MULTITESTER ELEKTRONIK BERBASIS MIKROKONTROLER ATMEGA 8

TUGAS AKHIR

Untuk memperoleh gelar Ahli Madya pada
Program Diploma III Teknik Elektro
Jurusan Teknik Elektro – Fakultas Teknik
Universitas Negeri Semarang

Oleh :

Nama	: Nanda Puji Arianto
NIM	: 5311311009
Program Studi	: Diploma III Teknik Elektro
Jurusan	: Teknik Elektro

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SEMARANG

2015

HALAMAN PENGESAHAN

Tugas Akhir ini telah dipertahankan dan disyahkan dihadapan sidang Panitia Ujian Tugas Akhir Fakultas Teknik Universitas Negeri Semarang pada:

Hari : Senin

Tanggal : 10 Agustus 2015

Panitia Ujian Tugas Akhir :

Ketua

Tatyantoro Andrasto, S.T, M.T
NIP. 196803161999031001

Sekretaris

Riana Defi Mahadji Putri S.T, M.T
NIP. 197609182005012001

Penguji 1

Drs. Agus Purwanto
NIP. 195909241986031003

Penguji 2/ Pembimbing Utama

Drs. Rafael Sri Wiyardi, M.T
NIP. 195011101979031001

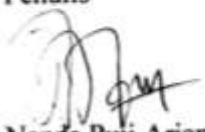
Mengetahui,
Atas Nama Pembantu Dekan 1

Drs. Djoko Adi Widodo, M. T.
NIP. 195909271986011001

PERNYATAAN

Saya menyatakan dengan sebenar-benarnya bahwa tugas akhir saya yang berjudul "*Multitester Elektronik Berbasis Mikrokontroler ATMega 8*" disusun berdasarkan dengan arahan dosen pembimbing. Argumen dan temuan orang lain yang terdapat di dalamnya dikutip dan dirujuk berdasarkan kode etik penulisan yang lazim dan ilmiah.

Penulis



Nanda Puji Arianto.

MOTTO DAN PERSEMBAHAN

Motto:

1. Tidak ada yang sia-sia jika kita bisa berpikir positif
2. Manusia pasti pernah jatuh, akan tetapi kita bisa memilih untuk bangkit atau tetap terpuruk.
3. Tidak menyerah selama harapan yang kita ingin capai belum 0 %
4. Kepuasan terletak pada usaha, bukan hasil. Berusahalah dengan keras adalah kemenangan yang hakiki (Mahatma Gandi)
5. Harga sebuah kegagalan dan kesuksesan bukan dinilai dari hasil akhir, tetapi dari proses perjuangannya (Andrie Wongso)

Persembahan:

Tugas akhir ini penulis persembahkan untuk:

1. Bapak dan ibu tercinta dan seluruh keluargaku
2. Teman-teman seperjuangan D3 T. Elektro UNNES Angkatan 2011.

KATA PENGANTAR

Alhamdulillah, puji syukur senantiasa penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufiq dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik dan lancar.

Ucapan terima kasih juga saya haturkan kepada Yth:

1. Bapak Drs. Rafael Sri Wiyardi, M.T., sebagai pembimbing yang telah memberikan bimbingan dan motivasi hingga terselesaikannya tugas akhir ini.
2. Ibu Riana Defi Mahadji Putri, S.T., M.T., selaku Ketua Program Studi D3 Teknik Elektro.
3. Bapak Drs. Suryono, M.T., selaku selaku Ketua Jurusan Teknik Elektro.
4. Drs. M. Harlanu, M.Pd, selaku Dekan Fakultas Teknik Universitas Negeri Semarang.
5. Bapak dan Ibu dosen Jurusan Teknik Elektro Universitas Negeri Semarang yang telah memberikan bekal ilmu.
6. Seluruh sahabat seperjuangan di jurusan Teknik Elektro.
7. Semua pihak yang telah memberikan bantuan moril maupun materil dalam penyusunan tugas akhir ini.

Penulis berharap semoga Tugas Akhir ini bermanfaat bagi para pembaca.

Terimakasih.

Semarang, Juni 2015.

Penulis.

Nanda Puji Arianto.

ABSTRAK

Arianto, Nanda Puji. 2014. ***Multitester Elektronik Berbasis Mikrokontroler ATMega 8***. Tugas Akhir, D3 Teknik Elektro Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang.
Pembimbing : Drs. Rafael Sri Wiyardi, M.T.

Kata kunci : *Multitester, Ohmmeter, Voltmeter, Amperemeter,Mikrokontroler.*

Multitester adalah suatu alat pengukur listrik yang sering dikenal sebagai VOM (*Volt-Ohm meter*) yang dapat mengukur tegangan (*voltmeter*), hambatan (*ohm-meter*), maupun arus (*amperemeter*). Ada dua kategori multimeter: *multimeter digital* atau DMM (*digital multi-meter*) dan *multimeter analog*. Sementara Mikrokontroler merupakan sebuah *system microprosesor* di mana di dalamnya sudah terdapat CPU, ROM, RAM, I/O, Clock dan peralatan internal lainnya yang sudah terhubung dan terorganisasi (teralamati) dengan baik oleh pembuatnya dan dikemas dalam satu *chip* yang siap pakai. ATMega 8 merupakan *chip* Mikrokontroler dalam bentuk IC (*integrated circuit*).

Tugas akhir ini merupakan pemanfaatan dari pemrograman *Mikrokontroler* ATMega 8 yang dikompilasi menggunakan bahasa C dan kemudian diprogramkan ke dalam sistem minimum *Mikrokontroler*. Rangkaian sistem minimum ini menggunakan 2 buah chip ATMega 8, masing-masing mempunyai fungsi sendiri yakni sebagai *ohm meter*/*component tester* dan *voltmeter/amperemeter*.

Pembuatan alat adalah dengan observasi mengenai program *chip Mikrokontroler* ATMega 8 yang bersumber dari media internet dan dipadukan menjadi sebuah fungsi multitester yang mencakup banyak fungsi yang dapat mengukur sebuah komponen elektronik Resistor, Kapasitor, Dioda, Bipolar Transistor, FET, dan MOSFET semuanya mencakup fungsi *ohmmeter* kemudian fungsi pengukur yang lan antara lain *DC Voltmeter* dan *DC Amperemeter* dari semua fungsi tersebut penulis menyebut "Multitester Elektronik Berbasis Mikrokontroler ATMega 8".

Hasil pengujian yakni berupa pengukuran terhadap besaran listrik dan dapat menunjukkan fungsinya dengan baik terlebih pada komponen elektronik alat ini dengan sendirinya dapat mengidentifikasi jenis komponen pada probe tester antara lain resistor, kapasitor, keluarga transistor (Bipolar, FET, MOSFET), dan juga dioda. Dan fungsi voltmeter dan amperemeter juga dapat mengukur dan menunjukkan hasilnya dengan baik yaitu dengan menggunakan sumber daya dari Batterai 9 VDC.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar : Halaman

DAFTAR TABEL

Tabel : Halaman

DAFTAR LAMPIRAN

Lampiran : Halaman

BAB I

PENDAHULUAN

A. Latar Belakang

Perkembangan zaman telah berkembang dengan pesat, melalui ilmu pengetahuan dan teknologi manusia kini semakin dimudahkan dengan segala aktifitas kehidupannya. Teknologi memang hal yang tidak bisa dipisahkan pada kehidupan sekarang ini, peralatan elektronik seperti: laptop, handphone, komputer, tv, radio, dan lain-lain telah menjadi bagian hidup manusia sekarang ini.

Salah satu ilmu yang mempelajari teknologi tersebut adalah ilmu teknik elektro yang mendasari semua peralatan elektronik diatas dapat bekerja atas beberapa blok rangkaian elektronika, ilmu teknik elektro mempelajari sifat dan juga gejala listrik. Salah satunya terdapat satuan/besaran listrik misalnya satuan arus listrik yaitu ampere, hambatan satuannya yaitu ohm, dan tegangan/beda potensial yaitu volt. Selain itu dalam aplikasinya terdapat bermacam-macam komponen dasar elektronika seperti: resistor, kondensator/kapasitor, dioda, dan beragam jenis transistor.

Dari satuan dan juga komponen listrik diatas makan diperlukan suatu instrumentasi pengukuran dan pengujian (*metering and testing*) yang mampu menampilkan harga yang sesungguhnya dari hasil pengukuran yang didapat dan juga mengetahui kondisi dan fungsi dari komponen tersebut apakah layak dinilai

baik atau buruknya sehingga dapat berfungsi dengan semestinya pada rangkaian elektronika.

Walaupun saat ini sudah terdapat instrumen pengukuran standart seperti: *multitester* atau *AVOmeter* atau multimeter akan tetapi alat ukur ini memiliki kekurangan:

1. Tidak dapat mengidentifikasi jenis komponen yang diuji/diukur, misalnya apakah itu komponen resistor, kondensator/kapasitor, dioda, bahkan transistor.
2. Tidak dapat membaca nilai dari satuan komponen yang diuji/diukur terkecuali resistor, misalnya satuan Farad untuk kapasitor/kondensator dan satuan penguatan Hfe pada transistor.
3. Pada pengujian/pengukuran transistor untuk menentukan kaki-kaki terminalnya membutuhkan waktu yang lama dan banyak kesulitan, misalnya menentukan kaki Basis, Collector, Emitor pada Transistor
4. Hasil pengukuran antara *multitester* satu dengan lainnya berbeda-beda.

Atas dasar itu maka penulis mencoba membuat alat ukur yang mampu mengukur satuan listrik ohm meter, ampere meter dan voltmeter dan dapat menguji/mengidentifikasi komponen elektronika sehingga dapat menutupi kekurangan dari multitester yang telah dijelaskan sebelumnya pada poin 1-4. Alat ukur/uji ini menggunakan *Mikrokontroler chip* AVR ATMega8 dengan struktur pemrograman yang diterapkan sehingga menghasilkan karya tugas akhir berupa instrumen pengukuran/pengujian dengan judul "**MULTITESTER ELEKTRONIK BERBASIS MIKROKONTROLER ATMEGA 8**"

B. Permasalahan

Berdasarkan latar belakang diatas maka muncul permasalahan, yaitu:

1. Bagaimana membuat alat ukur yang dapat mengukur Ohm meter yang mewakili pengukuran komponen-komponen elektronik dan menunjukkan hasilnya secara cepat dan dapat melakukan pengukuran secara mudah ?
2. Bagaimana membuat alat ukur yang dapat mengukur DC Voltmeteter dan Amperemeter menunjukkan hasilnya secara cepat dan dapat dilakukan dengan mudah?
3. Bagaimana alat ukur tersebut dapat dibangun diatas chip ATMega 8 dan dapat diaplikasikan pada laboratorium Teknik Elektro UNNES.

C. Batasan Masalah

Dikarenakan luasnya permasalahan di dalam pembahasan dan agar tidak terjadi kesalah pahaman maksud dari apa yang ada di dalam penulisan tugas akhir ini maka dibutuhkannya pembatasan masalah tersebut antara lain :

1. Hardware utama dalam pembuatan alat ini menggunakan chip Mikrokontroler IC ATMega 8
2. Jangkauan pengukuran pada komponen pasif yakni resistor dan kapasitor/kondensator
3. Jangkauan pengukuran pada komponen aktif yakni dioda dan transistor
4. Batas ukur pada pengukuran resistansi (Ohm meter) yakni: 20 Ohm – 20 M Ohm

5. Batas ukur pada pengukuran tegangan (Volt meter) yakni: 0 Volt DC – 20 Volt DC
6. Batas ukur pada pengukuran kuat arus (Ampere meter) yakni: 0 Ampere DC – 12 Ampere DC

D. Tujuan

Pembuatan tugas akhir ini diharapkan diperoleh hasil untuk membuat alat ukur yang dapat digunakan untuk mengukur dan menguji komponen elektronika seperti: resistor, kapasitor/kondensator, dioda maupun transistor, dan dapat juga sebagai ohm meter, dc voltmeter, dan dc amperemeter dengan berbasiskan Mikrokontroler

E. Manfaat

Manfaat dari tugas akhir ini adalah:

1. Dapat membuat alat ukur berbasis Mikrokontroler
2. Digunakan sebagai instrumen pengukuran di laboratorium teknik elektro
3. Menerapkan teknik pemrograman dengan bahasa C pada mata kuliah Mikrokontroler

F. Sistematika Tugas Akhir

Untuk memperjelas garis besar dalam penyusunan tugas akhir ini maka dicantumkan sistematikanya. Adapun susunan sistematikanya terdiri dari bagian awal, isi dan akhir.

1. Bagian awal berisi :

   Halaman judul, abstrak, halaman pengesahan, motto dan persembahan, kata pengantar, daftar isi, daftar tabel, daftar gambar dan daftar lampiran.

2. Bagian isi terdiri dari 3 bab yaitu :

   BAB I: PENDAHULUAN

   Bab ini menjelaskan tentang latar belakang, rumusan masalah, pembatasan masalah, tujuan, manfaat, dan sistematika penulisan tugas akhir.

   BAB II : LANDASAN TEORI

   Bab ini berisi teori-teori relevan yang melandasi pelaksanaan pembuatan alat dalam tugas akhir ini.

   BAB III : METODE PEMBUATAN

   Bab ini berisi tentang pembuatan, perancangan, analisa, dan pengujian dari alat yang dibuat

   BAB IV : PENUTUP

   Bab ini berisi uraian kesimpulan dan saran

3. Bagian akhir berisi :

   a) Daftar Pustaka

   b) Lampiran-lampiran

BAB II

LANDASAN TEORI

A. Multitester

Multitester merupakan instrumen alat ukur yang berfungsi mengukur bermacam-macam besaran listrik seperti: *Ohm meter* (Ω), *Voltmeter* (V), dan *Amperemeter* (A). Pada *Ohmmeter* berfungsi mengukur resistansi atau hambatan listrik, *Voltmeter* berfungsi mengukur tegangan atau beda potensial listrik, sedangkan *Amperemeter* berfungsi mengukur kuat arus listrik. Semua fungsi itu seluruhnya mencakup fungsi Multitester.

Multitester dapat disebut juga Multimeter maupun AVOmeter, karena mempunyai cakupan fungsi yang luas, multitester sering digunakan di dalam laboratorium elektronika. Terdapat 2 jenis multitester yakni multitester *analog* dan *digital*, secara umum keduanya memiliki fungsi yang sama.

Perbedaan dari keduanya adalah :

1. Multitester *analog* memiliki ketelitian pengukuran yang relatif kecil dibanding multitester *digital* yang memiliki ketelitian yang relatif tinggi.
2. Multitester *analog* memiliki batas ukur (BU) atau disebut range yang harus diposisikan diatas batas nilai pengukuran sedangkan multitester *digital* tidak memiliki batas ukur atau disebut juga autorange.

3. Multitester *analog* memiliki tampilan hasil dengan skala simpangan jarum dengan *moving coil* sebagai penggeraknya sedangkan multitester *digital* menggunakan layar kristal cair atau LCD sebagai media penampilnya.

Sedangkan kesamaan dari multitester analog dan *digital* adalah:

1. Memiliki fungsi yang sama dalam pengukuran hambatan/tahanan listrik (Ohm meter)
2. Memiliki fungsi yang sama dalam pengukuran tegangan (Volt meter)
3. Memiliki fungsi yang sama dalam pengukuran kuat arus listrik (Ampere meter)
4. Memiliki sumber tenaga yang sama yakni dari baterai
5. Memilki *probe testing* sebanyak dua buah yakni (+) dari warna merah dan (-) dari warna hitam.

Penampakan dari *multitester analog* ditunjukkan pada Gambar 1. Kemudian untuk *multitester digital* ditunjukkan pada Gambar 2.

Gambar 1. *Multitester Analog* SUNWA X360TRF
(*http://www.sanwa-meter.co.jp/prg_data/goods/img/PH21381898451.JPG*)

Gambar 2. *Multitester Digital* SUNWA CD800a
(*http://www.sanwa-meter.co.jp/prg_data/goods/img/PH21380674410.jpg*)

B. Mikrokontroler AVR ATMega 8

1. Arsitektur

Mikrokontroler adalah sebuah sistem *microprosessor* di mana didalamnya sudah terdapat CPU, RAM, ROM, I/O, Clock dan Peralatan Internal lainnya yang sudah saling terhubung dan terorganisasi (teralamati) dengan baik oleh pabrik pembuatnya dan dikemas dalam satu *chip* yang siap pakai.

Mikrokontroler ATMega 8 adalah Mikrokontroler AVR (*Alf and Vegard RISC*) 8-*bit* dengan 8 Kbytes FLASH, dan 512 Bytes EEPROM yang menggunakan teknologi RISC (*Reduce Instruction Set Computer*). Mikrokontroler ini dapat mengeksekusi perintah dalam satu periode clock untuk setiap instruksi. Mikrokontroler ini diproduksi oleh atmel (http://atmel.com). Penampakan dari Mikrokontroler ATMega 8 yakni seperti di Gambar 3. dengan 28-*pin* DIP.

Gambar 3. Mikrokontroler AVR ATMega 8 DIP 28*pin*

(*http://upload.wikimedia.org/wikipedia/commons/a/a9/ATmega8_01_Pengo.jpg*)

Beberapa fitur dari ATMega 8 adalah sebagai berikut:

8-*bit* AVR berbasis RISC dengan performa tinggi dan konsumsi daya rendah

Kecepatan maksimal 16 MHz

Memori:

a. 8 Kbyte Flash Program
b. 512 byte EEPROM
c. 1 Kbyte SRAM

2 *timer* 8-*bit* dan 1 timer 16-*bit*

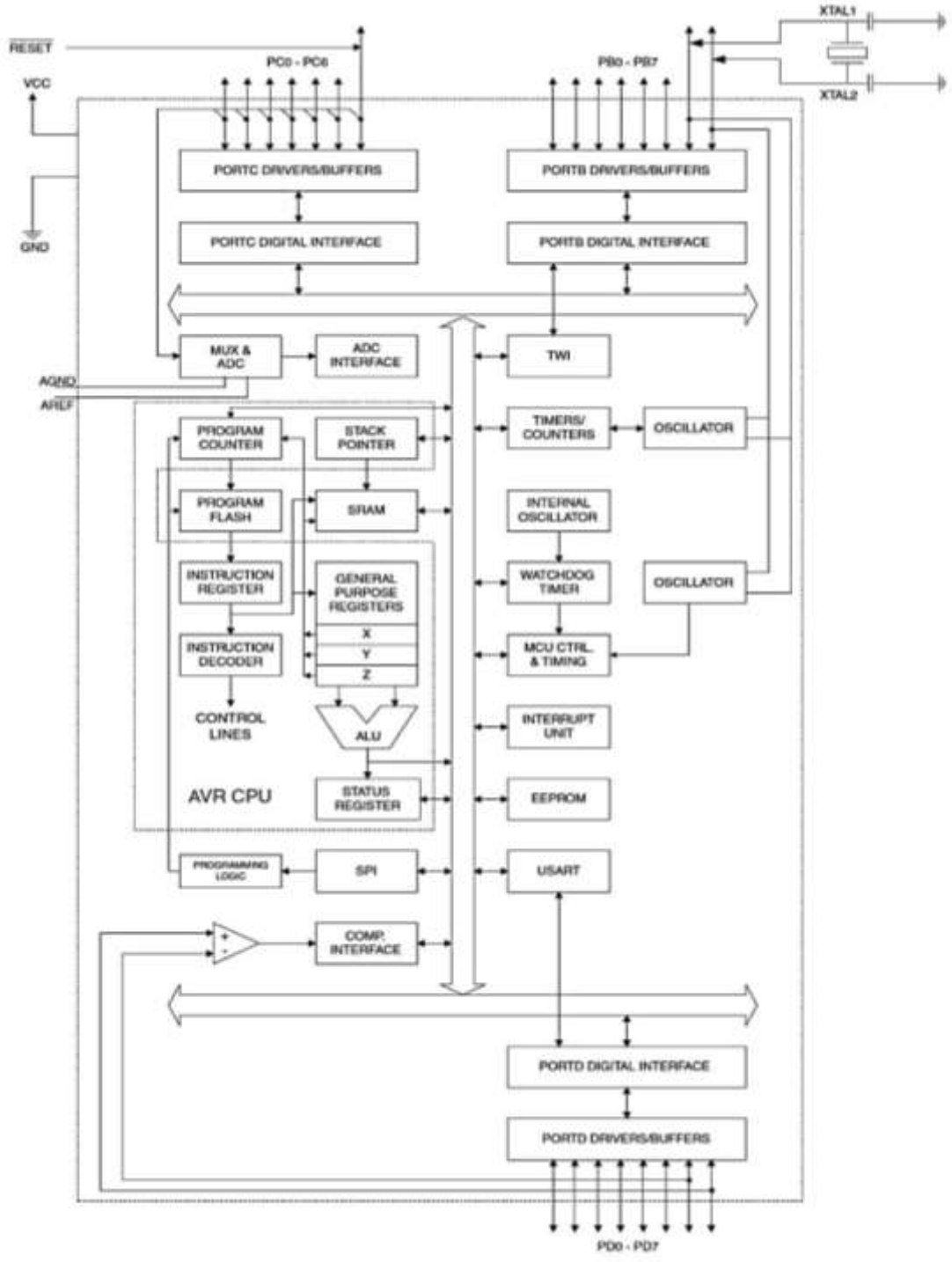
6 kanal 8/10-*bit* ADC (*Analog to Digital Converter*)

Programmable serial USART

Analog comparator

23 jalur I/O yang bisa diprogram.

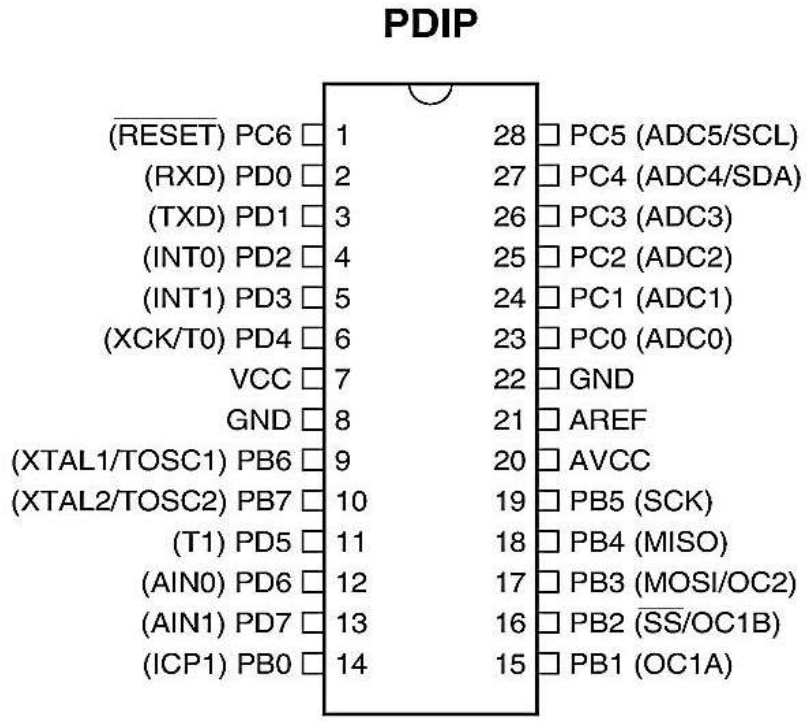
Mikrokontroler ATMega 8 ini dapat bekerja pada kecepatan (*clock*) maksimal sebesar 16 MHz akan tetapi pengunaan *clock* dalam pembuatan alat tugas akhir ini hanya sebesar 8 MHz pada *internal clock* dapat diatur dengan *fusebit* pada proses *burning* programnya. Selanjutnya, pada Gambar 4. dibawah ini ditampilkan sebuah *block diagram* Mikrokontroler ATMega 8 yang diperoleh dari *datasheet* Atmel AVR ATMega 8 , selanjutnya diperinci dimana sebuah sistem saling terhubung dan membentuk sebuah fungsi yang dipakai dalam pembuatan tugas akhir ini antara lain: ADC *Interface*, AVR CPU, Port *Drivers/Buffers* dan *Digital Interface.*



Gambar 4. *Block Diagram* Mikrokontroler ATMega 8 (*http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf*)

2. Konfigurasi Pin ATMega 8

Mikrokontroler ini dikemas dalam sebuah komponen elektronik berupa IC (*integrated circuit*) yang memiliki penghubung kedalam *chip* didalamnya berupa kaki komponen dengan total berjumlah 28 pin dengan kemasan (*package*) berupa DIP *package.* Gambar 5. Ini memperlihatkan konfigurasi/penataan tempat masing-masing dari pin tersebut diletakkan. DIP *package* banyak di pasaran dan sangat mudah dalam pemasangannya ke PCB.



Gambar 5. Pin Out ATMega 8 (DIP Package)

(*http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf*)

Keterangan masing-masing PIN dari DIP package pada Gambar 5. dijelaskan sebagai berikut:

VCC : Tegangan Supply

GND : Ground

Port B (PB7..PB0) : Port I/O 8-bit dengan resistor *pull-up* internal tiap pin. *Buffer* portB mempunyai kapasitas menyerap (*sink*) dan mencatu (*sorce*). Khusus **PB6** dapat digunakan sebagai *input* kristal (*inverting oscillator amplifier*) dan *input* ke rangkaian *clock* internal, bergantung pada pengaturan *Fuse bit* (ada dalam *software Programmer/downloader*) yang digunakan untuk memilih sumber clock. Khusus **PB7** dapat digunakan *output* kristal (*output inverting oscillator amplifier*) bergantung pada pengaturan *Fuse bit* yang digunakan untuk memilih sumber *clock*. Jika sumber clock yang dipilih dari *oscillator internal*, PB7 dan PB6 dapat digunakan sebagai I/O atau jika menggunakan *Asyncronous Timer/Counter2* maka PB6 dan PB7 (TOSC2 dan TOSC1) digunakan untuk saluran *input counter*.

Port C (PC5..PC0) : Port I/O 7-bit ([PC6],PC5…PC0) dengan *resistor pull-up internal* tiap pin. *Buffer* portC mempunyai kapasitas menyerap (Sink) dan mencatu (sorce).

RESET/PC6 : Jika *fuse* bit RSTDISBL di "programed", PC6 digunakan sebagai pin I/O. Jika *fuse* bit RSTDISBL di "unprogramed", PC6 digunakan sebagai pin RESET (aktif low).

Port D (PD7..PD0) : Port I/O 8-bit dengan resistor *pull-up internal* tiap pin. *Buffer* portC mempunyai kapasistas menyerap (Sink) dan mencatu (sorce).

Avcc : AVcc adalah pin tegangan catu untuk A/D *converter*, PC3..PC0, dan ADC(7..6). AVcc Harus dihibungkan ke Vcc, walaupun ADC tidak digunakan. Jika ADC digunakan, maka AVcc harus dihubungkan ke VCC melalui "*low pass filter*". Catatan: PC5, PC4, gunakan catu tegangan Vcc *digital.*

AREF : untuk pin tegangan referensi *analog* untuk ADC

ADC7..6(TQPF, QFN/MLF) : Hanya ada pada kemasan TQPF dan QFN/MLF, ADC 7..6 digunakan untuk pin input ADC.

3. ADC (*Analog To Digital Converter*) ATMega 8

Sesuai dengan namanya ADC (*Analog to Digital*) merupakan fungsi pengubah sinyal *analog* menjadi *digital*, untuk dapat memproses intruksi yang diberikan kepada Mikrokontroler besaran data yang diperoleh harus terlebih dahulu dikonversi ke *digital*.

Mikrokokontroller ATMega 8 telah mempunyai fasilitas ADC internal yang telah termasuk di dalam *chip* sehingga disini tinggal memprogramnya saja. Fitur dari ADC ATMega 8 disebutkan sebagai berikut:

Resolusi mencapai 10 *bit*

8 saluran ADC yang dapat digunakan secara bergantian

Waktu konversi 65 – 260 µs

0-Vcc *range input* ADC

3 Mode pemilihan tegangan referensi

(Vref, Vcc, Vref internal = 2,56V)

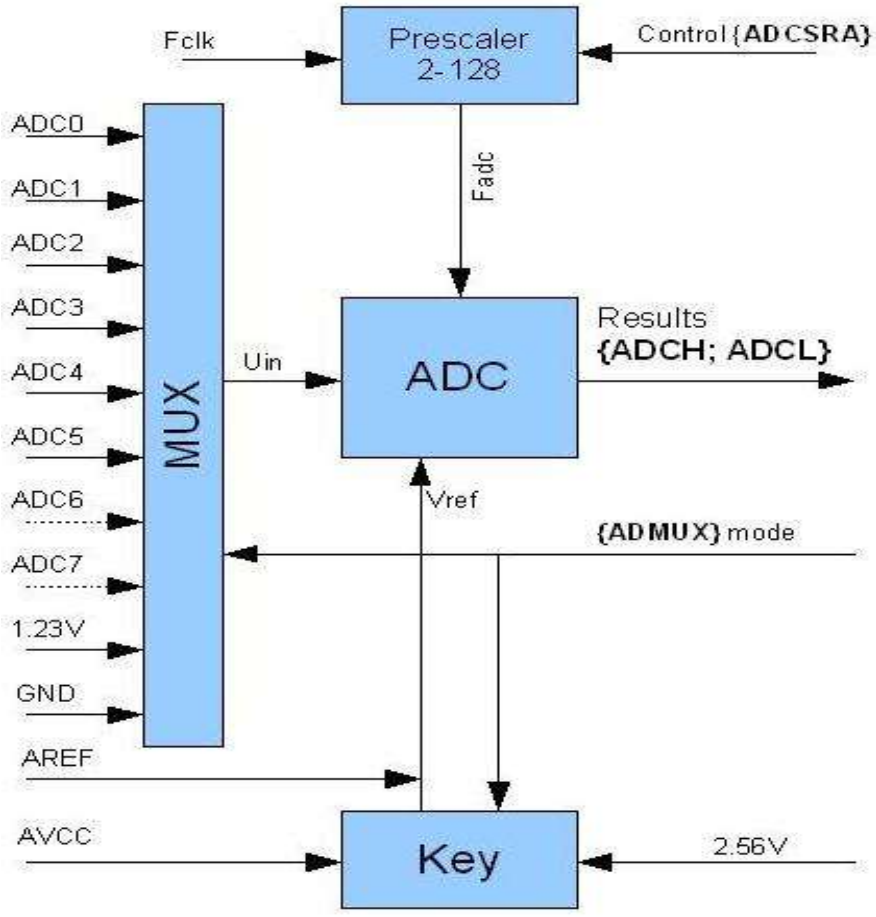
Mode konversi kontinyu (*free running*) atau mode konversi tunggal (*single conversion*)

Untuk memahami koneksi ADC didalan sebuah chip ATMega 8 berikut ini pada Gambar 6. dijelaskan mengenai gambar koneksi ADC yang terhubung dengan komponen penyusun ADC diantaranya: MUX (*multiplexer*), prescaler, Vref control (KEY).

Sinyal *input* dari *pin* adc akan dipilih oleh *multiplexer* (*register* ADMUX) untuk diproses oleh ADC. Karena *converter* ADC dalam *chip* hanya satu buah sedangkan sedangkan saluran *input*-nya ada delapan maka dibutuhkan *multiplexer* untuk memilih *input* pin adc secara bergantian. ADC mempunyai rangkaian untuk mengambil sampel dan *hold* (menahan) tegangan *input* ADC sehingga dalam keadaan konstan selama proses konversi. ADC mempunyai catu daya yang terpisah yaitu *pin* AVCC-AGND. AVCC tidak boleh berbeda ±0.3V dari Vcc.

Operasi ADC membutuhkan tegangan referensi vref dan clock Fadc (register ADCSRA). Tegangan referensi eksternal pada pin Aref tidak boleh melebihi AVCC. Tegangan referensi eksternal dapat di-decouple pada pin Aref dengan kapasitor untuk mengurangi derau. Atau dapat menggunakan tegangan referensi

internal sebesar 2,56V (pin AREF diberi kapasitor secara eksternal untuk menstabilkan tegangan referensi internal). ADC mengonversi tegangan input analog menjadi bilangan *digital* selebar 10-bit. GND (0 Volt) adalah nilai minimum yang mewakili ADC dan nilai *maximum* ADC diwakili oleh tegangan pada pin AREF minus 1 LSB. Hasil konversi ADC disimpan dalam register pasangan ADCH:ADCL.



Gambar 6. Diagram Blok ADC

Sinyal input ADC tidak boleh melebihi tegangan referensi. Nilai *digital* sinyal input ADC untuk resolusi 10-bit (1024) adalah:

Kode *digital* = (V input / Vref) x 1024

Sedangkan untuk resolusi 8-bit (256):

Kode *digital* = (V input / Vref) x 256

Misalnya input suatu pin ADC dengan resolusi 8-bit adalah 2,5 V dan tegangan referensi yang digunakan Vref internal sebesar 2,56 V sehingga kode digital-nya adalah:

Kode *digital* = (2500 mV / 2560mV) x 256 = 250(dec) = 0xFA(hex)

Akurasi ADC dalam *chip* tidak sempurna, akurasinya ±2LSB sehingga kemungkian kode yang dihasilkan tidak tepat 0xFA bisa jadi 0xF8, 0xF9, 0xFB, atau 0xFC

a) Inisialisasi ADC

Untuk menjalankan fitur pengubah sinyal *analog* ke *digital*, maka diperlukan inisialisasi yang melibatkan proses penentuan *clock*, tegangan referensi, format keluaran data serta mode pembacaan. Register yang terkait dengan penetapan inisialisasi tersebut adalah register ADMUX (ADC *Multiplexer Selection Register*), ADCSRA (ADC *Control and Status Register A*), dan SFIOR (*Special Function IO Register*). Semua register-register tersebut akan diperinci di halam berikutnya.

(1) ADMUX (ADC Multiplexer)

Register ADMUX adalah register 8 bit yang berfungsi menetapkan tegangan referensi ADC, format data keluaran, dan saluran ADC yang akan digunakan. Konfigurasi register ADMUX ditunjukkan pada Tabel 1.

Tabel 1. Register ADMUX

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

(a) REF_{0-1} bit-bit pengatur mode tegangan referensi ADC.

Bit-bit ini memilih tegangan referensi untuk ADC seperti yang ditunjukkan pada Tabel 2. Jika bit-bit ini diubah ketika sedang mengonversi, perubahannya tidak akan berpengaruh sampai konversi tersebut selesai (ADIF set di dalam ADCSRA). Pilihan-pilihan referensi tegangan internal tidak dapat digunakan jika tegangan referensi eksternal sedang terhubung ke pin AREF.

Tabel 2. Pemilihan Tegangan Referensi

$REFS_1$	$REFS_0$	Mode Tegangan Referensi
0	0	Pin Vref
0	1	Vcc
1	0	Tidak digunakan
1	1	Vref internal = 2,56V

(b) ADLAR adalah bit keluaran ADC

Bit ADLAR mempengaruhi hasil konversi ADC di dalam register data ADC (ADCH:ADCL). Tuliskan satu ke ADLAR untuk mengatur hasil ke kiri. Sebaliknya, hasilnya diatur ke kanan . Perubahan bit ADLAR akan mempengaruhi register data ADC dengan segera mengabaikan konversi yang sedang berlangsung.

Jika ADC telah selesai konversi maka data ADC akan diletakkan di 2 register, yaitu ADCH dan ADCL dengan format data sesuai ADLAR. Format data ini ditunjukkan pada Tabel 3 dan Tabel 4.

Tabel 3. Format Data ADC dengan ADLAR=0

-	-	-	-	-	-	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL

Tabel 4. Format Data ADC dengan ADLAR=1

ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	-	-	-	-	-	-	ADCL

(c) MUX_{0-4} adalah bit-bit pemilih saluran pembacaan ADC

Bit ini memilih kombinasi masukan analog yang dihubungkan ke ADC. Bit-bit ini juga memilih *gain* untuk *differential channel*. Untuk rinciannya dapat dilihat pada Tabel 5. Jika bit-bit ini diubah ketika

sedang mengkonversi, maka tidak akan ada perubahan sampai konversi tersebut selesai (ADIF di-set di dalam ADCSRA).

Tabel 5. Pemilihan Saluran Input dan Gain

MUX_{4-0}	Single Ended Input	Pos Differential Input	Neg Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22V (VBG)	N/A		
11111	0V (GND)			

(2) ADCSRA (ADC Control and Status Register)

ADCSRA adalah register 8 *bit* yang berfungsi untuk melakukan manajemen sinyal kontrol dan status ADC.Susunannya ditunjukkan pada Tabel 6.

Tabel 6. Register ADCSRA

ADEN	ADCS	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA

(a) ADEN adalah bit pengatur aktivasi ADC. Jika bernilai 1 maka ADC aktif

(b) ADCS merupakan bit penanda dimulainya bit penanda dimulainya konversi ADC. Selama konversi berlogika 1dan akan berlogika 0 jika selesai konversi.

(c) ADATE merupakan bit pengatur aktivasi picu otomatis. Jika bernilai 1 maka konversi ADC akan dimulai saat tepi positif pada sinyal trigger yang digunakan.

(d) ADIF merupakan bit penanda akhir konversi ADC. Jika bernilai 1 maka konversi ADC pada suatu saluran telah selesai dan siap diakses

(e) ADIE merupakan bit pengatur aktivasi interupsi. Jika bernilai 1 maka interupsi penandaan telah selesai. Konversi ADC diaktifkan

(f) $ADPS_{0-2}$ merupakan bit pengatur clock ADC

Bit-bit ini menentukan faktor pembagian antara frekuensi XTAL dan clock masukan ADC seperti ditunjukkan pada Tabel 7. Waktu konversi ADC mikrokonroler AVR dapat diatur melalui melalui bit ADPS2:0 di dalam register ADCSRA. Untuk memilih waktu konversi, dengan memilih salah satu dari fosc/2, fosc/4, fosc/8, fosc/16, fosc/32, fosc/64, fosc/128 untuk *clock* ADC, di mana fosc adalah frekuensi kristal yang digunakan pada AVR.

Tabel 7. Konfigurasi Clock ADC

$ADPS_{0-2}$	ADCS
000 - 001	$f_{osc}/2$
010	$f_{osc}/4$
011	$f_{osc}/8$
100	$f_{osc}/16$
101	$f_{osc}/32$
110	$f_{osc}/64$
111	$f_{osc}/128$

(3) SFIOR (Special Function I/O Register)

SFIOR adalah register 8 bit yang mengatur sumber pemicu ADC. Jika bit ADATE pada register ADCSRA bernilai 0maka $ADTS_{0-2}$ tidak berfungsi.

Susunan register ini dapat dilihat pada Tabel 8.

Tabel 8. Register SFIOR

ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR

Bit 7..5 – ADTS [2..0] adalah bit pengatur pemicu eksternal operasi ADC. Bit-bit ini hanya berfungsi jika bit ADATE pada register ADCSRA bernilai *high.* Bit-bit ini bernilai awal 000 yang menandakan ADC bekerja pada mode konversi kontinyu dan tidak ada interupsi yang akan dihasilkan. Rincian nilai ADTS [2..0] dapat dilihat pada Tabel 9. Untuk operasi ADC, bit ACME, PUD, PSR2 dan PSR0 tidak diaktifkan.

Tabel 9. Tabel Pemilihan Sumber Picu ADC

ADTS2	ADTS1	ADTS0	Sumber Pemicu
0	0	0	*Free Running Mode*
0	0	1	*Analog Comparator*
0	1	0	*External Interupt Request*
0	1	1	*Timer/Counter0 Compare Match*
1	0	0	*Timer/Counter0 Overflow*
1	0	1	*Timer/Counter1 Compare Match B*
1	1	0	*Timer/Counter1 Overflow*
1	1	1	*Timer/Counter1 Capture Event*

C. Bahasa C

Bahasa C luas digunakan untuk pemrograman berbagai jenis perangkat, termasuk mikrokontroler ATMega8. Bahasa ini sudah merupakan high level

language, dimana memudahkan programmer membuat algoritmanya. Dasar bahasa C adalah sebagai berikut:

1. Struktur penulisan program

```
#include <[library1.h]>
#include <[library2.h]>
void main (void)
{
Deklarasi local variable
Isi program Utama
}
```

2. Tipe Data

a) char : 1 byte (-128 s/d 127)

b) unsigned char : 1 byte (0 s/d 255)

c) int : 2 byte (-32768 s/d 32767)

d) unsigned int : 2 byte (0 s/d 65535)

e) long : 4 byte (-2147483648 s/d 2147483647)

f) unsigned long : 4 byte (0 s/d 4294967295)

g) float : bilangan desimal

h) array : kumpulan data-data yang sama tipenya.

3. Deklarasi variabel & konstanta

a) Variabel adalah memori penyimpanan data yang nilainya dapat diubah ubah.

b) Konstanta adalah memori penyimpanan data yang nilainya tidak dapat diubah.

4. Statement

Statement adalah setiap operasi dalam pemrograman, harus diakhiri dengan [;] atau [}]. *Statement* tidak akan dieksekusi bila diawali dengan tanda [//] untuk satu baris. Lebih dari 1 baris gunakan pasangan [/*] dan [*/]. *Statement* yang tidak dieksekusi disebut juga komentar.

5. Function

Function adalah bagian program yang dapat dipanggil oleh program utama.

6. *Conditional statement* dan *looping*

a) *if else* : digunakan untuk penyeleksian kondisi.

b) *For* : digunakan untuk looping dengan jumlah yang sudah diketahui.

c) *while* : digunakan untuk looping jika dan salama memenuhi syarat tertentu.

d) *do while* : digunakan untuk looping jika dan salama memenuhi syarat tertentu, namun min 1 kali.

e) *switch case* : digunakan untuk seleksi dengan banyak kondisi.

7. Operasi logika dan biner

a) Logika : AND (&&), OR (||), NOT (!)

b) Biner : AND (&), OR(|), XOR (^)

8. Operasi relasional (perbandingan)

a) Sama dengan : ==

b) Tidak sama dengan : !=

c) Lebih besar : >

d) Lebih besar sama dengan : >=

e) Lebih kecil : <

f) Lebih kecil sama dengan : <=

9. Operasi aritmatika

a) + , - , * , / : tambah,kurang,kali,bagi

b) ++ : tambah satu (increment)

c) -- : kurang satu (decrement)

D. Penampil LCD 20x4 HD44870

Untuk mengetahui sebuah Mikrokontroler dapat mengeluarkan proses berupa hasil informasi maka diperlukan instrumen penampil layar yang berupa LCD (*Liquid Crystal Display*) yang berguna untuk menampilkan hasil pengambilan data.

LCD yang dipakai dalam pembuatan tugas akhir ini mempunyai tipe *controller* Hitachi HD44780. yang memiliki 4 baris dimana masing-masing setiap baris memiliki 20 karakter.

Untuk rangkaian *interfacing*, LCD tidak banyak memerlukan komponen pendukung. Hanya diperlukan satu resistor dan satu variabel resistor untuk memberi tegangan kontras pada matriks LCD dari sumber tegangan Vcc 5V, Bentuk fisik dari LCD 20x4 ini dapat dilihat pada gambar 7.

Gambar 7. Bentuk Fisik LCD 20x4

(*http://site.gravitech.us/MicroResearch/Others/LCD-20x4B/LCD-20x4B_1R.jpg*)

Bentuk fisik LCD 20x4 dengan *backlight* atau lampu belakang layar berwarna biru ditunjukkan pada Gambar 7, LCD yang serupa di pasaran tersedia dalam banyak warna *backlight* diantaranya warna: hijau, biru, dan merah. Pembuatan tugas akhir ini dipilih LCD 20x4 dengan *backlight* biru. Selanjutnya dari LCD tersebut terdapat Matriks LCD ditunjukkan pada Gambar 8.

Gambar 8. Matriks LCD 20x4

(*http://site.gravitech.us/MicroResearch/Others/LCD-20x4B/LCD-20x4B_1R.jpg*)

Pada Matriks LCD yang ditunjukkan pada Gambar 8. Apabila dilihat terdapat *pinout* yang berguna untuk menghubungkan antara Mikrokontroler dan LCD, Data-data fungsi *pinout* dari LCD dinyatakan dalam Tabel 10. konfigurasi pin LCD.

Tabel 10. Konfigurasi Pin LCD 20x4

No.	Pin	Function
1	VSS	*Power* 0V (GND)
2	VDD	*Power* 5V
3	VO	LCD *Contrast Voltage*
4	RS	*Register Select*; H: *Data Input*; L: *Instruction Input*
5	R/W	H: *Read*; L: *Write*
6	E	*Enable Signal*
7	D0	*H/L Data Bus Line*
8	D1	*H/L Data Bus Line*
9	D2	*H/L Data Bus Line*

10	D3	*H/L Data Bus Line*
11	D4	*H/L Data Bus Line*
12	D5	*H/L Data Bus Line*
13	D6	*H/L Data Bus Line*
14	D7	*H/L Data Bus Line*
15	LED +	Positif *Backlight Voltage* (4-4,2 V; 50-200mA)
16	LED -	Negatif *Backlight Voltage* (0 V; GND)

E. Transistor

Transistor merupakan semikonduktor berbahan dasar Silicon atau Germanium dengan bentuk kemasan yang sangat banyak jenisnya (TO-92, TO-220, dll). Secara umum transistor memiliki 3 titik penyambungan, yaitu Basis (B), Kolektor (C), dan Emittor (E).

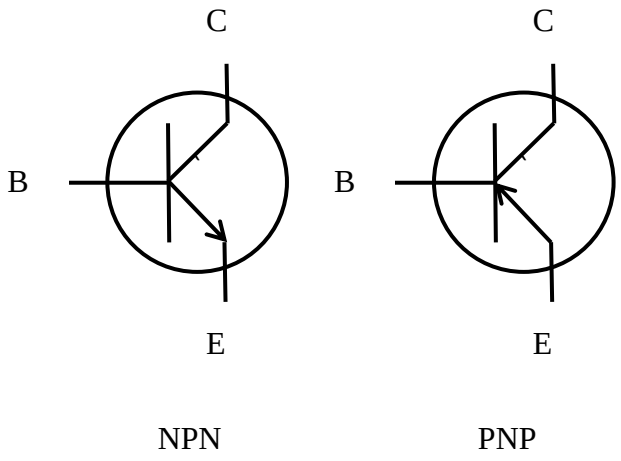
Pada prinsipnya transistor merupakan 2 buah dioda yang saling dipertemukan, yaitu dioda Basis-Emitor dan dioda Basis-Kolektor.

Kondisi tersebut menyebabkan transistor semacam ini disebut juga dengan pertemuan (*junctions*). Dengan adanya 2 kemungkinan untuk memnpertemukan kedua buah dioda tersebut, maka akan terdapat 2 jenis transistor yang dibentuk, yaitu transistor NPN (Negatif Positif Negatif) bila yang dipertemukan anodanya dan transistor PNP (Positif Negatif Positif) bila yang dipertemukan katodanya.

Bahan mentah yang digunakan untuk menghasilkan bahan N dan P adalah silikon dan germanium. Oleh karena itu, dikatakan:

1. Transistor germanium PNP dan NPN
2. Transistor silikon PNP dan NPN

Semua komponen didalam bagan transistor dinyatakan dengan simbol. anak panah yang terdapat didalam simbol menunjukkan arah yang melalui transistor.Selengkapnya pada Gambar 9. Tentang simbol Transistor NPN dan PNP



Gambar 9. Simbol Transistor NPN dan PNP

Berdasarkan jenisnya, identifikasi transistor dapat dengan mudah dilakukan dengan melihat *datasheet* transistor yang bersangkutan. Sebagai contoh, pengunaan transistor dalam pembuatan alat ini menggunakan spesifikasi dasar transistor daya dalam kondisi taraf maksimumnya.

Khusus transistor-transistor silikon yang memenuhi spesifikasi persyaratan TUP (*Transistor Unijunctions Positive*) dan TUN (*Transistor Unijunctions Negative*), memiliki karakteristik batas maksimal IC maksimum sebesar 150 mA dengan jangkauan tegangan kerja yang bervariasi, sedangkan transistor dengan arus kolektor maksimum lebih dari 150mA, dapat digolongkan dalam transistor penguat frekuensi audio atau radio (AF/RF) dan transistor daya (Po).

F. Relai

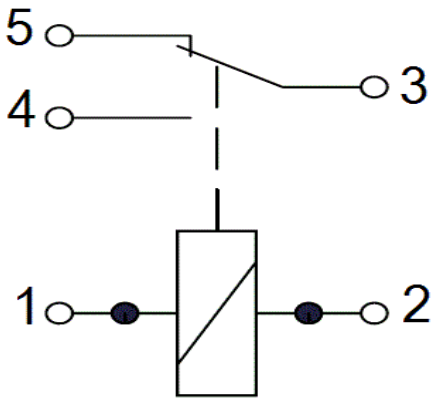
Relai merupakan komponen output yang paling sering digunakan pada beberapa peralatan elektronika dan di berbagai bidang lainnya. Relai berfungsi untuk menghubungkan atau memutuskan aliran arus listrik yang dikontrol dengan memberikan tegangan atau arus tertentu pada koilnya. Ada 2 macam relai berdasarkan tegangan untuk menggerakkan koilnya, yaitu AC dan DC.

Pada perangkat yang dibuat digunakan relay DC dengan tegangan koil 5VDC, jenis relai ini adalah HRS2H-S-DC5V. Jika dilihat dari kutubnya atau *Pole* relay ini menggunakan jenis *Double Pole Double Throw* (DPDT) yang berfungsi untuk menggerakkan peralatan di luar rangkaian. Bentuk fisik dari Relai HRS2H-S-DC5V tersebut ditunjukkan pada Gambar 10.

Gambar 10. Bentuk fisik Relai

Pada dasarnya relai adalah sebuah kumparan yang dialiri arus listrik sehingga kumparan mempunyai sifat sebagai magnet. Magnet sementara tersebut digunakan untuk menggerakkan suatu sistem saklar yang terbuat dari logam sehingga pada saat relai dialiri arus listrik maka kumparan akan terjadi

kemagnetan dan menarik logam tersebut, saat arus listrik diputus makan logam akan kembali pada posisi semula. Bentukdari Schematic relai ditunjukkan pada Gambar 11.



Gambar 11. Bentuk schematic Relai

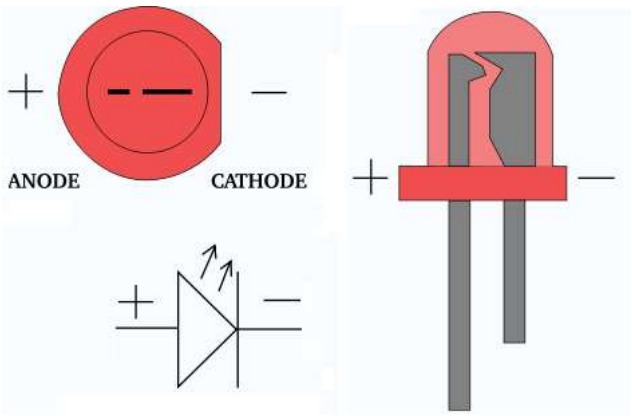
Pada saat ada arus yang mengalir pada kaki 1 dan 2 pada Gambar 11. Maka inti besi besi lunak akan menjadi magnet. Kemudian inti besi itu akan menarik kontak yang ada pada kaki 3, sehingga kaki 3 yang pada mulanya terhubung ke kaki 5 berubah kedudukan, yaitu terhubung ke kaki 4. Hal tersebut dapat terjadi jika kaki 5 relai bersifat NC (*Normally Close*) dan kaki 4 bersifat NO (*Normally Open*).

G. LED (*Light Emitting Dioda*)

LED merupakan komponen yang dapat mengeluarkan emisi cahaya. LED merupakan produk temuan lain setelah dioda. Strukturnya sama dengan dioda, tetapi belakangan ditemukan bahwa elektron yang menerjang sambungan P-N juga melepaskan energi panas dan energi cahaya. Karakteristik LED sama dengan

karakteristik dioda penyearah. Bedanya jika dioda membuang energi dalam bentuk panas, sedangkan LED membuang energi dalam bentuk cahaya.

Keuntungan menggunakan LED adalah struktur solid, ukurannya kecil, masa pakai tahan lama tidak terpengaruh oleh on/off pensaklaran, mudah dipakai dan mudah didapat. Karena tahan lama dan tidak terpengaruh on/off pensaklaran, maka LED banyak digunakan sebagai display atau indikator baik itu pada audio atau mesin-mesin kontrol.Bentuk fisik LED ditunjukkan pada Gambar 12.



Gambar 12. Bentuk fisik LED
(*http://www.societyofrobots.com/images/electronics_led_diagram.png*)

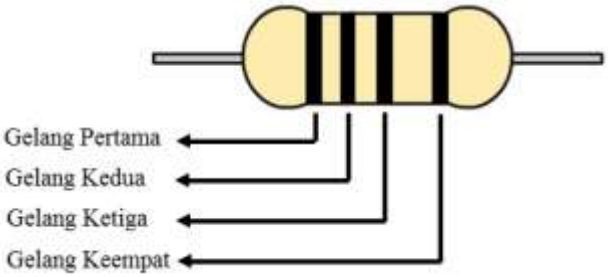
Radiasi cahaya yang dipancarkan LED tergantung dari materi dan susunan dioda P-N dan bahan semikonduktor penyusunan LED itu sendiri. Bahan semikonduktor yang sering digunakan dalam pembuatan LED adalah:

- Ga As (Galium Arsenide) meradiasikan sinar infra merah.
- Ga As P (Galium Arsenide Phospide) meradiasikan warna merah dan kuning.
- Ga P (Galium Phospide) meradiasikan warna merah dan kuning.

Seperti halnya sebuah dioda, salah satu karakteristik LED adalah harga ketergantungan antara I terhadap V. Grafik antara V-I untuk LED sama dengan grafik V-I untuk dioda penyearah. Perbedaanya terletak pada pengertian tegangan dan arus yang lewat. Harga arus I yang melewati LED menentukan intensitas cahaya yang dipancarkan, atau dengan kata lain arus LED sebanding dengan intensitas cahaya yang dihasilkan. Jika arus yang melewati LED besar, maka intensitas cahaya yang dihasilkan juga terang. Sebaliknya jika arus yang lewat kecil, maka nyala LED akan redup atau LED tidak akan menyala sama sekali.

H. Resistor

Resistor merupakan komponen elektronika pasif yang mempunyai fungsi dasar untuk menahan arus listrik atau membagi tegangan. Ada berbagai jenis resistor, namun disini hanya akan membahas resistor yang digunakan pada proses pembuatan alat tugas akhir ini, yakni resistor karbon. Resistor karbon terdiri atas sebuah unsur resistif berbentuk tabung dengan kawat atau tutup logam pada kedua ujungnya. Badan resistor dilindungi dengan cat atau plastik. Resistor komposisi karbon lawas mempunyai badan yang tidak terisolasi, kawat penghubung dililitkan di sekitar ujung unsur resistif dan kemudian disolder. Resistor yang sudah jadi dicat dengan kode warna nilainya, seperti ditunjukkan pada Gambar 13.



Gambar 13. Gelang-gelang resistor karbon

Gelang-gelang yang melekat pada badan resistor seperti pada Gambar 13. Memiliki kode warna yang dibuat untuk menjelaskaan besaran nilai hambatan pada resistor. Kode warna tersebut seperti dilihat pada Tabel 11. Yaitu tentang Nilai warna pada gelang resistor.

Tabel 11. Nilai warna gelang resistor karbon

Warna	Pita pertama	Pita kedua	Pita ketiga (pengali)	Pita keempat (toleransi)	Pita kelima (koefisien suhu)
Hitam	0	0	$\times 10^0$		
Cokelat	1	1	$\times 10^1$	±1% (F)	100 ppm
Merah	2	2	$\times 10^2$	±2% (G)	50 ppm
Oranye	3	3	$\times 10^3$		15 ppm
Kuning	4	4	$\times 10^4$		25 ppm
Hijau	5	5	$\times 10^5$	±0.5% (D)	
Biru	6	6	$\times 10^6$	±0.25% (C)	
Ungu	7	7	$\times 10^7$	±0.1% (B)	
Abu-abu	8	8	$\times 10^8$	±0.05% (A)	
Putih	9	9	$\times 10^9$		
Emas			$\times 10^{-1}$	±5% (J)	
Perak			$\times 10^{-2}$	±10% (K)	
Kosong				±20% (M)	

Untuk mengukur nilai hambatan pada resistor karbon dapat dilakukan dengan membaca gelang-gelang warna seperti pada Tabel 11. Dua gelang pertama merupakan informasi dua digit harga resistansi, gelang ketiga merupakan pengali (jumlah nol yang ditambahkan setelah dua digit resistansi), gelang keempat merupaka toleransi harga resistansi. Kadang-kadang gelang kelima menunjukkan koefisien suhu, tetapi ini harus dibedakan dengan sistem lima warna sejati yang menggunakan tiga digit resistansi.

Sebagai contoh, hijau-biru-kuning-merah adalah 56 x 10^4 Ω = 560 kΩ ±2%. Deskripsi yang lebih mudah adalah gelang pertama (hijau) mempunyai harga 5 dan gelang kedua (biru) mempunyai harga 6, dan keduanya dihitung 56. Gelang ketiga(kuning) mempunyai harga 10^4 , yang menambahkan empat nol dibelakang 56, sedangkan gelang keempat (merah) merupakan kode untuk toleransi ±2%, memberikan nilai 560.000 Ω pada keakuratan ±2%.

Metode paling mudah untuk mengukur hambatan adalah dengan menggunakan multimeter. Untuk multimeter *digital* dapat dengan mudah dilakukan dengan memindah sakelar posisi hambatan dan melihat nilai yang tertampil pada layar *display* pada saat kedua ujung kabel multimeter dihubungkan dengan resistor.

I. Kapasitor

Fungsi dasar kapasitor adalah menyimpan muatan listrik dan satuan kapasitansi yang digunakan adalah farad. Dengan fungsinya sebagai penyimpan muatan, maka pada aplikasi dalam rangkaian elektronika fungsi tersebut dapat

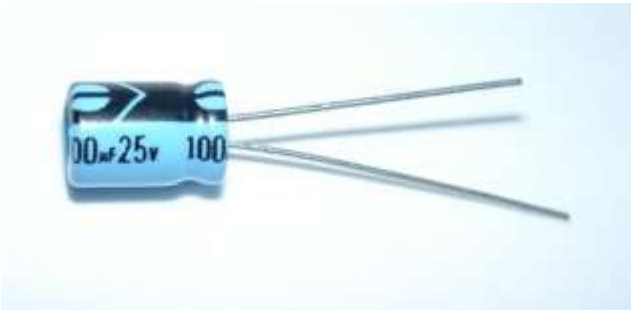
dikembangkan menjadi filter, waktu tunda (*delay*), pembangkit getaran (*oscillator*), dan coupling sinyal.

Pada pembuatan alat dalam tugas akhir ini, penggunaan kapasitor antara lain: kapasitor elektrolit, kapasitor keramik, dan kapsitor milar.

Berdasarkan bentuk fisik dan nilai kapasitansinya, kapasitor ini dibagi menjadi beberapa jenis. Berikut dibawah ini jenis-jenis kapasitor:

1. Kapasitor Elektrolit

Kapasitor elektrolit didesain dengan bahan elektrolit dan memiliki polaritas positif dan negatif. Polaritas negatif digambarkan pada tubuh kapasitor dan polaritas positif berada di kaki terpanjang. Kapasitor ini memiliki nilai yang cukup besar, yaitu mulai 0,1µF hingga puluhan ribu µF (mikrofarad) dan biasanya digunakan untuk *filter* pada power supply. Bentuk fisik dari kapasitor Elektrolit ditunjukkan pada Gambar 14.



Gambar 14. Kapasitor Elektrolit

2. Kapasitor Keramik

Kapasitor keramik didesain dengan menggunakan bahan keramik dan berkisar pada nilai 1 pF hingga 680 nF. Contoh dari bentuk fisik dari kapasitor keramik ditunjukkan pada Gamba15.

Gambar 15. Kapasitor Keramik

Cara menghitung nilai kapasitor jenis ini yaitu dua digit pertama adalah nilai dan digit ketiga adalah faktor pengali. Contoh, 224 adalah 22×10^4 pF atau 220 nF. Namun, sering kali diperoleh kapasitor dengan satu atau dua digit saja. Pada kapasitor jenis ini, nilai tersebut adalah nilai kapasitansi kapasitor dalam pF.

3. Kapasitor Milar

Kapasitor in biasanya berkisar antara 1 nF hingga 1 µF. Dibandingkan dengan kapasitor keramik, jenis ini lebih tahan terhadap suhu panas.

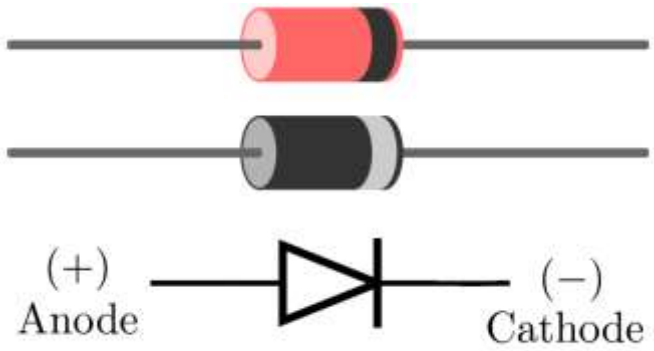
Gambar 16. Kapasitor Milar

J. Dioda

Dioda adalah komponen elektronika yang fungsi utamanya sebagai penyearah. Arus yang bergerak melalui dioda hanya dapat mengalir searah dari bagian positif ke bagian negatifnya, sedangkan arah sebaliknya akan terhambat.

Untuk Dioda jenis germanium penurunan tegangannya adalah 0,2 volt, sedangkan untuk dioda jenis silikon (paling banyak dijumpai di pasaran) adalah 0,7 volt. Oleh karena itu,fungsi dioda dapat dikembangkan menjadi penurun tegangan sebesar 0,7 volt dan penyearah tegangan (AC ke DC).

Simbol dari komponen Dioda ini ditunjukkan pada Gambar 17. Dimana terdapat dua buah kutub yaitu Anoda dan Katoda.



Gambar 17. Simbol dan bentuk fisik Dioda

BAB IV

PENUTUP

A. Kesimpulan

Berdasarkan data yang diperoleh dari pengujian alat yang dibuat, maka dapat dibuat sebuah kesimpulan sebagai berikut :

1. Alat tugas akhir yang dibuat yaitu "Multitester Elektronik Berbasis Mikrokontroler ATMega 8" dapat mengukur dan menampilkan hasil pengukuran kedalam tampilan LCD (*Liquid Crystal Display*) dan langsung menunjukkan harga pengukuran disertai dengan satuannya.
2. Dari hasil pengujian, melalui grafik pengujian alat ukur yang dibuat mampu mengukur dan menampilkan satuan yang sebenarnya dari banyak macam komponen elektronika, diantaranya: Resistor, Kapasitor, Dioda, dan Transistor.
3. Dari hasil pengujian, melalui grafik pengujian alat ukur yang dibuat dapat mengukur/membaca Tegangan DC dan Arus DC yang hasilnya mendekati multitester standar pabrikan.

B. Saran

1. Untuk pengujian Arus DC & Tegangan DC hendaknya tegangan yang diukur tidak melebihi: 20 Volt DC, dan arus yang diukur tidak melebihi: 12 Ampere.
2. Memperhatikan probe test ukur agar tidak salah mengukur komponen yang dapat menyebabkan kerusakan pada alat tugas akhir.

DAFTAR PUSTAKA

Budiharto, Widodo, dan Sigit Firmansyah. 2010. *Elektronika Digital dan Mikroprosesor*. Yogyakarta : Penerbit Andi.

Heryanto, M. Ary. 2008. P*emrograman Bahasa C untuk Mikrokontroler ATMega 8535*. Yogyakarta : Penerbit Andi.

Kurniawan, Dayat. 2009. *ATMega 8 dan Aplikasinya*. Jakarta : Penerbit Elex Media Komputindo.

Setiawan, Afrie. 2011. *20 Aplikasi Mikrokontroler ATMega 8535 & ATMega 16 Menggunakan BASCOM-AVR.* Yogyakarta : Penerbit Andi.

Syahrul. 2014. *Pemrograman Mikrokontroler AVR Bahasa Assembly dan C*. Bandung : Penerbit Informatika.

Winoto, Ardi. 2010. *Mikrokontroler AVR ATMega8/16/32/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR*. Bandung : Penerbit Informatika.

http://www.mikrocontroller.net/articles/AVR-Transistortester diakses tanggal 09-Juni 2014, Pukul 07:55 WIB.

http://www.circuitsdiy.com/atmega8-based-voltmeter-ampmeter-v2/ diakses tanggal 10 juni 2014, Pukul 09:15 WIB.

LAMPIRAN

Lampiran 1 : Datasheet Mikrokontroler AVR ATMega 8

Atmel

8-bit Atmel Microcontroller with 8KB In-System Programmable Flash

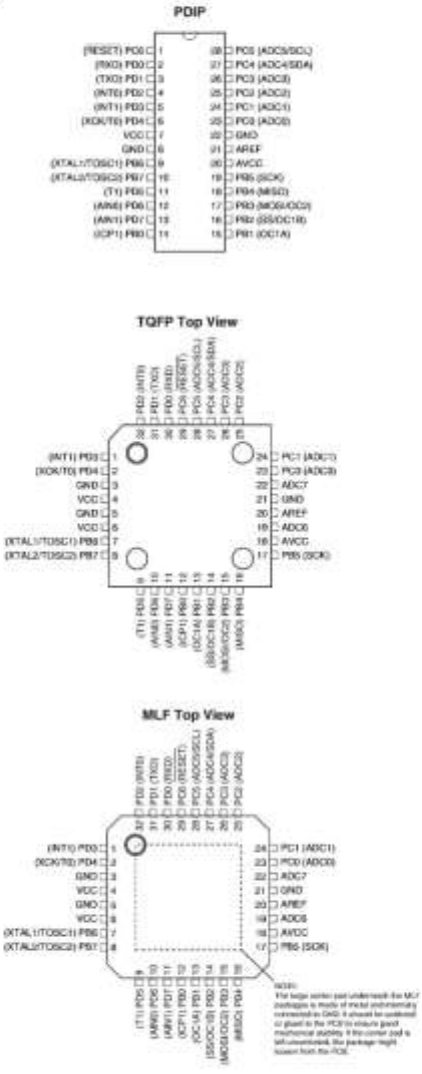
ATmega8A

Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 8KBytes of In-System Self-programmable Flash program memory
  - 512Bytes EEPROM
  - 1KByte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - Atmel QTouch and QMatrix acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    - Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    - Six Channels 10-bit Accuracy
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V
  - 0 - 16MHz
- Power Consumption at 4MHz, 3V, 25°C
  - Active: 3.6mA
  - Idle Mode: 1.0mA
  - Power-down Mode: 0.5µA

1. Pin Configurations

Figure 1-1. Pinout ATmega8A

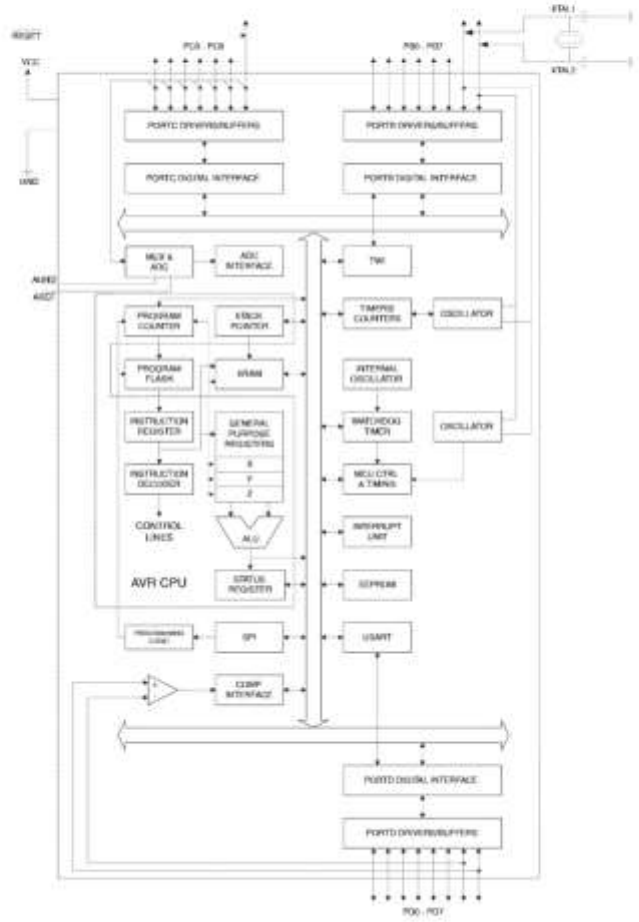


2. Overview

he Atmel®AVR® ATmega8A is a low-power CMOS 8-bit microcontrol er based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8A achieves throughputs approachi per MHz, al owing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The Atmel®AVR® AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8A provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8A is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The Atmel AVR ATmega8A is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program simulators and evaluation kits.

2.2 Pin Descriptions

2.2.1 VCC

Digital supply voltage.

2.2.2 GND

Ground.

2.2.3 Port B (PB7:PB0) – XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7:6 is used as TOSC2:1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 56 and "System Clock and Clock Options" on page 24.

2.2.4 Port C (PC5:PC0)

Port C is an 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

2.2.5 PC6/$\overline{\text{RESET}}$

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 26-3 on page 228. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated on page 59.

2.2.6 Port D (PD7:PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega8A as listed on page 61.

2.2.7 RESET

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 26-3 on page 228. Shorter pulses are not guaranteed to generate a reset.

2.2.8 AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3:0), and ADC (7:6). It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that Port C (5:4) use digital supply voltage, V_{CC}.

2.2.9 AREF

AREF is the analog reference pin for the A/D Converter.

2.2.10 ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

23. Analog-to-Digital Converter

23.1 Features

- 10-bit Resolution
- 0.5LSB Integral Non-linearity
- ± 2LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 15kSPS at Maximum Resolution
- 6 Multiplexed Single Ended Input Channels
- 2 Additional Multiplexed Single Ended Input Channels (TQFP and QFN/MLF Package only)
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

23.2 Overview

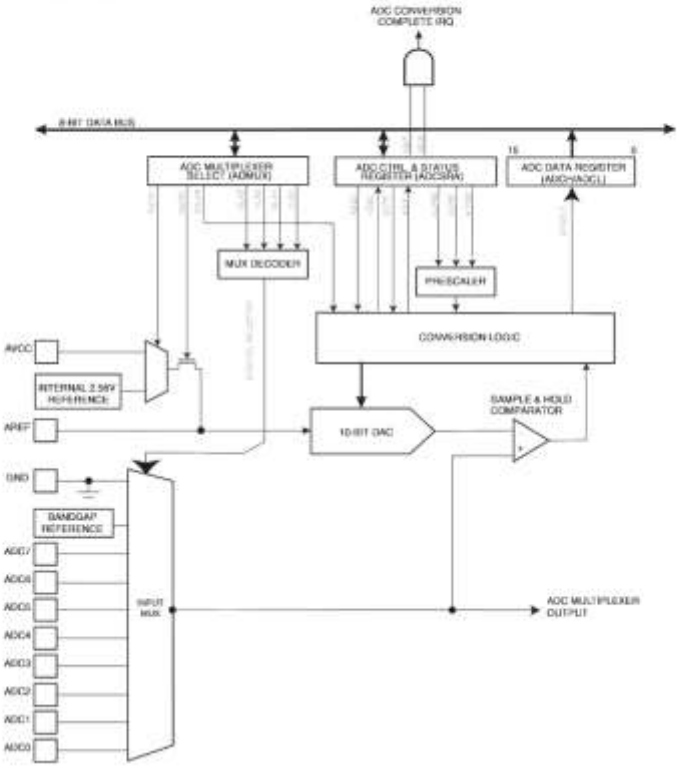
The ATmega8A features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port C. The single-ended voltage inputs refer to 0V (GND).

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 23-1.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ± 0.3V from V_{CC}. See

Internal reference voltages of nominally 2.56V or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Figure 23-1. Analog to Digital Converter Block Schematic Operation

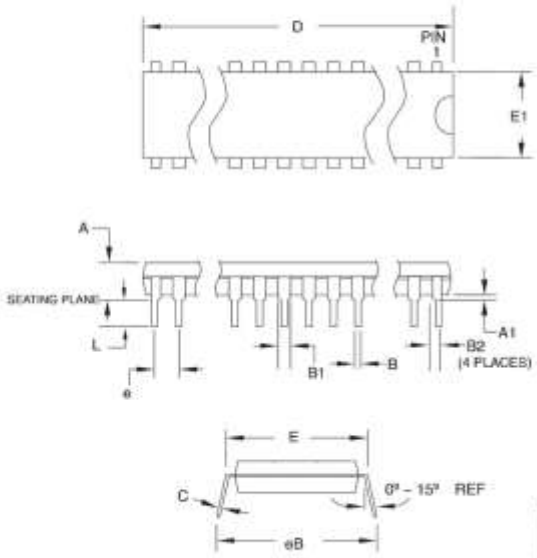


The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1LSB. Optionally, AV_{CC} or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

33.2 28P3



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.5724	
A1	0.508	–	–	
D	34.544	–	34.798	Note 1
E	7.620	–	8.255	
E1	7.112	–	7.493	Note 1
B	0.381	–	0.533	
B1	1.143	–	1.397	
B2	0.762	–	1.143	
L	3.175	–	3.429	
C	0.203	–	0.356	
eB	–	–	10.160	
e	2.540 TYP			

Note: 1. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**28P3**, 28-lead (0.300"/7.62mm Wide) Plastic Dual Inline Package (PDIP)	28P3	B

Lampiran 2 : *Source Code* Mikrokontroler 1

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Multitester Elektronik Berbasis Mikrokontroler ATMega
8
Version : v1.0
Date    : 10/28/2014
Author  : Nanda Puji Arianto
Company : ALIENTRONIC
Comments: SOURCE CODE MIKROKONTROLER 1


Chip type               : ATmega8
Program type            : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega8.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>

#define LED1 PORTB.6
#define LED2 PORTB.7
#define TUNDA delay_ms
// Alphanumeric LCD Module functions
#include <alcd.h>

#define ADC_VREF_TYPE 0xC0
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
// Declare your global variables here
unsigned dataADC,dataADC2,dataADC3;
```

```
int nilai,nilai2,nilai3;

void multitester(void)
{

tampil:
lcd_gotoxy(0,0);
lcd_putsf("  Silahkan Pilih !  ");
lcd_gotoxy(0,2);
lcd_putsf("VA Meter         (1)");
lcd_gotoxy(0,3);
lcd_putsf("Component Tester (2)");

LED1=1;
LED2=1;
TUNDA(100);
LED1=0;
LED2=0;
TUNDA(600);


if (PINB.1==0) {
LED1=1;
lcd_clear();
lcd_gotoxy(0,1);
lcd_putsf("OK!");
delay_ms(1000);
lcd_clear();
goto go;
}

if (PINB.2==0) {
LED2=1;
lcd_clear();
lcd_gotoxy(0,1);
lcd_putsf("OK!");
delay_ms(1000);
lcd_clear();
goto go2;
}

goto tampil;

go:
{

dataADC=read_adc(0);
dataADC2=read_adc(1);

nilai=(dataADC*5);
nilai2=(dataADC2*10);

lcd_gotoxy(6,2);
nilai%=10000;
```

```
lcd_putchar((nilai/1000)+48); //10.00
nilai%=1000;
lcd_putchar((nilai/100)+48);  //1.00
nilai%=100;
lcd_puts(".");
lcd_putchar((nilai/10)+48);   //0.1
lcd_putchar((nilai%10)+48);   //0.01

lcd_gotoxy(6,3);
nilai2%=10000;
lcd_putchar((nilai2/1000)+48); //10.00
nilai2%=1000;
lcd_putchar((nilai2/100)+48);  //1.00
nilai2%=100;
lcd_puts(".");
lcd_putchar((nilai2/10)+48);   //0.1
lcd_putchar((nilai2%10)+48);   //0.01



lcd_gotoxy(0,2);
lcd_putsf("Teg.:");
lcd_gotoxy(12,2);
lcd_putsf("Volt");

lcd_gotoxy(0,3);
lcd_putsf("Arus:");
lcd_gotoxy(12,3);
lcd_putsf("Ampere");

if (dataADC>=3&&dataADC2==0)
{
lcd_gotoxy(0,0);
delay_ms(500);
lcd_putsf("Tegangan Terdeteksi!");
delay_ms(500);
lcd_clear();
};

if(dataADC2>=3&&dataADC==0)
{
lcd_gotoxy(0,0);
delay_ms(500);
lcd_putsf("Arus Terdeteksi!");
delay_ms(500);
lcd_clear();
};

if(dataADC>=3&&dataADC2>=3)
{
lcd_gotoxy(0,0);
lcd_putsf("                    ");
delay_ms(500);
lcd_gotoxy(0,1);
```

```
lcd_putsf("Multi Ukur !");
delay_ms(500);
lcd_clear();
};
}
goto go;

go2:
{

PORTD.0=0;
delay_ms(10);
PORTB.0=1;

}
goto go2;
}



void startup_text(void)
{
delay_ms(100);
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("    Tugas Akhir   ");
lcd_gotoxy(0,1);
lcd_putsf("NANDA PUJI ARIANTO");
lcd_gotoxy(0,2);
lcd_putsf("NIM : 5311311009");
lcd_gotoxy(0,3);
lcd_putsf("Teknik Elektro, D3");
delay_ms(6000);
lcd_clear();

lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Multitester ");
lcd_gotoxy(0,1);
lcd_putsf("Elektronik Berbasis");
lcd_gotoxy(0,2);
lcd_putsf("Mikrokontroller");
lcd_gotoxy(0,3);
lcd_putsf("ATmega 8");
delay_ms(4000);
lcd_clear();


lcd_gotoxy(0,1);
lcd_putsf("Sdg. Periksa Baterai");
lcd_gotoxy(5,2);
lcd_putsf("Volt");

dataADC3=read_adc(2);
```

```
nilai3=(dataADC3*5);
lcd_gotoxy(0,2);
nilai3%=1000;
lcd_putchar((nilai3/100)+48);  //1.00
nilai3%=100;
lcd_puts(".");
lcd_putchar((nilai3/10)+48);   //0.1
lcd_putchar((nilai3%10)+48);   //0.01

lcd_gotoxy(0,0);
lcd_putsf("Harap Tunggu.");
delay_ms(1000);
lcd_gotoxy(0,0);
lcd_putsf("Harap Tunggu..");
delay_ms(800);
lcd_gotoxy(0,0);
lcd_putsf("Harap Tunggu...");
delay_ms(500);
lcd_gotoxy(0,0);
lcd_putsf("Harap Tunggu....");
delay_ms(500);
lcd_gotoxy(0,0);
lcd_putsf("Harap Tunggu.....");

}

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=Out Func6=Out Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=Out
// State7=0 State6=0 State5=T State4=T State3=T State2=P
State1=P State0=0
PORTB=0x06;
DDRB=0xC1;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T
State1=T State0=1
PORTD=0x01;
DDRD=0x01;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
```

```
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTD Bit 7
// RD - PORTD Bit 6
// EN - PORTD Bit 5
// D4 - PORTD Bit 4
// D5 - PORTD Bit 3
// D6 - PORTD Bit 2
// D7 - PORTD Bit 1
// Characters/line: 20
lcd_init(20);
lcd_clear();
startup_text();
while (1)
      {
      // Place your code here

      if (dataADC3>=150)
      {
      delay_ms(100);
      lcd_gotoxy(0,3);
      lcd_putsf("Baterai OK!");
      delay_ms(1000);
      lcd_clear();
      multitester();
      }

      if (dataADC3<=150)
      {
      delay_ms(100);
      lcd_gotoxy(0,3);
      lcd_putsf("Baterai Lemah !");
      delay_ms(1000);
      lcd_clear();
      lcd_gotoxy(0,0);
```

```
        lcd_putsf("Baterai Lemah ");
        lcd_gotoxy(0,1);
        lcd_putsf("Ganti Baterai !");
        lcd_gotoxy(0,2);
        lcd_putsf("Atau");
        lcd_gotoxy(0,3);
        lcd_putsf("Sambungkan Adaptor");
        break;
        }
        }
}
```

Lampiran 3 : *Source Code* Mikrokontroler 2

```
#include <avr/io.h>

#include "lcd-routines.h"
#include "config.h"
#include <util/delay.h>
#include <avr/sleep.h>
#include <stdlib.h>
#include <string.h>
#include <avr/eeprom.h>
#include <avr/wdt.h>
#include <math.h>

//AVR-Typenabhänggikeit
#if defined(__AVR_ATmega48__)
	#define MCU_STATUS_REG MCUCR
#elif defined(__AVR_ATmega88__)
	#define MCU_STATUS_REG MCUCR
	#define UseM8
#elif defined(__AVR_ATmega168__)
	#define MCU_STATUS_REG MCUCR
	#define UseM8
#else
	#define MCU_STATUS_REG MCUCSR
	#define UseM8
#endif


/*#########################################################################
##################
	Konfigurations-Einstellungen
*/

/* Port für die Test-Pins
	Dieser Port muss über einen ADC verfügen (beim Mega8 also PORTC).
	Für die Test-Pins müssen die unteren 3 Pins dieses Ports benutzt
werden.
	Bitte die Definitionen für TP1, TP2 und TP3 nicht ändern!
*/

#define ADC_PORT PORTC
#define ADC_DDR DDRC
#define ADC_PIN PINC
#define TP1 PC0
#define TP2 PC1
#define TP3 PC2

#ifdef UseM8
	/*Einstellungen für Kapazitätsmessung (nur für ATMega8
interessant)
	Der Test, ob ein Kondensator vorhanden ist, dauert relativ lange,
mit über 50ms je Testvorgang ist zu rechnen
```

```
	Bei allen 6 möglichen Testvorgängen ergibt das eine Verlängerung
der Testdauer um ca 0,3s bis 0,5s.
	Mit CAP_TEST_MODE lassen sich die durchgeführten Tests festlegen.

	Bedeutungen der Bits (7 = MSB):
	7:6 Nicht verwendet

	5:4 Test-Modus
	00: Kondensator-Messung deaktiviert
	01: Kondensator-Messung für eine einstellbare Pin-Kombination (in
beide Richtungen); verlängert Testdauer um ca. 120...200ms
	10: Kondensator-Messung für alle 6 Pin-Kombinationen; verlängert
Testdauer um ca. 300...500ms

	3:2 Erster Pin der gewählten Pin-Kombination (0...2), nur
entscheidend wenn Bits 5:4 = 01

	1:0 Zweiter Pin der gewählten Pin-Kombination (0...2), nur
entscheidend wenn Bits 5:4 = 01
	*/
	uint8_t CapTestMode EEMEM = 0b00100010;//Messung für alle 6 Pin-
Kombinationen
#endif

#ifdef UseM8
	//3 EEPROM-Bytes für zukünftige Verwendung reserviert
	uint8_t RFU1 EEMEM = 0;
	uint8_t RFU2 EEMEM = 0;
	uint8_t RFU3 EEMEM = 0;
#endif


/*
	Genaue Werte der verwendeten Widerstände in Ohm.
	Der Nennwert für R_L ist 680 Ohm, für R_H 470kOhm
	Um das Programm auf Abweichungen von diesen Werten (z.B. durch
Bauteiltoleranzen)
	zu kalibrieren, die Widerstandswerte in Ohm in die folgenden
Defines eintragen:
*/
#ifdef UseM8
	unsigned int R_L_VAL EEMEM = 680;		//R_L;
Normwert 680 Ohm
	unsigned int R_H_VAL EEMEM = 4700;		//R_H;
Normwert 470000 Ohm, durch 100 dividiert angeben

#else
	#define M48_RH_RL_RATIO 691	//Verhältnis von R_H zu R_L; nur
für Mega48 nötig. Beim Mega8 wird dieser Wert aus den EEPROM-Konstanten
berechnet.
#endif

#ifdef UseM8
```

```
	/*	Faktoren für die Kapatitätsmessung bei Kondensatoren
		Diese Faktoren hängen von Fertigungstoleranzen des AVR ab
und müssen somit ggf. angepasst werden
		H_CAPACITY_FACTOR ist für die Messung mit 470k-Widerstand
(geringe Kapazität)
		L_CAPACITY_FACTOR ist für die Messung mit 680-Ohm-
Widerstand (hohe Kapazität)
		Der gesamte Messbereich ist ca. 0,2nF bis 7300µF.
	*/
	unsigned int H_CAPACITY_FACTOR EEMEM = 394;
	unsigned int L_CAPACITY_FACTOR EEMEM = 283;
#endif


/*##############################################################################
##################
Ende der Konfigurations-Einstellungen
*/

/*Strings im EEPROM
Beim Hinzufügen weiter Sprachen müssen alle Strings mit "€"-Zeichen
(ASCII 0x80) auf gleiche Länge gebracht werden,
sonst gibt es Probleme mit der LCD-Anzeige bei den verschiedenen
Sprachen!
*/


#elif defined(ENGLISH)    //englisch
	unsigned char TestRunning[] EEMEM = "Testing ...€€€€€€";
	unsigned char Bat[] EEMEM = "Battery €";
	unsigned char BatWeak[] EEMEM = "weak€€€";
	unsigned char BatEmpty[] EEMEM = "empty!€€";
	unsigned char TestFailed1[] EEMEM = "No, unknown, or€";
	unsigned char TestFailed2[] EEMEM = "damaged €€€€";
	unsigned char Bauteil[] EEMEM = "part€€€€€€";
	unsigned char Unknown[] EEMEM = " unknown€";
	unsigned char Diode[] EEMEM = "Diode: ";
	unsigned char DualDiode[] EEMEM = "Double diode €";
	unsigned char TwoDiodes[] EEMEM = "2 diodes";
	unsigned char Antiparallel[] EEMEM = "anti-parallel";
	unsigned char InSeries[] EEMEM = "serial A=€€";
	unsigned char K1[] EEMEM = ";C1=";
	unsigned char K2[] EEMEM = ";C2=";
	unsigned char GAK[] EEMEM = "GAC=";
	unsigned char NextK[] EEMEM = ";C=";
	unsigned char K[] EEMEM = "C=";
	unsigned char Triac[] EEMEM = "Triac";
	unsigned char Thyristor[] EEMEM = "Thyristor";


	#ifdef UseM8
		unsigned char OrBroken[] EEMEM = "or damaged €€";
		unsigned char Resistor[] EEMEM = "Resistor: €€";
		unsigned char Capacitor[] EEMEM = "Capacitor: €€";
```

```
      #endif

#endif



//Sprachunabhängige EEPROM-Strings
unsigned char mosfet[] EEMEM = "-MOS";
unsigned char emode[] EEMEM = "-E";
unsigned char dmode[] EEMEM = "-D";
unsigned char jfet[] EEMEM = "-JFET";
unsigned char A1[] EEMEM = ";A1=";
unsigned char A2[] EEMEM = ";A2=";
unsigned char NullDot[] EEMEM = "0,";
unsigned char GateCap[] EEMEM = " C=";
unsigned char hfestr[] EEMEM ="hFE=";
unsigned char NPN[] EEMEM = "NPN";
unsigned char PNP[] EEMEM = "PNP";
unsigned char bstr[] EEMEM = " B=";
unsigned char cstr[] EEMEM = ";C=";
unsigned char estr[] EEMEM = ";E=";
unsigned char gds[] EEMEM = "GDS=";
unsigned char Uf[] EEMEM = "Uf=";
unsigned char vt[] EEMEM = "Vt=";
unsigned char mV[] EEMEM = "mV";
unsigned char Anode[] EEMEM = "A=";
unsigned char Gate[] EEMEM = "G=";
unsigned char CA[] EEMEM = "CA";
unsigned char CC[] EEMEM = "CC";
unsigned char TestTimedOut[] EEMEM = "Timeout!";

unsigned char DiodeIcon[] EEMEM = {4,31,31,14,14,4,31,4};  //Dioden-Icon

#ifdef LCD_CYRILLIC //Omega- und µ-Zeichen als Custom-Zeichen erzeugen,
weil diese Zeichen im kyrillischen Zeichensatz nicht enthalten sind
       unsigned char CyrillicOmegaIcon[] EEMEM = {0,0,14,17,17,10,27,0};
       //Omega
       unsigned char CyrillicMuIcon[] EEMEM = {0,17,17,17,19,29,16,16};
       //µ
#endif

//Ende der EEPROM-Strings

//Watchdog
#define WDT_enabled
/* Wird das Define "WDT_enabled" entfernt, wird der Watchdog beim
Programmstart
 nicht mehr aktiviert. Das ist für Test- und Debuggingzwecke sinnvoll.
 Für den normalen Einsatz des Testers sollte der Watchdog aber unbedingt
aktiviert werden!
*/


struct Diode {
```

```
	uint8_t Anode;
	uint8_t Cathode;
	int Voltage;
};

void CheckPins(uint8_t HighPin, uint8_t LowPin, uint8_t TristatePin);
void DischargePin(uint8_t PinToDischarge, uint8_t DischargeDirection);
unsigned int ReadADC(uint8_t mux);
void lcd_show_format_cap(char outval[], uint8_t strlength, uint8_t
CommaPos);

#ifdef UseM8
	void ReadCapacity(uint8_t HighPin, uint8_t LowPin);
	//Kapazitätsmessung nur auf Mega8 verfügbar
#endif


#define R_DDR DDRB
#define R_PORT PORTB

/* Port für die Testwiderstände
	Die Widerstände müssen an die unteren 6 Pins des Ports
angeschlossen werden,
	und zwar in folgender Reihenfolge:
	RLx = 680R-Widerstand für Test-Pin x
	RHx = 470k-Widerstand für Test-Pin x

	RL1 an Pin 0
	RH1 an Pin 1
	RL2 an Pin 2
	RH2 an Pin 3
	RL3 an Pin 4
	RH3 an Pin 5


*/



#define ON_DDR DDRD
#define ON_PORT PORTD
#define ON_PIN_REG PIND
#define ON_PIN PD6  //Pin, der auf high gezogen werden muss, um
Schaltung in Betrieb zu halten
#define RST_PIN PD7 //Pin, der auf low gezogen wird, wenn der Einschalt-
Taster gedrückt wird

//Bauteile
#define PART_NONE 0
#define PART_DIODE 1
#define PART_TRANSISTOR 2
#define PART_FET 3
#define PART_TRIAC 4
#define PART_THYRISTOR 5
```

```
#define PART_RESISTOR 6
#define PART_CAPACITOR 7

//Ende (Bauteile)
//Spezielle Definitionen für Bauteile
//FETs
#define PART_MODE_N_E_MOS 1
#define PART_MODE_P_E_MOS 2
#define PART_MODE_N_D_MOS 3
#define PART_MODE_P_D_MOS 4
#define PART_MODE_N_JFET 5
#define PART_MODE_P_JFET 6

//Bipolar
#define PART_MODE_NPN 1
#define PART_MODE_PNP 2


struct Diode diodes[6];
uint8_t NumOfDiodes;

uint8_t b,c,e;                  //Anschlüsse des Transistors
unsigned long lhfe;       //Verstärkungsfaktor
uint8_t PartReady;        //Bauteil fertig erkannt
unsigned int hfe[2];            //Verstärkungsfaktoren
unsigned int uBE[2];      //B-E-Spannung für Transistoren
uint8_t PartMode;
uint8_t tmpval, tmpval2;
#ifdef UseM8 //Widerstands- und Kondensatormessung nur auf dem Mega8
verfügbar
      uint8_t ra, rb;                       //Widerstands-Pins
      unsigned int rv[2];             //Spannungsabfall am Widerstand
      unsigned int radcmax[2];  //Maximal erreichbarer ADC-Wert
(geringer als 1023, weil Spannung am Low-Pin bei Widerstandsmessung über
Null liegt)
      uint8_t ca, cb;                       //Kondensator-Pins
      uint8_t cp1, cp2;               //Zu testende Kondensator-Pins,
wenn Messung für einzelne Pins gewählt
      uint8_t ctmode;                       //Kondensator-Test-Modus
(siehe ab Zeile 40)
      #ifdef SWUART_INVERT
            #define TXD_VAL 0
      #else
            #define TXD_VAL (1<<TxD)
      #endif
#endif

unsigned long cv;

uint8_t PartFound, tmpPartFound; //das gefundene Bauteil
char outval[8];
unsigned int adcv[4];
unsigned int gthvoltage;  //Gate-Schwellspannung
uint8_t tmpval, tmpval2;
```

```
#ifdef UseM8
	char outval2[6];
#endif

//Programmbeginn
int main(void) {
	//Einschalten
	ON_DDR = (1<<ON_PIN);
	ON_PORT = (1<<ON_PIN) | (1<<RST_PIN);  //Strom an und Pullup für
Reset-Pin
	uint8_t tmp;
	//ADC-Init
	ADCSRA = (1<<ADEN) | (1<<ADPS1) | (1<<ADPS0); //Vorteiler=8
	lcd_init();

	#ifdef UseM8
		//Konstanten aus EEPROM laden
		unsigned int rhval = eeprom_read_word(&R_H_VAL);	//R_H
		unsigned int rlval = eeprom_read_word(&R_L_VAL);	//R_L
		ctmode = eeprom_read_byte(&CapTestMode);
		cp1 = (ctmode & 12) >> 2;
		cp2 = ctmode & 3;
		ctmode = (ctmode & 48) >> 4;
	#endif

	wdt_disable();
	if(MCU_STATUS_REG & (1<<WDRF)) {
		/*
		Überprüfen auf Watchdog-Reset
		Das tritt ein, wenn der Watchdog 2s nicht zurückgesetzt
wurde
		Kann vorkommen, wenn sich das Programm in einer
Endlosschleife "verheddert" hat.
		*/
		lcd_eep_string(TestTimedOut);	//Timeout-Meldung
		_delay_ms(3000);
		ON_PORT = 0; //Abschalten!
		return 0;
	}
	LCDLoadCustomChar(0);	//Custom-Zeichen
	//Diodensymbol in LCD laden
	lcd_eep_customchar(DiodeIcon);

	#ifdef LCD_CYRILLIC
		//bei kyrillischen LCD-Zeichensatz Omega- und µ-Zeichen
laden
		LCDLoadCustomChar(1);	//Custom-Zeichen
		//Omega-Zeichen in LCD laden
		lcd_eep_customchar(CyrillicOmegaIcon);
		LCDLoadCustomChar(2);	//Custom-Zeichen
		//µ-Zeichen in LCD laden
		lcd_eep_customchar(CyrillicMuIcon);
	#endif
```

```
	Line1();	//1. Zeile

	//Einsprungspunkt, wenn Start-Taste im Betrieb erneut gedrückt
wird
	start:
	#ifdef WDT_enabled
		wdt_enable(WDTO_2S);	//Watchdog an
	#endif
	PartFound = PART_NONE;
	tmpPartFound = PART_NONE;
	NumOfDiodes = 0;
	PartReady = 0;
	PartMode = 0;
	#ifdef UseM8
	ca = 0;
	cb = 0;
	#endif
	lcd_clear();
	ADC_DDR = (1<<TxD); //Software-UART aktivieren
	uart_newline();
	//Versorgungsspannung messen
	ReadADC(5 | (1<<REFS1));  //Dummy-Readout
	hfe[0] = ReadADC(5 | (1<<REFS1));	//mit interner Referenz
	if (hfe[0] < 650) {		//Vcc < 7,6V; Warnung anzeigen
		lcd_eep_string(Bat);		//Anzeige: "Batterie"
		if(hfe[0] < 600) {			//Vcc <7,15V;
zuverlässiger Betrieb nicht mehr möglich
			lcd_eep_string(BatEmpty);	//Batterie leer!
			_delay_ms(1000);
			PORTD = 0;   //abschalten
			return 0;
		}
		lcd_eep_string(BatWeak);	//Batterie schwach
		Line2();
	}
	//Test beginnen
	lcd_eep_string(TestRunning);	//String: Test läuft
	//Alle 6 Kombinationsmöglichkeiten für die 3 Pins prüfen
	CheckPins(TP1, TP2, TP3);
	CheckPins(TP1, TP3, TP2);
	CheckPins(TP2, TP1, TP3);
	CheckPins(TP2, TP3, TP1);
	CheckPins(TP3, TP2, TP1);
	CheckPins(TP3, TP1, TP2);
	#ifdef UseM8
		//Separate Messung zum Test auf Kondensator
		if(((PartFound == PART_NONE) || (PartFound ==
PART_RESISTOR) || (PartFound == PART_DIODE)) && (ctmode > 0)) {
			//Kondensator entladen; sonst ist evtl. keine
Messung möglich
			R_PORT = 0;
			R_DDR = (1<<(TP1 * 2)) | (1<<(TP2 * 2)) | (1<<(TP3 *
2));
			_delay_ms(10);
```

```
                R_DDR = 0;
                //Kapazität in allen 6 Pin-Kombinationen messen
                if(ctmode == 1) {
                        ReadCapacity(cp1, cp2);
                        ReadCapacity(cp2, cp1);
                } else {
                        ReadCapacity(TP3, TP1);
                        ReadCapacity(TP3, TP2);
                        ReadCapacity(TP2, TP3);
                        ReadCapacity(TP2, TP1);
                        ReadCapacity(TP1, TP3);
                        ReadCapacity(TP1, TP2);
                }
        }
#endif
//Fertig, jetzt folgt die Auswertung
lcd_clear();
if(PartFound == PART_DIODE) {
        if(NumOfDiodes == 1) {
                //Standard-Diode
                lcd_eep_string(Diode);    //"Diode: "
                lcd_eep_string(Anode);
                lcd_data(diodes[0].Anode + 49);
                lcd_eep_string(NextK);//";K="
                lcd_data(diodes[0].Cathode + 49);
                Line2();     //2. Zeile
                lcd_eep_string(Uf); //"Uf = "
                lcd_string(itoa(diodes[0].Voltage, outval, 10));
                lcd_eep_string(mV);
                goto end;
        } else if(NumOfDiodes == 2) {
        //Doppeldiode
                if(diodes[0].Anode == diodes[1].Anode) {
                        //Common Anode
                        lcd_eep_string(DualDiode);//Doppeldiode
                        lcd_eep_string(CA); //"CA"
                        Line2(); //2. Zeile
                        lcd_eep_string(Anode);
                        lcd_data(diodes[0].Anode + 49);
                        lcd_eep_string(K1); //";K1="
                        lcd_data(diodes[0].Cathode + 49);
                        lcd_eep_string(K2); //";K2="
                        lcd_data(diodes[1].Cathode + 49);
                        goto end;
                } else if(diodes[0].Cathode == diodes[1].Cathode) {
                        //Common Cathode
                        lcd_eep_string(DualDiode);//Doppeldiode
                        lcd_eep_string(CC); //"CC"
                        Line2(); //2. Zeile
                        lcd_eep_string(K);  //"K="
                        lcd_data(diodes[0].Cathode + 49);
                        lcd_eep_string(A1);       //";A1="
                        lcd_data(diodes[0].Anode + 49);
                        lcd_eep_string(A2);       //";A2="
```

```
                        lcd_data(diodes[1].Anode + 49);
                        goto end;
                    } else if ((diodes[0].Cathode == diodes[1].Anode) &&
(diodes[1].Cathode == diodes[0].Anode)) {
                        //Antiparallel
                        lcd_eep_string(TwoDiodes);//2 Dioden
                        Line2(); //2. Zeile
                        lcd_eep_string(Antiparallel);
      //Antiparallel
                        goto end;
                    }
             } else if(NumOfDiodes == 3) {
                    //Serienschaltung aus 2 Dioden; wird als 3 Dioden
erkannt
                    b = 3;
                    c = 3;
                    /* Überprüfen auf eine für eine Serienschaltung von
2 Dioden mögliche Konstellation
                        Dafür müssen 2 der Kathoden und 2 der Anoden
übereinstimmen.
                        Das kommmt daher, dass die Dioden als 2
Einzeldioden und ZUSÄTZLICH als eine "große" Diode erkannt werden.
                    */
                    if((diodes[0].Anode == diodes[1].Anode) ||
(diodes[0].Anode == diodes[2].Anode)) b = diodes[0].Anode;
                    if(diodes[1].Anode == diodes[2].Anode) b =
diodes[1].Anode;

                    if((diodes[0].Cathode == diodes[1].Cathode) ||
(diodes[0].Cathode == diodes[2].Cathode)) c = diodes[0].Cathode;
                    if(diodes[1].Cathode == diodes[2].Cathode) c =
diodes[1].Cathode;
                    if((b<3) && (c<3)) {
                        lcd_eep_string(TwoDiodes);//2 Dioden
                        Line2(); //2. Zeile
                        lcd_eep_string(InSeries); //"in Serie A="
                        lcd_data(b + 49);
                        lcd_eep_string(NextK);
                        lcd_data(c + 49);
                        goto end;
                    }
             }
      } else if (PartFound == PART_TRANSISTOR) {
             if(PartReady == 0) {      //Wenn 2. Prüfung nie gemacht,
z.B. bei Transistor mit Schutzdiode
                    hfe[1] = hfe[0];
                    uBE[1] = uBE[0];
             }
             if((hfe[0]>hfe[1])) {     //Wenn der Verstärkungsfaktor
beim ersten Test höher war: C und E vertauschen!
                    hfe[1] = hfe[0];
                    uBE[1] = uBE[0];
                    tmp = c;
                    c = e;
```

```
                e = tmp;
            }

            if(PartMode == PART_MODE_NPN) {
                lcd_eep_string(NPN);
            } else {
                lcd_eep_string(PNP);
            }
            lcd_eep_string(bstr);     //B=
            lcd_data(b + 49);
            lcd_eep_string(cstr);     //;C=
            lcd_data(c + 49);
            lcd_eep_string(estr);     //;E=
            lcd_data(e + 49);
            Line2(); //2. Zeile
            //Verstärkungsfaktor berechnen
            //hFE = Emitterstrom / Basisstrom
            lhfe = hfe[1];

            #ifdef UseM8
                lhfe *= (((unsigned long)rhval * 100) / (unsigned
long)rlval); //Verhältnis von High- zu Low-Widerstand
            #else
                lhfe *= M48_RH_RL_RATIO;
            #endif
            if(uBE[1]<11) uBE[1] = 11;
            lhfe /= uBE[1];
            hfe[1] = (unsigned int) lhfe;
            lcd_eep_string(hfestr);   //"hFE="
            lcd_string(utoa(hfe[1], outval, 10));
            SetCursor(2,7);                  //Cursor auf Zeile 2,
Zeichen 7
            if(NumOfDiodes > 2) {     //Transistor mit Schutzdiode
                lcd_data(LCD_CHAR_DIODE); //Diode anzeigen
            } else {
                #ifdef UseM8
                    lcd_data(' ');
                #endif
            }
            #ifdef UseM8
                for(c=0;c<NumOfDiodes;c++) {
                    if(((diodes[c].Cathode == e) &&
(diodes[c].Anode == b) && (PartMode == PART_MODE_NPN)) ||
((diodes[c].Anode == e) && (diodes[c].Cathode == b) && (PartMode ==
PART_MODE_PNP))) {
                        lcd_eep_string(Uf); //"Uf="
                        lcd_string(itoa(diodes[c].Voltage,
outval, 10));
                        lcd_data('m');
                        goto end;
                    }
                }
            #endif
            goto end;
```

```
	} else if (PartFound == PART_FET) {    //JFET oder MOSFET
		if(PartMode&1) {    //N-Kanal
			lcd_data('N');
		} else {
			lcd_data('P');      //P-Kanal
		}
		if((PartMode==PART_MODE_N_D_MOS) ||
(PartMode==PART_MODE_P_D_MOS)) {
			lcd_eep_string(dmode);    //"-D"
			lcd_eep_string(mosfet);   //"-MOS"
		} else {
			if((PartMode==PART_MODE_N_JFET) ||
(PartMode==PART_MODE_P_JFET)) {
				lcd_eep_string(jfet);     //"-JFET"
			} else {
				lcd_eep_string(emode);    //"-E"
				lcd_eep_string(mosfet);   //"-MOS"
			}
		}
		#ifdef UseM8 //Gatekapazität
			if(PartMode < 3) {  //Anreicherungs-MOSFET
				lcd_eep_string(GateCap);  //" C="
				ReadCapacity(b,e);  //Messung
				hfe[0] = (unsigned int)cv;
				if(hfe[0]>2) hfe[0] -= 3;
				utoa(hfe[0], outval2, 10);

				tmpval = strlen(outval2);
				tmpval2 = tmpval;
				if(tmpval>4) tmpval = 4;  //bei Kapazität
>100nF letze Nachkommastelle nicht mehr angeben (passt sonst nicht auf
das LCD)
				lcd_show_format_cap(outval2, tmpval,
tmpval2);
				lcd_data('n');
			}
		#endif
		Line2(); //2. Zeile
		lcd_eep_string(gds);      //"GDS="
		lcd_data(b + 49);
		lcd_data(c + 49);
		lcd_data(e + 49);
		if((NumOfDiodes > 0) && (PartMode < 3)) {     //MOSFET mit
Schutzdiode; gibt es nur bei Anreicherungs-FETs
			lcd_data(LCD_CHAR_DIODE); //Diode anzeigen
		} else {
			lcd_data(' ');      //Leerzeichen
		}
		if(PartMode < 3) {  //Anreicherungs-MOSFET
			gthvoltage=(gthvoltage/8);
			lcd_eep_string(vt);
			lcd_string(utoa(gthvoltage, outval, 10));
	//Gate-Schwellspannung, wurde zuvor ermittelt
			lcd_data('m');
```

```
		}
		goto end;
	} else if (PartFound == PART_THYRISTOR) {
		lcd_eep_string(Thyristor);//"Thyristor"
		Line2(); //2. Zeile
		lcd_eep_string(GAK);	//"GAK="
		lcd_data(b + 49);
		lcd_data(c + 49);
		lcd_data(e + 49);
		goto end;
	} else if (PartFound == PART_TRIAC) {
		lcd_eep_string(Triac);	//"Triac"
		Line2(); //2. Zeile
		lcd_eep_string(Gate);
		lcd_data(b + 49);
		lcd_eep_string(A1);	//";A1="
		lcd_data(e + 49);
		lcd_eep_string(A2);	//";A2="
		lcd_data(c + 49);
		goto end;
	#ifdef UseM8 //Widerstandsmessung nur mit Mega8 verfügbar
		} else if(PartFound == PART_RESISTOR) {
			lcd_eep_string(Resistor); //"Widerstand: "
			lcd_data(ra + 49);  //Pin-Angaben
			lcd_data('-');
			lcd_data(rb + 49);
			Line2(); //2. Zeile
			if(rv[0]>512) {		//Überprüfen, wie weit die
an den Testwiderständen anliegenden Spannungen von 512 abweichen
				hfe[0] = (rv[0] - 512);
			} else {
				hfe[0] = (512 - rv[0]);
			}
			if(rv[1]>512) {
				hfe[1] = (rv[1] - 512);
			} else {
				hfe[1] = (512 - rv[1]);
			}
			if(hfe[0] > hfe[1])  {
				radcmax[0] = radcmax[1];
				rv[0] = rv[1];	//Ergebnis verwenden,
welches näher an 512 liegt (bessere Genauigkeit)
				rv[1] = rhval;	//470k-Testwiderstand
			} else {
				rv[1] = rlval;	//680R-Testwiderstand
			}
			if(rv[0]==0) rv[0] = 1;
			lhfe = (unsigned long)((unsigned long)((unsigned
long)rv[1] * (unsigned long)rv[0]) / (unsigned long)((unsigned
long)radcmax[0] - (unsigned long)rv[0]));	//Widerstand berechnen
			ultoa(lhfe,outval,10);

			if(rv[1]==rhval) {  //470k-Widerstand?
```

```
                        ra = strlen(outval);      //Nötig, um Komma
anzuzeigen
                        for(rb=0;rb<ra;rb++) {
                              lcd_data(outval[rb]);
                              if(rb==(ra-2)) lcd_data('.');
      //Komma
                        }
                        lcd_data ('k'); //Kilo-Ohm, falls 470k-
Widerstand verwendet
                  } else {
                        lcd_string(outval);
                  }
                  lcd_data(LCD_CHAR_OMEGA); //Omega für Ohm
                  goto end;

            } else if(PartFound == PART_CAPACITOR) {
      //Kapazitätsmessung auch nur auf Mega8 verfügbar
                  lcd_eep_string(Capacitor);
                  lcd_data(ca + 49);  //Pin-Angaben
                  lcd_data('-');
                  lcd_data(cb + 49);
                  Line2(); //2. Zeile
                  tmpval2 = 'n';
                  if(cv > 99999) {    //ab 1µF
                        cv /= 1000;
                        tmpval2 = LCD_CHAR_U;
                  }
                  ultoa(cv, outval, 10);
                  tmpval = strlen(outval);
                  lcd_show_format_cap(outval, tmpval, tmpval);
                  lcd_data(tmpval2);
                  lcd_data('F');
                  goto end;
      #endif
      }
      #ifdef UseM8 //Unterscheidung, ob Dioden gefunden wurden oder
nicht nur auf Mega8
            if(NumOfDiodes == 0) {
                  //Keine Dioden gefunden
                  lcd_eep_string(TestFailed1); //"Kein,unbek. oder"
                  Line2(); //2. Zeile
                  lcd_eep_string(TestFailed2); //"defektes "
                  lcd_eep_string(Bauteil);
            } else {
                  lcd_eep_string(Bauteil);
                  lcd_eep_string(Unknown); //" unbek."
                  Line2(); //2. Zeile
                  lcd_eep_string(OrBroken); //"oder defekt"
                  lcd_data(NumOfDiodes + 48);
                  lcd_data(LCD_CHAR_DIODE);
            }
      #else  //auf Mega48 keine Anzeige der evtl. gefundenen Dioden
            lcd_eep_string(TestFailed1); //"Kein,unbek. oder"
            Line2(); //2. Zeile
```

```
			lcd_eep_string(TestFailed2); //"defektes "
			lcd_eep_string(Bauteil);
		#endif
		end:
		while(!(ON_PIN_REG & (1<<RST_PIN)));		//warten ,bis
Taster losgelassen
		_delay_ms(200);
		for(hfe[0] = 0;hfe[0]<10000;hfe[0]++) {
			if(!(ON_PIN_REG & (1<<RST_PIN))) {
				/*Wenn der Taster wieder gedrückt wurde...
				wieder zum Anfang springen und neuen Test
durchführen
				*/
				goto start;
			}
			wdt_reset();
			_delay_ms(1);
		}
		ON_PORT &= ~(1<<ON_PIN);  //Abschalten
		wdt_disable();		//Watchdog aus
		//Endlosschleife
		while(1) {
			if(!(ON_PIN_REG & (1<<RST_PIN))) {
				/* wird nur erreicht,
				wenn die automatische Abschaltung nicht eingebaut
wurde */
				goto start;
			}
		}
		return 0;
}

void CheckPins(uint8_t HighPin, uint8_t LowPin, uint8_t TristatePin) {
		/*
		Funktion zum Testen der Eigenschaften des Bauteils bei der
angegebenen Pin-Belegung
		Parameter:
		HighPin: Pin, der anfangs auf positives Potenzial gelegt wird
		LowPin: Pin, der anfangs auf negatives Potenzial gelegt wird
		TristatePin: Pin, der anfangs offen gelassen wird

		Im Testverlauf wird TristatePin natürlich auch positiv oder
negativ geschaltet.
		*/
		unsigned int adcv[6];
		uint8_t tmpval, tmpval2;
		/*
			HighPin wird fest auf Vcc gelegt
			LowPin wird über R_L auf GND gelegt
			TristatePin wird hochohmig geschaltet, dafür ist keine
Aktion nötig
		*/
		wdt_reset();
		//Pins setzen
```

```
	tmpval = (LowPin * 2);                  //nötig wegen der
Anordnung der Widerstände
	R_DDR = (1<<tmpval);                    //Low-Pin auf Ausgang und
über R_L auf Masse
	R_PORT = 0;
	ADC_DDR = (1<<HighPin) | (1<<TxD);                  //High-Pin
auf Ausgang
	ADC_PORT = (1<<HighPin) | TXD_VAL;            //High-Pin fest auf
Vcc
	_delay_ms(5);
	//Bei manchen MOSFETs muss das Gate (TristatePin) zuerst entladen
werden
	//N-Kanal:
	DischargePin(TristatePin,0);
	//Spannung am Low-Pin ermitteln
	adcv[0] = ReadADC(LowPin);
	if(adcv[0] < 200) goto next;     //Sperrt das Bauteil jetzt?
	//sonst: Entladen für P-Kanal (Gate auf Plus)
	DischargePin(TristatePin,1);
	//Spannung am Low-Pin ermitteln
	adcv[0] = ReadADC(LowPin);

	next:

	if(adcv[0] > 19) {//Bauteil leitet ohne Steuerstrom etwas
		//Test auf N-JFET oder selbstleitenden N-MOSFET
		R_DDR |= (2<<(TristatePin*2));   //Tristate-Pin (vermutetes
Gate) über R_H auf Masse
		_delay_ms(20);
		adcv[1] = ReadADC(LowPin);       //Spannung am vermuteten
Source messen
		R_PORT |= (2<<(TristatePin*2));  //Tristate-Pin (vermutetes
Gate) über R_H auf Plus
		_delay_ms(20);
		adcv[2] = ReadADC(LowPin);       //Spannung am vermuteten
Source erneut messen
		//Wenn es sich um einen selbstleitenden MOSFET oder JFET
handelt, müsste adcv[1] > adcv[0] sein
		if(adcv[2]>(adcv[1]+100)) {
			//Spannung am Gate messen, zur Unterscheidung
zwischen MOSFET und JFET
			ADC_PORT = TXD_VAL;
			ADC_DDR = (1<<LowPin) | (1<<TxD);      //Low-Pin
fest auf Masse
			tmpval = (HighPin * 2);          //nötig wegen der
Anordnung der Widerstände
			R_DDR |= (1<<tmpval);                  //High-Pin
auf Ausgang
			R_PORT |= (1<<tmpval);                 //High-Pin
über R_L auf Vcc
			_delay_ms(20);
			adcv[2] = ReadADC(TristatePin);        //Spannung am
vermuteten Gate messen
			if(adcv[2]>800) {   //MOSFET
```

```
                        PartFound = PART_FET;                   //N-
Kanal-MOSFET
                        PartMode = PART_MODE_N_D_MOS;    //Verarmungs-
MOSFET
                  } else {     //JFET (pn-Übergang zwischen G und S
leitet)
                        PartFound = PART_FET;                   //N-
Kanal-JFET
                        PartMode = PART_MODE_N_JFET;
                  }
                  b = TristatePin;
                  c = HighPin;
                  e = LowPin;
            }
            ADC_PORT = TXD_VAL;

            //Test auf P-JFET oder selbstleitenden P-MOSFET
            ADC_DDR = (1<<LowPin) | (1<<TxD);      //Low-Pin
(vermuteter Drain) fest auf Masse, Tristate-Pin (vermutetes Gate) ist
noch über R_H auf Plus
            tmpval = (HighPin * 2);                //nötig wegen der
Anordnung der Widerstände
            R_DDR |= (1<<tmpval);                  //High-Pin auf
Ausgang
            R_PORT |= (1<<tmpval);                 //High-Pin über R_L
auf Vcc
            _delay_ms(20);
            adcv[1] = ReadADC(HighPin);            //Spannung am
vermuteten Source messen
            R_PORT = (1<<tmpval);                  //Tristate-Pin
(vermutetes Gate) über R_H auf Masse
            _delay_ms(20);
            adcv[2] = ReadADC(HighPin);            //Spannung am
vermuteten Source erneut messen
            //Wenn es sich um einen selbstleitenden P-MOSFET oder P-
JFET handelt, müsste adcv[0] > adcv[1] sein
            if(adcv[1]>(adcv[2]+100)) {
                  //Spannung am Gate messen, zur Unterscheidung
zwischen MOSFET und JFET
                  ADC_PORT = (1<<HighPin) | TXD_VAL;     //High-Pin
fest auf Plus
                  ADC_DDR = (1<<HighPin) | (1<<TxD);
      //High-Pin auf Ausgang
                  _delay_ms(20);
                  adcv[2] = ReadADC(TristatePin);        //Spannung am
vermuteten Gate messen
                  if(adcv[2]<200) {   //MOSFET
                        PartFound = PART_FET;                   //P-
Kanal-MOSFET
                        PartMode = PART_MODE_P_D_MOS;    //Verarmungs-
MOSFET
                  } else {     //JFET (pn-Übergang zwischen G und S
leitet)
```

```
                        PartFound = PART_FET;                   //P-
Kanal-JFET
                        PartMode = PART_MODE_P_JFET;
                }
                b = TristatePin;
                c = LowPin;
                e = HighPin;
            }
        }
        //Pins erneut setzen
        tmpval = (LowPin * 2);                  //nötig wegen der
Anordnung der Widerstände
        R_DDR = (1<<tmpval);                    //Low-Pin auf Ausgang und
über R_L auf Masse
        R_PORT = 0;
        ADC_DDR = (1<<HighPin) | (1<<TxD);                  //High-Pin
auf Ausgang
        ADC_PORT = (1<<HighPin) | TXD_VAL;            //High-Pin fest auf
Vcc
        _delay_ms(5);

        if(adcv[0] < 200) { //Wenn das Bauteil keinen Durchgang zwischen
HighPin und LowPin hat
            //Test auf pnp
            tmpval2 = (TristatePin * 2);            //nötig wegen der
Anordnung der Widerstände
            R_DDR |= (1<<tmpval2);                  //Tristate-Pin über
R_L auf Masse, zum Test auf pnp
            _delay_ms(2);
            adcv[1] = ReadADC(LowPin);       //Spannung messen
            if(adcv[1] > 700) {
                //Bauteil leitet => pnp-Transistor o.ä.
                //Verstärkungsfaktor in beide Richtungen messen
                R_DDR = (1<<tmpval);             //Tristate-Pin
(Basis) hochohmig
                tmpval2++;
                R_DDR |= (1<<tmpval2);           //Tristate-Pin
(Basis) über R_H auf Masse

                _delay_ms(10);
                adcv[1] = ReadADC(LowPin);       //Spannung am Low-
Pin (vermuteter Kollektor) messen
                adcv[2] = ReadADC(TristatePin);  //Basisspannung
messen
                //Prüfen, ob Test schon mal gelaufen
                if((PartFound == PART_TRANSISTOR) || (PartFound ==
PART_FET)) PartReady = 1;
                hfe[PartReady] = adcv[1];
                uBE[PartReady] = adcv[2];

                if(PartFound != PART_THYRISTOR) {
                    if(adcv[2] > 200) {
                        PartFound = PART_TRANSISTOR;     //PNP-
Transistor gefunden (Basis wird "nach oben" gezogen)
```

```
                                    PartMode = PART_MODE_PNP;
                              } else {
                                    if(adcv[0] < 20) {  //Durchlassspannung
im gesperrten Zustand gering genug? (sonst werden D-Mode-FETs
fälschlicherweise als E-Mode erkannt)
                                          PartFound = PART_FET;
      //P-Kanal-MOSFET gefunden (Basis/Gate wird NICHT "nach oben"
gezogen)
                                          PartMode = PART_MODE_P_E_MOS;
                                          //Messung der Gate-
Schwellspannung
                                          tmpval = (1<<LowPin);
                                          tmpval2 = R_DDR;
                                          ADMUX = TristatePin |
(1<<REFS0);
                                          gthvoltage = 0;
                                          for(b=0;b<13;b++) {
                                                wdt_reset();

      DischargePin(TristatePin,1);
                                                while (!(ADC_PIN&tmpval));
// Warten, bis der MOSFET schaltet und Drain auf high geht
                                                R_DDR = 0;
                                                ADCSRA |= (1<<ADSC);
                                                while (ADCSRA&(1<<ADSC));
                                                gthvoltage += (1023 -
ADCW);
                                                R_DDR = tmpval2;
                                          }
                                          gthvoltage *= 3;    //Umrechnung
in mV, zusammen mit der Division durch 8 (bei der LCD-Anzeige)
                                    }
                              }
                              b = TristatePin;
                              c = LowPin;
                              e = HighPin;
                        }
                  }

                  //Tristate (vermutete Basis) auf Plus, zum Test auf npn
                  ADC_PORT = TXD_VAL;                           //Low-Pin
fest auf Masse
                  tmpval = (TristatePin * 2);            //nötig wegen der
Anordnung der Widerstände
                  tmpval2 = (HighPin * 2);         //nötig wegen der
Anordnung der Widerstände
                  R_DDR = (1<<tmpval) | (1<<tmpval2);
      //High-Pin und Tristate-Pin auf Ausgang
                  R_PORT = (1<<tmpval) | (1<<tmpval2);          //High-Pin
und Tristate-Pin über R_L auf Vcc
                  ADC_DDR = (1<<LowPin) | (1<<TxD);                   //Low-
Pin auf Ausgang
                  _delay_ms(10);
```

```
            adcv[1] = ReadADC(HighPin);            //Spannung am High-
Pin messen
            if(adcv[1] < 500) {
                  if(PartReady==1) goto testend;
                  //Bauteil leitet => npn-Transistor o.ä.

                  //Test auf Thyristor:
                  //Gate entladen

                  R_PORT = (1<<tmpval2);                //Tristate-
Pin (Gate) über R_L auf Masse
                  _delay_ms(10);
                  R_DDR = (1<<tmpval2);                 //Tristate-
Pin (Gate) hochohmig
                  //Test auf Thyristor
                  _delay_ms(5);
                  adcv[3] = ReadADC(HighPin);           //Spannung am
High-Pin (vermutete Anode) erneut messen

                  R_PORT = 0;
      //High-Pin (vermutete Anode) auf Masse
                  _delay_ms(5);
                  R_PORT = (1<<tmpval2);                //High-Pin
(vermutete Anode) wieder auf Plus
                  _delay_ms(5);
                  adcv[2] = ReadADC(HighPin);           //Spannung am
High-Pin (vermutete Anode) erneut messen
                  if((adcv[3] < 500) && (adcv[2] > 900)) {     //Nach
Abschalten des Haltestroms muss der Thyristor sperren
                        //war vor Abschaltung des Triggerstroms
geschaltet und ist immer noch geschaltet obwohl Gate aus => Thyristor
                        PartFound = PART_THYRISTOR;
                        //Test auf Triac
                        R_DDR = 0;
                        R_PORT = 0;
                        ADC_PORT = (1<<LowPin) | TXD_VAL;     //Low-
Pin fest auf Plus
                        _delay_ms(5);
                        R_DDR = (1<<tmpval2);     //HighPin über R_L
auf Masse
                        _delay_ms(5);
                        if(ReadADC(HighPin) > 50) goto savenresult;
      //Spannung am High-Pin (vermuteter A2) messen; falls zu hoch:
Bauteil leitet jetzt => kein Triac
                        R_DDR |= (1<<tmpval);     //Gate auch über
R_L auf Masse => Triac müsste zünden
                        _delay_ms(5);
                        if(ReadADC(TristatePin) < 200) goto
savenresult; //Spannung am Tristate-Pin (vermutetes Gate) messen;
Abbruch falls Spannung zu gering
                        if(ReadADC(HighPin) < 150) goto savenresult;
//Bauteil leitet jetzt nicht => kein Triac => Abbruch
                        R_DDR = (1<<tmpval2);     //TristatePin
(Gate) wieder hochohmig
```

```
                        _delay_ms(5);
                        if(ReadADC(HighPin) < 150) goto savenresult;
//Bauteil leitet nach Abschalten des Gatestroms nicht mehr=> kein Triac
=> Abbruch
                        R_PORT = (1<<tmpval2);    //HighPin über R_L
auf Plus => Haltestrom aus
                        _delay_ms(5);
                        R_PORT = 0;                       //HighPin
wieder über R_L auf Masse; Triac müsste jetzt sperren
                        _delay_ms(5);
                        if(ReadADC(HighPin) > 50) goto savenresult;
      //Spannung am High-Pin (vermuteter A2) messen; falls zu hoch:
Bauteil leitet jetzt => kein Triac
                        PartFound = PART_TRIAC;
                        PartReady = 1;
                        goto savenresult;
                  }
                  //Test auf Transistor oder MOSFET
                  tmpval++;
                  R_DDR |= (1<<tmpval);           //Tristate-Pin
(Basis) auf Ausgang
                  R_PORT |= (1<<tmpval);          //Tristate-Pin
(Basis) über R_H auf Plus
                  _delay_ms(50);
                  adcv[1] = ReadADC(HighPin);           //Spannung am
High-Pin (vermuteter Kollektor) messen
                  adcv[2] = ReadADC(TristatePin);  //Basisspannung
messen

                  if((PartFound == PART_TRANSISTOR) || (PartFound ==
PART_FET)) PartReady = 1; //prüfen, ob Test schon mal gelaufen
                  hfe[PartReady] = 1023 - adcv[1];
                  uBE[PartReady] = 1023 - adcv[2];
                  if(adcv[2] < 500) {
                        PartFound = PART_TRANSISTOR;     //NPN-
Transistor gefunden (Basis wird "nach unten" gezogen)
                        PartMode = PART_MODE_NPN;
                  } else {
                        if(adcv[0] < 20) {  //Durchlassspannung im
gesperrten Zustand gering genug? (sonst werden D-Mode-FETs
fälschlicherweise als E-Mode erkannt)
                              PartFound = PART_FET;
      //N-Kanal-MOSFET gefunden (Basis/Gate wird NICHT "nach unten"
gezogen)
                              PartMode = PART_MODE_N_E_MOS;
                              //Gate-Schwellspannung messen
                              tmpval2 = R_DDR;
                              tmpval=(1<<HighPin);
                              ADMUX = TristatePin | (1<<REFS0);
                              gthvoltage = 0;
                              for(b=0;b<13;b++) {
                                    wdt_reset();
                                    DischargePin(TristatePin,0);
```

```
                                        while ((ADC_PIN&tmpval));  //
Warten, bis der MOSFET schaltet und Drain auf low geht
                                        R_DDR = 0;
                                        R_PORT = 0;
                                        ADCSRA |= (1<<ADSC);
                                        while (ADCSRA&(1<<ADSC));
                                        gthvoltage += ADCW;
                                        R_DDR = tmpval2;
                                        R_PORT = tmpval2;
                                }
                                gthvoltage *= 3;    //Umrechnung in mV,
zusammen mit der Division durch 8 (bei der LCD-Anzeige)
                        }
                }
                savenresult:
                b = TristatePin;
                c = HighPin;
                e = LowPin;
        }
        ADC_DDR = (1<<TxD);
        ADC_PORT = TXD_VAL;
        //Fertig
    } else {     //Durchgang
        //Test auf Diode
        tmpval2 = (2<<(2*HighPin));      //R_H
        tmpval = (1<<(2*HighPin));//R_L
        ADC_PORT = TXD_VAL;
        ADC_DDR = (1<<LowPin) | (1<<TxD);      //Low-Pin fest auf
Masse, High-Pin ist noch über R_L auf Vcc
        DischargePin(TristatePin,1);     //Entladen für P-Kanal-
MOSFET
        _delay_ms(5);
        adcv[0] = ReadADC(HighPin) - ReadADC(LowPin);
        R_DDR = tmpval2;    //High-Pin über R_H auf Plus
        R_PORT = tmpval2;
        _delay_ms(5);
        adcv[2] = ReadADC(HighPin) - ReadADC(LowPin);
        R_DDR = tmpval;     //High-Pin über R_L auf Plus
        R_PORT = tmpval;
        DischargePin(TristatePin,0);     //Entladen für N-Kanal-
MOSFET
        _delay_ms(5);
        adcv[1] = ReadADC(HighPin) - ReadADC(LowPin);
        R_DDR = tmpval2;    //High-Pin über R_H  auf Plus
        R_PORT = tmpval2;
        _delay_ms(5);
        adcv[3] = ReadADC(HighPin) - ReadADC(LowPin);
        /*Ohne das Entladen kann es zu Falscherkennungen kommen, da
das Gate eines MOSFETs noch geladen sein kann.
                Die zusätzliche Messung mit dem "großen" Widerstand
R_H wird durchgeführt, um antiparallele Dioden von
                Widerständen unterscheiden zu können.
                Eine Diode hat eine vom Durchlassstrom relativ
unabhängige Durchlassspg.
```

```
                  Bei einem Widerstand ändert sich der Spannungsabfall
stark (linear) mit dem Strom.
            */
            if(adcv[0] > adcv[1]) {
                  adcv[1] = adcv[0];  //der höhere Wert gewinnt
                  adcv[3] = adcv[2];
            }

            if((adcv[1] > 30) && (adcv[1] < 950)) { //Spannung liegt
über 0,15V und unter 4,64V => Ok
                  if((PartFound == PART_NONE) || (PartFound ==
PART_RESISTOR)) PartFound = PART_DIODE;//Diode nur angeben, wenn noch
kein anderes Bauteil gefunden wurde. Sonst gäbe es Probleme bei
Transistoren mit Schutzdiode
                  diodes[NumOfDiodes].Anode = HighPin;
                  diodes[NumOfDiodes].Cathode = LowPin;
                  diodes[NumOfDiodes].Voltage = (adcv[1]*54/11);
      // ca. mit 4,9 multiplizieren, um aus dem ADC-Wert die Spannung
in Millivolt zu erhalten
                  NumOfDiodes++;
                  for(uint8_t i=0;i<NumOfDiodes;i++) {
                        if((diodes[i].Anode == LowPin) &&
(diodes[i].Cathode == HighPin)) {      //zwei antiparallele Dioden:
Defekt oder Duo-LED
                              if((adcv[3]*64) < (adcv[1] / 5)) {
      //Durchlassspannung fällt bei geringerem Teststrom stark ab =>
Defekt
                                    if(i<NumOfDiodes) {
                                          for(uint8_t
j=i;j<(NumOfDiodes-1);j++) {
                                                diodes[j].Anode =
diodes[j+1].Anode;
                                                diodes[j].Cathode =
diodes[j+1].Cathode;
                                                diodes[j].Voltage =
diodes[j+1].Voltage;
                                          }
                                    }
                                    NumOfDiodes -= 2;
                              }
                        }
                  }
            }
      }

      #ifdef UseM8 //Widerstandsmessung nur auf dem Mega8 verfügbar
            //Test auf Widerstand
            tmpval2 = (2<<(2*HighPin));      //R_H
            tmpval = (1<<(2*HighPin));//R_L
            ADC_PORT = TXD_VAL;
            ADC_DDR = (1<<LowPin) | (1<<TxD);      //Low-Pin fest auf
Masse
            R_DDR = tmpval;     //High-Pin über R_L auf Plus
            R_PORT = tmpval;
```

```
            adcv[2] = ReadADC(LowPin);
            adcv[0] = ReadADC(HighPin) - adcv[2];
            R_DDR = tmpval2;    //High-Pin über R_H auf Plus
            R_PORT = tmpval2;
            adcv[3] = ReadADC(LowPin);
            adcv[1] = ReadADC(HighPin) - adcv[3];

            //Messung der Spannungsdifferenz zwischen dem Pluspol von
R_L und R_H und Vcc
            tmpval2 = (2<<(2*LowPin));//R_H
            tmpval = (1<<(2*LowPin)); //R_L
            ADC_DDR = (1<<HighPin) | (1<<TxD);            //High-Pin
auf Ausgang
            ADC_PORT = (1<<HighPin) | TXD_VAL;     //High-Pin fest auf
Plus
            R_PORT = 0;
            R_DDR = tmpval;                        //Low-Pin über R_L
auf Masse
            adcv[2] += (1023 - ReadADC(HighPin));
            R_DDR = tmpval2;                       //Low-Pin über R_H
auf Masse
            adcv[3] += (1023 - ReadADC(HighPin));

            if(((adcv[0] - adcv[2]) < 900) && ((adcv[1] - adcv[3]) >
20)) goto testend;  //Spannung fällt bei geringem Teststrom nicht weit
genug ab
            if(((adcv[1] * 32) / 31) < adcv[0]) {  //Abfallende
Spannung fällt bei geringerem Teststrom stark ab und es besteht kein
"Beinahe-Kurzschluss" => Widerstand
                  if((PartFound == PART_DIODE) || (PartFound ==
PART_NONE) || (PartFound == PART_RESISTOR)) {
                        if((tmpPartFound == PART_RESISTOR) && (ra ==
LowPin) && (rb == HighPin)) {
                              /* Das Bauteil wurde schon einmal mit
umgekehrter Polarität getestet.
                              Jetzt beide Ergebnisse miteinander
vergleichen. Wenn sie recht ähnlich sind,
                              handelt es sich (höchstwahrscheinlich)
um einen Widerstand. */
                              if(!((((adcv[0] + 100) * 6) >= ((rv[0]
+ 100) * 5)) && (((rv[0] + 100) * 6) >= ((adcv[0] + 100) * 5)) &&
(((adcv[1] + 100) * 6) >= ((rv[1] + 100) * 5)) && (((rv[1] + 100) * 6)
>= ((adcv[1] + 100) * 5)))) {
                                    //min. 20% Abweichung => kein
Widerstand
                                    tmpPartFound = PART_NONE;
                                    goto testend;
                              }
                              PartFound = PART_RESISTOR;
                        }
                        rv[0] = adcv[0];
                        rv[1] = adcv[1];
```

```
                        radcmax[0] = 1023 - adcv[2];     //Spannung am
Low-Pin ist nicht ganz Null, sondern rund 0,1V (wird aber gemessen). Der
dadurch entstehende Fehler wird hier kompenisert
                        radcmax[1] = 1023 - adcv[3];
                        ra = HighPin;
                        rb = LowPin;
                        tmpPartFound = PART_RESISTOR;
                  }
            }
      #endif
      testend:
      ADC_DDR = (1<<TxD);
      ADC_PORT = TXD_VAL;
      R_DDR = 0;
      R_PORT = 0;
}

#ifdef UseM8 //Kapazitätsmessung nur auf Mega8 verfügbar
void ReadCapacity(uint8_t HighPin, uint8_t LowPin) {
      //Test auf Kondensator (auch nur auf ATMega8 möglich)
      if(PartFound == PART_CAPACITOR) goto end;     //Schon einen
Kondensator gefunden
      unsigned long gcval = 0;
      unsigned int tmpint = 0;
      uint8_t extcnt = 0;
      uint8_t tmpx = 0;

      tmpval2 = (2<<(2*HighPin));      //R_H
      tmpval = (1<<(2*HighPin));//R_L
      ADC_PORT = TXD_VAL;
      R_PORT = 0;
      R_DDR = 0;
      ADC_DDR = (1<<LowPin) | (1<<TxD);      //Low-Pin fest auf Masse
      R_DDR = tmpval2;          //HighPin über R_H auf Masse
      _delay_ms(5);
      adcv[0] = ReadADC(HighPin);
      DischargePin(HighPin,1);
      adcv[2] = ReadADC(HighPin);
      _delay_ms(5);
      adcv[1] = ReadADC(HighPin);
      wdt_reset();
      if((adcv[1] > (adcv[0] + 1)) || (adcv[2] > (adcv[0] + 1))) {
      //Spannung ist gestiegen
            R_DDR = tmpval;                  //High-Pin über R_L auf
Masse
            while(ReadADC(HighPin) > (ReadADC(LowPin) + 10)) {
                  wdt_reset();
                  tmpint++;
                  if(tmpint==0) {
                        extcnt++;
                        if(extcnt == 30) break; //Timeout für
Entladung
                  }
            }
```

```
            tmpint = 0;
            extcnt = 0;
            R_PORT = tmpval;                 //High-Pin über R_L auf
Plus
            _delay_ms(5);
            adcv[2] = ReadADC(HighPin);
            _delay_ms(80);
            adcv[3] = ReadADC(HighPin);
            if((adcv[3] < (adcv[2] + 3)) && (adcv[3] < 850)) goto end;
      //Spannung ist nicht nennenswert gestiegen => Abbruch
            if((NumOfDiodes > 0) && (adcv[3] > 950) && (PartFound !=
PART_FET)) goto end; //höchstwahrscheinlich eine (oder mehrere) Diode(n)
in Sperrrichtung, die sonst fälschlicherweise als Kondensator erkannt
wird
            R_PORT = 0;
            while(ReadADC(HighPin) > (ReadADC(LowPin) + 10)) {
                  wdt_reset();
                  tmpint++;
                  if(tmpint==0) {
                        extcnt++;
                        if(extcnt == 30) break; //Timeout für
Entladung
                  }
            }
            tmpint = 0;
            extcnt = 0;
            ADC_DDR = 7 | (1<<TxD);                              //alle
Pins auf Ausgang und aus Masse
            R_PORT = tmpval;                        // HighPin über R_L
auf Plus
            tmpval=(1<<HighPin);
            _delay_ms(2);
            ADC_DDR = (1<<LowPin) | (1<<TxD);          // Kondensator
über R_L langsam laden
            while (!(ADC_PIN & tmpval)) {  // Warten, bis HighPin auf
High geht; Schleife dauert 7 Zyklen
                  wdt_reset();
                  tmpint++;
                  if(tmpint==0) {
                        extcnt++;
                        if(extcnt == 30) break; //Timeout für Ladung
                  }
            }
            if((extcnt == 0) && (tmpint<256)) {    //Niedrige
Kapazität
                  ADC_DDR = (1<<LowPin) | (1<<TxD);
                  //mit R_H erneut messen
                  R_PORT = 0;
                  tmpint = 0;
                  extcnt = 0;
                  while(ReadADC(HighPin) > (ReadADC(LowPin) + 10)) {
                        wdt_reset();
                        tmpint++;
                        if(tmpint==0) {
```

```
                                extcnt++;
                                if(extcnt == 30) break; //Timeout für
Entladung
                        }
                }
                tmpint = 0;
                extcnt = 0;
                ADC_DDR = 7 | (1<<TxD);
        //alle Pins auf Ausgang
                ADC_PORT = TXD_VAL;                             //alle
Pins fest auf Masse
                R_DDR = tmpval2;                // HighPin über R_H
auf Ausgang
                R_PORT = tmpval2;                       // HighPin
über R_H auf Plus
                _delay_ms(2);
                if(PartFound == PART_FET) {
                        ADC_DDR = (7 & ~tmpval) | (1<<TxD);
// Kondensator über R_H langsam laden, dabei freien Pin (Drain) für
Gate-Kapazitäts-Messung auf Masse
                } else {
                        ADC_DDR = (1<<LowPin) | (1<<TxD);          //
Kondensator über R_H langsam laden
                }
                while (!(ADC_PIN & tmpval)) {  // Warten, bis
HighPin auf High geht; Schleife dauert 7 Zyklen
                        wdt_reset();
                        tmpint++;
                        if(tmpint==0) {
                                extcnt++;
                                if(extcnt == 30) break; //Timeout für
Kapazitätsmessung
                        }
                }
                tmpx = 1;
        }
        if(tmpx) {
                gcval = eeprom_read_word(&H_CAPACITY_FACTOR);
                if((extcnt == 0) && (tmpint < 5)) goto end;
        //Kapazität zu gering
                cv = 1;
        } else {
                gcval = eeprom_read_word(&L_CAPACITY_FACTOR);
                cv = 1000;
        }

        gcval *= (unsigned long)(((unsigned long)extcnt * 65536) +
(unsigned long)tmpint);   //Wert unrechnen und speichern
        gcval /= 100;
        cv *= gcval;

        PartFound = PART_CAPACITOR;      //Kondensator gefunden

        ca = HighPin;
```

```
            cb = LowPin;
            //Kondensator wieder entladen
            tmpint = 0;
            extcnt = 0;
            R_DDR = (1<<(2*HighPin));              //High-Pin über R_L
auf Masse
            R_PORT = 0;
            while(ReadADC(HighPin) > (ReadADC(LowPin) + 10)) {
                  wdt_reset();
                  tmpint++;
                  if(tmpint==0) {
                        extcnt++;
                        if(extcnt == 30) break; //Timeout für
Entladung
                  }
            }
            ADC_DDR = 7 | (1<<TxD);   //komplett entladen
            ADC_PORT = 7 | TXD_VAL;
            _delay_ms(10);
            //Fertig
      }
      end:
      ADC_DDR =  (1<<TxD);
      ADC_PORT = TXD_VAL;
      R_DDR = 0;
      R_PORT = 0;
}
#endif


unsigned int ReadADC(uint8_t mux) {
      //ADC-Wert des angegebenen Kanals auslesen und als unsigned int
zurückgegen
      unsigned int adcx = 0;
      ADMUX = mux | (1<<REFS0);
      for(uint8_t j=0;j<20;j++) {      //20 Messungen; für bessere
Genauigkeit
            ADCSRA |= (1<<ADSC);
            while (ADCSRA&(1<<ADSC));
            adcx += ADCW;
      }
      adcx /= 20;
      return adcx;
}


void DischargePin(uint8_t PinToDischarge, uint8_t DischargeDirection) {
      /*Anschluss eines Bauelementes kurz(10ms) auf ein bestimmtes
Potenzial legen
            Diese Funktion ist zum Entladen von MOSFET-Gates
vorgesehen, um Schutzdioden u.ä. in MOSFETs erkennen zu können
            Parameter:
            PinToDischarge: zu entladender Pin
```

```
            DischargeDirection: 0 = gegen Masse (N-Kanal-FET), 1= gegen
Plus(P-Kanal-FET)
      */
      uint8_t tmpval;
      tmpval = (PinToDischarge * 2);         //nötig wegen der
Anordnung der Widerstände

      if(DischargeDirection) R_PORT |= (1<<tmpval);              //R_L
aus
      R_DDR |= (1<<tmpval);                  //Pin auf Ausgang und über
R_L auf Masse
      _delay_ms(10);
      R_DDR &= ~(1<<tmpval);                 //Pin wieder auf Eingang
      if(DischargeDirection) R_PORT &= ~(1<<tmpval);
      //R_L aus
}

#ifdef UseM8
void lcd_show_format_cap(char outval[], uint8_t strlength, uint8_t
CommaPos) {
      if(strlength < 3) {
            if(strlength==1) {
                  lcd_string("0.");
                  lcd_data('0');
                  lcd_data(outval[0]);
            } else {
                  lcd_string("0.");
                  lcd_data(outval[0]);
                  lcd_data(outval[1]);
            }
      } else {
            for(PartReady=0;PartReady<strlength;PartReady++) {
                  if((PartReady + 2) == CommaPos) lcd_data('.');
                  lcd_data(outval[PartReady]);
            }
      }
}
#endif
```

KEPUTUSAN
DEKAN FAKULTAS TEKNIK
UNIVERSITAS NEGERI SEMARANG

Nomor : 510 /FT – UNNES/2014
Tentang
PENETAPAN DOSEN PEMBIMBING TUGAS AKHIR SEMESTER GENAP
TAHUN AKADEMIK 2013/2014

Menimbang : Bahwa untuk memperlancar mahasiswa Jurusan Teknik Elektro/Prodi Teknik Elektro DIII Fakultas Teknik Universitas Negeri Semarang membuat Tugas Akhir, maka perlu menetapkan Dosen-dosen Jurusan Teknik Elektro/Prodi Teknik Elektro DIII Fakultas Teknik UNNES untuk menjadi pembimbing.

Mengingat :

1. Undang-undang No. 20 Tahun 2003 tentang Sistem Pendidikan Nasional (Tambahan Lembaran Negara RI No.4301, penjelasan atas Lembaran Negara RI Tahun 2003, Nomor 78);
2. SK. Rektor UNNES No. 164/O/2004 tentang Pedoman penyusunan Skripsi/Tugas Akhir Mahasiswa Diploma III UNNES;
3. SK Rektor UNNES No. 162/O/2004 tenteng penyelenggaraan Pendidikan UNNES;
4. SK Rektor Universitas Negeri Semarang Nomor. 362/P/2011, tanggal 24 Oktober 2011 tentang Pemberhentian dan Pengangkatan Dekan Fakultas Teknik Universitas Negeri Semarang.

Memperhatikan : Usul Ketua Jurusan Teknik Elektro Tanggal 20 Maret 2014

M E M U T U S K A N

Menetapkan :

PERTAMA : Menunjuk dan menugaskan kepada :

1. Nama : Drs. Rafael Sri Wiyardi, M.T.
   NIP : 195011101979031001
   Pangkat/Golongan : Pembina Tk. I, IV/b
   Jabatan Akademik : Lektor Kepala
   Sebagai Pembimbing

Untuk membimbing mahasiswa penyusun Tugas Akhir :

Nama : Nanda Puji Arianto
NIM : 5311311009
Prodi : D3 Teknik Elektro
Judul : Multitester Elektronik Berbasis Mikrokontroler Atmega 8

KEDUA : Keputusan ini mulai berlaku sejak tanggal ditetapkan.

DITETAPKAN DI : SEMARANG
PADA TANGGAL : 20 Maret 2014
DEKAN

Drs. Muhammad Harlanu, MPd.
NIP. 196602151991021001

Tembusan :
1. Pembantu Dekan Bidang Akademik
2. Ketua Jurusan TE
3. Dosen Pembimbing
4. Pertinggal

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS NEGERI SEMARANG
FAKULTAS TEKNIK
Gedung E1Kampus Sekaran Gunungpati Semarang 50229
Telepon/Fax (024) 8508101 – 8508009
Laman : http://www.ft.unnes.ac.id, email: ft_unnes@yahoo.com

Nomor : 6121 /UN.37.1.5/DT/2015
Lampiran : -
Hal : Surat Tugas Panitia Ujian Diploma

Dengan ini kami tetapkan bahwa ujian Diploma Fakultas Teknik UNNES untuk Jurusan Teknik Elektro adalah sebagai berikut :

I.Susunan Panitia Ujian :

a. Ketua : Tatyantoro Andrasto, ST,MT
b. Sekretaris : Riana Defi Mahadji Putri, ST,MT
c. Pembimbing Utama : Drs. Rafael Sri Wiyardi, MT
d. Penguji : 1. Drs. Agus Purwanto
2. Drs. Rafael Sri Wiyardi, MT

II.Calon yang diuji

Nama	NIM/Jurusan/Program Studi	Judul Tugas Akhir
Nanda Puji Arianto	5311311009/Teknik Elektro/D3 Teknik Elektro	Multimeter Elektronik Berbasis Mikrokontroller Atmega 8

III.Waktu dan Tempat Ujian

Hari/Tanggal : Senin, 10 Agustus 2015
Jam : 08.00.WIB s.d. selesai
Tempat : E8. 201
Pakaian : Hitam Putih Berjaket Almamater

Demikian surat tugas ini kami buat untuk dilaksanakan sebaik-baiknya.

Semarang, 7 Agustus 2015
Dekan

Drs. H.Muhammad Harlanu, M.Pd
NIP. 196602151991021001

Tembusan :
1.Ketua Jurusan Teknik Elektro
2.Calon yang diuji