV	Diode, 719mV, 1pF	Diode, 734mV, 1pF
1N4150	Diode, 682mV	Diode, 685mV, 1pF	Diode, 691mV, 2pF
BA157	Diode,628mV	Diode, 631mV, 18pF	Diode, 636mV, 18pF
BY398	Diode, 549mV	Diode, 549mV, 0pF	Diode, 553mV, 0pF
1N4007	Diode, 657mV	Diode, 665mV, 10pF	Diode, 674mV, 12pF
LED grün	Diode, 1973mV	Diode, 1966mV, 4pF	Diode, 1975mV, 4pF
ZPD2,7	2Dioden Antiparallel	2xDi, 734mV, 2664mV	2xDi, 744mV, 2679mV
BU508A	NPN, B=5, 618mV	NPN, B=9, 617mV	NPN, B=10, 616mV
BU508A B+E	Diode, 618mV	Diode, 617mV, 5203pF	Diode, 616mV, 5208pF
BU508A B+C	Diode, 598mV	Diode, 602mV, 257pF	Diode, 602mV, 266pF
2N3055	NPN, B=6, 623mV	NPN, B=21, 626mV	NPN, B=22, 617mV
BC546B	NPN, B=388,785mV	NPN, B=372, 792mV	NPN, B=476, 788mV
BC556B	PNP, B=271, 795mV	PNP, B=284, 802mV	PNP, B=428, 803mV
BC639	NPN, B=171, 731mV	NPN, B=179, 734mV	NPN, B=206, 733mV
BC640	PNP, B=190, 726mV	PNP, B=215, 730mV	PNP, B=288, 722mV
AC128 (Ge.)	PNP, B=58, 274mV	PNP, B=59, 281mV	PNP, B=63, 284mV
BC517	NPN, B=792, 1423mV	NPN, B=26637, 1424mV	NPN, B=28227, 1423mV
BC516	PNP, B=795, 1433mV	PNP, B=21410, 1430mV	PNP, B=39664, 1413mV
BS170	N-E-MOS,D,2217mV,0pF	N-E-MOS,D,2628mV, 68pF	N-E-MOS,D,2576mV, 66pF
J310	N-JFET	N-JFET	N-JFET
BRY55/200	Thyristor	Thyristor	Thyristor
IRFU120N	N-E-MOS,D,3314mV,1.07n	N-E-MOS,D,4170mV, 913pF	N-E-MOS,D,4175mV, 909pF
IRFU9024	P-E-MOS, D,3024mV,1.23nF	P-E-MOS, D,1152mV,948pF	P-E-MOS, D,1147mV,942pF
ZVP2106A	P-E-MOS,D,2388mV,0.00nF	P-E-MOS,D,837mV,115pF	P-E-MOS,D,834mV,127pF
ZVNL120A	N-E-MOS,D,1101mV,0.16nF	N-E-MOS,D,1575mV,147pF	N-E-MOS,D,1540mV,151pF

Anhang E
Bilder



Testversion mit 4x20 Display offen



Modifikationen 8MHz Quarz (Pin9+10) und Pull-up Widerstand (Pin12+13)

Anhang F

Bekannte Fehler

Software Version 0.92k

Anhang T

Arbeitsliste (to do)

Software Version 0.92k

- Dokumentation im Programm verbessern, sinnvolle Namen für Funktionen ausdenken.
- Eventuell Verbesserung der Messgenauigkeit kleiner Spannungen durch automatische Wahl der Referenzspannung. (Bis ca. 1V die Bandgap-Referenz verwenden.)
  Wie groß muss die Wartezeit beim Hin und Her-Schalten sein?
  Das bringt insbesondere bei den neueren Prozessoren mega168 oder mega328 im unteren Bereich was, bei mega8 ist die ADC-Referenz nur auf 2.54V schaltbar, was nicht so viel bringt. Bringt vielleicht etwas Zugewinn an Genauigkeit bei der Messung kleiner Widerstandwerte, hilft bei der Erkennung, ob es sich um einen „Dicken“ Kondensator oder um einen Kurzschluß handelt.
- Bei mir weichen die gelesenen Spannungswerte (Test 1) der internen Referenz gegen VCC mehr ab, als es das Datenblatt erwarten lässt, Ursache muss geklärt werden. VCC?, ADC-Abweichung von Ideallinie?
- Test, ob man durch „Rauschquellen“ die Zwischeninterpolation der ADC-Werte durch Mittelwertbildung verbessern kann, wenn zusätzliches Rauschen erzeugt wird.
  Wenn die ADC-Werte dauerhaft stehenbleiben, kann man auch durch die beste Statistik (Mittelwert-Bildung) nicht entscheiden, an welcher Schaltstufe der Spannungwert näher ist. Hier hilft Rauschen, was man ja sonst gerne vermeidet. Geplant ist z.B. der Einsatz der internen Zähler. Natürlich können damit keine Linearitätsfehler des ADC beseitigt werden und die Methode macht an den Bereichsgrenzen des ADC zusätzliche Probleme. Im unteren Bereich hilft vielleicht die Bereichsumschaltung auf interne Referenz, für die obere Grenze habe ich noch keine Idee.
- Verifikation der ADC-Rausch Einflüsse auf die Mittelwertbildung durch Spezial Selbsttest, der es erlaubt das Laden von großen Kondensatoren mit hochohmigen Widerstand zu beobachten. Hier könnte dann auch die ReadADC-Varianten 44/9, 22*2/9 oder 11*4/9 beobachtet werden.
- Methode ausdenken, wie man auch den individuellen Innen-Widerstand der Pinschalter der Widerstands Ports bestimmen kann.
- Möglichkeit untersuchen, die Entladezeit von Kondensatoren in der Schluss-Phase dadurch zu beschleunigen, dass der Minus-Pol über den 680Ω Widerstand nach +angehoben werden kann.
- Wer nutzt die Seriell-Ausgabe? Für mich bedeutet es ziemlichen Aufwand, das zu testen.
- Methode für Induktivitäts-Messung ausdenken.
- Einfluss der Spannungsversorgung auf Messergebnisse untersuchen 4,5 – 5V
- Organisation der Files ändern. Mir wäre es lieber, wenn die Makefile im Verzeichnisbaum nach vorne, vor die Source Files rücken würde. Eventuell automatisch Kopie in einem anderen Verzeichnis (für den Fall, daß man eine Makefile vermurkst hat.
- Mehr Anweisungen aus der main.c in eine neue transistortester.h oder so ähnlich.
- Prüfen, ob für zukünfige Versionen Gleitkomma (float) eingesetzt werden kann, die Multiplikation und Division in Integer kann so durch eine Multiplikation ersetzt werden. Platz braucht float auch nicht mehr als long int, aber ich habe einen riesigen Darstellungsbereich. Da dies mein erstes AVR-Projekt ist, wo ich C-Code verwende, habe ich noch keine Ahnung, wieviel Speicher die Bibliothek frisst.

- Bedienungsanleitung für den Übersetzungsvorgang, damit möglichst jeder Benutzer seine Parameter nach Bedarf einstellen kann. Anwendungsbeispiel: entweder ein Tester mit allen Erkennungsfunktionen auf einem Atmega8 oder einer mit der Grundfunktion und dem Selbsttest.
- Entwickeln einer neuen Platine für Quarzbetrieb (oder Quarz-Oszillator)??
- Test, ob bei Verdacht auf Thyristor oder Triac der Haltestrom für sehr kurze Zeit im Kurzschlussbetrieb, Kathode nur mit Pinwiderstand (18Ω) nach -, Anode nur mit Pinwiderstand (22Ω) nach + dem AVR zugemutet werden kann. Was passiert dann mit der Betriebsspannung?
  Port-Parameter in jedem Fall mit Selbsttest prüfen.
- Erkennung der Anschluss-Belegung von Spannungsreglern?
- Erkennung und Funktionstest von Optokopplern
- Generell besteht die Möglichkeit, den Transistortester durch zeitweiliges Umprogrammieren auch als Funktions oder Clockgenerator zu benutzen, z.B. um einen zerschossenen (falsche Fusebits) AVR mit dem Transistortester als externe Takt-Quelle zum Leben zu erwecken. (Dazu sollte beim Platinen-Entwurf mit Quarz-Clock der ISP-Stecker am Rande sitzen, damit man ihn durch eine Öffnung im Gehäuse erreichen kann.)
- Abbruch der Selbsttestfunktion durch Taste

Nachtrag:

Wie man sieht, ist die Liste der Ideen und Aufgaben ziemlich lang, mein Ziel ist es, den Transistor-Tester (Bauelemente-Tester) genauer, schneller, zuverlässiger und universeller werden zu lassen. Vielleicht lege ich mir mal ein zweites Board zu, damit ich leichter die verschiedenen Versionen mega8 und mega168 testen kann. Ich kann natürlich nicht versprechen, das ich die Liste jemals vollständig abarbeiten kann und ich kann auch keine Garantie auf die Funktionsfähigkeit geben. Aber ich werde nie eine Version ohne Quellcode veröffentlichen!

Inzwischen entwickle ich unter Linux (Ubuntu) weiter, da es mir dank eines nützlichen Hinweises aus einem anderen Thread auf www.mikrocontroller.net gelungen ist, den Diamex ALL-AVR Programmer unter Linux zu betreiben. Eigentümlicherweise kann ich unter Linux auch das EEprom der ATmega8 Bausteine mit dem gleichen Programmer fehlerfrei beschreiben. Damit bin ich jetzt auch in der Lage die Originalsoftware von Markus F. aufzuspielen. Ich habe nach wie vor die Hoffnung nicht aufgegeben, dass meine Verbesserungen in die offizielle Version von Markus F. einfließen, obwohl ich auf meine Nachricht vom 9.2. an 5volt und am 14.2. an linuxgeek bisher keine Reaktion erhalten habe.