Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler AT-mega 16

PROYEK AKHIR

Diajukan Kepada Fakultas Teknik Universitas Negeri Yogyakarta
Untuk Memenuhi Sebagian Persyaratan
Guna Memperoleh Gelar Ahli Madya

Oleh:
Zaenal Muttaqien
NIM. 09507131015

PROGRAM STUDI TEKNIK ELEKTRONIKA
JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
2012

PERSETUJUAN

Proyek akhir yang berjudul "Mesin Ketik Huruf *Braille* Elektronik Berbasis ATmega16" ini telah disetujui oleh pembimbing untuk diujikan.

Yogyakarta,
31. Oktober , 2012
Dosen Pembimbing,

Dr. Putu Sudira, M.P.
196412311987021063

PENGESAHAN

Proyek akhir yang berjudul "Mesin Ketik Huruf *Braille* Berbasis Mikrokontroler ATmega16" ini telah dipertahankan di depan Dewan Penguji pada tanggal 29 November 2012 dan dinyatakan lulus.

DEWAN PENGUJI

Nama	Jabatan	Tandatangan	Tanggal
Dr. Putu Sudira, MP.	Ketua Penguji	.....	7/12 2012
Handaru Jati, S.T.,MM.,MT.Ph.D	Sekretaris	.....	6/12 '12
Drs. Halim Sunawi	Penguji	.....	28/11 '12

Yogyakarta, 17 Desember 2012
Fakultas Teknik
Universitas Negeri Yogyakarta
Dekan

Dr. MOCH. BRURI TRIYONO
NIP. 19560216 198603 1 003

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Proyek Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Ahli Madya atau gelar lainnya di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 21 Oktober 2012

Yang menyatakan,

Zaenal Muttaqien

MOTTO

Nasib terbaik adalah tidak pernah dilahirkan

Yang kedua, dilahirkan tapi mati muda....

Dan yang tersial adalah yang berumur tua..

-Soe Hok Gie-

Buktikan! Buktikan!!!

Kalau hanya omong, burung beo pun bisa...

-Iwan Fals-

PERSEM BAHAN

Proyek Akhir ini saya persembahkan pada:

Alloh SWT yang senantiasa memberikan hidayah dalam pembuatan proyak akhir ini. Kedua Orangtuaku, Bapak Nasam R.S. dan Ibu Komariyah atas doa, perhatian, semangat dan kasih sayangnya dalam mendidik saya sampai dewasa. Istri saya, Supriyanti, S.Pd yang selalu sabar, menemani, berbagi dan memberikan dukungan. Kedua Mertua saya, Bapak Hadi S. Dan Ibu Sumarmi yang telah memberikan do'a dan semangat. Kakak dan adikku Sixteen Muzakki,S.Pd dan Ratih Azizatul Wahyu yang selalu memberikan sema-ngat..Teman-teman Kelas B 2009 Teknik Elektronika FT-UNY. Terima kasih atas dukungan, bantuan, dan dorongannya dalam penyelesaian Proyek Akhir ini. Teman-teman Kerja saya yang senantiasa mengerti dan mendukung saya.

Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16

Oleh :
Zaenal Muttaqien
09507131015

ABSTRAK

Perhatian bagi para penyandang tuna netra saat ini masih kurang. Masih sangat jarang ditemui pengumuman publik atau sejenisnya dalam bentuk huruf *braille*. Hal ini terjadi karena pembuatan huruf *braille* dengan menggunakan cara manual membutuhkan waktu yang relatif lama, disamping itu juga pembuat harus menghafal motif huruf *braille*. Memang sudah terdapat *printer* untuk huruf *braille* tetapi harganya masih sangat mahal. Alasan-alasan tersebut mendasari pembuatan mesin ketik huruf *braille* elektronik berbasis mikrokontroller ATmega16 guna memudahkan pembuatan huruf *braille* bagi orang awas tanpa harus menghafal motif-motif huruf *braille.*

Pembuatan alat ini dimulai dengan merancang dan merakit *hardware* yang dilanjutkan dengan membuat *software*. Alat ini menggunakan *keyboard* sebagai *input* dan dua buah *motor stepper* serta solenoid sebagai *output*. Perancangan perangkat lunak sebagai pengendali pada mikrokontroller ATmega16 menggunakan bahasa C dan *software* CodeVsion AVR sebagai *compiler*-nya. Proses akhir pembuatan alat ini adalah dengan mendownloadkan *software* ke sistim minimum AT-mega 16.

Hardware telah berhasil dibuat dengan menggunakan sistem minimum mikrokontroler ATmega16 yang digabungkan dengan instrumen pendukung lainnya. *Software* Mesin Ketik Huruf *Braille* Berbasis Mikrokontroller ATmega16 berhasil dibuat dengan menggunakan CodeVisionAVR berupa satu program utama. Alat ini membaca data dari *input keyboard* yang ditekan, menggerakan *motor stepper* dan atau solenoid sehingga dapat menghasilkan motif huruf *Braille* sesuai tombol *keyboard* yang ditekan pada selembar kertas.

Kata kunci : Mesin Ketik, Huruf *Braille*

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur ke hadirat Allah SWT yang senantiasa memberikan ridho dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan laporan Proyek Akhir ini dengan judul **"**Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16**",** sebagai salah satu syarat kelulusan di Jurusan Pendidikan Teknik Elektronika dan Informatika, Universitas Negeri Yogyakarta.

Penulis menyadari bahwa dalam penyusunan laporan Proyek Akhir ini masih terdapat banyak kekurangan dan kelemahan serta jauh dari sempurna. Penulis berharap semua pihak dapat memberikan kritik dan saran yang dapat membangun penulis menjadi lebih baik.

Selanjutnya dengan segala kerendahan dan ketulusan hati, perkenankanlah penulis menyampaikan ucapan terima kasih kepada semua pihak yang telah memberikan bantuan dan dorongannya baik secara langsung maupun tidak langsung sehingga penulis dapat menyelesaikan laporan Proyek Akhir ini. Ucapan terima kasih ditujukan kepada :

1. Dr. Moch. Bruri Triyono selaku Dekan Fakultas Teknik Universitas Negeri Yogyakarta.
2. Muhammad Munir, M.Pd. selaku Ketua Jurusan Pendidikan Teknik Elektronika dan Informatika

3. Dr. Putu Sudira, M.P. selaku dosen pembimbing yang telah meluangkan waktu untuk memberikan petunjuk dan arahan dalam menyelesaikan laporan Proyek Akhir ini.
4. Kedua orang tua penulis yang telah memberikan do'a, fasilitas dan dorongan kepada penulis baik secara moril maupun materiil.
5. Supriyanti, S.Pd. dan Nathania atas semangat, doa, kesabaran dan pengorbanannya dalam membantu menyusun laporan Proyek Akhir ini.
6. Kakak dan adik penulis atas dukungannya kepada penulis.
7. Rekan–rekan mahasiswa Teknik Elektronika kelas B 2009 semuanya yang tidak dapat penulis sebutkan satu persatu.
8. Atasan dan rekan-rekan kerja penulis atas dukungan dan semangat kepada penulis.

Akhir kata semoga ketulusan serta bantuan dari semua pihak tersebut kiranya mendapat berkah dan anugerah dari Allah SWT.

Yogyakarta, 20 Oktober 2012

Penulis

DAFTAR ISI

Halaman

LEMBAR PERSETUJUAN i

LEMBAR PENGESAHAN ii

SURAT PERNYATAAN KEASLIAN iii

MOTTO iv

PERSEMBAHAN v

ABSTRAK vi

KATA PENGANTAR vii

DAFTAR ISI ix

DAFTAR TABEL xi

DAFTAR GAMBAR xii

DAFTAR LAMPIRAN xiv

BAB I. PENDAHULUAN 1

A. Latar Belakang Masalah 1

B. Identifikasi Masalah 2

C. Batasan Masalah 2

D. Rumusan Masalah 3

E. Tujuan 3

F. Manfaat 3

G. Keaslian Gagasan 4

BAB II. PENDEKATAN PEMECAHAN MASALAH 6

A. Huruf *Braille* 6

B. Mesin Ketik Huruf *Braille* Elektronik 10

C. Hardware 11

D. Pemrograman Bahasa C 21

BAB III. KONSEP RANCANGAN 31

A. Identifikasi Kebutuhan 31

B. Analisis Kebutuhan 31

C. Perancangan Alat 32

D. Perancangan Program 34

E. Langkah Kerja 39

F. Penjadwalan 39

G. Biaya 40

BAB IV.PROSES, HASIL DAN PEMBAHASAN..... 41

A. Proses 41

B. Hasil 49

C. Pembahasan 55

BAB V. KESIMPULAN DAN SARAN 56

A. Kesimpulan 56

B. Kelemahan Alat 57

C. Saran 57

DAFTAR PUSTAKA 58

LAMPIRAN 59

DAFTAR TABEL

Halaman

Tabel 1. Tipe Data 24

Tabel 2. Simbol dan Aritmatika 27

Tabel 3. Simbol dan Pembanding 27

Tabel 4. Manipulasi Bit 28

Tabel 5. Data *Keyboard* 37

Tabel 6. Rencana Kegiatan 43

Tabel 7. Rincian Biaya 43

Tabel 8. Hasil Pengujian IC ULN2003 57

Tabel 9. Keadaan Logika Input-Output 59

DAFTAR GAMBAR

Halaman

Gambar 1. Motif Huruf *Braille* 10

Gambar 2. Diagram Blok ATmega16 12

Gambar 3. Susunan Pin ATmega16 12

Gambar 4. *Keyboard* 14

Gambar 5. Pin Port PS2 14

Gambar 6. Motor Stepper 15

Gambar 7. Lilitan pada Solenoid 15

Gambar 8. Konfigurasi L293D 17

Gambar 9. Blok Diagram L293D 17

Gambar 10. IC ULN2003 18

Gambar 11. Pin Koneksi IC ULN2003 19

Gambar 12. IRF 3205 19

Gambar 13. Bentuk Fisik IC Regulator 7805 20

Gambar 14. Simbol Kapasitor 21

Gambar 15. Diagram Blok Mesin Ketik *Braille* 33

Gambar 16. Skema Rangkaian Sistem Minimum 33

Gambar 17. Diagram Alir Program Mesin Ketik *Braille* 38

Gambar 18. Rangkaian Sistem Minimum 50

Gambar 19. Bentuk Fisik Mesin Ketik Huruf *Braille* 50

Gambar 20. Menguji Data *Keyboard* 52

Gambar 21. Sinyal Clock dan Data 52

Gambar 22. Pengujian IC ULN2003 53

Gambar 23. Rangkaian Simulasi IC L293D 54

Gambar 24. Pulsa Input dan Output..... 54

DAFTAR LAMPIRAN

Halaman

Lampiran 1. Rangkaian Blok Sistem Minimum 59

Lampiran 2. Layout PCB Rangkaian Sistem Minimum 60

Lampiran 3. Dokumentasi Proses Pembuatan..... 61

Lampiran 4. Listing Program 62

Lampiran 5. Datasheet Mikrokontroler ATmega16 68

Lampiran 6. Datasheet 7805 89

Lampiran 7. Datasheet IC ULN2003 92

Lampiran 9. Datasheet IC L293D..... 100

Lampiran 10.Datasheet IRF3205 107

Lampiran 11.SK Pembimbing 115

Lampiran 12.Kartu Bimbingan 116

Lampiran 13.Surat Persetujuan Ujian TA 117

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Manusia merupakan makhluk paling sempurna yang diciptakan Tuhan. Manusia diberi akal oleh Tuhan untuk membedakannya dengan makhluk lain. Namun demikian, tidak semua manusia tercipta dengan fisik sempurna. Sebagian manusia memiliki kekurangan fisik, seperti tuna netra, tuna rungu, dan sebagainya. Keterbatasan fisik tersebut kadang membuat mereka terpinggirkan karena kurangnya perhatian bagi mereka. Seharusnya kita sadar bahwa mereka juga berhak untuk hidup layak seperti kita.

Huruf *braille* merupakan solusi bagi para tuna netra untuk dapat berkomunikasi. Seiring berkembangnya teknologi masa kini huruf *braille* dapat juga digunakan untuk memberikan pembelajaran bagi para penyandang tuna netra. Sayangnya kenyataan yang ada saat ini masih sangat jarang ditemui buku-buku dengan menggunakan huruf *braille*. Bahkan, sekedar pengumuman publik pun jarang ditemui dalam huruf *braille.*

Membuat tulisan *braille* memang membutuhkan proses yang lebih susah dan membutuhkan waktu lama jika pengerjaannya masih menggunakan cara manual. Kita harus membuat motif huruf dengan jarum pada kertas untuk setiap huruf. Sebenarnya, sudah terdapat *printer* untuk huruf *braille*, tetapi harganya masih sangat mahal. Kendala tersebutlah yang mendorong penulis untuk membuat mesin ketik huruf *braille* elektronik berbasis ATmega 16. Mesin ketik tersebut

akan membantu orang awas dalam membuat huruf *braille* dengan mudah, tanpa harus menghafal setiap motif huruf.

B. Identifikasi Masalah

Dari latar belakang masalah yang telah di uraikan di atas, maka dapat diidentifikasikan masalah sebagai berikut:

1. Kurangnya perhatian bagi penyandang tuna netra dalam bidang ilmu pengetahuan dan teknologi.
2. Kurangnya media baca bagi penyandang tuna netra dengan huruf *braille*.
3. Belum adanya mesin pembuat huruf *braille* yang praktis dan terjangkau yang memudahkan orang awas membuat huruf *braille*.
4. Kurangnya pemanfaatan ATmega 16 sebagai mesin ketik huruf *braille*.

C. Batasan Masalah

Berbagai permasalahan yang di kemukakan dalam identifikasi masalah dapat dibatasi menjadi beberapa hal sebagai berikut :

1. Rancang bangun *hardware*
2. Rancang bangun *software*
3. Unjuk kerja alat

D. Rumusan Masalah

Dalam proyek ini, penulis merumuskan permasalahan sebagai berikut:

1. Bagaimana merancang *hardware* mesin ketik huruf *braille* elektronik berbasis ATmega 16?
2. Bagaimana merancang *software* mesin ketik huruf *braille* elektronik berbasis ATmega 16?
3. Bagaimana unjuk kerja mesin ketik huruf *braille* elektronik berbasis ATmega 16?

E. Tujuan

Pembuatan proyek akhir mesin ketik huruf braille elektronik berbasis ATmega 16 mempunyai tujuan sebagai berikut:

1. Merealisasikan rancangan *hardware* mesin ketik huruf *braille* elektronik berbasis ATmega 16.
2. Merealisasikan rancangan *software* mesin ketik huruf *braille* elektronik berbasis ATmega 16.
3. Mengetahui unjuk kerja mesin ketik huruf *braille* elektronik berbasis ATmega 16.

F. Manfaat

Adapun manfaat dari pembuatan proyek akhir ini adalah:

1. Bagi mahasiswa

a. Untuk memenuhi persyaratan guna memperoleh gelar ahli madya.

b. Memberi bekal pengalaman untuk mengaplikasikan ilmu pengetahuan yang didapat selama di bangku kuliah kedalam suatu karya nyata.

c. Sebagai bentuk kontribusi terhadap universitas baik dalam cita maupun daya tawar terhadap masyarakat luas.

2. Bagi jurusan P.T.Elektronika

a. Sebagai tolak ukur daya serap mahasiswa yang bersangkutan selama menempuh pendidikan dan kemampuan ilmunya secara praktis.

b. Sebagai bahan referensi atau kajian utuk pengembangan selanjutnya.

c. Menambah khasanah ilmu pengtahuan bidang elektronika dengan menggunakan teknologi *chip programmable*.

3. Bagi masyarakat

a. Memberikan kemudahan menulis huruf *braille*.

b. Mengurangi rasa diskriminasi bagi penyandang tuna netra.

c. Mempermudah proses pembelajaran bagi penyandang tuna netra.

G. Keaslian Gagasan

Mesin ketik huruf *braille* elektronik berbasis ATmega 16 ini belum pernah dibuat oleh siapapun, baik dalam civitas akademik Universitas Negeri Yogyakarta maupun instansi lain. Adapun karya-karya sejenis, misalanya Desain *keyboard* dengan *output* suara sebagai alat *bantu* pengenalan huruf *braille*, oleh Rizky Yunanta, Akhmad Hendriawan, Reesa Akbar, Rika Rokhana, Jurusan Teknik Elektronika, Kampus PENS-ITS Sukolilo, Surabaya. Selain itu, ada juga media pembelajaran huruf braille dengan tombol tekan dan penyuaraan, Mashoedah,

S.Pd, M.T. , Universitas Negeri Yogyakarta. Namun, kedua alat tersebut merupakan media pembelajaran untuk tuna netra, tanpa menghasilkan tulisan *braille* seperti proyek akhir ini.

BAB II

PENDEKATAN PEMECAHAN MASALAH

A. Huruf *Braille*

1. Sejarah Tulisan *Braille*

Seorang tunanetra asal Prancis bernama Louis Braille lahir pada 4 Januari 1809. Berkat kelahiran anak inilah maka lebih dari 40 juta orang tunanetra di seluruh dunia dapat belajar membaca dan menulis, serta dapat mengenyam pendidikan sebagaimana rekan-rekannya yang awas. Karena itulah, tanggal 4 januari diperingati oleh dunia internasional sebagai "Hari Braille".

Usaha untuk menciptakan tulisan bagi orang tunanetra telah dimulai sekurang-kurangnya 16 abad yang lalu. Seorang cendekiawan tunanetra Jepang pada abad ke-4 mengukir huruf-huruf pada kayu dan mendirikan sebuah perpustakaan untuk menghimpun karya-karyanya tersebut. Sementara, orang-orang di Eropa masih memusatkan usaha membantu tunanetra belajar membaca dan menulis dengan cara memperbesar huruf Latin atau Romawi menggunakan tali-temali, potongan-potongan logam, kulit atau kertas hingga awal abad ke-19. Puncak keberhasilan usaha-usaha tersebut dicapai oleh Louis Braille, seorang anak tukang sepatu yang menjadi buta akibat matanya tergores pisau pemotong kulit milik ayahnya.

Louis Braille mendapatkan inspirasi bagi ciptaannya itu dari mantan perwira artileri Napoleon bernama Kapten Charles Barbier. Dalam peperangan Napoleon, Barbier menciptakan tulisan sandi yang terdiri dari titik-titik dan garis-garis

timbul yang dinamakannya "tulisan malam". Dia menggunakan tulisan tersebut untuk memungkinkan pasukannya membaca perintah-perintah militer dalam kegelapan malam dengan merabanya melalui ujung-ujung jari.

Meskipun ciptaan Barbier ini telah terbukti berhasil untuk keperluan militer, tetapi tidak cocok untuk keperluan membaca dan menulis biasa. Akan tetapi, penemuan tersebut sangat berharga bagi Louis Braille ke arah apa yang sedang dicari-carinya.

Setelah bertemu dengan Charles Barbier, Louis Braille selalu mencoba menciptakan tulisan yang cocok bagi tunanetra dengan membuat titik-titik dan garis-garis pada kartu-kartu. Dia mencoba setiap perkembangan tulisannya kepada teman-temannya yang tunanetra. Louis Braille menyadari bahwa jari teman-temannya lebih peka terhadap titik daripada terhadap garis. Dia memutuskan untuk menggunakan titik-titik dan mengesampingkan garis-garis bagi tulisannya.

Akhirnya, ketika Louis Braille berusia 25 tahun, sempurnalah sistem tulisan yang terdiri dari titik-titik timbul itu pada tahun 1834. Louis Braille hanya menggunakan enam titik domino sebagai kerangka sistem tulisannya. Satu atau beberapa dari enam titik itu divariasikan letaknya sehingga dapat membentuk bermacam-macam kombinasi yang cukup untuk menggambarkan abjad, angka, tanda baca, dan sebagainya.

Ketika Louis Braille masih menyederhanakan sistem tulisannya, dia diangkat sebagai guru di *L'Institution Nationale des Jeunes Aveugles* (Lembaga Nasional untuk Anak-anak Tunanetra) di Paris yang didirikan oleh Monsieur Valentin Hauy pada tahun 1783. Dia merupakan guru yang sangat disukai. Dia

dipercaya untuk mengajar sejarah, geografi, matematika, tata bahasa Perancis, dan musik.

Dia mulai mencoba kepada murid-muridnya saat sistem tulisannya sudah cukup sempurna. Merekapun menyambutnya dengan gembira dan sangat merasakan manfaatnya. Dr. Pignier sebagai kepala lembaga tersebut mengizinkan sistem tulisan itu dipergunakan dalam pengajaran di sekolah itu, tetapi tidak seorang pun di luar lembaga mau menerima keberadaannya. Mereka beranggapan mengajarkan tulisan yang berbeda dari tulisan umum sebagai sesuatu yang ganjil dan tidak masuk akal. Bahkan, badan pembina lembaga juga tidak menyukai sistem tulisan Braille sehingga mereka memecat Dr. Pignier saat ia merencanakan menyalin buku sejarah ke dalam braille.

Kepala yang baru, Dr. Dufau tidak menyetujui sistem Braile itu dan melarang keras penggunaannya. Namun, murid-muridnya telah merasakan manfaat tulisan Braille. Mereka meminta Braille mengajarkannya secara diam-diam. Demi murid-muridnya, dia setuju mengajar mereka di luar jam sekolah.

Guru dan semua murid di dalam kelas Braille merupakan tunanetra sehingga mudah bagi guru guru lain untuk mengintip kelas rahasia itu dan memperhatikannya tanpa mereka ketahui. Kepala staf pengajar, Dr. Guadet sering mengamati pelajaran rahasia tersebut dengan penuh minat dan simpati. Setelah melihat betapa cepatnya murid-murid itu memahami pengajaran yang disampaikan oleh Braille, maka Dr. Guadet mengimbau kepada Dr. Dufau agar mengubah pendiriannya dan mengizinkan penggunaan sistem tulisan Braille.

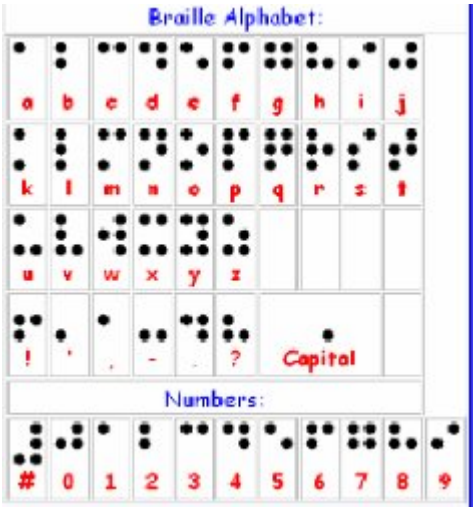
Akhirnya Dr. Dufau sejuju dan menjelang tahun 1847 Louis Braille kembali dapat mengajarkan ciptaannya tersebut secara leluasa.

Pada tahun 1851 Dr. Dufau mengajukan ciptaan Braille itu kepada Pemerintah Perancis dengan permohonan agar ciptaan tersebut mendapat pengakuan pemerintah dan Louis Braille diberi tanda jasa. Tetapi hingga Braille meninggal pada tanggal 6 Januari 1852, tanda jasa dan pengakuan resmi terhadap ciptaannya tersebut tidak pernah diterimanya. Ciptaan Louis Braille diakui secara resmi di L'institution Nationale des Jeunes Aveugles baru beberapa bulan setelah wafatnya. Beberapa tahun kemudian tulisan Braille digunakan di beberapa sekolah tunanetra di negara-negara lain dan menjelang akhir abad ke-19 sistem tulisan ini diterima secara universal dengan nama tulisan "Braille".

Untuk mengenang jasa Braille tersebut maka pada tahun 1956 *The World Council for the Welfare of the Blind* (Dewan Dunia untuk Kesejahteraan Tunanetra) menjadikan bekas rumah kediaman Louis Braille yang terletak di Coupvray, 40 km sebelah timur Paris sebagai museum Louis Braille. Pada tahun 1984 *WCWB* melebur diri dengan *International Federation of the Blind* (Federasi Tunanetra Internasional) menjadi *World Blind Union* (Perhimpunan Tunanetra Dunia) dan sejak saat itu pemeliharaan dan pengembangan museum ini menjadi tanggung jawab *WBU*.

2. Bentuk Huruf *Braille*

Huruf *Braille* merupakan huruf yang terdiri dari motiv titik yang timbul. Motiv ini terdiri dari enam buah titik yang divariasi sesuai abjad yang diinginkan. Adapun bentuk-bentuk huruf *braille* adalah sebagai berikut :



Gambar 1. Motif Huruf *Braille*

Ukuran huruf *braille* yang umum digunakan adalah dengan tinggi sepanjang 0.5 mm, serta spasi horizontal dan vertikal antar titik dalam sel sebesar 2.5 mm.

B. Mesin Ketik Huruf *Braille* Elektronik

Mesin Ketik Huruf *Braille* Elektronik diharapkan memiliki fitur-fitur sebagai berikut:

1. *Keyboard* sebagai masukan dalam mengetik hururf *braille*,
2. Jarum yang bekerja secara otomatis guna membentuk motif huruf *braille* sesuai tombol *keyboard* yang ditekan.
3. *Roller* yang akan berputar menarik kertas secara otomatis untuk memudahkan pengguna.

Untuk itu, mesin ketik dirancang memiliki tiga bagian, yaitu rangkaian sistem minimum, rangkaian *input* dan rangkaian *output*.

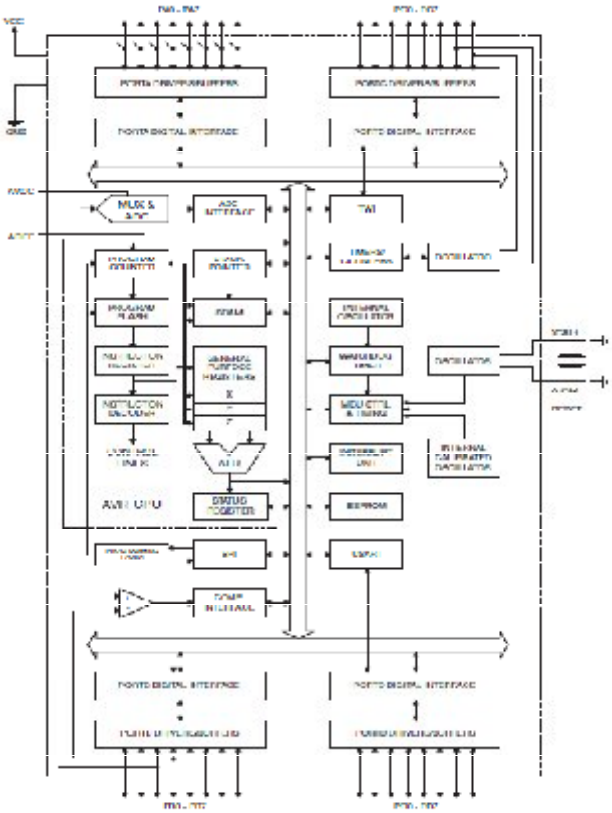
Rangkaian sistem minimum pada alat ini menggunakan mikrokon-troler ATmega16 sebagai pengendali utama. Rangkaian *input* menggunakan *keyboard* dengan konektor PS/2. Penggunaan *keyboard* ini bertujuan untuk memudahkan pengguna. Sedangkan rangkaian *output* merupakan modivikasi dari printer bekas yang ditambahkan penitik untuk membuat motiv huruff *braille.*

C. Hardware

1. Mikrokontroler AVR

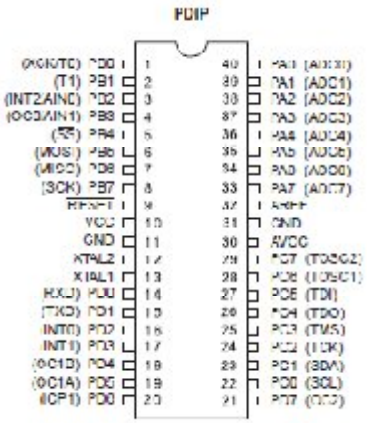
AVR singkatan dari *Advanced Versatile RISC* atau *Alf and Vegard's Risc processor* yang merupakan gabungan dua nama, yaitu Alf Egil Bogen dan Vegard Wollan. AVR memiliki kemampuan eksekusi program lebih cepat. Mikrokontroler AVR dapat diprogram menggunakan *low level language* dan *high level language*. Mikrokontroler AVR memiliki tiga kelompok, yaitu seri AT90Sxx, ATmega, dan ATtiny. Pada proyek akhir ini Mikrokontroler yang digunakan ATmega16. ATmega16 memiliki 32 Input/Output yang terdiri dari empat *Port*.

a. Arsitektur Mikrokontroler ATmega16



Gambar 2. Diagram Blok ATmega16

b. Konfigurasi Pin Atmega16



Gambar 3. Susunan Pin ATmega16

Susunan pin/kaki dari ATmega16, yaitu:

a. Pin 1 – 8, PORT B ATmega16.

b. Pin 9, merupakan pin RESET untuk me-*reset* mikrokontroler.

c. Pin 10, VCC untuk masukan positif catu daya.

d. Pin 11 dan 31, GND sebagai pin Ground

e. Pin 12-13, XTAL merupakan masukan *crystal* eksternal jika diperlukan.

f. Pin 14-21, PORT D ATmega16.

g. Pin 22-29, PORT C ATmega16.

h. Pin 30, AVCC

i. Pin 32, AREF

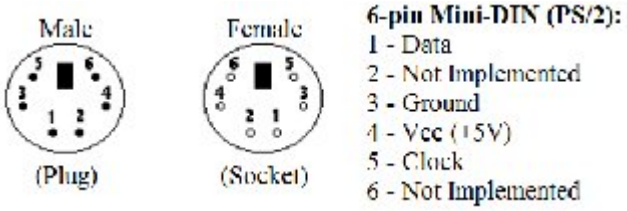
j. Pin 33-40, PORT A ATmega16. (Heri Andrianto: 2008)

2. *Keyboard PS/2*

Keyboard merupakan unit input yang paling penting dalam suatu pengolahan data dengan komputer. *Keyboard* dapat berfungsi memasukkan huruf, angka, karakter khusus serta sebagai media bagi *user* (pengguna) untuk melakukan perintah-perintah lainnya yang diperlukan, seperti mengetik, menyimpan file dan membuka file.

Keyboard yang kita kenal sekarang memiliki beberapa jenis port, yaitu port serial, ps2, usb dan wireless. Konektor PS2 sendiri merupakan konektor dengan menggunakan 4 buah terminal aktif, meskipun biasanya terdiri dari 5 atau 6 terminal. Adapun 4 terminal tersebut merupakan terminal VCC, ground, *clock,* dan data.

Gambar 4. *Keyboard*



Gambar 5. Pin Port PS2

3. *Motor Stepper*

Motor stepper adalah perangkat elektromekanis yang bekerja dengan mengubah pulsa elektronis menjadi gerakan mekanis diskrit. *Motor stepper* bergerak berdasarkan urutan pulsa yang diberikan kepada motor. Karena itu, untuk menggerakkan *motor stepper* diperlukan pengendali *motor stepper* yang membangkitkan pulsa-pulsa periodik. Penggunaan *motor stepper* memiliki beberapa keunggulan dibandingkan dengan penggunaan motor DC biasa.

Keunggulannya antara lain adalah :

a. Sudut rotasi motor proporsional dengan pulsa masukan sehingga lebih mudah diatur.
b. Motor dapat langsung memberikan torsi penuh pada saat mulai bergerak
c. Posisi dan pergerakan repetisinya dapat ditentukan secara presisi
d. Memiliki respon yang sangat baik terhadap mulai, stop dan berbalik (perputaran)

e. Sangat realibel karena tidak adanya sikat yang bersentuhan dengan rotor seperti pada motor DC

f. Dapat menghasilkan perputaran yang lambat sehingga beban dapat dikopel langsung ke porosnya

g. Frekuensi perputaran dapat ditentukan secara bebas dan mudah pada range yang luas.

Gambar 6. Motor Stepper

4. Solenoid

Solenoid adalah salah satu jenis kumparan terbuat dari kabel panjang yang dililitkan secara rapat dan dapat diasumsikan bahwa panjangnya jauh lebih besar daripada diameternya. Dalam kasus *solenoid ideal*, panjang kumparan adalah tak hingga dan dibangun dengan kabel yang saling berhimpit dalam lilitannya, dan medan magnet di dalamnya adalah seragam dan paralel terhadap sumbu solenoid.

Gambar 7. Lilitan Pada Solenoid

Kuat medan magnet untuk solenoid ideal adalah:

$B = \mu_0 i n$

di mana:

B adalah kuat medan magnet,

μ_0 adalah permeabilitas ruang kosong,

i adalah kuat arus yang mengalir,

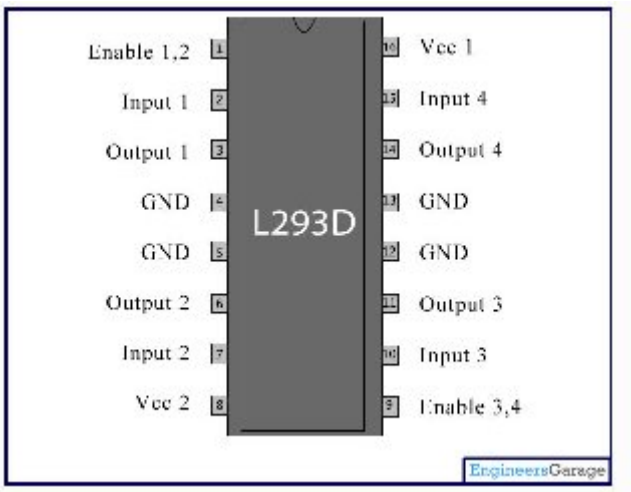
n dan adalah jumlah lilitan.

Jika terdapat batang besi dan ditempatkan sebagian panjangnya di dalam solenoid, batang tersebut akan bergerak masuk ke dalam solenoid saat arus dialirkan. Hal ini dapat dimanfaatkan untuk menggerakkan tuas, membuka pintu, atau mengoperasikan relai.

5. IC L293D

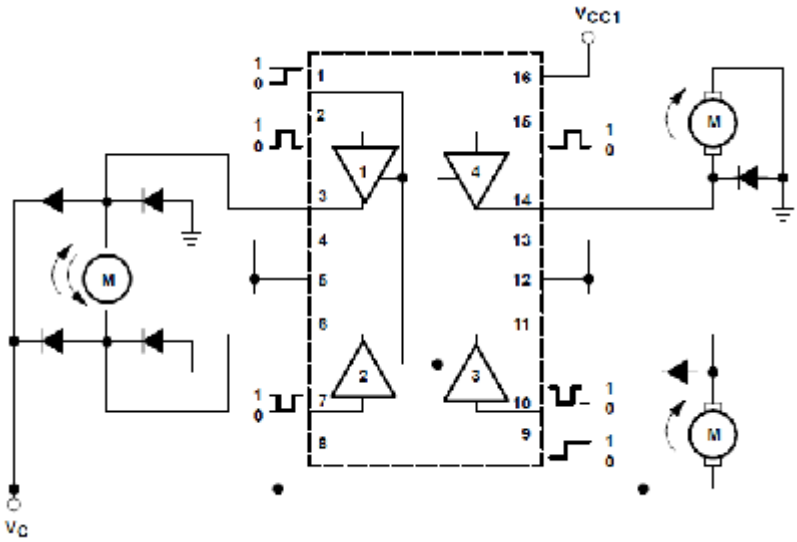
IC L293D adalah sebuah chip H-Bridge yang mempunyai 2 buah rangkaian H-bridge didalamnya sehingga bisa mengendalikan kecepatan dan arah 2 buah motor. Mendukung operasi motor 4.5V – 36V dengan arus 600 mA (arus maksimal 1.2A). IC ini bisa digunakan untuk mengendalikan relay, solenoid, motor DC dan *motor stepper*.

Keuntungan lain IC l293D adalah ic ini telah mempunyai proteksi arus balik dari beban berupa diode didalam IC.



Gambar 8. Konfigurasi L293D

Setiap *driver* diaktifkan secara berpasangan, dengan *driver* 1 dan 2 diaktifkan oleh 1,2 EN dan *driver* 3 dan 4 diaktifkan oleh 3,4 EN.



Gambar 9. Blok Diagaram L293D.

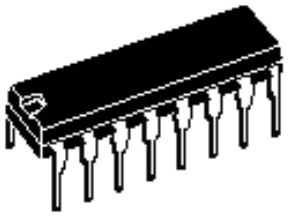
Bila input *enable* tinggi, driver terkait diaktifkan dan output mereka aktif dan dalam fase dengan input mereka. Bila input enable rendah, mereka driver

dinonaktifkan dan output mereka tidak aktif dan dalam keadaan impedansi tinggi. Dengan input data yang tepat, setiap pasangan driver membentuk-H penuh (atau jembatan) *hard reversibel* cocok untuk aplikasi solenoid atau motor.

6. IC ULN 2003

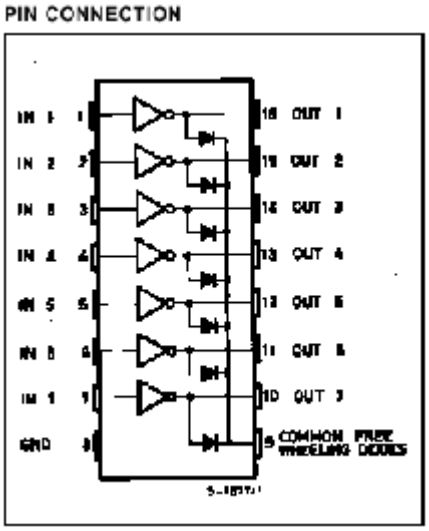
IC ULN 2003 adalah sebuah IC yang memiliki 7 bit input, tegangan maksimal 50 *volt* dan arus 500 mA. IC ini termasuk jenis TTL. Didalam IC ini terdapat transistor darlinton. Transistor darlington merupakan 2 buah transistor yang dirangkai dengan konfigurasi khusus untuk mendapatkan penguatan ganda sehingga dapat menghasilkan penguatan arus yang besar.

IC ULN 2003 merupakan IC yang mempunyai 16 buah pin. *output* yang akan dihubungkan ke motor stepper dan ada yang berfungsi sebagai catu daya. Catu daya ini terdiri dari catu daya (+) dan *ground.* Besar catu daya yang dihubungkan tergantung pada tipe motor stepper. Bentuk fisik IC ULN 2003 adalah seperti gambar dibawah.



Gambar 10. IC ULN 2003

Sedang isi dari IC ULN 2003 dan fungsi dari masing-masing pin adalah sebagai berikut :



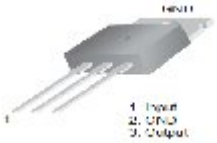
Gambar 11. Pin Koneksi IC ULN 2003

7. Mosfet IRF3205

Mosfet adalah salah satu jenis transistor efek medan. Prinsip dasar perangkat ini pertama kali diusulkan oleh Julius Edgar Lilienfeld pada tahun 1925 . Mosfet mencakup kanal dari bahan semikonduktor tipe-N dan tipe-P, dan biasa disebut nMOS dan pMOS. Transistor ini biasa digunakan untuk menaikan atau menurunkan tegangan. Namun, paling sering digunakan sebagai *switch* atau pensaklaran.

Gambar 12. IRF3205

8. IC Regulator 7805



Gambar 13. Bentuk Fisik IC Regulator 7805

IC ini berfungsi untuk menstabilkan tegangan input menjadi output 5V.

9. Resistor

Resistor adalah komponen elektronik dua kutub yang didesain untuk menahan arus listrik dengan memproduksi tegangan listrik di antara kedua kutubnya, nilai tegangan terhadap resistansi berbanding dengan arus yang mengalir, berdasarkan hukum Ohm:

$$V = IR$$

$$I = \frac{V}{R}$$

Karakteristik utama dari resistor adalah resistansi dan daya listrik yang dapat dihantarkan. Karakteristik lain termasuk koefisien suhu dan induktansi.

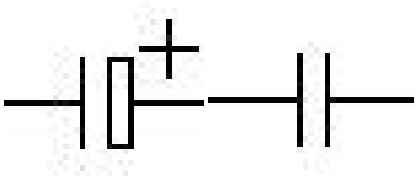
10. Kapasitor

Kondensator atau sering disebut sebagai kapasitor adalah suatu alat yang dapat menyimpan energi di dalam medan listrik, dengan cara mengumpulkan ketidakseimbangan internal dari muatan listrik. Kondensator memiliki satuan yang disebut Farad dari nama Michael Faraday.

Kapasitor dapat dibagi menjadi 2 jenis berdasarkan kutub, yaitu :

a. Kapasitor bipolar (memiliki 2 kutub)

b. Kapasitor nonpolar (tanpa kutub)



Gambar 14. Simbol Kapasitor Bipolar dan Nonpolar

11. Dioda

Dioda adalah komponen yang berfungsi sebagai penyearah arus, agar tidak terdapat arus AC yang masuk ke dalam rangkaian

D. Pemrograman Bahasa *C*

Bahasa C adalah bahasa mesin tingkat tinggi. Dimana dapat dengan mudah untuk melakukan pemrograman terhadap mikrokontroler. Dengan instruksi-instruksi yang mudah dipahami dan mudah diakses. Secara umum pemrograman mikrokontroler terdiri atas empat blok, yaitu :

1. Header,
2. Deklarasi kostanta global,
3. Fungsi dan atau prosedur (biosa dibawah program utama),
4. Program utama.

Secara umum, pemrograman C paling sederhana dapat dilakukan dengan hanya menuliskan program utamanya saja. Beberapa peraturan yang ada dalam bahasa C adalah:

1. Header

Header berisi include file (.hex), yaitu *library* (pustaka) yang akan digunakan dalam pemrograman. Perhatikan contoh dibawah ini:

```
#include <mega16.h>

#include <delay.h>

#incude <stdio.h>
```

2. Tipe Data

Berikut ini adalah tabel tipe-tipe variabel data yang dapat digunakan di *compiler Code Vision AVR*:

Tabel 1. Tipe Data

Tipe	Size (Bits)	Range
Bit	1	0,1
Char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
Int	16	-32768 to 32767
short int	16	-23768 to32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
Float	32	±1.175e-38 to ±3.402e38
Double	32	±1.175e-38 to ±3.402e38

3. Konstanta

Penulisan konstanta adalah sebagai berikut:

a. *Integer* atau *long integer* dapat ditulis dengan format decimal (contoh 1234), biner dengan awalan 0b (contoh 0b101001), heksadesimal dengan awalan 0x (contoh 0xff) atau octal dengan awalan 0 (contoh 0777).
b. *Unsigned integer* ditulis dengan diakhiri U (contoh 10000U).
c. Long integer ditulis dengan diakhiri L (contoh 99L).
d. *Unsigned long integer* ditulis dengan diakhiri UL (contoh 99UL).
e. *Floating* poin ditulis dengan diakhiri F (contoh 1.234F).

Karakter konstanta harus ditulis dalam tanda kutip (contoh 'z'), sedangkan konstanta string harus dalam tanda kutip dua (contoh "Saya Pelajar").

4. Label, Variabel, Fungsi

Identifikasi label, variabel dan fungsi dapat berupa huruf (A…Z, a…z) Dan angka (0…9), juga karakter *underscore* (_). Meskipun begitu identifikasi hanya bisa dimulai dengan huruf atau karakter *underscore*. Yang lebih penting lagi, identifikasi ini *Case is significant*, yaitu huruf besar dan kecil berbeda. Misal Variable1 tidak sama dengan variabel1. Identifikasi bisa memuat sebanyak 32 karakter.

5. Komentar

Komentar diawali dengan tanda '/*' dan diakhiri dengan '*/'. Perhatikan contoh dibawah:

```
/* ini komentar */
```

6. *Reserved Keywords*

Berikut ini adalah daftar kata baku yang tidak bisa dipakai (*reserved keywords*) untuk label, identifikasa atau variable:

Break	flash	signed	do	int	typedef
Bib	float	sizeof	double	interrupt	union
Case	for	sfrb	eeprom	long	unsigned
Char	funcused	sfrw	else	register	void
const	goto	static	enum	return	volatile
continue	if	struct	extern	short	while
default	inline	witch			

7. Operator

Suatu intruksi pasti mengandung operator dan operand. Operand adalah variabel atau konstanta yang merupakan bagian pernyataan sedangkan operator adalah suatu simbol yang menyatakan operasi mana yang akan dilakukun oleh operand tersebut.

Contoh: c = a + b ;

Ada tiga operand (a, b dan c) dan dua operator (= dan +). Operator dalam bahasa C dibagi menjadi 3

kelompok, yaitu :

a. *unary* operator yang beroperasi pada satu operand, misal: -n

b. *binary* operator yang beroperasi pada dua operand, misal: a-n

c. *ternary* operator memerlukan tiga / lebih operand, missal: a = (b*c) + d

8. Aritmatika

Tabel 2. Simbol dan Aritmatika

simbol	Contoh	aritmatika
+	c = a + b n = n + 2	penjumlahan
-	c = a - b n = n - 2	pengurangan
++	++i	Kenaikan (increment), sama dengan i = i + 1
--	--i	Penurunan (decrement), sama dengan i = i - i
*	c = a * b	perkalian
/	c = a / b n = n / 2	pembagian
%	Sisa = a % b	Menghasilkan sisa dari pembagi. a dan b bilangan bulat
=	a = b	Pemberian nilai
+=	A += 2	Penambahan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a = a + 2
-=	A -= 2	Pengurangan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a = a - 2
*=	A *= 2	Pengalian suatu nilai pada suatu variabel yang sudah ada sebelumnya.nSama dengan a = a * 2
/=	a /= 2	Pembagian suatu nilai pada suatu variabel yang sudah ada sebelumnya.nSama dengan a = a / 2
%=	A %= 2	Sisa dari suatu nilai pada suatu variabel yang sudah ada sebelumnya yang dibagi oleh nilai atau variable lainnya. Sama dengan a = a % 2
*	*pointer	Menunjukkan isi dari pointer

9. Logika

Tabel 3. Simbol dan Pembanding

Simbol	Contoh	Logika pembanding
==	If (a == b)	Logika sama dengan, digunakan untuk pembanding. Menghasilkan nilai *true* jika a = b
!=	If (a != b)	Tidak sama dengan. Menghasilkan nilai *true* jika a ≠ b.
<	If (a < b)	Logika lebih kecil dari. Menghasilkan nilai *true* jika a < b.
<=	If (a <= b)	Logika lebih kecil sama dengan dari. Menghasilkan nilai true jika a ≤ b.
>	If (a > b)	Logika lebih besar dari. Menghasilkan nilai *true* jika a > b.
>=	If (a >= b)	Logika lebih beser sama dengan dari. Menghasilkan nilai *true* jika a > b.
!	If (!a)	NOT
&&	If (a = = b && a = c)	AND
||	If (a = b || a = c)	OR

10. Manipulasi Bit

Tabel 4. Manipulasi Bit

~	A = ~ b	*Complement.* b 1100; a 0011.
&	c a & b	AND untuk manupulasi *bit.* a 1100; b 1001; c 1000.
|	c = a | b	OR untuk manipulasi bit. a 1100; b 1001; c 0101.
<<	c = a << n	*Shift left*, manipulasi bit menggeser kekiri sejauh n bit a 1101; n 2; maka c 110100.
>>	c = a >> n	Shift Right, manipulasi bit menggeser kekanan sejauh n bit a 11010;n 2; c 0110.

11. Percabangan

a. *if – then*

bentuk umum dari percabangan ini adalah:

```
if (kondisi) {
// pernyataan
};
```

b. *if – then – else*

bentuk umum dari percabangan ini adalah:

```
if (kondisi)
{
// pernyataan a
}
else
{
// pernyataan b
};
```

c. *switch – case*

Pernyataan *switch – case* digunakan jika terjadi banyak percabangan. struktur penulisan pernyataan ini adalah sebagai berikut:

```
…..
switch (ekspresi)
{
pernyataan1
`break;
case konstanta2:
pernyataan2
break;
……
case konstantaN:
pernyataanN
break;
}
```

d. *switch – case – default*

Pernyataan *switch – case – default* hampir sama dengan *switch – case.* Hal yang membedakannya adalah bahwa dengan adanya *default* maka jika tidak terdapat kondisi *case* yang sesuai dengan ekspresi *switch* maka akan menuju pernyataan yang terdapat pada bagian *default*. Struktur penulisan pernyataan ini adalah sebagai berikut:

```
switch (ekspresi)
{
case konstanta1
```

```
break;
case konstanta2:
pernyataan2
break;
….
case konstantaN:
pernyataanN
break;
default:
Pernyataan-pernyataan;
}
```

e. Perulangan *for*

Pernyataan *for* akan melakukan perulangan berapa kali sesuai yang diinginkan. Struktur penulisan perulangan for adalah sebagai berikut:

```
….
For (mulai ; kondisi ; penambahan atau pengurangan)
{
Pernyataan-pernyataan;
};
```

f. *while*

Bentuk dari perulangan *while* adalah sebagai berikut:

```
while (kondisi)
{
pernyataan-pernyataan;
}
```

g. *do – while*

Bentuk perulangan ini kebalikan dari *while – do*, yaitu pernyataan dilakukan terlebih dahulu kemudian diuji kondisinya

```
do
{
pernyataan-pernyataan;
} while (kondisi);
```

12. Konversi pola (%)

Karakter % dipakai sebagai operator konversi pola. Konversi pola akan sangat berguna pada saat kita menampilkan hasil ke LCD.

a. %d menampilkan bilangan bulat positif. Contoh: Sprintf(buf,"angka%d",14);

b. %o menampilkan bilangan octal bulat.

c. %x menampilkan bilangan heksadesimal bulat.

d. %u menampilkan bilangan decimal tanpa tanda.

e. %f menampilkan bilangan pecahan.

f. %i menampilkan bilangan integer.

g. %c menampilkan karakter yang ditunjukkan bilangan ASCII.

13. Prosedur

Prosedur adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu tanpa mengembalikan suatu nilai.

```
…..
void nama_prosedur (parameter1, parameter2,……parameterN)
```

```
{
} …..
```

14. Fungsi

Fungsi adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu dengan hasil akhir pengembalian nilai dari keperluan tersebut.

```
…..
Type data nama_fungsi (parameter1, parameter2, ….parameterN)
{
Pernyataan-pernyataan;
return variable_hasil;
} ….
```

15. Memasukkan Bahasa *Assembly*

Sering disebut juga dengan *in-line assembly*. Pemrograman dengan bahasa C ini masih dapat memasukkan bahasa *assembly* ke dalam program C. Struktur penulisannya juga mudah, yaitu:

```
…
#asm //dimulai dengan #asm
nop // blok bahasa assembly
nop //
#endasm // diakhiri dengan #endasm
….
```

Atau jika hanya beberapa instruksi maka kita bisa melakukannya dengan cara:

```
#asm("nop\nop\nop") ….
```

BAB III

KONSEP RANCANGAN

Perancangan mesin ketik huruf *braille* elektronik berbasis Mikrokontroller ATmega16 menggunakan metode rancang bangun. Secara urut metode tersebut adalah identifikasi kebutuhan yang diperlukan. Kemudian kebutuhan tersebut dianalisis untuk mendapatkan komponen secara spesifik. Selanjutnya dilakukan perancangan perangkat keras dan perangkat lunak, pembuatan serta pengujian.

A. Identifikasi Kebutuhan

1. Dibutuhkan *keyboard* sebagai input.
2. Dibutuhkan bahasa pemrograman yang sesuai untuk mikrokontroler ATmega16.
3. Dibutuhkan *motor stepper* untuk menggerakan kertas sebagai media tulis.
4. Dibutuhkan *motor stepper* untuk menggeser solenoid.
5. Dibutuhkan solenoid sebagai penggerak jarum penitik kertas.
6. Dibutuhkan *power supply* untuk mendukung kerja sistem tersebut.
7. Dibutuhkan sistem minimum untuk mengatur kerja rangkaian keseluruhan.

B. Analisis Kebutuhan

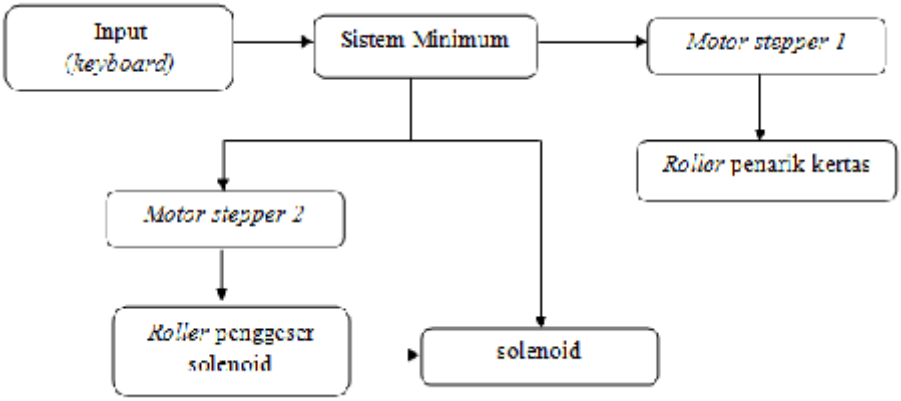
Dari beberapa identifikasi kebutuhan di atas maka diperoleh beberapa analisis kebutuhan terhadap pengembangan alat yang akan dibuat sebagai berikut:

1. Menggunakan ATmega16 dengan *clock* 12 Mhz untuk mendapatkan *output* yang diinginkan.
2. Menggunakan bahasa pemrograman C untuk mengaplikasikan algoritma untuk mesin ketik huruf *braille* elektronik berbasis ATmega16.
3. Menggunakan *keyboard* sebagai input.
4. Menggunakan *regulator power supply* dengan spesifikasi output DC +5V dan input 220 AC
5. Menggunakan *motor stepper* 5 kaki yang terhubung pada *roller* untuk menarik kertas.
6. Menggunakan *motor stepper* tipe CH4-4252 yang terhubung pada *roller* untuk menggeser solenoid.
7. Memodifikasi rangka printer bekas sebagai *body* mesin ketik.

C. Perancangan Alat

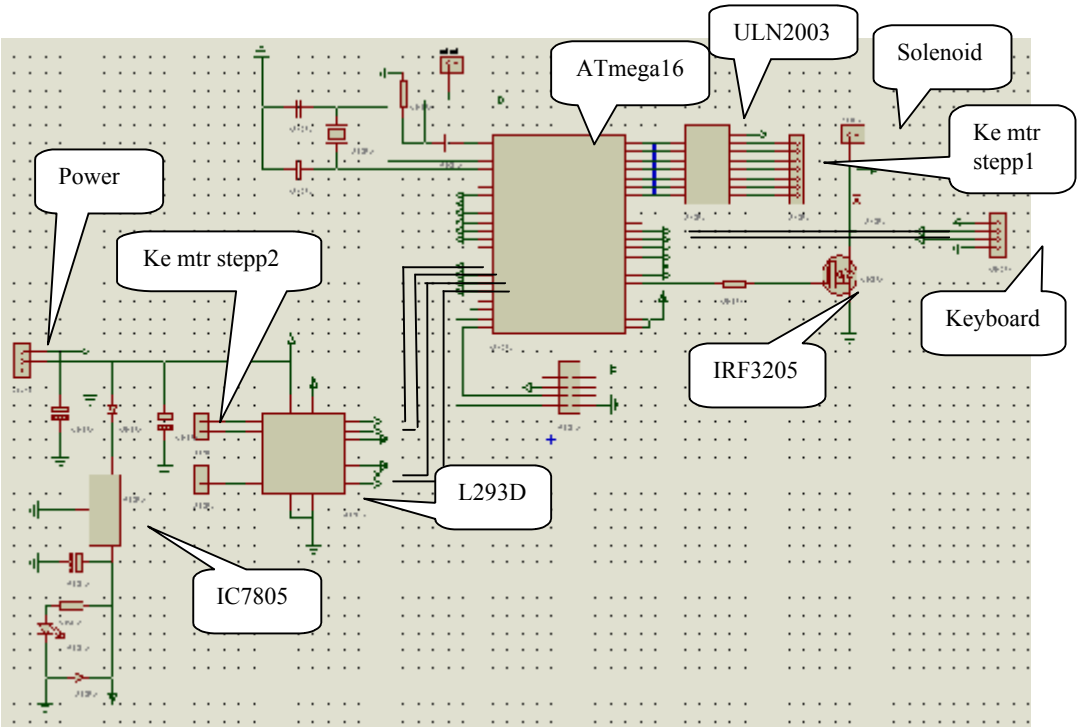
Mesin ketik yang akan penulis buat terdiri atas tiga blok, yaitu *input*, mikrokontroler dan *output*. Blok *input* terdiri dari sebuah *keyboard* yang akan terhubung dengan mikrokontroller menggunakan komunikasi PS2. Mikrokontroller terdiri dari ATmega16 sebagai pengatur utama. Sedangkan blok *output* terdiri dari dua buah *motor stepper* yang berfungsi sebagai penarik kertas dan penggeser solenoid.

Diagram blok mesin ketik huruf braille elektronik berbasis ATmega16 dapat dilihat pada gambar berikut.



Gambar 15. Diagram Blok Mesin Ketik *Braille*

Adapun skema komponen dari sistem minimum tersebut adalah sebagai berikut :



Gambar 16. Skema Rangkaian Sistem Minimum

D. Perancangan Program

Dalam perancangan 34ystem mikrokontroller, perancangan perangkat lunak menjadi salah satu hal yang sangat penting karena tanpa perangkat lunak 34ystem tidak akan bekerja. Perancangan perangkat lunak berisi langkah kerja yang sistematis, sesuai bahasa pemrograman yang digunakan.

Dalam pembuatan perangkat lunak selalu di awali dengan penentuan langkah kerja yang di gambarkan dengan algoritma dan diagram alir (*flowchart*) agar dapat diperoleh langkah-langkah yang paling efektif dan efisien. Kemudian dari langkah-langkah tersebut akan direalisasikan kedalam sebuah program. Program inilah yang akan di *download*-kan kedalam *flash memory* pada IC Atmega 16. Algoritma ntuk program mesin ketik huruf *braille* elektronik berbasis ATmega16 adalah sebagai berikut :

1. Step1 : mulai .
2. Step2 : inisialisasi Variabel, Port, Interupt.
3. Step3 : konfigurasi *keyboard.*
4. Step4 : baca data dari *keyboard*.
5. Step5 : periksa apakah data terdapat pada tabel data *keyboard*. Jika tidak, maka ke step berikutnya, jika ya maka panggil sub program .

Tabel 5. Data *Keyboard*

Data	Huruf	Data	Huruf	Data	Huruf	Data	Huruf
57	A	117	M	107	Y	83	SPC
101	B	99	N	53	Z	49	?
67	C	9	O	97	0	37	!
71	D	27	P	45	1	25	.
73	E	43	Q	61	2	41	,
87	F	91	R	77	3	29	-
105	G	55	S	73	4	21	ENT
103	H	89	T	93	5	3	DOWN
7	I	121	U	109	6	19	UP
119	J	85	V	121	7	113	LEFT
5	K	59	W	125	8	15	RIGHT
23	L	69	X	13	9		

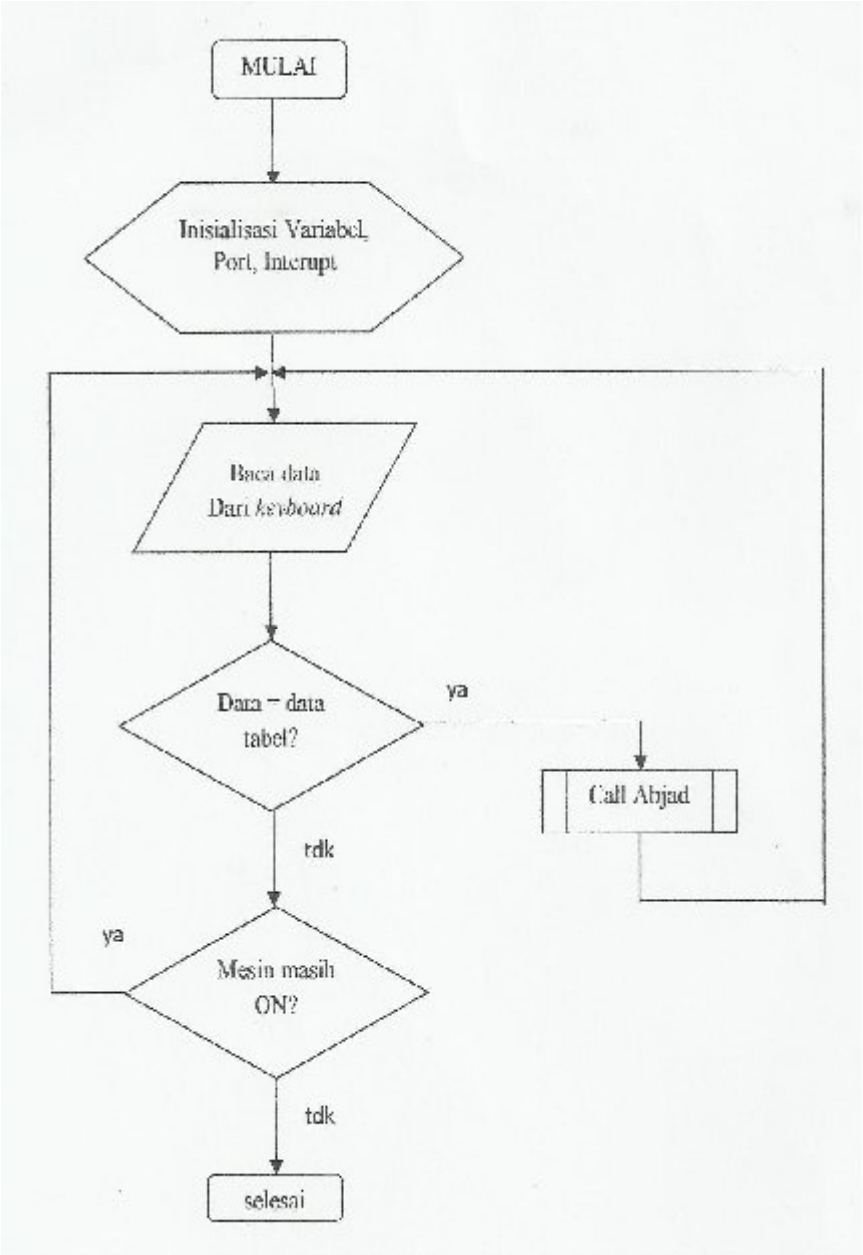
6. Step6 : apakah mesin masih ON?
7. Step7 : jika ya, kembali ke step 4. Jika tidak, ke step 8.
8. Step8 : selesai.

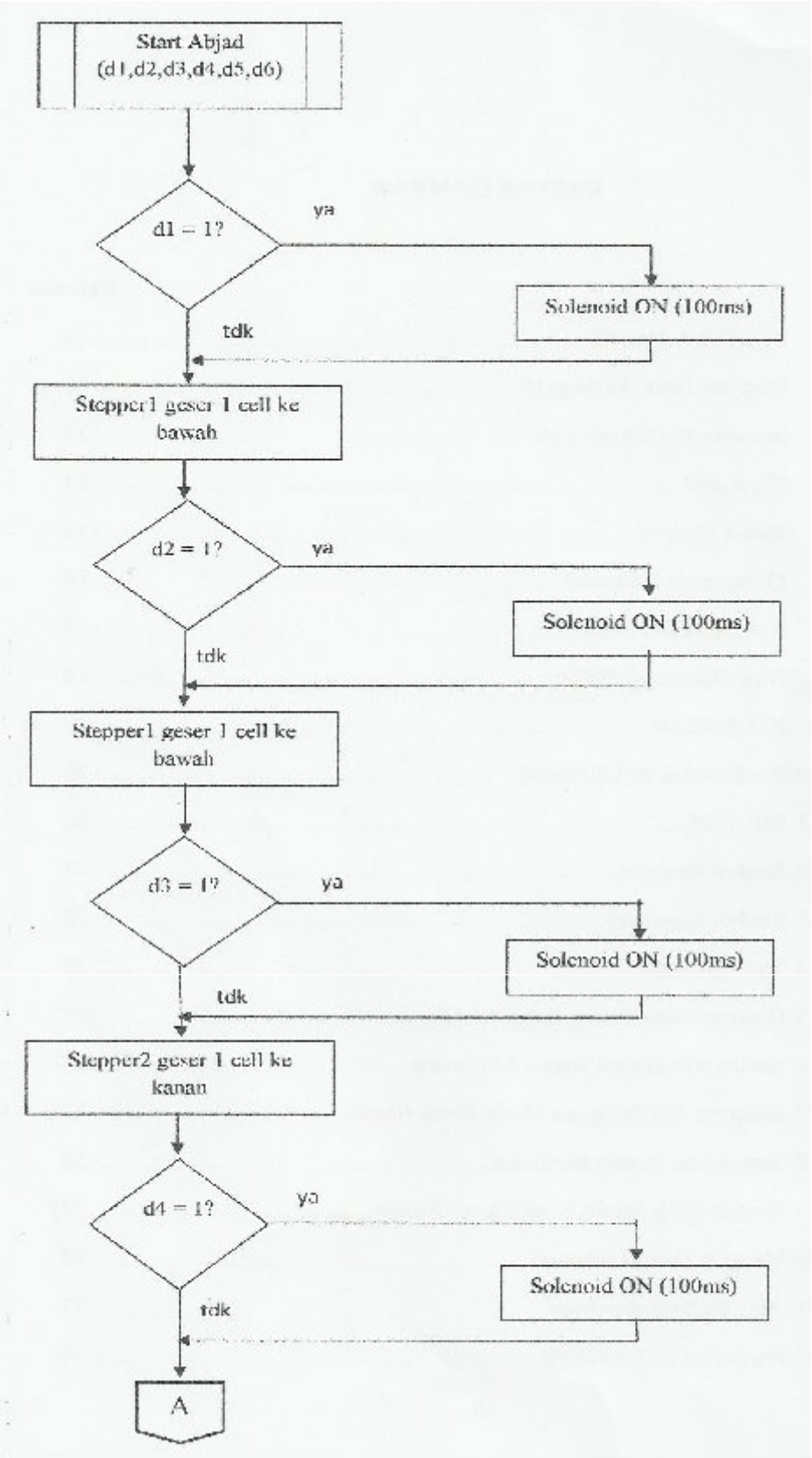
Sub program abjad dapat dideklarasikan sebagai berikut :

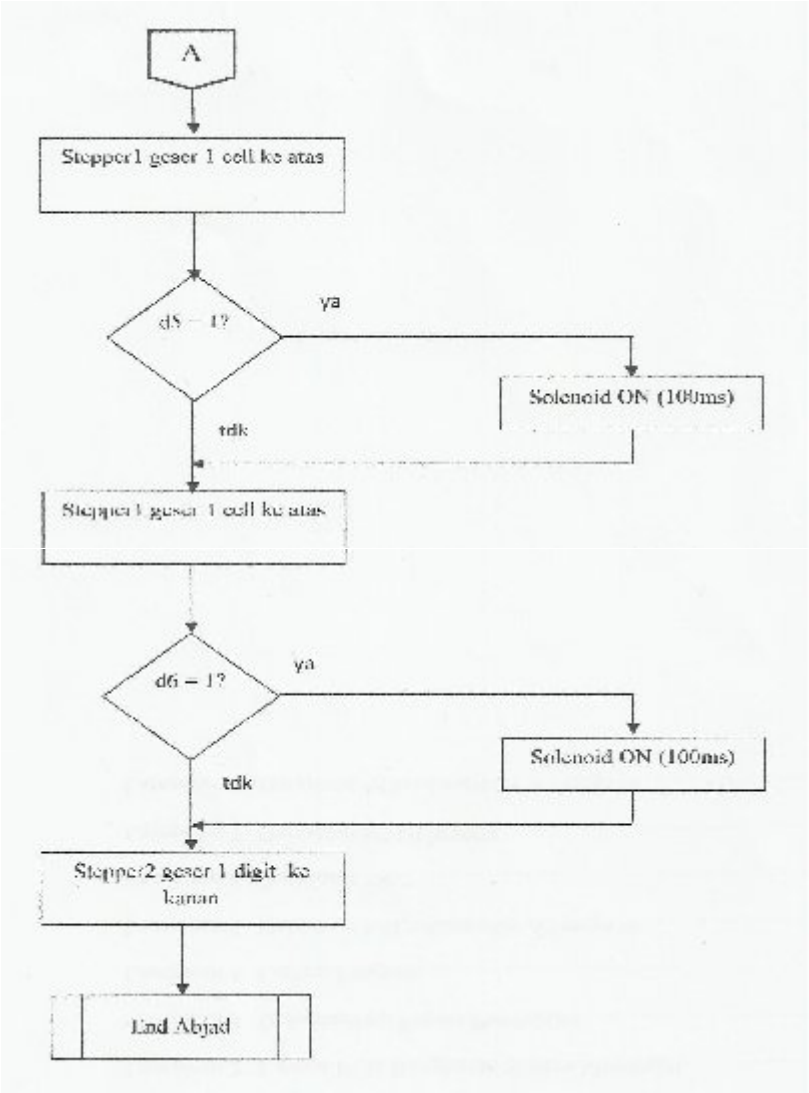
1. Step1 : Star Abjad (d1,d2,d3,d4,d6)
2. Step2 : apakah d1=1?
3. Step3 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.
4. Step4 : stepper 1 putar ke bawah 1cell.
5. Step5 : apakah d2=1?
6. Step6 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.
7. Step7 : stepper 1 putar ke bawah 1cell.
8. Step8 : apakah d3=1?
9. Step9 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.
10. Step10 : stepper2 putar ke kanan 1cell.
11. Step11 : apakah d4=1?
12. Step12 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.
13. Step12 : stepper 1 putar ke atas 1cell.
14. Step12 : apakah d5=1?
15. Step12 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.
16. Step12 : stepper 1 putar ke atas 1cell.
17. Step12 : apakah d6=1?
18. Step12 : jika ya, solenoid ON (100ms), jika tidak solenoid tetap OFF.

19. Step12 : stepper 2 putar ke kanan 1digit.
20. Step12 : selesai

Flowchart atau diagram alir dari program tersebut dapat digambarkan sebagai berikut :




Start Abjad
(d1,d2,d3,d4,d5,d6)
d1 = 1?
ya
Solenoid ON (100ms)
tdk
Stepper1 geser 1 cell ke bawah
d2 = 1?
ya
Solenoid ON (100ms)
tdk
Stepper1 geser 1 cell ke bawah
d3 = 1?
ya
Solenoid ON (100ms)
tdk
Stepper2 geser 1 cell ke kanan
d4 = 1?
ya
Solenoid ON (100ms)
tdk
A



Gambar 17. Diagram Alir Program Mesin Ketik *Braille*

Dalam perancangan program terdapat berbagai macam metode seperti penggunaan bahasa pemrograman C/C++, *Bascom*, *Basic* dan *Asembler*. Namun dalam proyek ini penulis menggunakan bahasa C dengan bantuan program *software Code Vision AVR* (*CVAVR*). Dalam *CVAVR* pemrograman dapat dibuat dengan lebih mudah dengan tersedianya fitur generator pada *CVAVR* seperti penggunaan *PORT*, *ADC, PWM* dan komunikasi yang lain dapat diseting dengan mudah. *CVAVR* akan menghasilkan *code-code* bahasa pemrograman dalam bentuk *file-file* seperti hex, c, asm, coff. *File* hex inilah yang akan di *download*-kan kedalam *flash memory* pada IC ATmega 16.

E. Langkah Kerja

Langkah kerja pembuatan mesin ketik huruf *braille* elektronik berbasis Mikrokontroller ATmega16 adalah sebagai berikut :

1. Pembuatan skema rangkaian.
2. Pembuatan *layout* jalur PCB
3. Pembuatan jalur PCB
4. Pemasangan komponen
5. Pembuatan *software*
6. Pemasangan rangkaian
7. Pemrograman mikrokontroler ATmega16
8. Pengujian unjuk kerja

F. Penjadwalan

Dalam penyelesaian proyek akhir ini, mahasiswa akan melakukan tahapan sebagai berikut :

1. Studi Pustaka

   Mencari dan mempelajari sumber referensi mengenai teori-teori yang dapat mendukung terselesaikannya proyek akhir ini

2. Konsultasi

   Mahasiswa akan berkonsultasi dengan pembimbing maupun dosen terkait dengan proyek akhir ini.

3. Praktik Langsung

   Mahasiswa akan merancang, merakit, dan mengevaluasi secara langsung konsep rancangan rangkaian yang akan dibuat.

Tabel 6. Rencana Kegiatan

NO	KEGIATAN	Minggu ke-									
		1	2	3	4	5	6	7	8	9	10
1	Studi Pustaka										
2	Perencanaan dan pembuatan *Hardware*										
3	Perencanaan dan pembuatan *Software*										
4	Uji coba dan analisis data										
5	Penyusunan laporan										

F. Biaya

Pembuatan mesin ketik huruf *braille* elektronik berbasis mikrokontroller ATmega16 memerlukan beberapa komponen tertentu. Rincian biaya pembuatan dapat dilihat pada tabel berikut.

Tabel 7. Rincian Biaya

No	Nama Barang	Jumlah	Harga Satuan (Rp)	Jumlah (Rp)
1	ATmega16	1	50.000	50.000
2	*Keyboard*	1	30.000	30.000
3	PCB 20cm x 10cm	1	5.000	5.000
4	*Motor stepper*	2	200.000	400.000
5	Solenoid	1	85.000	85.000
6	IC ULN 2003	1	15.000	15.000
7	IC L293D	1	22.000	22.000
8	IC AN 7805	1	2.000	2.000
9	IRF 3205	1	12.000	12.000
23	Lain-lain			30.000
Jumlah Total				**651.000**

BAB IV

PROSES, HASIL DAN PEMBAHASAN

A. Proses

Proses pembuatan Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 terdiri dari perancangan dan pembuatan alat yang dilanjutkan dengan perancangan dan pembuatan program.

1. Perancangan dan Pembuatan Alat

Proses pembuatan *hardware* Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 meliputi langkah-langkah berikut :

a. Pembuatan Skema Rangkaian.

Pembuatan skema rangkaian dilakukan menggunakan perangkat lunak Isis Proteus. Hasil penggambaran skema rangkaian Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 dapat dilihat pada Lampiran 1.

b. Pembuatan *Layout* PCB

Pembuatan *layout* PCB dilakukan menggunakan perangkat lunak Ares Proteus. Hasil penggambaran *layout* PCB dapat dilihat pada Lampiran 2.

c. Pembuatan PCB

Proses pembuatan PCB Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 terdiri atas langkah-langkah sebagai berikut :

1) Pencetakan *layout* pada kertas.
2) Fotokopi xerox *layout* menggunakan kertas *glossy*.
3) Penyablonan *layout* ke PCB dengan cara disetrike selama ± 10 menit.

4) Perendaman dengan air untuk menghilangkan kertas yang tidak perlu.
5) Pelarutan lapisan tembaga pada PCB menggunakan larutan feriklorit (*ferrichloride,* $FeCl_3$) selama ± 20 menit.
6) Pembersihan sisa sablon menggunakan bensin.
7) Pengeboran PCB pada titik-titik kaki komponen, menggunakan mata bor 0,8mm dan 1mm.

d. Pemasangan Komponen

Memasang seluruh komponen yang terdapat pada rangkaian dengan urutan:

1) Menyiapkan komponen yang dibutuhkan.
2) Memasang komponen dari ukuran paling kecil terlebih dahulu.
3) Menyolder kaki komponen sampai semua komponen terpasang.
4) Memeriksa hasil pemasangan komponen baik secara visual maupun menggunakan alat ukur.

e. Pemasangan Rangkaian

Rangkaian selanjutnya dihubungkan dengan komponen-komponen eksternal seperti *motor stepper*, solenoid, dan *keyboard.* PCB dipasang pada *body* printer menggunakan baut agar lebih ringkas. Dokumentasi proses pembuatan PCB hingga pemasangan rangkaian pada *printer* dapat dilihat pada lampiran 3.

2. Perancangan dan Pembuatan Program

Program Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 dibuat mengacu pada diagram blok yang terdapat pada Bab III.

```
#include <mega16.h>

#include <delay.h>

int i=1;
unsigned char dat[9];

int x;
unsigned char data, q, w, d, ent;

int slide[5]={0b00000010,  0b00001000, 0b00000001,
0b00000100}; //data slider
int step[5]={0b00010010, 0b00000110, 0b00001100,
0b00011000}; //data steper

void nol()
{dat[1]=0; dat[2]=0; dat[3]=0; dat[4]=0;  dat[5]=0;
dat[6]=0; dat[7]=0; dat[8]=0;}
```

Script 1 berisi inisialisasi beberapa variabel yang digunakan. *Script* tersebut juga menunjukan bahwa mikrokontroller yang digunakan adalah ATmega16.

```
PORTA=0x00;
DDRA=0xFF;

PORTB=0x00;
DDRB=0xFF;

PORTC=0x00;
DDRC=0xFF;

PORTD=0x06;
DDRD=0x00;
```

Script 2. Inisialisasi PORT

Script diatas menjunjukan bahwa PORT A, PORT B dan PORT C didefinisikan sebagai output sedangkan PORT D berfungsi sebagai input. input.

```
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x03;    //R 101 0101 mcu2
MCUCSR=0x00;
GIFR=0x40;
```

Script 3. Inisialisasi Interupt

Interupt pada program ini bermode *falling edge* yang artinya akan mengeksekusi program saat terjadi penurunan clock dari 1 ke 0.

```
i=i+1;

     if (i==2 & PIND.1==1){dat[1]=1; delay_us(1);}
else if (i==2 & PIND.1==0){dat[1]=0; delay_us(1);}
else if (i==3 & PIND.1==1){dat[2]=2; delay_us(1);}
else if (i==3 & PIND.1==0){dat[2]=0; delay_us(1);}
else if (i==4 & PIND.1==1){dat[3]=4; delay_us(1);}
else if (i==4 & PIND.1==0){dat[3]=0; delay_us(1);}
else if (i==5 & PIND.1==1){dat[4]=8; delay_us(1);}
else if (i==5 & PIND.1==0){dat[4]=0; delay_us(1);}
else if (i==6 & PIND.1==1){dat[5]=16;  delay_us(1);}
else if (i==6 & PIND.1==0){dat[5]=0;   delay_us(1);}
else if (i==7 & PIND.1==1){dat[6]=32;  delay_us(1);}
else if (i==7 & PIND.1==0){dat[6]=0;   delay_us(1);}
else if (i==8 & PIND.1==1){dat[7]=64;  delay_us(1);}
else if (i==8 & PIND.1==0){dat[7]=0;   delay_us(1);}
else if (i==9 & PIND.1==1){dat[8]=128; delay_us(1);}
else if (i==9 & PIND.1==0){dat[8]=0;   delay_us(1);}
//konfigurasi keyboard
else if (i==10)
    {i=0;

            d=d+1;


                if (d==3) {
                data = dat[1] + dat[2] + dat[3] + dat[4] +
dat[5] + dat[6] + dat[7] + dat[8];
                PORTA=data; delay_ms(200); d=0; }
      }
```

Pada saat *keyboard* ditekan data yang akan masuk berjumlah sangat banyak tetapi data yang diambil untuk selanjutnya dibaca adalah data ketiga yang masuk. Hal ini dikarenakan data yang masuk ketiga lebih valid dari data pertama dan kedua.

```
while (1)
      {

       if (data==57)   //a
         {abjad(1,0,0,0,0,0);}
       if (data==101)   //b
          {abjad(1,1,0,0,0,0);}
       if (data==67)  //c
          {abjad(1,0,0,0,0,1);}
       if (data==71)   //d
          {abjad(1,0,0,0,1,1);}
       if (data==73)  //e
          {abjad(1,0,0,0,1,0);}
       if (data==87)  //f
          {abjad(1,1,0,0,0,1);}
       if (data==105)   //g
          {abjad(1,1,0,0,1,1);}
       if (data==103)  //h
          {abjad(1,1,0,0,1,0);}
       if (data==7)   //i
          {abjad(0,1,0,0,0,1);}
       if (data==119)   //j
          {abjad(0,1,0,0,1,1);}
       if (data==5)   //k
          {abjad(1,0,1,0,0,0);}
```

```
      if (data==45)   //1
          {abjad(0,0,1,1,1,1);
           abjad(1,0,0,0,0,0);}
       if (data==61)   //2
          {abjad(0,0,1,1,1,1);
           abjad(1,1,0,0,0,0);}
       if (data==77)   //3
          {abjad(0,0,1,1,1,1);
           abjad(1,0,0,0,0,1);}
       if (data==75)   //4
          {abjad(0,0,1,1,1,1);
           abjad(1,0,0,0,1,1);}
       if (data==93)   //5
          {abjad(0,0,1,1,1,1);
           abjad(1,0,0,0,1,0);}
       if (data==109)   //6
          {abjad(0,0,1,1,1,1);
           abjad(1,1,0,0,0,1);}
       if (data==123)   //7
          {abjad(0,0,1,1,1,1);
           abjad(1,1,0,0,1,1);}
       if (data==125)   //8
          {abjad(0,0,1,1,1,1);
           abjad(1,1,0,0,1,0);}
       if (data==13)   //9
          {abjad(0,0,1,1,1,1);
           abjad(0,1,0,0,0,1);}
       if (data==11)   //0
          {abjad(0,0,1,1,1,1);
           abjad(0,1,0,0,1,1);}

       ///tanda baca
       if (data==37)   // !      "'
           {abjad(0,1,1,0,1,0);}
       if (data==25)   // .      :;
           {abjad(0,0,1,0,0,0);}
       if (data==41)   // ,      {[
           {abjad(0,1,0,0,0,0);}
       if (data==29)   // .      _-
           {abjad(0,0,1,1,0,0);}
       if (data==51)   // .      shift
           {abjad(0,1,0,1,1,0);}
       if (data==49)   // ?     capsl
           {abjad(0,1,1,1,0,0);}
       if (data==135)   // !     F7
           {abjad(0,0,0,1,0,0);}


       if (data==83)   //sepasi
          {abjad(0,0,0,0,0,0);}

       if (data==193)   //ctrl
          {stpper (0,32);
           slider (0,130); data=0; PORTB=0; PORTC=0;  }
       if (data==21)   //F8
          {stpper (0,32);
           data=0; PORTB=0; PORTC=0;
           }
       if (data==3) //   _    f9
          { stpper(0,10); PORTB=0; PORTC=0;}
       if (data==19) //   _    f10
         { stpper(1,10); PORTB=0; PORTC=0;}

      if (data==113) //  _ f11
         { slider(0,5);  PORTB=0; PORTC=0; }
      if (data==15) //  _ f12
         { slider(1,5);  PORTB=0; PORTC=0; }
     }
}
```

Script 5. Subroutine abjad

Pada masing-masing tombol *keyboard* yang ditekan maka data yang diterbaca akan berbeda-beda. Hasil pengujian data *keyboard* yang telah tertuang pada tabel data *keyboard* selanjutnya dimasukan ke dalam program seperti pada *Script* di atas.

```
void solenoid()
 {
 PORTD.7=1;
 delay_ms(80);
 PORTD.7=0;
 delay_ms(20);
 }
```

Script 6. Subroutine Solenoid

Solenoid akan aktif jika PORTD.7 berilai 1 dan tidak aktif jika PORTD.7 bernilai 0.

```
 unsigned char stpper(unsigned char a, unsigned char b)
 {  for (q=0;q<=b;q++)
 {
 if (a==0)
 {
 for (x=0;x<=3;x++)
 { PORTC=step[x]; delay_ms(4);}
 }
 if (a==1)
 {
 for (x=0;x<=3;x++)
 { PORTC=step[3-x]; delay_ms(4);}
 }
 }

 return;
 }
```

Script 7. Subroutine stepper 1

Script di atas menunjukan bahwa jika nilai variabel a=0 maka *motor stepper1* akan berputar ke bawah, dan jika a=1 *motor stepper1* akan berputar ke atas.

```
unsigned char slider(unsigned char r, unsigned
char s)
 {  for (w=0;w<=s;w++)
    {
    if (r==0)
      { for (x=0;x<=3;x++)
        {
        PORTB=slide[x];
        delay_ms(4);
        }
      }
    if (r==1)
      { for (x=0;x<=3;x++)
        {
        PORTB=slide[3-x];
        delay_ms(4);
        }
      }
     }
```

Script 8. Subroutine *Stepper2*

Script di atas menunjukan bahwa jika nilai variabel r=0 maka *motor stepper2* akan berputar ke kiri, dan jika r=1 *motor stepper2* akan berputar ke kanan.

```
void abjad(unsigned char d1, unsigned char d2, unsigned char
d3, unsigned char d4, unsigned char d5, unsigned char d6)
{       if (d1==1)
           {solenoid();}
            delay_ms(100);
            stpper (0,8);         //0 step down
        if (d2==1)
           {solenoid();}
           delay_ms(100);
            stpper (0,8);
        if (d3==1)
           {solenoid();}
            slider (1,1);     //1 step left   for self
        if (d4==1)
           {solenoid();}
           delay_ms(100);
            stpper (1,8);
        if (d5==1)
           {solenoid();}
           delay_ms(100);
            stpper (1,8);
        if (d6==1)
           {solenoid();}
            slider (1,2);         //1 step left
            data=0;
            nol();
              ent=ent+1
            PORTB=0; PORTC=0;
}
       If (ent==20)
           { stpper (0,32);
              slider(0,60);
              data=0; PORTB=0; PORTC=0;
              ent=0;
              }
```

Script 9. Program Eksekusi

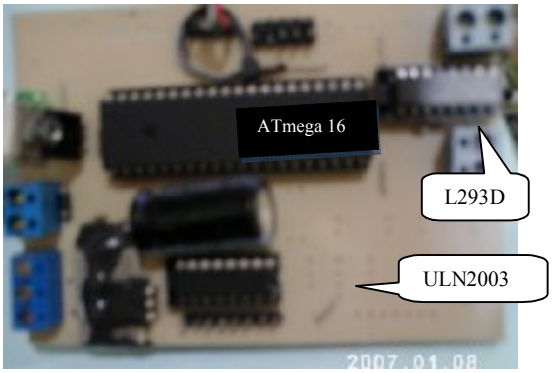
Script di atas merupakan program utama di mana terjadi proses eksekusi. Dengan membaca data dari *keyboard* maka Solenoid, *stepper1,* dan *stepper2* akan bekerja sesuai fungi yang maksud.

B. Hasil

1. *Hardware* Tombol Kuis Berbasis Mikrokontroler ATmega16

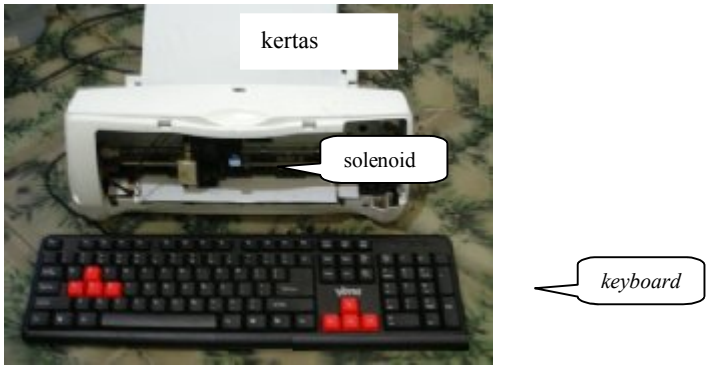
a. Bentuk Fisik Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroller ATmega16

Rangkaian sistem minimum yang dilengkapi dengan IC ULN2003 dan IC L293D tampak seperti gambar berikut :



Gambar 18. Rangkaian Sistem Minimum

Sedangkan bentuk fisik secara keseluruhannya adalah sebagai berikut :



Gambar 19. Bentuk Fisik Mesin Ketik Huruf *Braille* Elektronik berbasis mikrokontroller ATmega 16

b. Cara Kerja Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroller ATmega16.

Semua perintah kerja pada mesin ketik ini terdapat pada *keyboard.* Adapun beberapa tombol utama, yaitu F9 yang berfungsi menarik kertas kebawah dan tombol F10 untuk menarik kertas arah sebaliknya. Tombol F11 dan F12 berfungsi

untuk menggeser solenoid ke kiri dan ke kanan guna mendapatkan posisi yang tepat pada kertas. Untuk menghentikan perintah dari tombol-tombol tersebut kita dapat menggunakan tombol spasi, sementara untuk menurunkan solenoid satu baris kita dapat menggunakan tombol F8.

Pada mesin ketik ini, setiap tombol huruf, angka, dan tanda baca yang ditekan akan menghasilkan *output* berupa gerakan dua buah *motor stepper* dan solenoid sehingga akan menghasilkan motif huruf *braille* yang dimaksud pada kertas.

2. Program

Listing program Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroller ATmega16 dapat dilihat pada Lampiran 4.

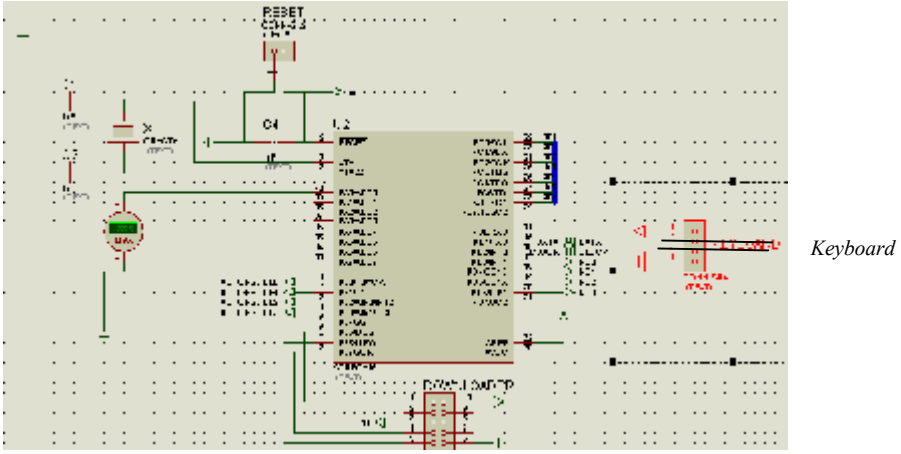
3. Pengujian

Hasil pengujian Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroller ATmega16 dapat diuraikan dengan menguji komonen-komponen utamanya. Beberapa komponen tersebut perlu diuji untuk menunjang pambuatan alat.

a. *Keyboard Votre sturdy ps2*

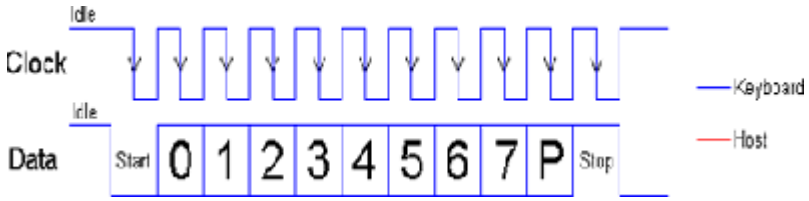
Pengujian data *keyboard* dilakukan dengan menghubungkan terminal data dari *keyboard* dengan PORTD.1 dan terminal clock dengan PORTD.2 pada ATmega 16. Kemudian output ATmega dikeluarkan melalui PORTA menggunakan program bahasa C. Ketika masing-masing tombol pada *keyboard*

ditekan kita dapat mengetahui data yang keluar dengan menggunakan Voltmeter pada masing-masing PORTA secara bergantian.



Gambar 20. Menguji Data *Keyboard*

Transmisi data, dari keyboard ke sistem, dilakukan dengan frame 11 bit. Bit pertama adalah bit start (logika 0), diikuti dengan 8 bit data (LSB first), satu bit paristas (paritas ganjil) dan bit stop (logika 1). Setiap bit harus dibaca pada sisi turun dari clock.

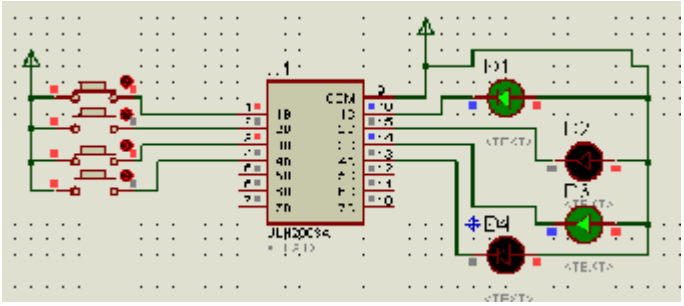


Gambar 21. Sinyal clock dan data

b. IC ULN2003

Pengujian IC ULN2003 ini menggunakan 4 saklar dengan tegangan 5V dan 4 buah lampu led dengan tegangan 5V untuk menguji *driver motor stepper*.

Keempat saklar dihubungkan dengan kaki 1 sampai 4 pada IC ULN2003 dan keempat lampu led dihubungkan dengan kaki 13 sampai 16 pada IC ULN2003.



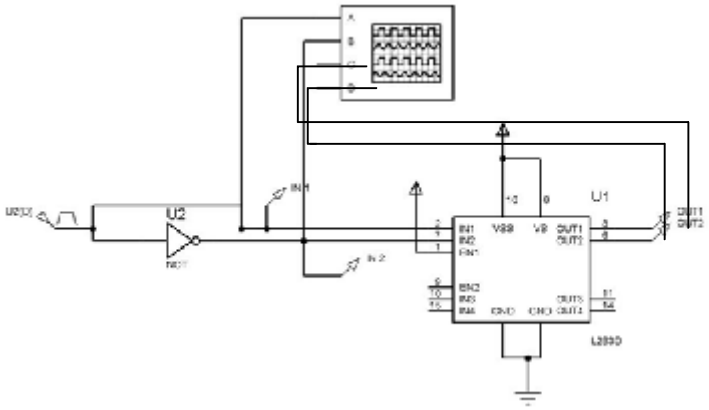
Gambar 22. Pengujian IC ULN2003

Adapun tabel hasil pengujiannya adalah sebagai berikut:

Tabel 8. Hasil Pengujian IC ULN2003

No	Keadaan saklar	Led yang menyala
1	0000	0000
2	0001	0001
3	0010	0010
4	0011	0011
5	0100	0100
6	0101	0101
7	0110	0110
8	0111	0111
9	1000	1000
10	1001	1001
11	1010	1010
12	1011	1011
13	1100	1100
14	1101	1101
15	1110	1110
16	1111	1111

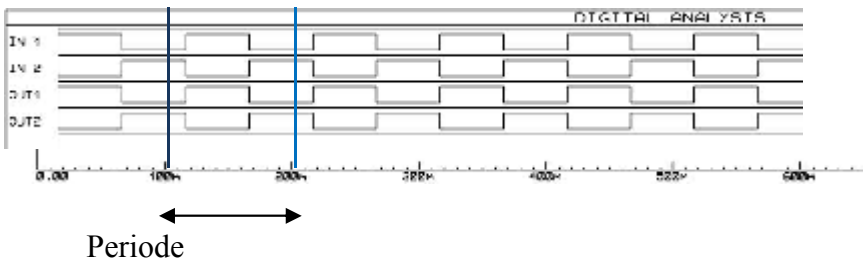
c. IC L293D



Gambar 23. Rangkaian simulasi IC L293D

Gambar diatas merupakan rangkaian simulasi sebagai pengamatan dari kerja alat dan dapat di ketahui output yang dihasilkan dengan pengamatan output pada gambar dibawah ini.

Pengamatan output rangkaian analisa digital.



Gambar 24. Pulsa input dan output

Analisa gambar diatas merupakan hasil dari output rangkaian simulasi yang dapat di hitung dengan perhitugan teori dan menggunakan logika input 1 dan 0 (not).

Tabel 9. Keadaan logika input output

IN	OUT
0	0
1	1

C. Pembahasan

Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16 terdiri atas 3 bagian. Bagian pertama terdiri dari sebuah *keybard* dengan spesifikasi *votre sturdy ps2. Keyboard* tersebut menghasilkan data serial pada setiap tombol yang ditekan. Hasil penjumlahan dari dat-data tersebut yang selanjutnya diakumulasi menjadi sebuah data desimal guna memudahkan dalam membedakannya. Data yang dihasilkan tiap merk dan tipe *keyboad* tidak sama pada setiap tombol yang sama. Pada *keyboard* juga biasanya terdapat beberapa data yang sama untuk tombol yang berbeda. Hasil pengujian data *keyboard Votre* dapat dilihat pada tabel 7 yang terdapat pada BAB III.

Bagian kedua yaitu sistem minimum. Pada rangkaian ini terdapat sebuah mikrokontrolller ATmega16 sebagai pengontrol kerja seluruh bagian mesin ketik. Rangkaian ini juga dilengkapi IC ULN2003 sebagai *driver* untuk *motor stepper1* dan IC L293D sebagai *driver* untuk *motor stepper2.*

Bagian ketiga adalah tiga buah *output* mekanik. Pada bagian ini terdapat dua buah *motor stepper* dan sebuah solenoid. *Motor stepper* pertama berfungsi untuk menarik kertas sedangkan *motor stepper* kedua berfungsi menggeser posisi solenoid.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Dari hasil pengujian yang telah dilakukan tehadap "Mesin Ketik Huruf *Braille* Elektronik Berbasis Mikrokontroler ATmega16", didapat beberapa kesimpulan sebagai berikut :

1. Hasil *hardware* yang telah dibuat dapat bekerja dengan baik sesuai dengan konsep rancangan. Kendala yang terjadi pada pembuatan *hardware* antara lain adalah penyesuaian rangkaian sistem minimum dengan rangka printer bekas sebagai *body* alat. *Input* dan *Output* dari sistem minimum telah disesuaikan dengan printer bekas yang digunakan sehingga alat dapat bekerja secara maksimal.
2. Hasil *software* yang telah dibuat dapat bekerja dengan baik sesuai dengan algoritma program yang telah dibuat sebelumnya. Hal yang harus dilakukan sebelum membuat program secara keseluruhan adalah memeriksa data yang keluar pada setiap tomol *keyboard* secara manual sehingga membutuhkan waktu cukup lama.
3. Hasil penulisan mesin ketik dengan tulisan yang diharapkan dapat dibandingkan sebagai berikut :

a. Jarak antara titik-titik vertikal pada masing-masing huruf adalah 2,5mm sesuai dengan tulisan yang diharapkan.

b. Jarak antara titik-titik horisontal pada masing-masing huruf adalah 3mm sementara tulisan yang diharapkan adlah 2,5mm.

B. Kelemahan Alat

Mesin Ketik Huruf *Braille* Elektronik Nilai Berbasis Mikrokontroler ATmega16 memiliki beberapa kelemahan, yaitu :

1. Hanya dapat menggunakan *keyboard Votre sturdy ps2*, jika *keyboard* diganti dengan tipe lein maka hasil *output* akan berbeda.
2. Tidak terdapat pengatur ketebalan kertas sehingga jika kertas yang digunakan terlalu tipis maka bisa berlubang.

C. Saran

Dalam pembuatan proyek akhir ini tentu saja terdapat kekurangan, sehingga diperlukan pengembangan guna menyempurnakan proyek akhir ini. Oleh karena itu penulis memberikan saran :

1. Memodifikasi program agar semua tipe *keyboard* dapat digunakan.
2. Menambahkan rangkaian pengatur gerakan solenoid sehingga dapat disesuaikan dengan ketebalan kertas yang digunakan.

DAFTAR PUSTAKA

Andrianto Heri. (2008). *Pemrograman Mikrokontroler AVR ATmega16*. Bandung: Informatika.

Daryanto. (2010). *Ketrampilan Kejuruan Teknik Elektronika.* Bandung: Satu Nusa.

Anonim. (2009). *L293D Sebagai Driver Motor.* Diambil pada tanggal 3 September 2012, dari http://ini-robot.blogspot.com/2012/06/driver-motor-l293d-motor-driver-ic.html

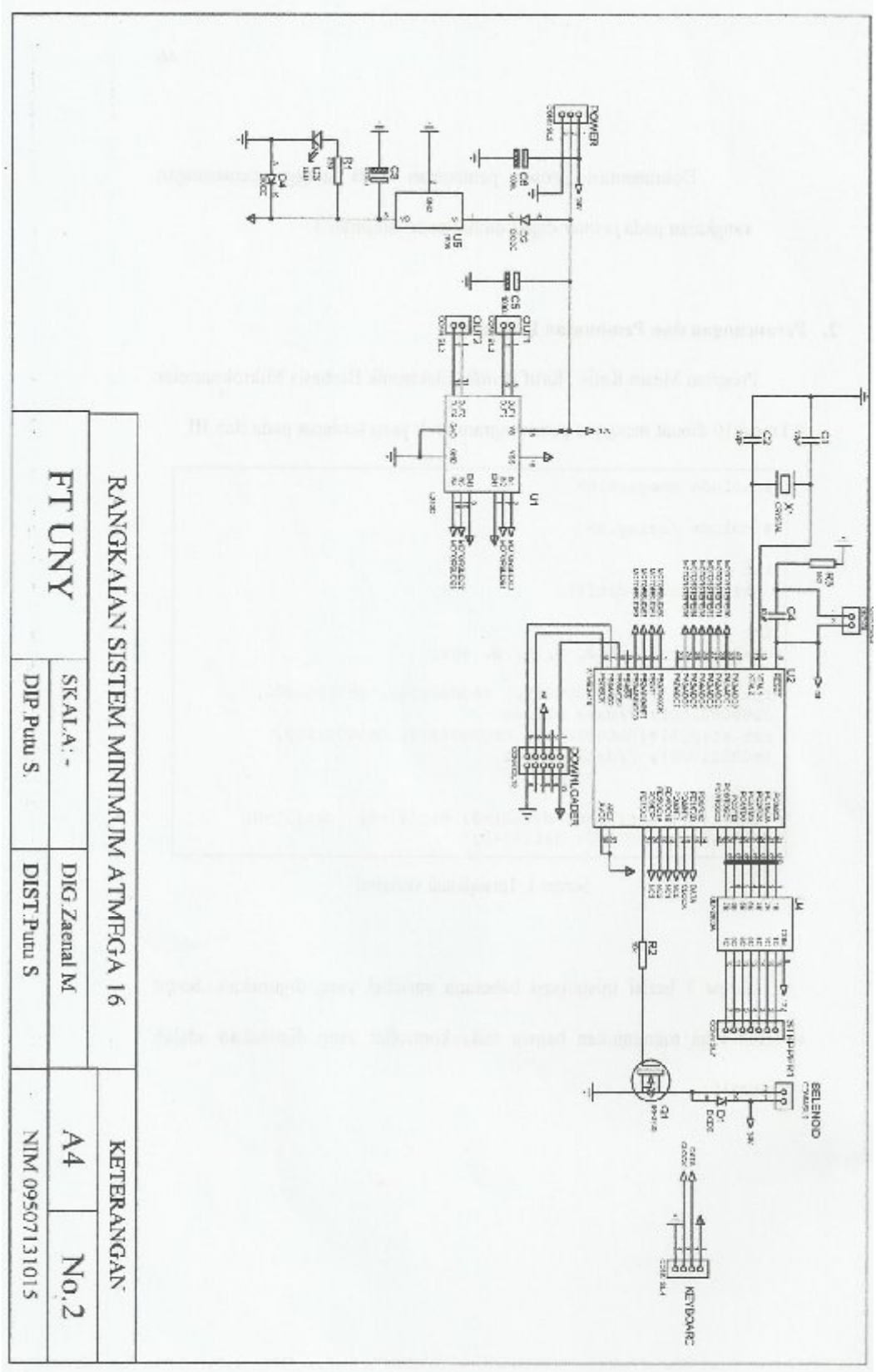
Heriani. (2010). *Huruf Braille.* Diambil pada tanggal 3 September 2012, dari http://pendidikanalb.blogspot.com/2011/03/sistem-huruf-braille-gar-0606094320.html

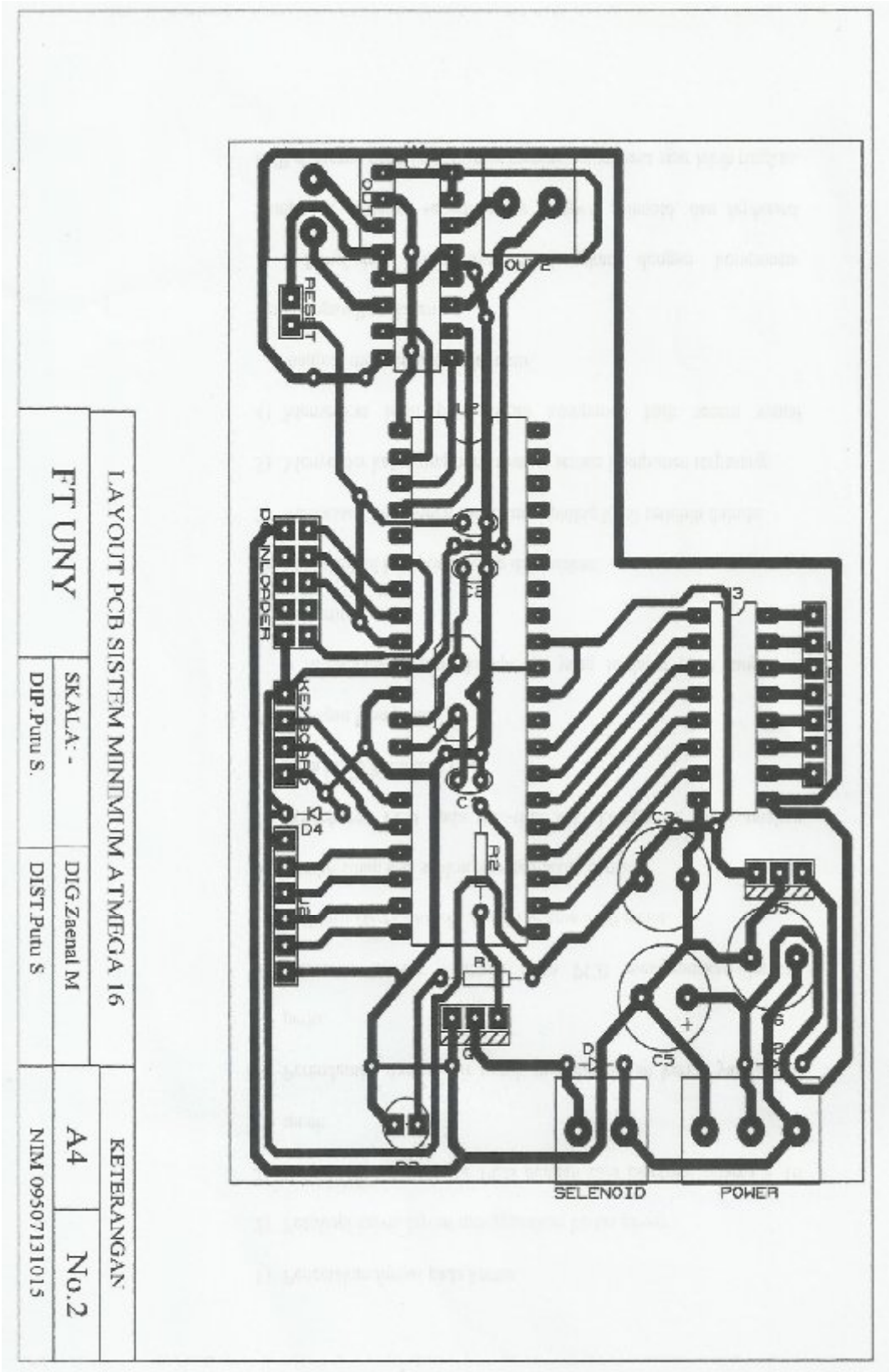
Himawan Al. (2011). *Dasar Motor Stepper.* Diambil pada tanggal 14 September 2012, dari http://elektronika-elektronika.blogspot.com/2007/04/motor-stepper.html

Kandar Adhyzal. (2010). *Cara Kerja Keyboard.* Diambil pada tanggal 3 September 2012, dari http://id.shvoong.com/writing-and-speaking/2012511-cara-kerja-keyboard/

Nawazir. (2012). *Pengertian Solenoida.* Diambil pada tanggal 14 September, dari http://www.elektronikamania.com/2011/12/solenoid.html

LAMPIRAN


POWER
OUT1
OUT2
U1
U2
U5
U4
X1
SELENOID
KEYBOARD
RANGKAIAN SISTEM MINIMUM ATMEGA 16
FT UNY
SKALA : -
DIP.Putu S.
DIG.Zaenal M.
DIST.Putu S
KETERANGAN
A4
No.2
NIM 09507131015


LAYOUT PCB SISTEM MINIMUM ATMEGA 16
FT UNY
SKALA: -
DIG.Zaenal M
DIP.Putu S.
DIST.Putu S
A4
No.2
KETERANGAN
NIM 09507131015
RESET
DOWNLOADER
KEYBOARD
SELENOID
POWER
D4
R2
R1
C3
C5
C
D

Doumentasi Pembuatan Alat

Penyablonan *Layout pada PCB*

Pembersihan Kertas dari PCB

Perendaman dengan *ferrichloride*

Proses Pengeboran PCB

Proses Penyolderan

Pemasangan Rangkaian pada *Printer*

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 9/4/2012
Author  : Zaenal Muttaqien
Company : research unit
Comments:

edited 7 oktober 2012
Chip type               : ATmega16
Program type            : Application
AVR Core Clock frequency: 12.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega16.h>

#include <delay.h>

int i=1;
unsigned char dat[9];

int x;
unsigned char data, q, w, d;

int slide[6]={0b00000010, 0b00001000, 0b00000001, 0b00000100, 0b0000000,0b0000000}; //data slider
int step[6]={0b00010010, 0b00000110, 0b00001100, 0b00011000}; //data steper


void nol()
{dat[1]=0; dat[2]=0; dat[3]=0; dat[4]=0;  dat[5]=0; dat[6]=0; dat[7]=0; dat[8]=0;}

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
i=i+1;
```

```
    if (i==2 & PIND.1==1){dat[1]=1; delay_us(1);}
else if (i==2 & PIND.1==0){dat[1]=0; delay_us(1);}
else if (i==3 & PIND.1==1){dat[2]=2; delay_us(1);}
else if (i==3 & PIND.1==0){dat[2]=0; delay_us(1);}
else if (i==4 & PIND.1==1){dat[3]=4; delay_us(1);}
else if (i==4 & PIND.1==0){dat[3]=0; delay_us(1);}
else if (i==5 & PIND.1==1){dat[4]=8; delay_us(1);}
else if (i==5 & PIND.1==0){dat[4]=0; delay_us(1);}
else if (i==6 & PIND.1==1){dat[5]=16;  delay_us(1);}
else if (i==6 & PIND.1==0){dat[5]=0;   delay_us(1);}
else if (i==7 & PIND.1==1){dat[6]=32;  delay_us(1);}
else if (i==7 & PIND.1==0){dat[6]=0;   delay_us(1);}
else if (i==8 & PIND.1==1){dat[7]=64;  delay_us(1);}
else if (i==8 & PIND.1==0){dat[7]=0;   delay_us(1);}
else if (i==9 & PIND.1==1){dat[8]=128; delay_us(1);}
else if (i==9 & PIND.1==0){dat[8]=0;   delay_us(1);}          //konfigurasi keyboard
else if (i==10)
   {i=0;
         d=d+1;
            if (d==3) {
            data = dat[1] + dat[2] + dat[3] + dat[4] + dat[5] + dat[6] + dat[7] + dat[8];
            PORTA=data; delay_ms(200); d=0; }
     }

}

//////////////////////steper up (1) down(0)///////////////
unsigned char stpper(unsigned char a, unsigned char b)
 {  for (q=0;q<=b;q++)
   {
   if (a==0)
    {
     for (x=0;x<=3;x++)
      { PORTC=step[x]; delay_ms(4);}
    }
   if (a==1)
    {
     for (x=0;x<=3;x++)
      { PORTC=step[3-x]; delay_ms(4);}
    }
   }

  return;
 }

//////////////////slider run to left (1) right (0) from self////////////
unsigned char slider(unsigned char r, unsigned char s)
```

```
{  for (w=0;w<=s;w++)
   {
   if (r==0)
     { for (x=0;x<=3;x++)
       {
       PORTB=slide[x];
       delay_ms(4);
       }
     }
   if (r==1)
     { for (x=0;x<=3;x++)
       {
       PORTB=slide[3-x];
       delay_ms(4);
       }
     }
    }
   return;
}

//////////////////////////////solenoida//////////
void solenoid()
{
PORTD.7=1;
delay_ms(80);
PORTD.7=0;
delay_ms(20);
}

/////////////////////////////delete sisa data////////
void abjad(unsigned char d1, unsigned char d2, unsigned char d3, unsigned char d4,
unsigned char d5, unsigned char d6)
{      if (d1==1)
         {solenoid();}
          delay_ms(100);
          stpper (0,8);        //0 step down
       if (d2==1)
         {solenoid();}
         delay_ms(100);
          stpper (0,8);
       if (d3==1)
         {solenoid();}
          slider (1,1);        //1 step left   for self
       if (d4==1)
         {solenoid();}
         delay_ms(100);
          stpper (1,8);
```

```
      if (d5==1)
        {solenoid();}
        delay_ms(100);
         stpper (1,8);
      if (d6==1)
        {solenoid();}
         slider (1,2);        //1 step left
         data=0;
         nol();
         PORTB=0; PORTC=0;
}
void main(void)
{
PORTA=0x00;
DDRA=0xFF;
PORTB=0x00;
DDRB=0xFF;
PORTC=0x00;
DDRC=0xFF;
PORTD=0x06;
DDRD=0x00;

nol();
while (1)
    {
     if (data==57)   //a
       {abjad(1,0,0,0,0,0);}
     if (data==101)   //b
       {abjad(1,1,0,0,0,0);}
     if (data==67)  //c
       {abjad(1,0,0,0,0,1);}
     if (data==71)   //d
       {abjad(1,0,0,0,1,1);}
     if (data==73)  //e
       {abjad(1,0,0,0,1,0);}
     if (data==87)  //f
       {abjad(1,1,0,0,0,1);}
     if (data==105)   //g
       {abjad(1,1,0,0,1,1);}
     if (data==103)  //h
       {abjad(1,1,0,0,1,0);}
     if (data==7)   //i
       {abjad(0,1,0,0,0,1);}
     if (data==119)   //j
       {abjad(0,1,0,0,1,1);}
     if (data==5)   //k
       {abjad(1,0,1,0,0,0);}
```

```
if (data==23)   //l
  {abjad(1,1,1,0,0,0);}
if (data==117)   //m
  {abjad(1,0,1,0,0,1);}
if (data==99)   //n
  {abjad(1,0,1,0,1,1);}
if (data==9)   //o
  {abjad(1,0,1,0,1,0);}
if (data==27)   //p
  {abjad(1,1,1,0,0,1);}
if (data==43)   //q
  {abjad(1,1,1,0,1,1);}
if (data==91)  //r
  {abjad(1,1,1,0,1,0);}
if (data==55)   //s
  {abjad(0,1,1,0,0,1);}
if (data==89)   //t
  {abjad(0,1,1,0,1,1);}
if (data==121)  //u
  {abjad(1,0,1,1,0,0);}
if (data==85)   //v
  {abjad(1,1,1,1,0,0);}
if (data==59)   //w
  {abjad(0,1,0,1,1,1);}
if (data==69)   //x
  {abjad(1,0,1,1,0,1);}
if (data==107)   //y
  {abjad(1,0,1,1,1,1);}
if (data==53)   //Z
  {abjad(1,0,1,1,1,0);}
if (data==45)   //1
  {abjad(0,0,1,1,1,1);
   abjad(1,0,0,0,0,0);}
if (data==61)   //2
  {abjad(0,0,1,1,1,1);
   abjad(1,1,0,0,0,0);}
if (data==77)   //3
  {abjad(0,0,1,1,1,1);
   abjad(1,0,0,0,0,1);}
if (data==75)   //4
  {abjad(0,0,1,1,1,1);
   abjad(1,0,0,0,1,1);}
if (data==93)   //5
  {abjad(0,0,1,1,1,1);
   abjad(1,0,0,0,1,0);}
if (data==109)   //6
  {abjad(0,0,1,1,1,1);
```

```
        abjad(1,1,0,0,0,1);}
      if (data==123)   //7
        {abjad(0,0,1,1,1,1);
        abjad(1,1,0,0,1,1);}
      if (data==125)   //8
        {abjad(0,0,1,1,1,1);
        abjad(1,1,0,0,1,0);}
      if (data==13)   //9
        {abjad(0,0,1,1,1,1);
        abjad(0,1,0,0,0,1);}
      if (data==11)   //0
        {abjad(0,0,1,1,1,1);
        abjad(0,1,0,0,1,1);}

      ///tanda baca
      if (data==37)   // !     "'
        {abjad(0,1,1,0,1,0);}
      if (data==25)   // .     :;
        {abjad(0,0,1,0,0,0);}
      if (data==41)   // ,     {[
        {abjad(0,1,0,0,0,0);}
      if (data==29)   // .     _-
        {abjad(0,0,1,1,0,0);}
      if (data==51)   // .     shift
        {abjad(0,1,0,1,1,0);}
      if (data==49)   // ?     capsl
        {abjad(0,1,1,1,0,0);}
      if (data==135)   // !     F7
        {abjad(0,0,0,1,0,0);}
     if (data==83)   //sepasi
        {abjad(0,0,0,0,0,0);}
      if (data==193)   //ctrl
        {stpper (0,32);
        slider (0,130); data=0; PORTB=0; PORTC=0;  }
      if (data==21)   //F8
        {stpper (0,32);
        data=0; PORTB=0; PORTC=0;  }
      if (data==3) //  _   f9
        { stpper(0,10); PORTB=0; PORTC=0;}
      if (data==19) //  _   f10
        { stpper(1,10); PORTB=0; PORTC=0;}
      if (data==113) //  _ f11
        { slider(0,5);  PORTB=0; PORTC=0; }
      if (data==15) //  _ f12
        { slider(1,5);  PORTB=0; PORTC=0; }
     }
}
```

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - 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
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

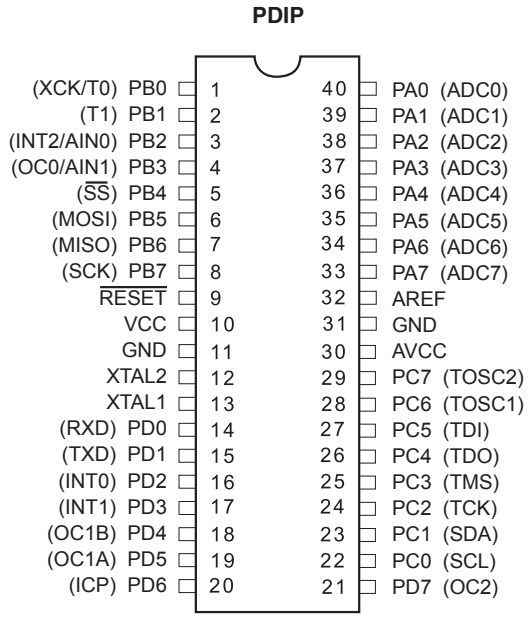
ATmega16
ATmega16L

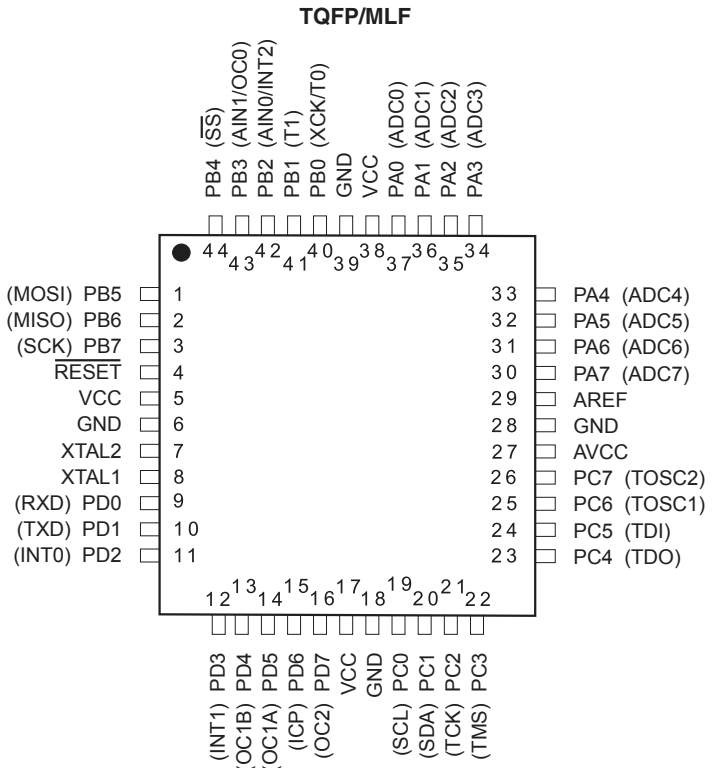
Preliminary

Rev. 2466E–AVR–10/02

Pin Configurations

Figure 1. Pinouts ATmega16





Disclaimer

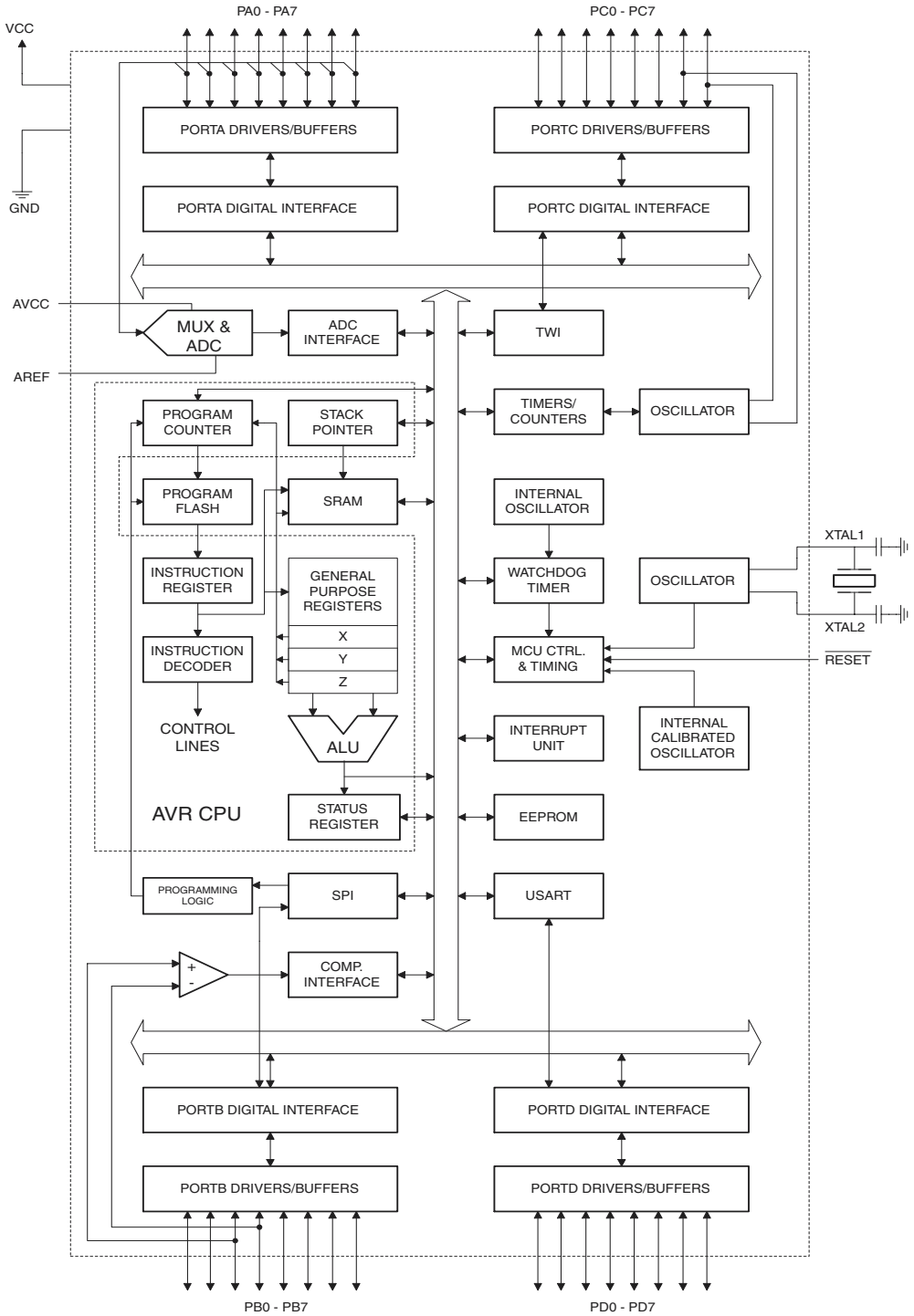
Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The 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 ATmega16 provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, 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 External 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. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile 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 ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)

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.

Port B also serves the functions of various special features of the ATmega16 as listed on page 55.

Port C (PC7..PC0)

Port C is an 8-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. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 58.

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 ATmega16 as listed on page 60.

$\overline{\text{RESET}}$

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 15 on page 35. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. 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.

AREF

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

About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C Compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.

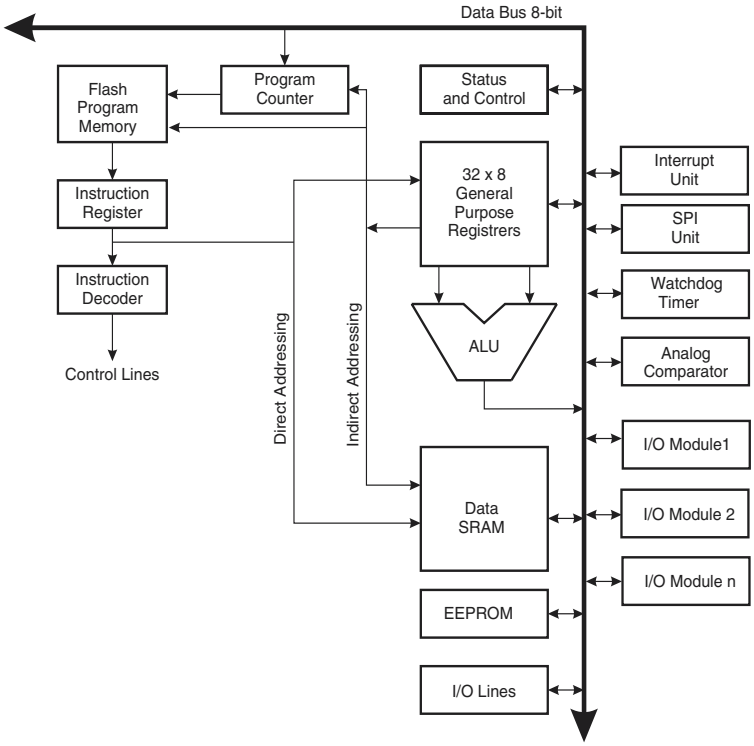
AVR CPU Core

Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register file, the operation is executed, and the result is stored back in the Register file – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After

an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register file, $20 - $5F.

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register file by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a half carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two’s complement overflow flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register file:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers

	7 0	Addr.	
	R0	$00	
	R1	$01	
	R2	$02	
	…		
	R13	$0D	
General	R14	$0E	
Purpose	R15	$0F	
Working	R16	$10	
Registers	R17	$11	
	…		
	R26	$1A	X-register Low Byte
	R27	$1B	X-register High Byte
	R28	$1C	Y-register Low Byte
	R29	$1D	Y-register High Byte
	R30	$1E	Z-register Low Byte
	R31	$1F	Z-register High Byte

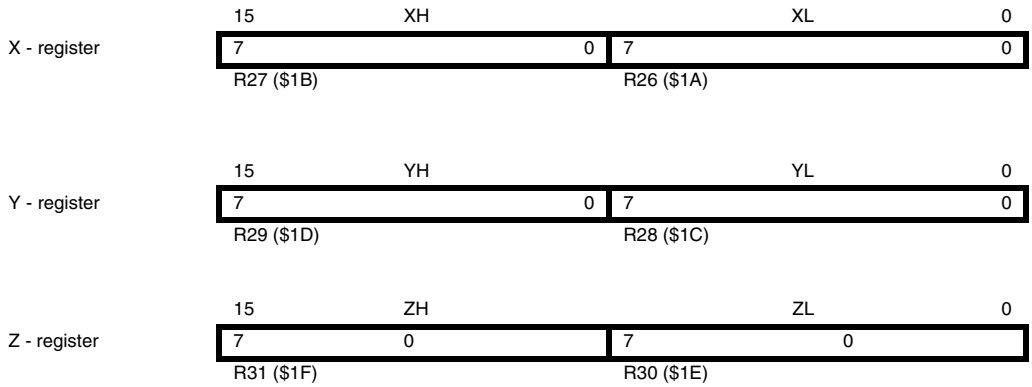
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

The X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the Instruction Set Reference for details).

Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the stack. Note that the stack is implemented as growing from higher memory locations to lower memory locations. This implies that a stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

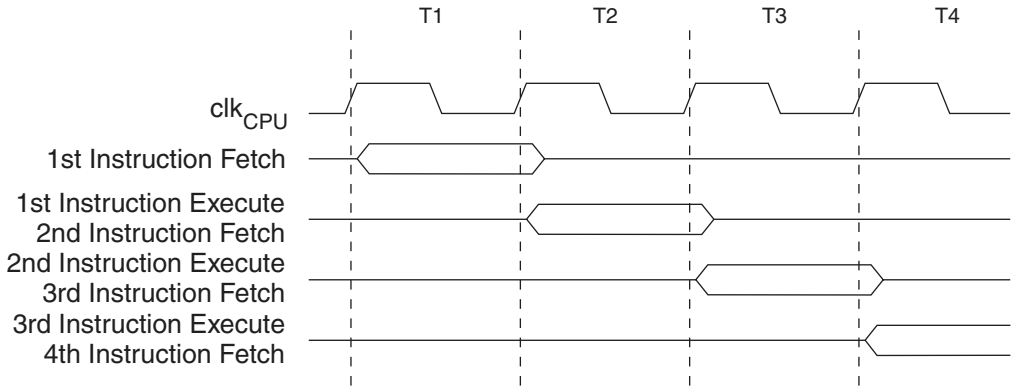
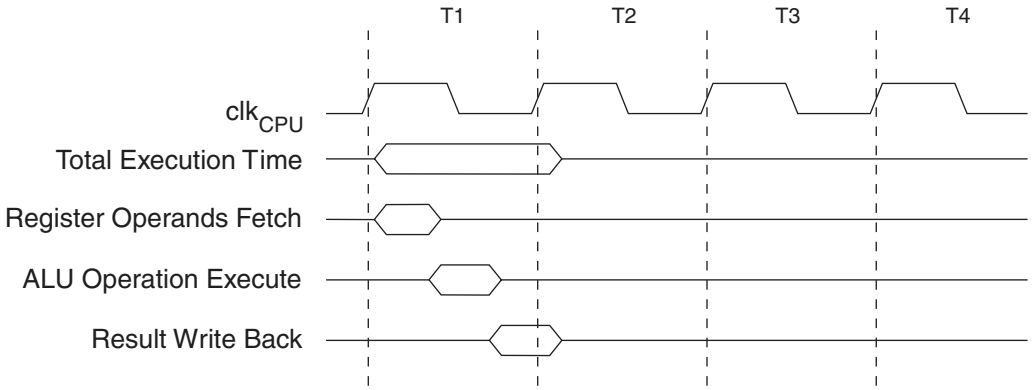


Figure 7 shows the internal timing concept for the Register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the program counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section “Memory Programming” on page 254 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 42. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0

– the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to “Interrupts” on page 42 for more information. The Reset Vector can also be moved to the start of the boot Flash section by programming the BOOTRST fuse, see “Boot Loader Support – Read-While-Write Self-Programming” on page 241.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in  r16, SREG      ; store SREG value
cli     ; disable interrupts during timed sequence
sbi EECR, EEMWE    ; start EEPROM write
sbi EECR, EEWE
out SREG, r16      ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;  /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei   ; set global interrupt enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

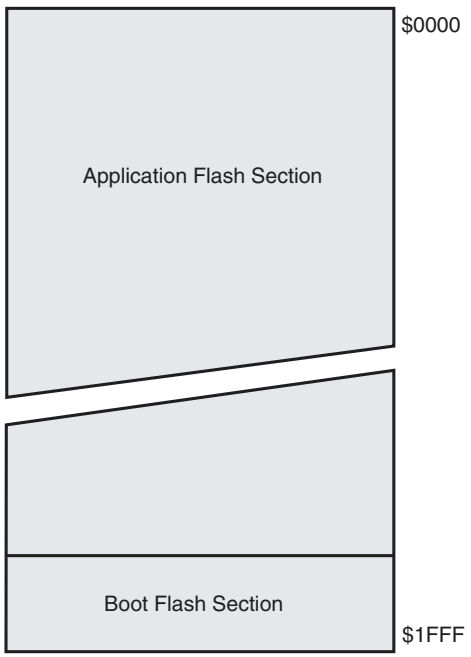
The ATmega16 contains 16K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in “Boot Loader Support – Read-While-Write Self-Programming” on page 241. “Memory Programming” on page 254 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory Instruction Description).

Timing diagrams for instruction fetch and execution are presented in “Instruction Execution Timing” on page 11.

Figure 8. Program Memory Map



SRAM Data Memory

Figure 9 shows how the ATmega16 SRAM Memory is organized.

The lower 1120 Data Memory locations address the Register file, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register file and I/O Memory, and the next 1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register file, registers R26 to R31 feature the indirect addressing pointer registers.

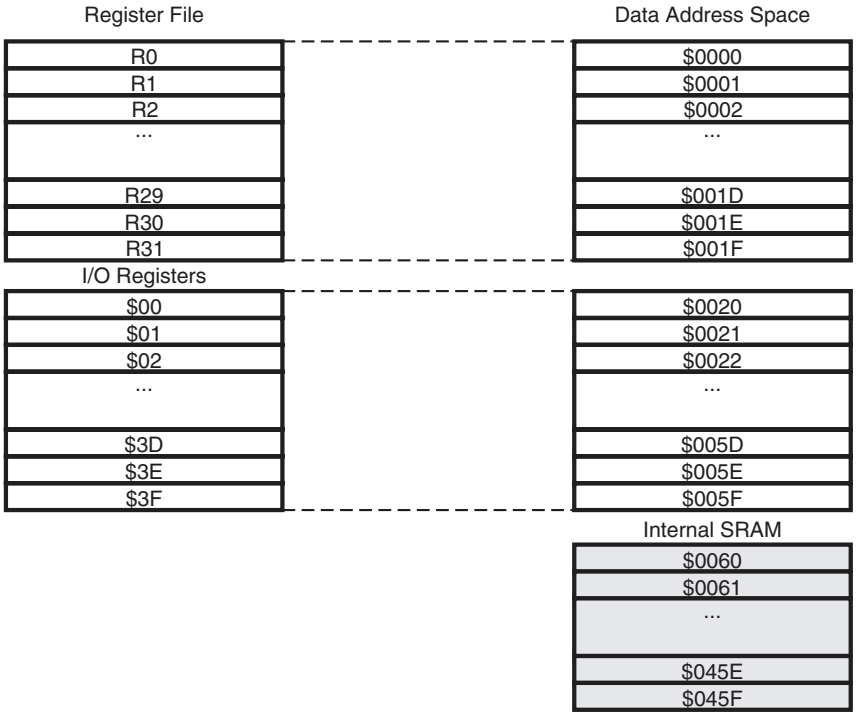
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes. The Register file is described in “General Purpose Register File” on page 8.

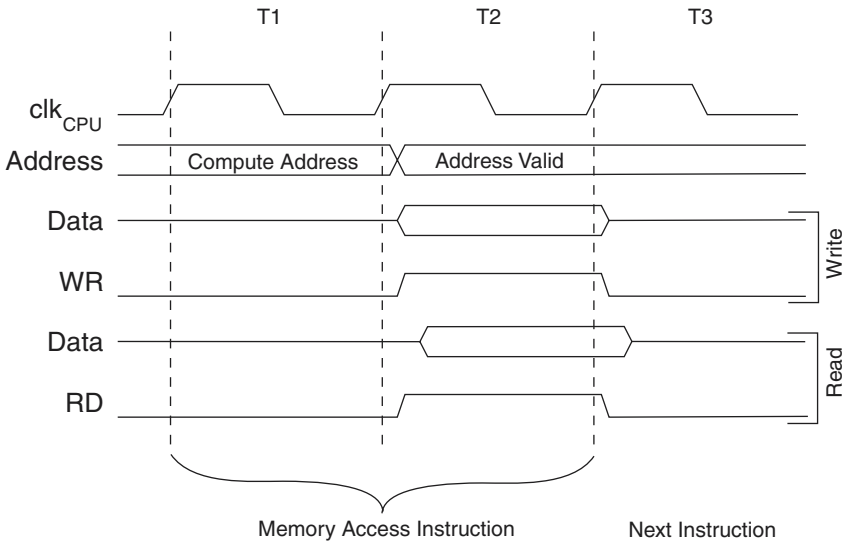
Figure 9. Data Memory Map



Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI and JTAG data downloading to the EEPROM, see page 268 and page 272, respectively.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See “Preventing EEPROM Corruption” on page 20 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect.

When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See “Boot Loader Support – Read-While-Write Self-Programming” on page 241 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM Access, the EEAR or EEDR reGister will be modified, causing the interrupted EEPROM Access to fail. It is recommended to have the global interrupt flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles(1)	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse setting.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

Preventing EEPROM Corruption

During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

I/O Memory

The I/O space definition of the ATmega16 is shown in “Register Summary” on page 298.

All ATmega16 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the Instruction Set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O Registers as data space using LD and ST instructions, $20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

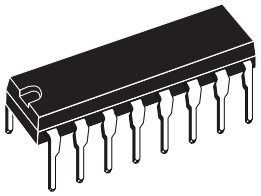
Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

The I/O and peripherals control registers are explained in later sections.

ULN2001A-ULN2002A
ULN2003A-ULN2004A

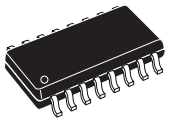
SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A



SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

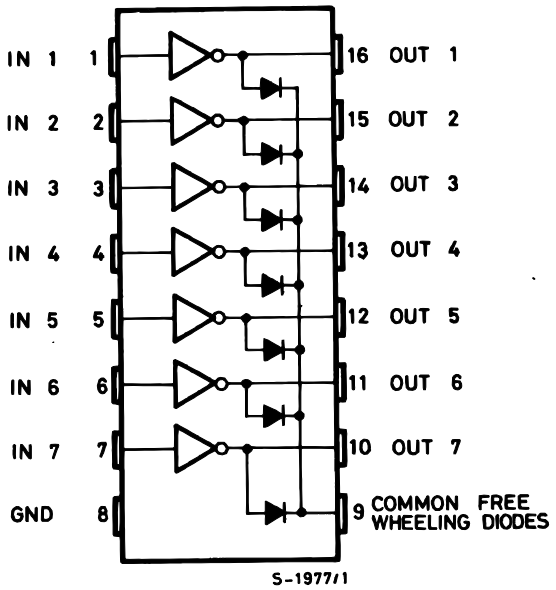
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

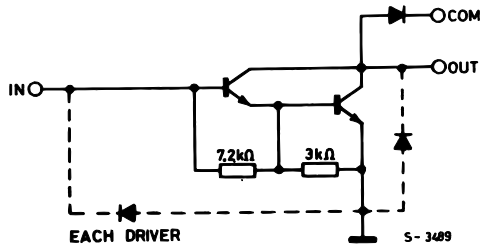
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

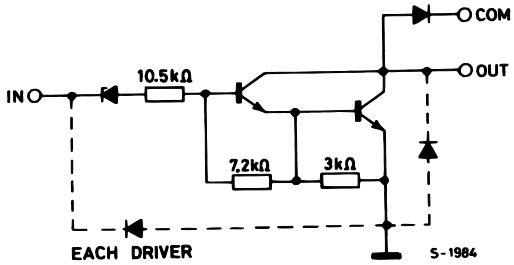
PIN CONNECTION



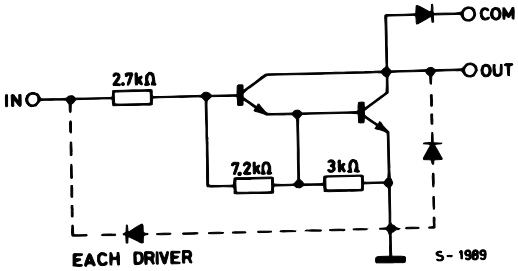
SCHEMATIC DIAGRAM



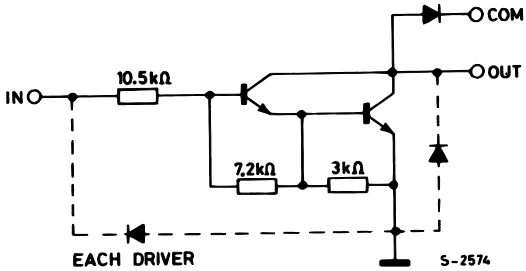
Series ULN-2001A
(each driver)



Series ULN-2002A
(each driver)



Series ULN-2003A
(each driver)



Series ULN-2004A
(each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	120	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V T_{amb} = 70°C, V_{CE} = 50V T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_i = 6V for ULN2004A V_{CE} = 50V, V_i = 1V			50 100 500 500	µA µA µA µA	1a 1a 1b 1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA I_C = 200 mA, I_B = 350µA I_C = 350mA, I_B = 500µA		0.9 1.1 1.3	1.1 1.3 1.6	V V V	2 2 2
$I_{i(on)}$	Input Current	for ULN2002A, V_i = 17V for ULN2003A, V_i = 3.85V for ULN2004A, V_i = 5V V_i = 12V		0.82 0.93 0.35 1	1.25 1.35 0.5 1.45	mA mA mA mA	3 3 3 3
$I_{i(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{i(on)}$	Input Voltage	V_{CE} = 2V for ULN2002A I_C = 300mA for ULN2003A I_C = 200mA I_C = 250mA I_C = 300mA for ULN2004A I_C = 125mA I_C = 200mA I_C = 275mA I_C = 350mA			 13 2.4 2.7 3 5 6 7 8	V	5
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	 1000				 2
C_i	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V T_{amb} = 70°C, V_R = 50V			50 100	µA µA	6 6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

TEST CIRCUITS

Figure 1a.

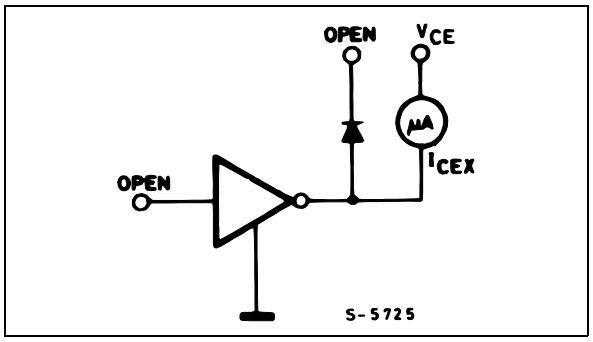


Figure 1b.

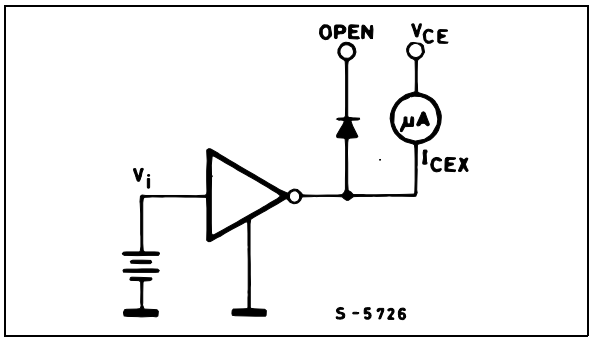


Figure 2.

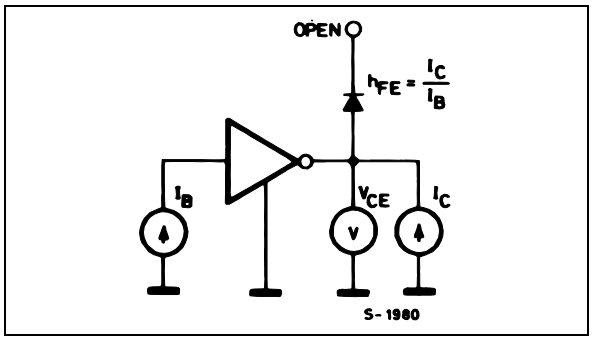


Figure 3.

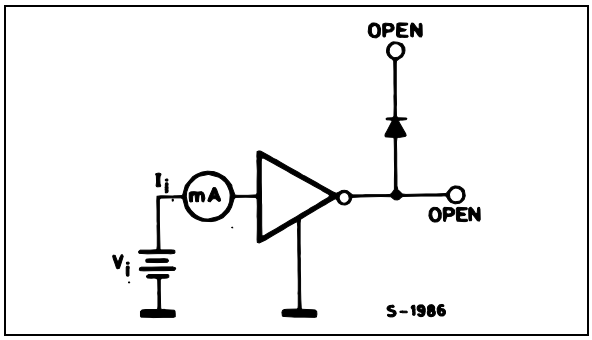


Figure 4.

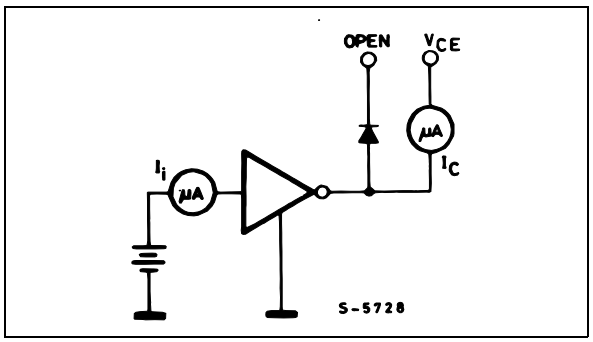


Figure 5.

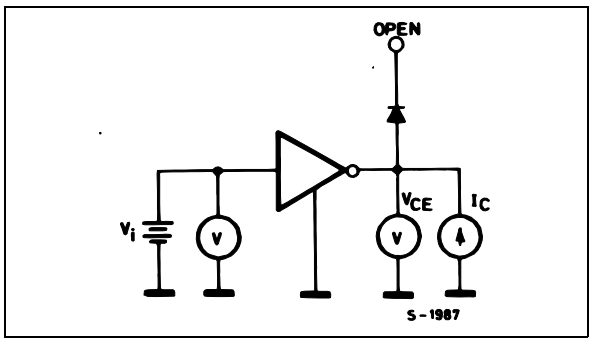


Figure 6.

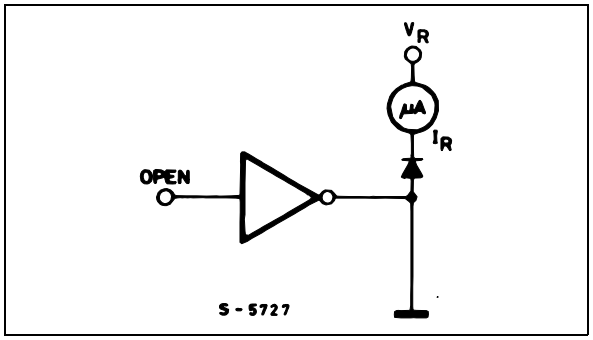


Figure 7.

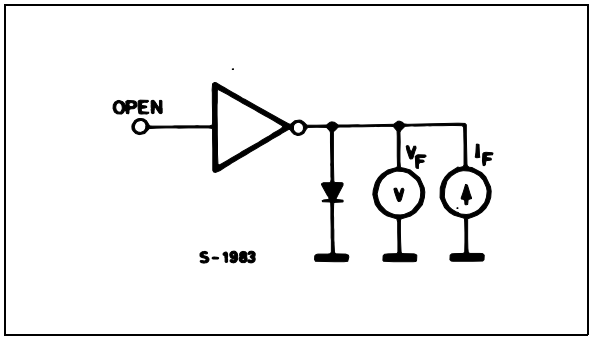


Figure 8: Collector Current versus Input Current

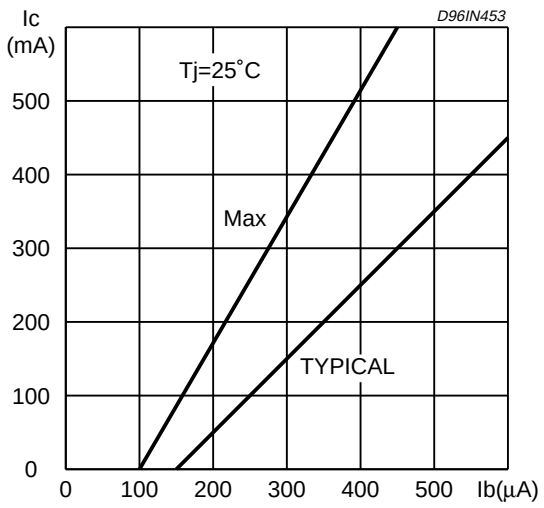


Figure 9: Collector Current versus Saturation Voltage

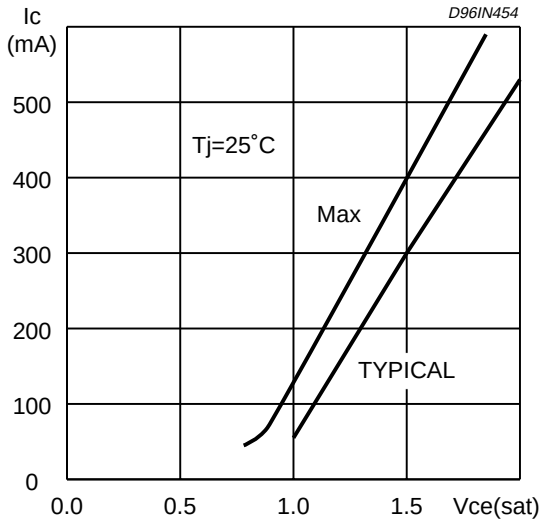


Figure 10: Peak Collector Current versus Duty Cycle

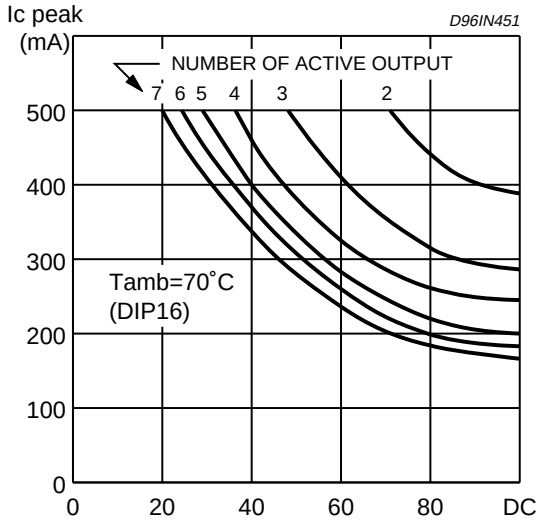
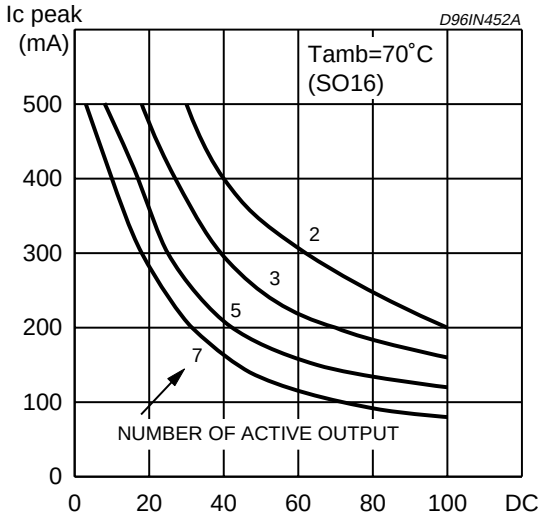
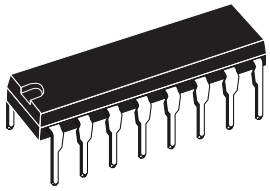


Figure 11: Peak Collector Current versus Duty Cycle

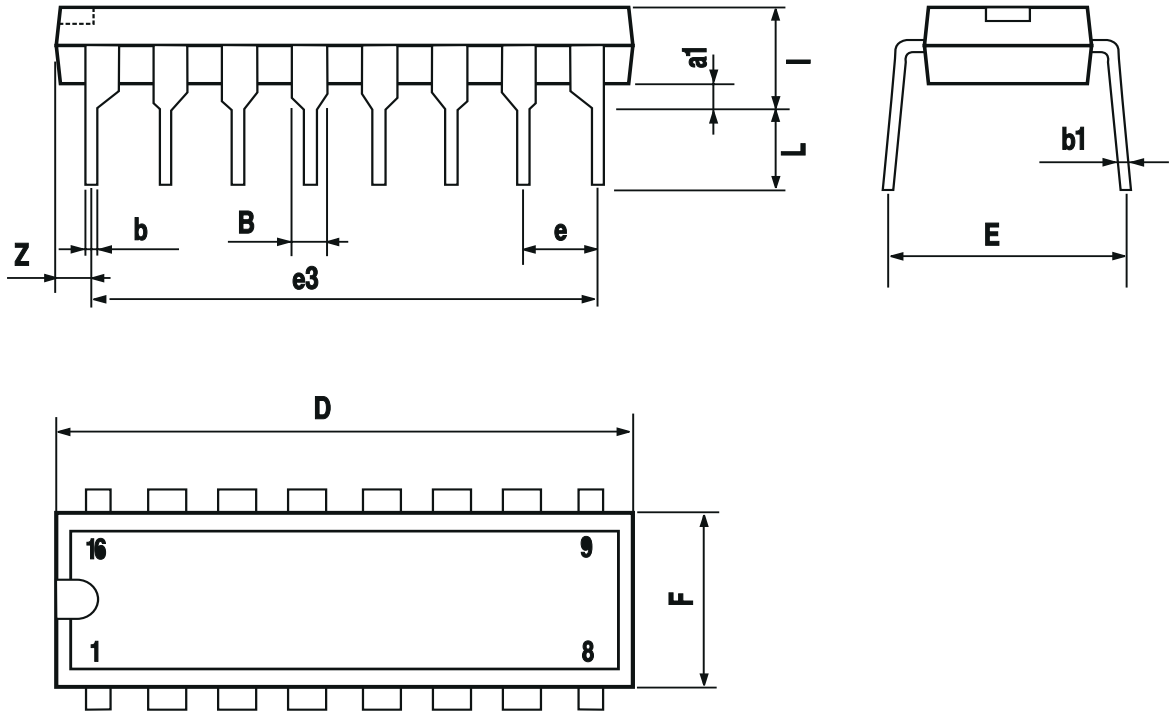


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.77		1.65	0.030		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		17.78			0.700	
F			7.1			0.280
I			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA

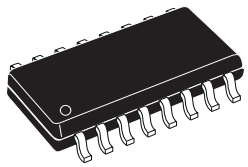


DIP16



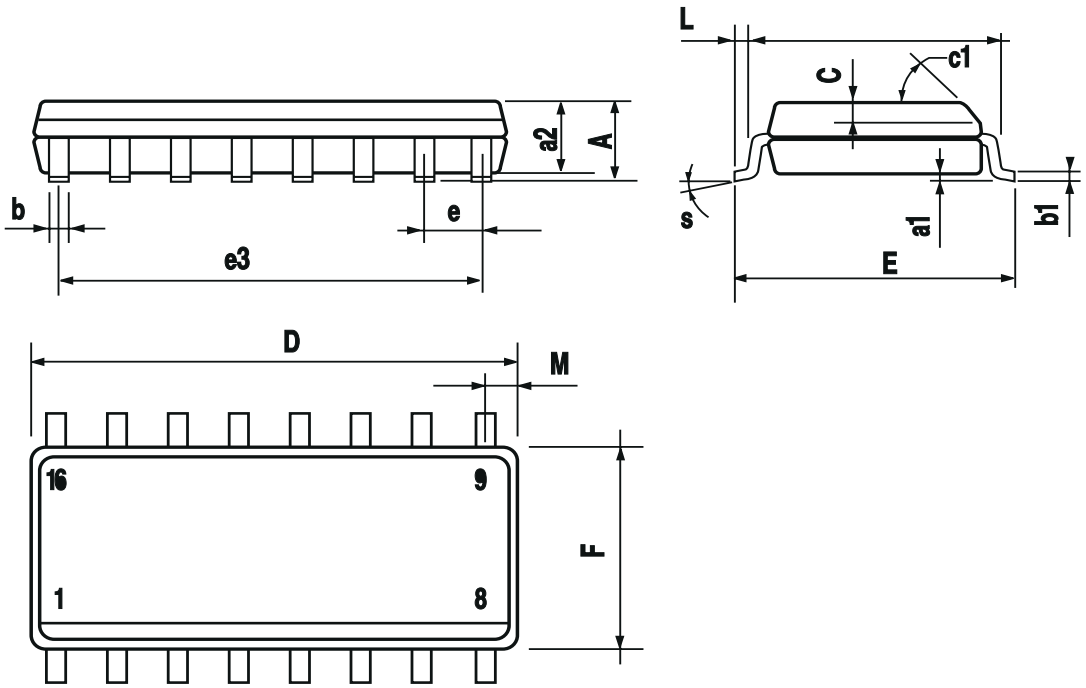
DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			1.75			0.069
a1	0.1		0.25	0.004		0.009
a2			1.6			0.063
b	0.35		0.46	0.014		0.018
b1	0.19		0.25	0.007		0.010
C		0.5			0.020	
c1	45˚ (typ.)					
D (1)	9.8		10	0.386		0.394
E	5.8		6.2	0.228		0.244
e		1.27			0.050	
e3		8.89			0.350	
F (1)	3.8		4	0.150		0.157
G	4.6		5.3	0.181		0.209
L	0.4		1.27	0.016		0.050
M			0.62			0.024
S	8˚(max.)					

OUTLINE AND MECHANICAL DATA



SO16 Narrow

(1) D and F do not include mold flash or protrusions. Mold flash or potrusions shall not exceed 0.15mm (.006inch).



Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

© 2002 STMicroelectronics – Printed in Italy – All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States.

http://www.st.com

This datasheet has been download from:

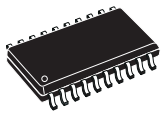
www.datasheetcatalog.com

Datasheets for electronics components.

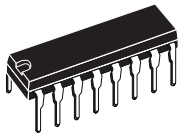
L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES



SO(12+4+4)



Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

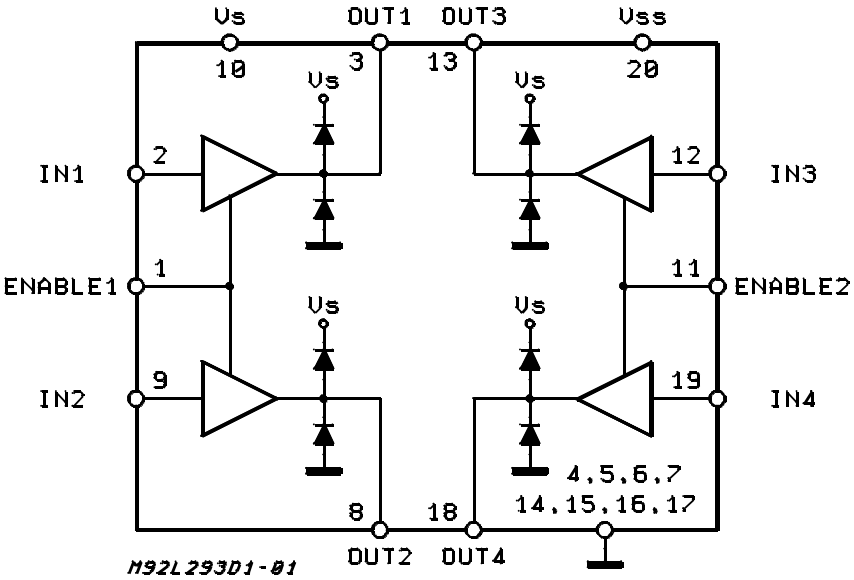
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

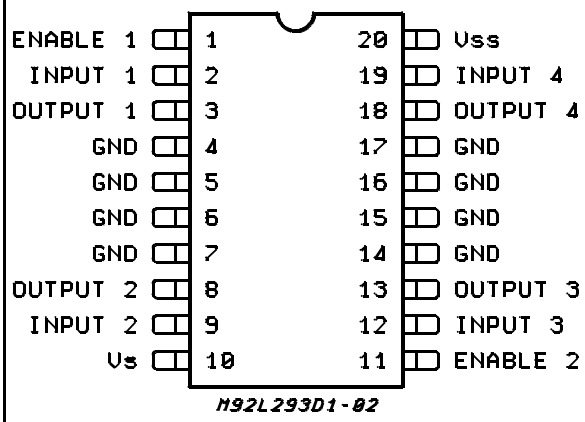
BLOCK DIAGRAM



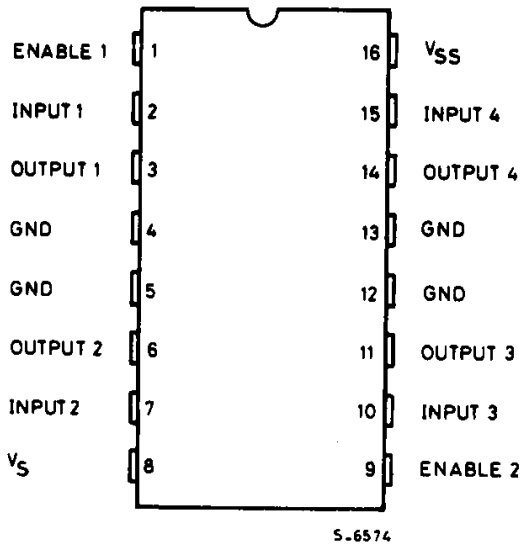
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4)



Powerdip(12+2+2)

THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	$2.3 \text{ V} \leq V_{IH} \leq V_{SS} - 0.6$ V		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	$2.3 \text{ V} \leq V_{en\,H} \leq V_{SS} - 0.6$ V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

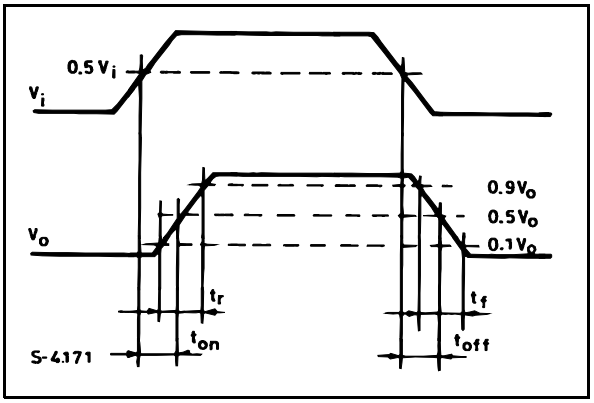
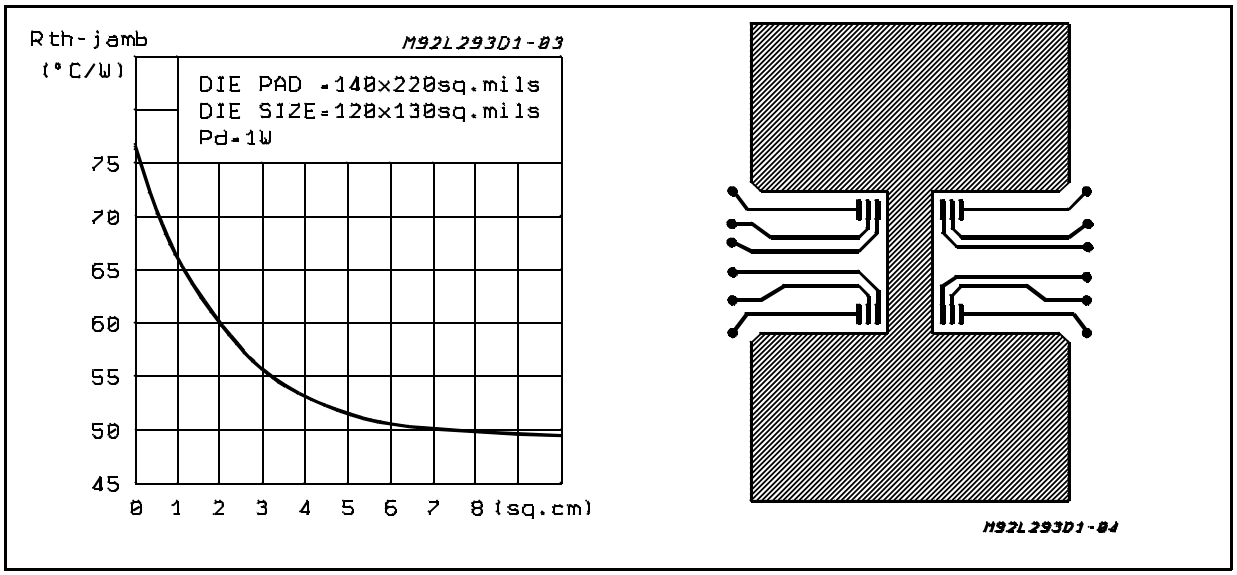
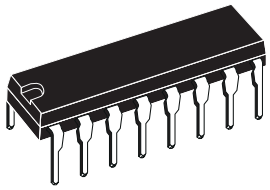


Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)

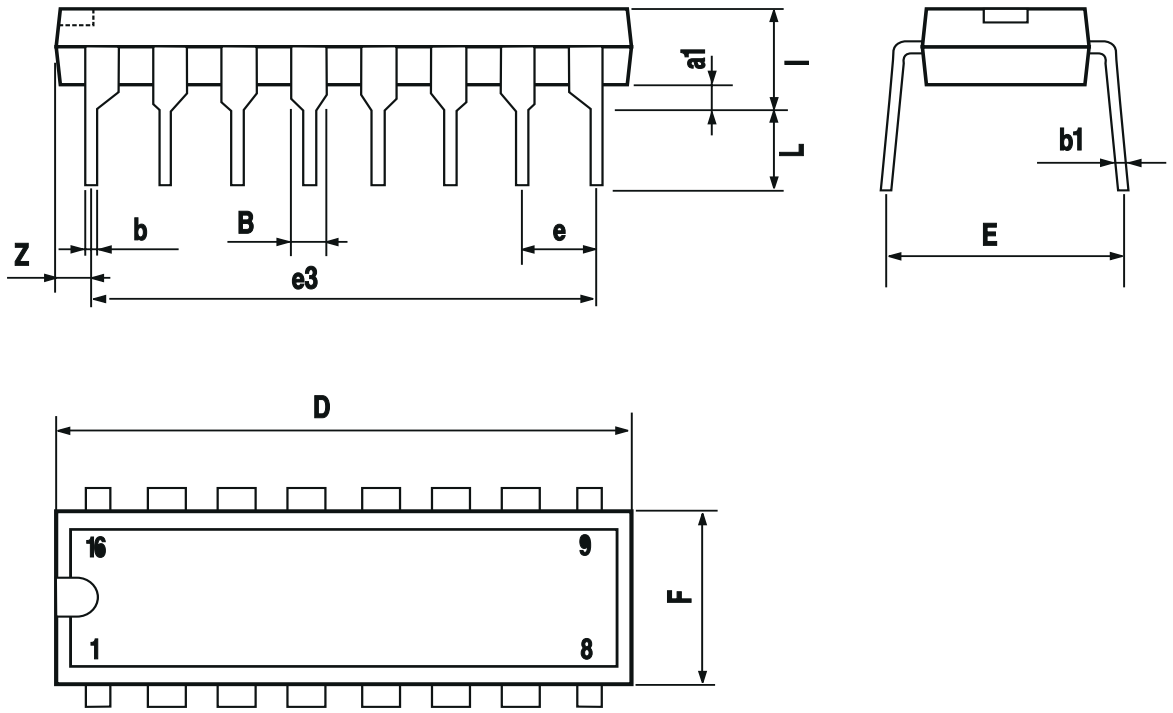


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA

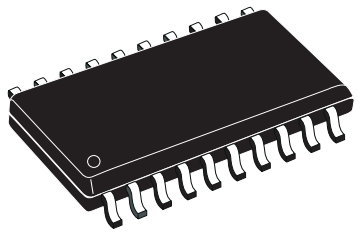


Powerdip 16

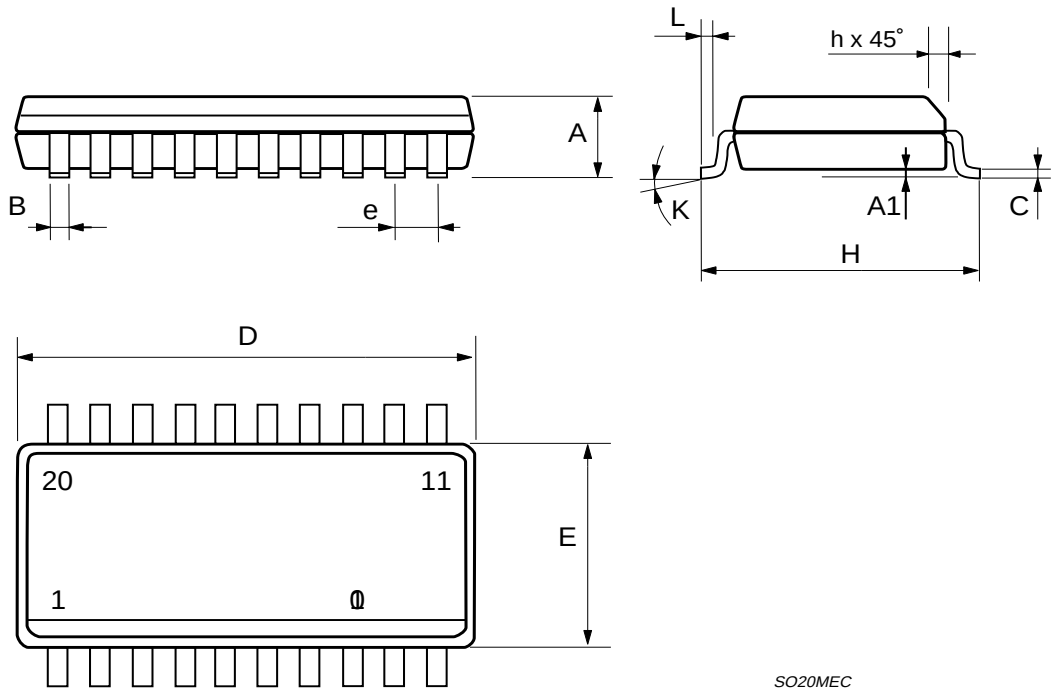


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	2.35		2.65	0.093		0.104
A1	0.1		0.3	0.004		0.012
B	0.33		0.51	0.013		0.020
C	0.23		0.32	0.009		0.013
D	12.6		13	0.496		0.512
E	7.4		7.6	0.291		0.299
e		1.27			0.050	
H	10		10.65	0.394		0.419
h	0.25		0.75	0.010		0.030
L	0.4		1.27	0.016		0.050
K	0° (min.)8° (max.)					

OUTLINE AND MECHANICAL DATA



SO20



Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

© 2003 STMicroelectronics – Printed in Italy – All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States.

http://www.st.com

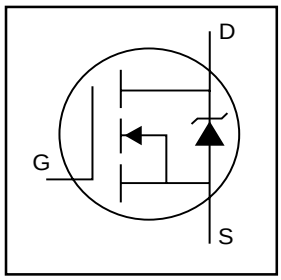
International
IOR Rectifier

PD-91279E

IRF3205

HEXFET® Power MOSFET

- Advanced Process Technology
- Ultra Low On-Resistance
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated



V_{DSS} = 55V

$R_{DS(on)}$ = 8.0mΩ

I_D = 110A⑤

Description

Advanced HEXFET® Power MOSFETs from International Rectifier utilize advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in a wide variety of applications.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.

TO-220AB

Absolute Maximum Ratings

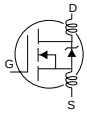
	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	110 ⑤	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	80	
I_{DM}	Pulsed Drain Current ①	390	
P_D @T_C = 25°C	Power Dissipation	200	W
	Linear Derating Factor	1.3	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
I_{AR}	Avalanche Current①	62	A
E_{AR}	Repetitive Avalanche Energy①	20	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.0	V/ns
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to + 175	°C
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 srew	10 lbf•in (1.1N•m)	

Thermal Resistance

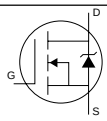
	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	——	0.75	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	0.50	——	
$R_{\theta JA}$	Junction-to-Ambient	——	62	

01/25/01

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	55	——	——	V	V_{GS} = 0V, I_D = 250µA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	——	0.057	——	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	——	——	8.0	mΩ	V_{GS} = 10V, I_D = 62A ④
$V_{GS(th)}$	Gate Threshold Voltage	2.0	——	4.0	V	V_{DS} = V_{GS}, I_D = 250µA
g_{fs}	Forward Transconductance	44	——	——	S	V_{DS} = 25V, I_D = 62A④
I_{DSS}	Drain-to-Source Leakage Current	——	——	25	µA	V_{DS} = 55V, V_{GS} = 0V
		——	——	250		V_{DS} = 44V, V_{GS} = 0V, T_J = 150°C
I_{GSS}	Gate-to-Source Forward Leakage	——	——	100	nA	V_{GS} = 20V
	Gate-to-Source Reverse Leakage	——	——	-100		V_{GS} = -20V
Q_g	Total Gate Charge	——	——	146	nC	I_D = 62A
Q_{gs}	Gate-to-Source Charge	——	——	35		V_{DS} = 44V
Q_{gd}	Gate-to-Drain ("Miller") Charge	——	——	54		V_{GS} = 10V, See Fig. 6 and 13
$t_{d(on)}$	Turn-On Delay Time	——	14	——	ns	V_{DD} = 28V
t_r	Rise Time	——	101	——		I_D = 62A
$t_{d(off)}$	Turn-Off Delay Time	——	50	——		R_G = 4.5Ω
t_f	Fall Time	——	65	——		V_{GS} = 10V, See Fig. 10 ④
L_D	Internal Drain Inductance	——	4.5	——	nH	Between lead, 6mm (0.25in.) from package and center of die contact 
L_S	Internal Source Inductance	——	7.5	——		
C_{iss}	Input Capacitance	——	3247	——	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	——	781	——		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	——	211	——		ƒ = 1.0MHz, See Fig. 5
E_{AS}	Single Pulse Avalanche Energy②	——	1050⑥	264⑦	mJ	I_{AS} = 62A, L = 138µH

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	——	——	110	A	MOSFET symbol showing the integral reverse p-n junction diode. 
I_{SM}	Pulsed Source Current (Body Diode)①	——	——	390		
V_{SD}	Diode Forward Voltage	——	——	1.3	V	T_J = 25°C, I_S = 62A, V_{GS} = 0V ④
t_{rr}	Reverse Recovery Time	——	69	104	ns	T_J = 25°C, I_F = 62A
Q_{rr}	Reverse Recovery Charge	——	143	215	nC	di/dt = 100A/µs ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				

Notes:

① Repetitive rating; pulse width limited by max. junction temperature. (See fig. 11)

② Starting T_J = 25°C, L = 138µH R_G = 25Ω, I_{AS} = 62A. (See Figure 12)

③ I_{SD} ≤ 62A, di/dt ≤ 207A/µs, V_{DD} ≤ $V_{(BR)DSS}$, T_J ≤ 175°C

④ Pulse width ≤ 400µs; duty cycle ≤ 2%.

⑤ Calculated continuous current based on maximum allowable junction temperature. Package limitation current is 75A.

⑥ This is a typical value at device destruction and represents operation outside rated limits.

⑦This is a calculated value limited to T_J = 175°C.

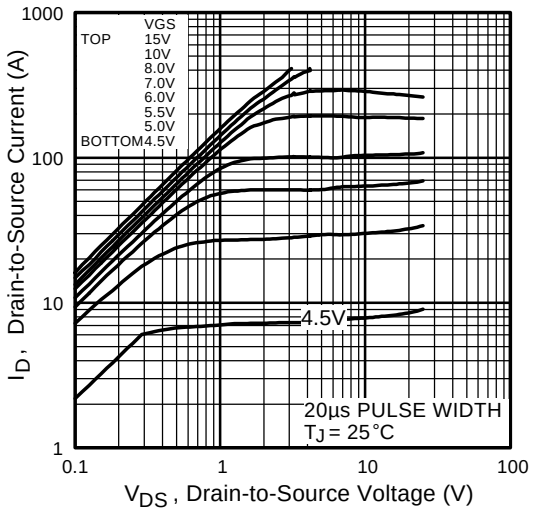


Fig 1. Typical Output Characteristics

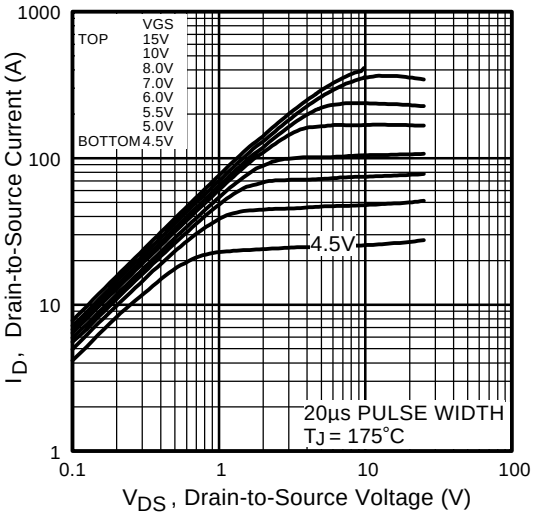


Fig 2. Typical Output Characteristics

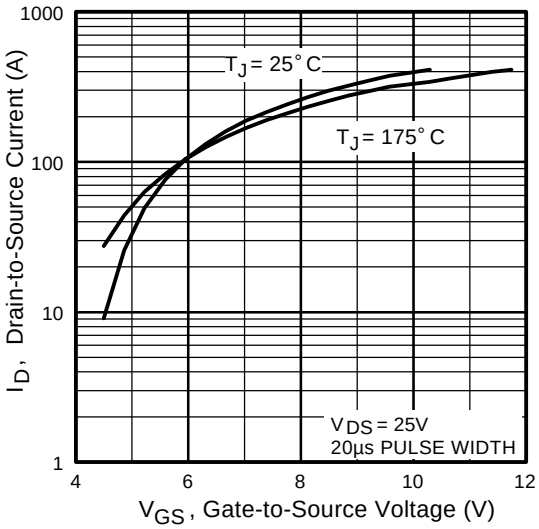


Fig 3. Typical Transfer Characteristics

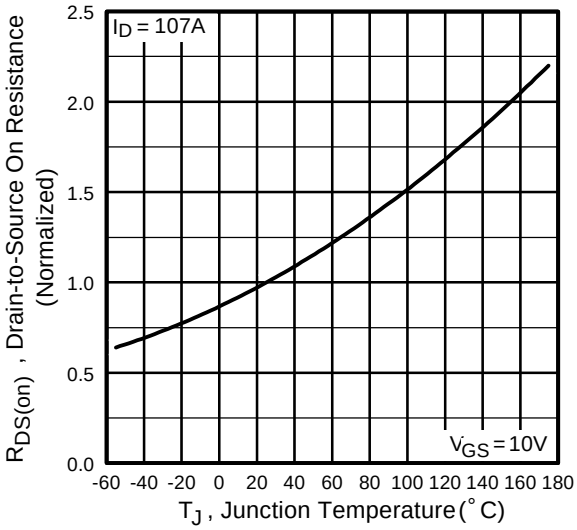


Fig 4. Normalized On-Resistance Vs. Temperature

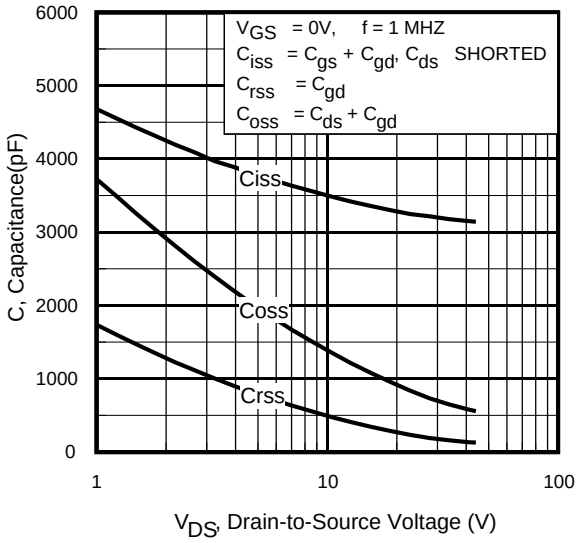


Fig 5. Typical Capacitance Vs. Drain-to-Source Voltage

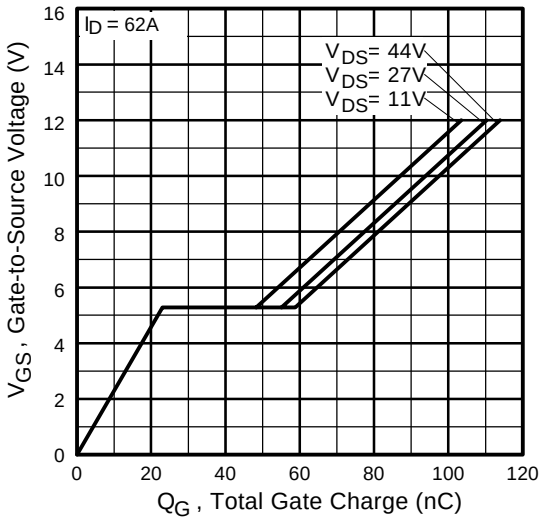


Fig 6. Typical Gate Charge Vs. Gate-to-Source Voltage

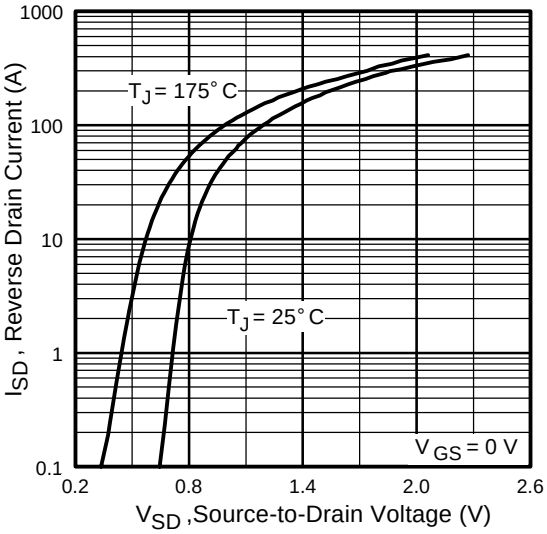


Fig 7. Typical Source-Drain Diode Forward Voltage

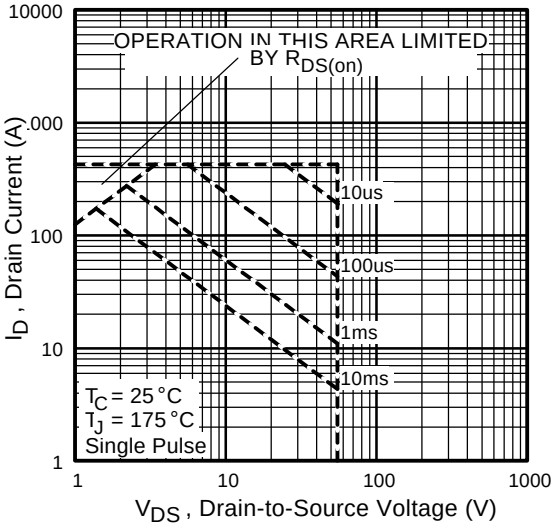


Fig 8. Maximum Safe Operating Area

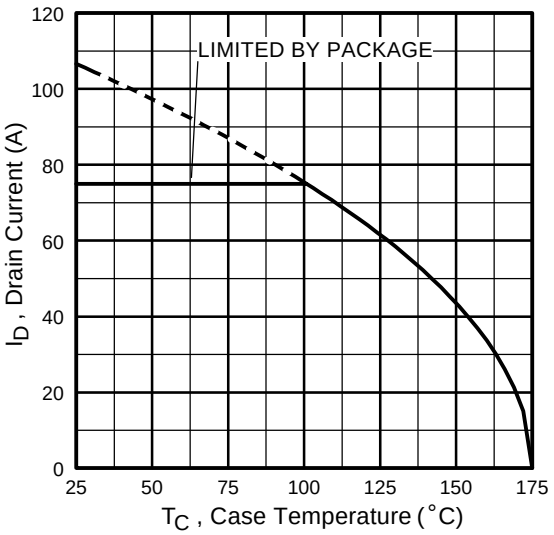


Fig 9. Maximum Drain Current Vs. Case Temperature

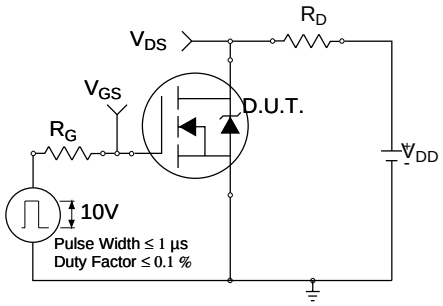


Fig 10a. Switching Time Test Circuit

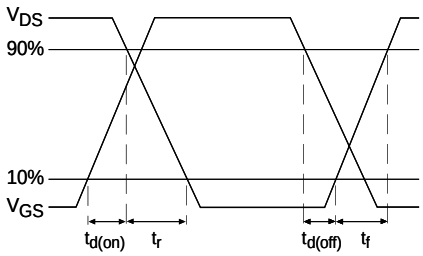


Fig 10b. Switching Time Waveforms

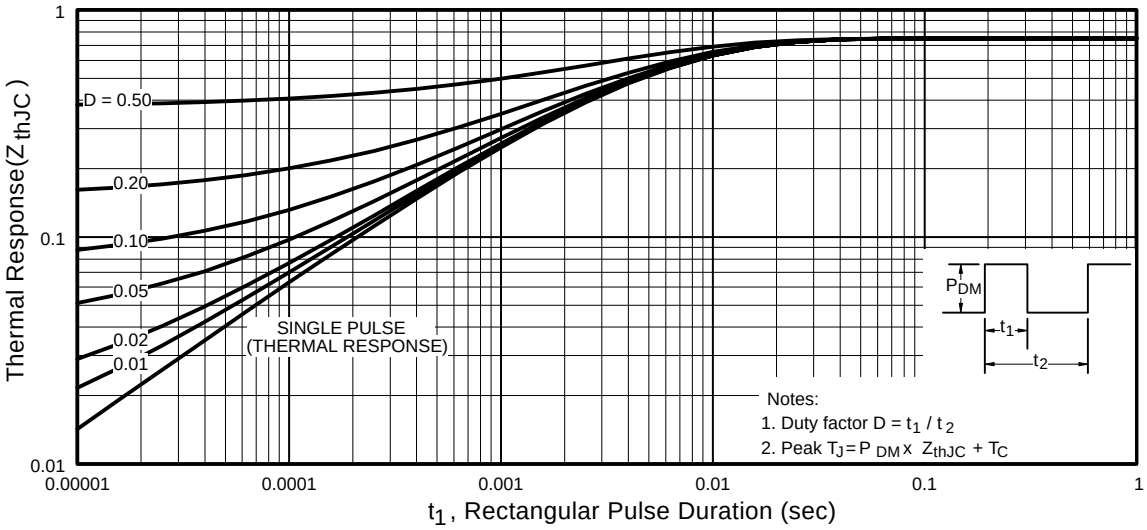


Fig 11. Maximum Effective Transient Thermal Impedance, Junction-to-Case

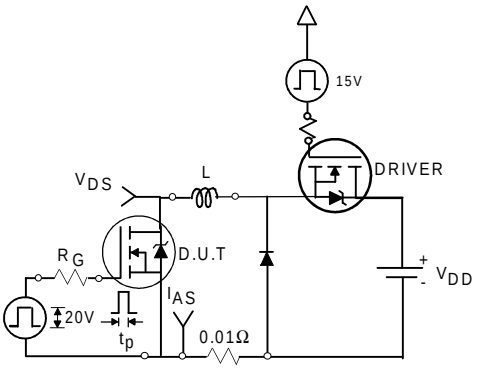


Fig 12a. Unclamped Inductive Test Circuit

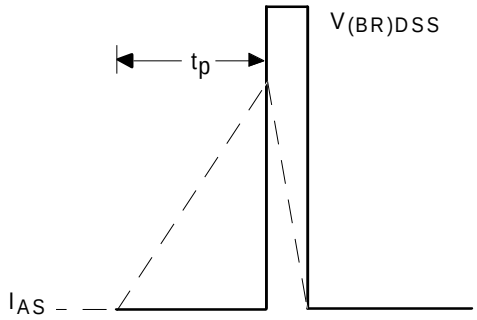


Fig 12b. Unclamped Inductive Waveforms

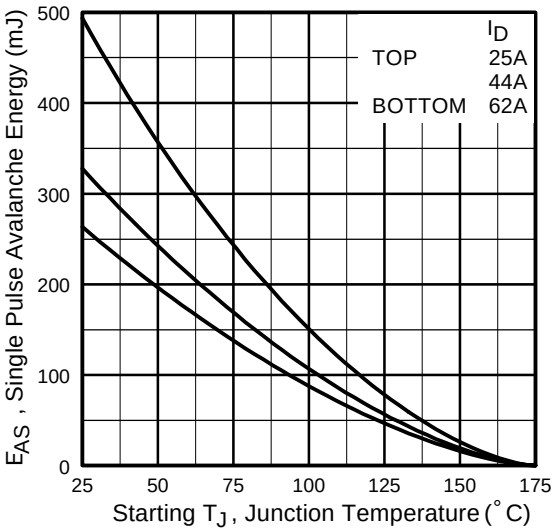


Fig 12c. Maximum Avalanche Energy Vs. Drain Current

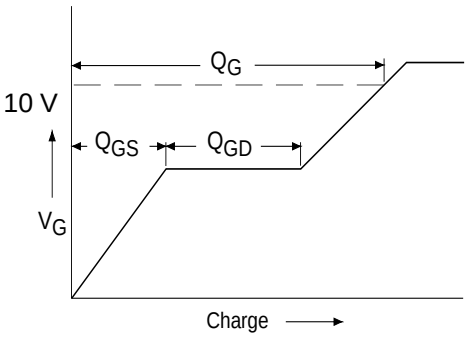


Fig 13a. Basic Gate Charge Waveform

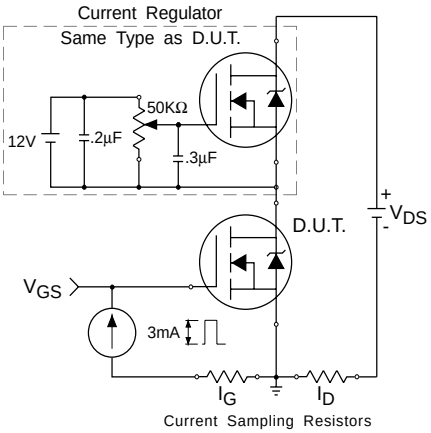
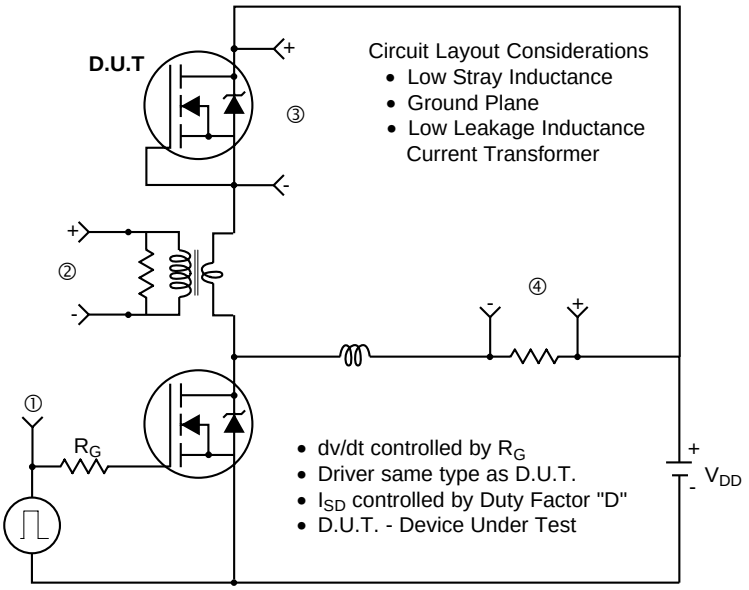
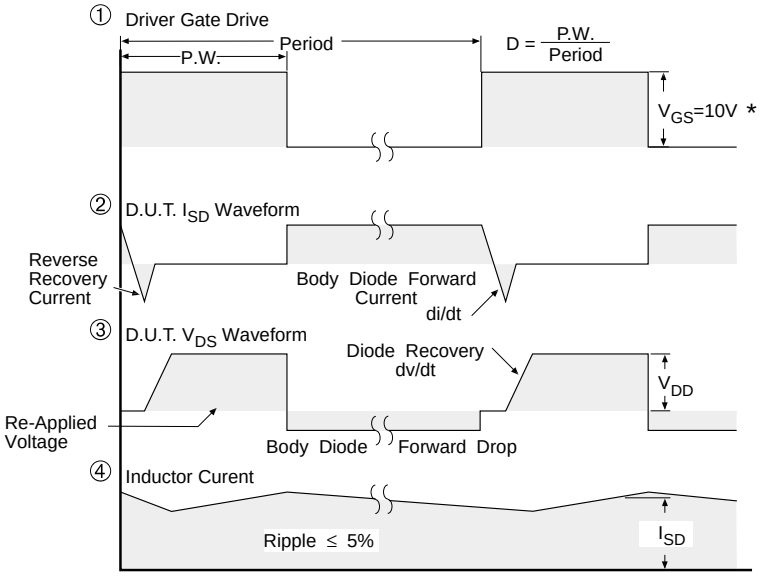


Fig 13b. Gate Charge Test Circuit

Peak Diode Recovery dv/dt Test Circuit





* V_{GS} = 5V for Logic Level Devices

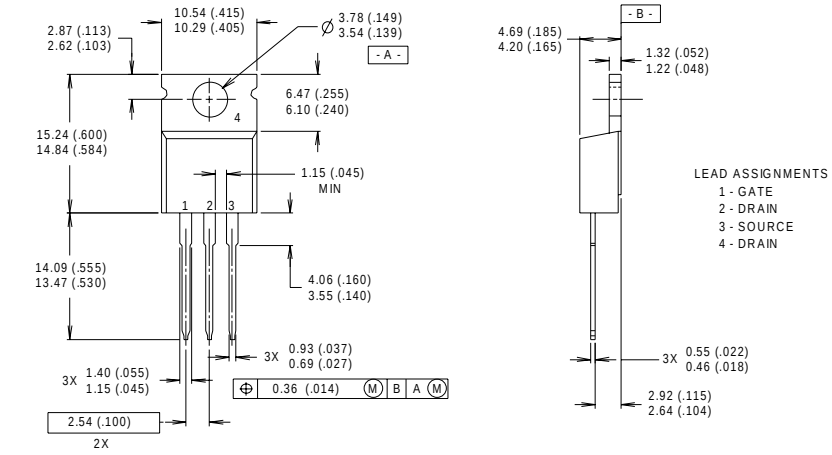
Fig 14. For N-Channel HEXFETS

International
IOR Rectifier

Package Outline

TO-220AB Outline

Dimensions are shown in millimeters (inches)



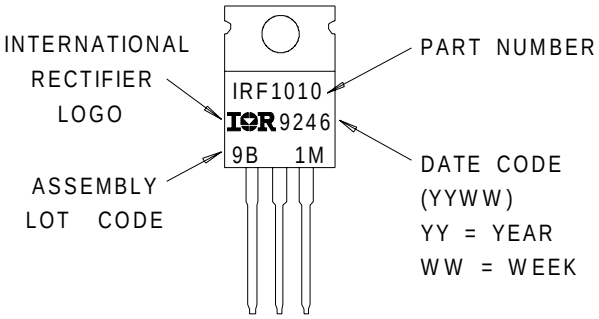
NOTES:

1 DIMENSIONING & TOLERANCING PER ANSI Y14.5M, 1982.
2 CONTROLLING DIMENSION : INCH
3 OUTLINE CONFORMS TO JEDEC OUTLINE TO-220AB.
4 HEATSINK & LEAD MEASUREMENTS DO NOT INCLUDE BURRS.

Part Marking Information

TO-220AB

EXAMPLE : THIS IS AN IRF1010
WITH ASSEMBLY
LOT CODE 9B1M



Data and specifications subject to change without notice.
This product has been designed and qualified for the automotive [Q101] market.
Qualification Standards can be found on IR's Web site.

International
IOR Rectifier

IR WORLD HEADQUARTERS: 233 Kansas St., El Segundo, California 90245, USA Tel: (310) 252-7105
TAC Fax: (310) 252-7903
Visit us at www.irf.com for sales contact information.01/01

Note: For the most current drawings please refer to the IR website at:
http://www.irf.com/package/

KEPUTUSAN DEKAN
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
NOMOR : 127/ELK/Q-I/VI/2012
TENTANG
PENGANGKATAN PEMBIMBING TUGAS AKHIR D.3
BAGI MAHASISWA FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA

DEKAN FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA

Menimbang :
1. Bahwa sehubungan dengan telah dipenuhi syarat untuk penulisan Tugas Akhir D3 bagi mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta, perlu diangkat pembimbing.
2. Bahwa untuk keperluan dimaksud perlu ditetapkan dengan Keputusan Dekan.

Mengingat :
1. Undang-undang Nomor 20 tahun 2003
2. Peraturan Pemerintah RI Nomor 60 tahun 1999
3. Keputusan Presiden RI: a. Nomor 93 tahun 1999; b. 305/M tahun 1999.
4. Keputusan Menteri Pendidikan dan Kebudayaan RI: Nomor 274/O/1999.
5. Keputusan Mendiknas RI Nomor 003/O/2001.
6. Keputusan Rektor UNY Nomor : 1150/UN34/KP/2011

M E M U T U S K A N

Menetapkan

Pertama : Mengangkat Pembimbing Tugas Akhir D3 bagi mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta sebagai berikut :

Nama Pembimbing	: Dr. Putu Sudira
Bagi mahasiswa	
Nama/No.Mahasiswa	: **Zaenal Muttaqien / 09507131015**
Jurusan	: Pendidikan Teknik Elektronika / Teknik Elektronika

Kedua : Dosen pembimbing diserahi tugas membimbing penulisan Tugas Akhir sesuai dengan Pedoman Tugas Akhir.

Ketiga : Keputusan ini berlaku sejak ditetapkan

Keempat : Segala sesuatu akan diubah dan dibetulkan sebagaimana mestinya apabila di kemudian hari ternyata terdapat kekeliruan dalam Keputusan ini.

Ditetapkan : di Yogyakarta
Pada tanggal : 1 Juni 2012
Dekan

Dr. Moch. Bruri Triyono
NIP. 19560216 198603 1 003

Tembusan Yth :
1. Wakil Dekan I FT UNY
2. Ketua Program Studi Teknik Elektronika
3. Kasub. Bag. Pendidikan FT UNY
4. Yang bersangkutan.

JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA
Alamat : Kampus Karangmalang Yogyakarta 55281
Telp. : (0274) 554686 ; (0274) 586168 ext. 293

KARTU BIMBINGAN PROYEK AKHIR (Untuk Mahasiswa)

FRM/EKA/03-00
25 Januari 2008

Nama Mahasiswa : Zaenal Muttaqien
No. Mahasiswa : 09507131015
Program Studi *) : 1. ~~Pendidikan Teknik Elektronika~~ Jenjang : ~~S1~~
2. ~~Pendidikan Teknik Informatika~~ Jenjang : ~~S1~~
3. Teknik Elektronika Jenjang : D3

Kelas : B
Dosen Pembimbing : Dr. Putu Sudira, M.P. No. Telp. / HP. :
Judul : Mesin Ketik Huruf Braille Elektronik Berbasis Mikrokontroller ATmega 16

No	Tanggal	Uraian Bimbingan	Tandatangan Pembimbing
1.	24/9 2012	Laporan BAB I - BAB III	
2.	18/10 2012	Demonstrasi alat	
3.	22/10 2012	Revisi Laporan	
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Rekomendasi Pembimbing :

1. Mahasiswa yang bersangkutan siap untuk diuji.

Tanggal Persetujuan : 5-11-2012 Tandatangan Pembimbing :

2. Kartu Bimbingan ini wajib dilampirkan pada saat pendaftaran ujian Proyek Akhir.

UNIVERSITAS NEGERI YOGYAKARTA
FAKULTAS TEKNIK

Certificate No: 05C-06592

SURAT KETERANGAN PERSETUJUAN UJIAN TUGAS AKHIR

FRM/TKF/36-00
02 Juli 2007

Yang bertanda tangan di bawah ini:

Nama : Dr. Putu Sudira, MP

NIP. : 1964 1231 1987 021 063

sebagai pembimbing I, dan

Nama : .....

NIP. : .....

sebagai pembimbing II

menerangkan bahwa Tugas Akhir bagi mahasiswa:

Nama : Zaenal Muttaqien

No. Mhs. : 09507 131 015

Judul TA : Mesin Ketik Huruf Braille Elektronik Berbasis Mikrokontroller ATmega 16.

sudah layak untuk diujikan di depan Dewan Penguji.

Demikian surat keterangan ini dibuat, untuk digunakan sebagaimana mestinya.

Pembimbing I	Pembimbing II
Dr. Putu Sudira, MP	
NIP. 1964 1231 1987 021 063	