# SKRIPSI

## PERENCANAAN DAN PEMBUATAN ALAT PENGENDALI MOTOR DC BERBASIS FUZZY LOGIC DENGAN MENGGUNAKAN MIKROKONTROLLER AT89S51 DENGAN TAMPILAN LCD


**Disusun Oleh :**

**TULUS JUNAEDHI**
**NIM 0112123**





**JURUSAN TEKNIK ELEKTRO S-1**
**KONSENTRASI TEKNIK ENERGI LISTRIK**
**FAKULTAS TEKNOLOGI INDUSTRI**
**INSTITUT TEKNOLOGI NASIONAL MALANG**



**SEPTEMBER 2007**

# ГОСТ 2.601-68

## ЕДИНАЯ СИСТЕМА КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ

## ЭКСПЛУАТАЦИОННЫЕ ДОКУМЕНТЫ

Издание официальное

ИЗДАТЕЛЬСТВО СТАНДАРТОВ
Москва

# LEMBAR PERSETUJUAN

# PERENCANAAN DAN PEMBUATAN ALAT PENGENDALI MOTOR DC BERBASIS FUZZY LOGIC DENGAN MENGGUNAKAN MIKROKONTROLLER AT89S51 DENGAN TAMPILAN LCD

## SKRIPSI

*Disusun Dan Diajukan Untuk Memenuhi Persyaratan Guna Mencapai Gelar Sarjana Teknik Elektro Strata Satu (S-1)*


**Disusun Oleh :**
**TULUS JUNAEDHI**
**NIM : 01.12.123**


**Diperiksa dan disetujui,**

| Disetujui,<br>Dosen Pembimbing I, | Disetujui,<br>Dosen Pembimbing II, |
|---|---|
| **Ir. Widodo Pudji Muljanto, MT** | **I Komang Somawirata, ST, MT** |
| **NIP. Y. 102 870 0171** | **NIP. 1030100361** |

**Mengetahui,**
**Ketua Jurusan Teknik Elektro S-1**

**Ir. F. Yudi Limpraptono, MT**
**NIP. Y. 103 950 0274**

**KONSENTRASI TEKNIK ENERGI LISTRIK**
**JURUSAN TEKNIK ELEKTRO S-1**
**FAKULTAS TEKNOLOGI INDUSTRI**
**INSTITUT TEKNOLOGI NASIONAL MALANG**
**2007**

# KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yang Maha Esa karena oleh anugrah-Nya,penyusun dapat menyelesaikan tugas akhir ini.

Tugas skripsi ini dibuat untuk melengkapi ujian sarjana Jurusan Teknik Elektro S-1 , konsentrasi Teknik Energi Listrik pada Institut Teknologi Nasional Malang ( ITN Malang ).Penulisan tugas akhir ini berjudul " Perencanaan Pembuatan Alat control Kecepatan Motor DC Berbasis Logika Fuzzy dengan Mikrokontroller AT89S51".

Dengan selesainya tugas akhir ini penyusun ingin menyampaikan terima kasih sebesar-besarnya kepada yang terhormat :

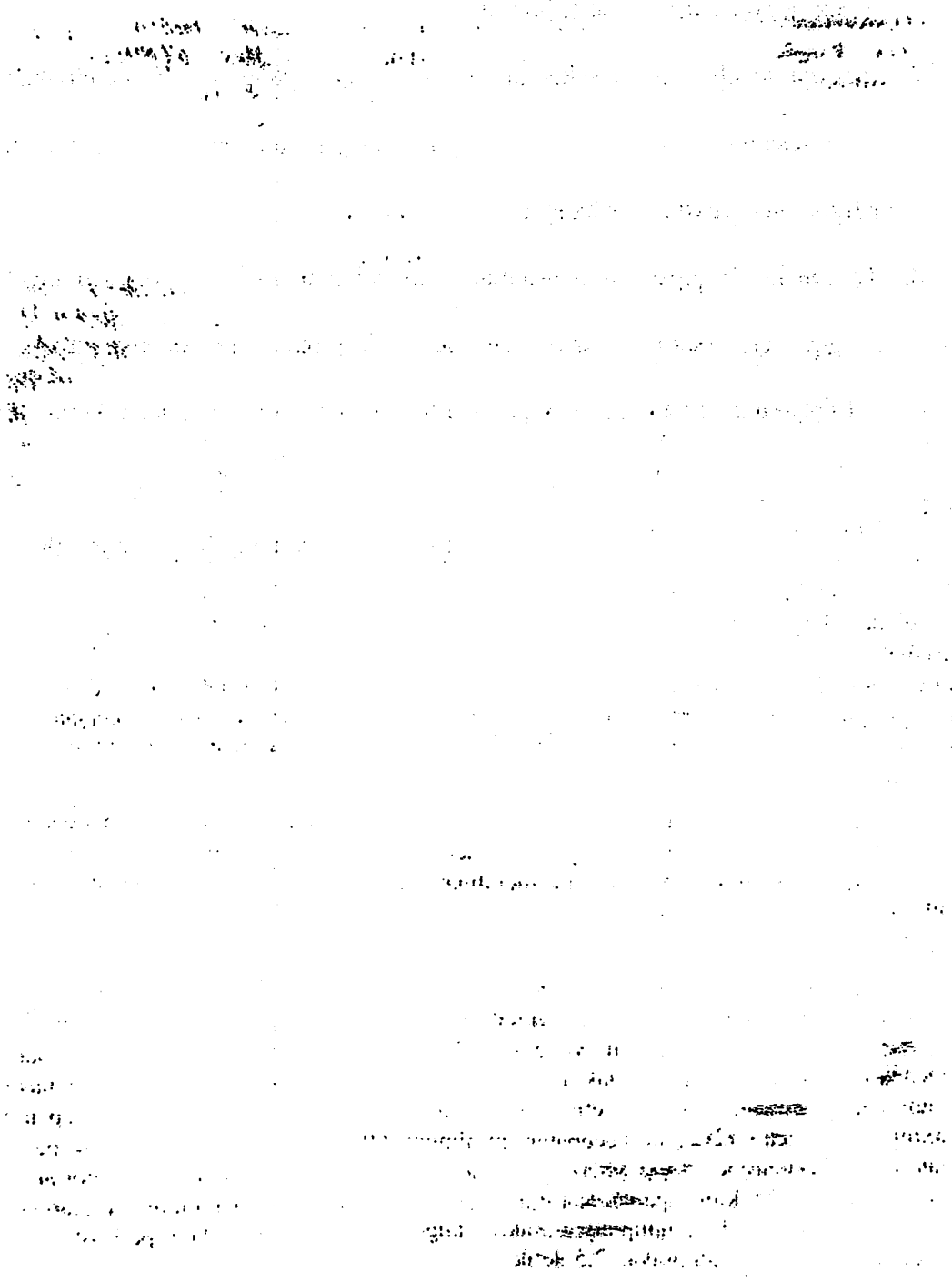
1. **Frof.Dr.Ir.Abraham Lomi,MSEE,**selaku rector Institut Teknologi Malang.
2. **Ir.Mochtar Ansori MSEE,**selaku Dekan Fakultas Teknologi Industri, Institut Teknologi Nasional Malang.
3. **Ir.F Yudi Limpraptono,MT,**selaku Ketua Jurusan Elektro, Institut Teknologi Nasional Malang.
4. **Bapak Ir.Widodo Pudji M,MT,**selaku Desen Pembimbing yang telah memberi ilmu,dorongan semangat dan bantuan.
5. **Bapak I Komang Somawirata,ST,MT,**selaku Dosen Pembimbing kedua yang telah memberi masukan yang sangat berguna..
6. Rekan – rekan di Institut Teknologi Nasional Malang.Dalam penyusunan skripsi ini penyusun menyadari bahwa skripsi ini masih memiliki

kekurangan ,karena itu masukan-masukannya akan penyusun nantikan untuk penyempurnaan selanjutnya.

7. Terima kasih atas doa kedua orang tua aku yang selalu mendoakan kesuksesanku serta doa Y 42 KU yuni yang setiap malam setiap waktu berdoa buat ku thank's banget
8. Terima kasih kepada teman-teman yang telah memberi suport semangat hingga terselesainya skripsi ini serta dorongan dan doanya hingga sukses,aku doakan juga semua teman-teman cepat lulus dan dapat kerja....

Malang,September 2007

Tulus j



# ABSTRAK

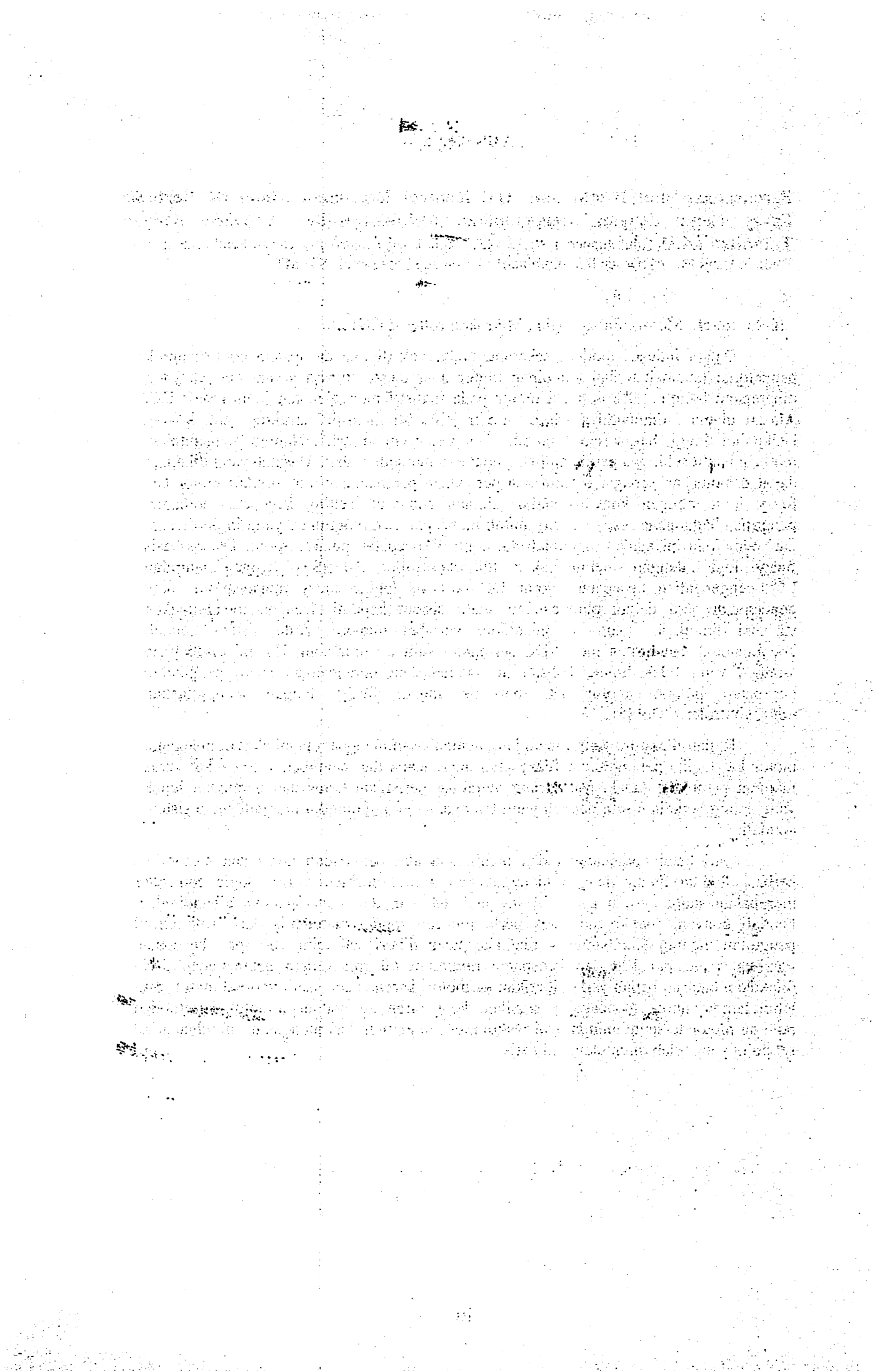
**Perencanaan dan Pembuatan Alat Kontrol Kecepatan Motor DC Berbasis Fuzzy Logic dengan Menggunakan Mikrokontroller AT89S51 dengan Tampilan LCD,**Tulus Junaedhi, 01.12.123,Teknik Energi Listrik S-1,Dosen Pembimbing I: Ir. Widodo Pudji M, MT,Dosen Pembimbing II : I Komang Somawirata,ST, MT

**Kata kunci :** Motor , Fuzzy logic , Mikrokontroller AT89S51

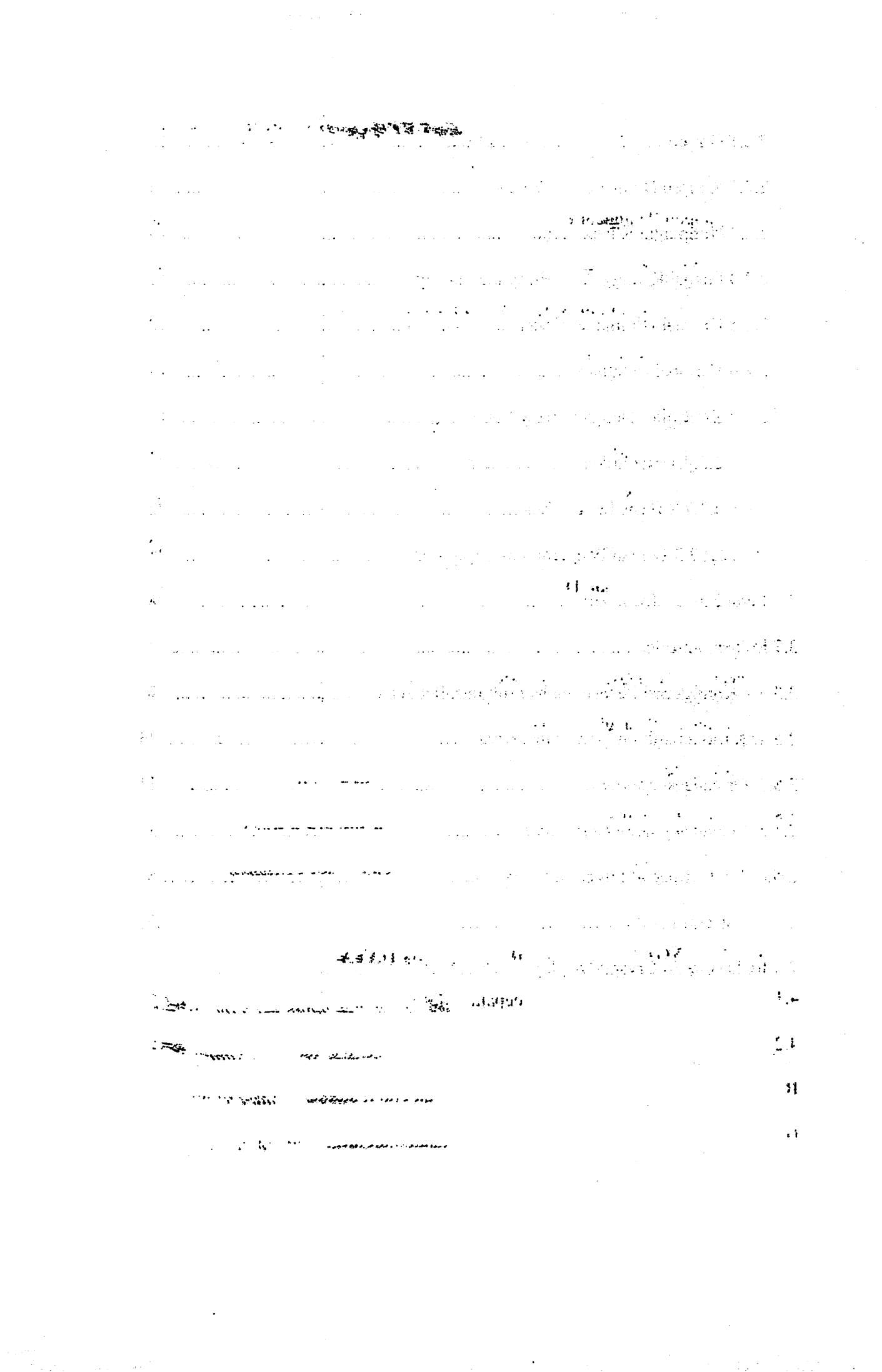
Dalam industri modern sekarang ini,banyak digunakan system control untuk pengaturan kelemahan dan kelebihan motor agar dapat bekerja sesuai dengan yang diharapkan hampir 90% aplikasi motor pada industri menggunakan jenis motor DC. Alasan utama , diantaranya karena motor jenis ini memiliki struktur yang kokoh, keandalan tinggi, harga relatif murah, dan perawatan mudah . Namun pengendalian motor ini sulit sehingga peralatan pengendalinya menjadi mahal. Masalah yang dihadapi dapat dirumuskan sebagai berikut apa perbedaan perancangan alat kendali motor DC fuzzy logic dengan konvensional,bagaimana pengaruh setting kecepatan terhadap pengujian,bagaimana respon waktu untuk mencapai sertpoint,tujuan yang ingin dicapai dari penelitian ini adalah membuat alat control kecepatan putaran motor DC berbasis fuzzy logic dengan menggunakan mikrokontroller AT89S51 dengan tampilan LCD.pengendalian kecepatan motor DC berbasis logika fuzzy menawarkan suatu pendekatan baru dalam perancangan suatu sistem kendali. Dengan menggunakan variabel linguistik, yang menggantikan variabel numerik pada sistem kendali konvensional. Kecepatan motor DC seringkali sulit dikendalikan, Hal ini disebabkan sifatnya yang tidak linier. Tulisan ini menjelaskan perancangan suatu pengendali kecepatan putaran motor DC berbasis logika fuzzy dengan menggunakan mikrokontroler AT89S51.

Kajian literature yang menunjang pembahasan mengenai pembahasan mengenai motor DC,logika pemograman fuzzy guna merencana dan pembuatan perangkat keras maupun perangkat lunak. Merancang membuat perangkat keras dan perangkat lunak yang saling berkaitan satu dengan yang lainnya,kemudian merakit menjadi suatu sistem kendali.

Dari hasil perancangan dan pembuatan alat pengendali kecepatan motor DC berbasis logika Fuzzy dengan menggunakan mikrokontroler : Fuzzy logic controller merupakan suatu system kendali yang jauh lebih mudah penerapannya dibandingkan kendali konvensional,karena tidak perlu mencari model matematis dari system,hasil pengujian menunjukan bahwa ketika kecepatan diberi set point 100 rpm, kecepatan maximal mencapai 120 dan kecepatan minimum 60 rpm dalam pembebanan perlu dilakukan batasan beban yang diberikan ke motor, karena pada saat overload motor akan lebih lambat untuk mencapai kestabilan. Pengaturan kecepatannya untuk kesetabilan putaran motor tersebut,mempunyai waktu yang ditempuh dari mengalami perubahan ke set point yang telah ditentukan 2,5 detik.



# DAFTAR ISI



# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Dalam industri modern sekarang ini,banyak digunakan system control untuk pengaturan kelemahan dan kelebihan motor agar dapat bekerja sesuai dengan yang diharapkan hampir 90% aplikasi motor pada industri menggunakan jenis motor DC. Alasan utama , diantaranya karena motor jenis ini memiliki struktur yang kokoh, keandalan tinggi, harga relatif murah, dan perawatan mudah . Namun pengendalian motor ini sulit sehingga peralatan pengendalinya menjadi mahal.

Logika fuzzy menawarkan suatu pendekatan baru yang lebih mudah dalam pemecahan suatu masalah termasuk perancangan suatu sistem kendali. Kemudahan tersebut ditawarkan melalui penggunaan variabel linguistik, yang lebih dekat dengan domain pemikiran manusia, menggantikan variabel numerik pada sistem kendali konvensional. Jadi perancangan sistem kendali dapat terhindar dari perhitungan-perhitungan matematis yang terkadang sangat rumit terutama untuk sistem kendali yang kompleks. Sistem kendali berbasis logika fuzzy tidak terlalu membutuhkan banyak perhitungan sehingga dapat diterapkan pada mikrokontroler .Tulisan ini membahas proses perancangan suatu perangkat pengendali kecepatan putaran motor DC berbasis fuzzy logic dengan menggunakan mikrokontroller AT89S51 . Alat yang dibuat ini memiliki batasan yaitu kecepatan yang dapat dikendalikannya adalah dari 100 rpm hingga 1500 rpm.

3. Tidak membahas pengasutan motor DC.

4. Tidak membahas system proteksi dari motor DC.

5. Mikrokontroller keluarga ATMEL 89S51.

## 1.5 Metologi Pembahasan

- Kajian literature yang menunjang pembahasan mengenai pembahasan mengenai motor DC,logika pemograman fuzzy guna merencana dan pembuatan perangkat keras maupun perangkat lunak.
- Merancang membuat perangkat keras dan perangakat lunak yang saling berkaitan satu dengan yang lainnya,kemudian merakit menjadi suatu system kendali.
- Melakukan pengujian sistem kendali.

## 1.6 Sistematika Penulisan

Skripsi ini dibagi menjadi lima BAB,yaitu :

1. **BAB I : PENDAHULUAN**
2. **BAB II : LANDASAN TEORI**
3. **BAB III : PERANCANGAN PERANGKAT KERAS DAN LUNAK**
4. **BAB IV : ANALISA PENGUJIAN ALAT.**
5. **BAB V : KESIMPULAN DAN SARAN.**

# BAB II

# LANDASAN TEORI

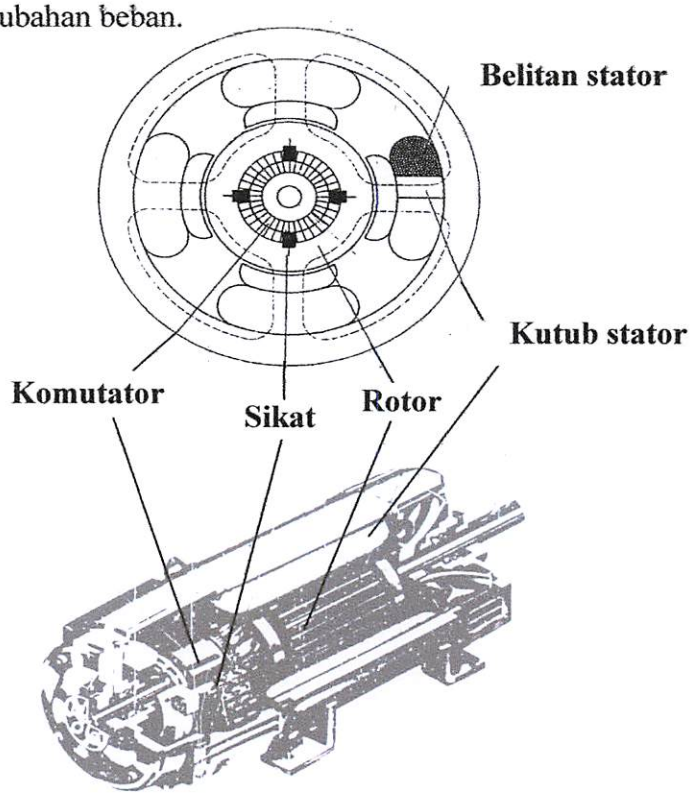
## 2.1. Motor DC

### 2.1.1 Kontruksi

Motor DC adalah peralatan elektromekanis yang mengubah daya listrik menjadi daya mekanis dengan arus searah sebagai suplai energi listriknya. Motor DC terdiri dari dua bagian dasar yaitu *stator* dan *rotor*. Stator merupakan bagian dari motor DC yang tidak bergerak sedangkan rotor merupakan bagian yang bergerak. Pada stator terdapat belitan yang dinamakan belitan medan karena berfungsi menghasilkan medan magnet, sedangkan pada rotor terdapat belitan yang dinamakan belitan jangkar karena berfungsi membawa arus beban. Pada poros rotor terdapat komutator dan sikat, komutator bergerak bersamaan dengan poros rotor sedangkan sikat tidak bergerak tetapi menyentuh komutator. Komutator berupa silinder yang terbuat dari beberapa segmen tembaga yang terisolasi satu sama lain, dan sikat terbuat dari bahan karbon. Komutator dan sikat secara bersamaan berfungsi sebagai penyearah. *Gambar 2.1* merupakan gambar konstruksi motor DC.

Penggunaan motor arus searah sudah sangat dikenal secara luas. Keuntungan-keuntungan yang menonjol akan penggunaan motor-motor arus searah tersebut timbul dengan penunjukan karakteristik operasinya. Motor DC secara luas dipergunakan dalam berbagai macam penerapan yang memerlukan putaran yang dapat diatur dan beberapa penerapannya digunakan pada industri tekstil, industri kertas dan lain-lain.

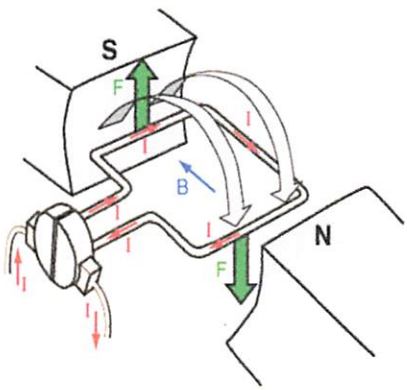
Motor DC terutama motor DC penguatan terpisah mempunyai keunggulan-keunggulan tersendiri, yang mana motor DC penguatan terpisah ini mempunyai kecepatan yang hampir konstan pada tegangan jepit yang konstan meskipun terjadi perubahan beban.




*Gambar 2-1. Konstruksi Motor DC*
*Sumber: www.NEETS.com*

### 2.1.2. Prinsip Kerja Motor DC

Prinsip kerja motor DC berdasarkan pada penghantar yang dialiri arus ditempatkan dalam suatu medan magnet sehingga penghantar tersebut akan mengalami gaya. Gaya menimbulkan torsi sehingga menghasilkan putaran. Penghantar yang berputar akan menimbulkan tegangan AC sehingga diubah menjadi tegangan DC oleh komutator dan sikat.




*Gambar 2-2. Interaksi Antara Medan Magnet Dan Penghantar Yang Dialiri Arus*
*Sumber: www.HyperPhysics.com*

Gaya yang dihasilkan sebesar: *(Cathey, 2001:50)*

$$F = B.I.l \tag{2-1}$$

Gaya itu menimbulkan torsi sebesar:

$$T = F.r \tag{2-2}$$

$$T = B.I.l.r$$

dengan:

$F$ = Gaya (N).

$B$ = Rapat fluks (T).

$I$ = Arus yang mengalir pada penghantar (A).

$l$ = Panjang penghantar (m).

$r$ = Jari-jari inti jangkar (m).

$T$ = Torsi (Nm).

Jangkar memiliki jumlah penghantar dan cabang paralel penghantar sehingga dari Persamaan (2-1) dan (2-2) didapatkan:

$$T = \frac{Z}{a} B.I_a.l.r \qquad (2\text{-}3)$$

dengan:

$Z$ = Jumlah penghantar jangkar.

$a$ = Jumlah cabang paralel penghantar jangkar yang berada di antara sikat.

$I_a$ = Arus jangkar (A).

Rapat fluks yang dihasilkan sebesar:

$$B = \frac{\phi.p}{2\pi.r.l} \qquad (2\text{-}4)$$

Jika Persamaan (2-4) diberikan ke Persamaan (2-3) didapatkan:

$$T = \frac{z}{a} B.I_a.l.r = \frac{z}{a} \frac{\phi.p}{2\pi.r.l} B.I_a.l.r$$

maka akan didapatkan nilai T sebesar :

$$T = \frac{p.Z}{2\pi.a} \phi.I_a \qquad (2\text{-}5)$$

Dimana telah diketahui bahwa besarnya nilai K pada motor DC sebagai berikut :

$$K = \frac{p.z}{2\pi.a}$$

Sehingga persamaan (2-5) dapat ditulis juga sebagai berikut :

$$T = K.\phi.I_a \qquad (2\text{-}6)$$

dengan:

$p$ = Jumlah kutub stator.

$\phi$ = Fluks tiap kutub stator (Wb).

$K$ = Konstanta mesin.

Putaran jangkar yang berada dalam medan magnet akan menghasilkan gaya gerak listrik lawan sebesar:

$$E_a = K.\phi.\omega_m \quad (2\text{-}7)$$

Daya yang dihasilkan sebesar:

$$P = E_a.I_a \quad (2\text{-}8)$$

Dari persamaan (2-7) dan (2-8):

$$P = K\ .\phi.I_a.\omega_m \quad (2\text{-}9)$$

$$P = T.\omega_m \quad (2\text{-}10)$$

dengan:

$E_a$ = Gaya gerak listrik lawan (V).

$P$ = Daya (W).

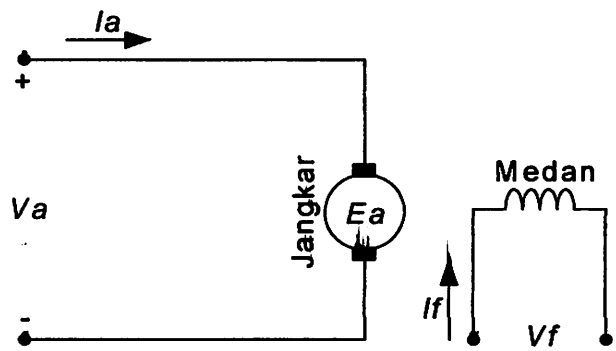
$\omega_m$ = Putaran (rad/s).

**2.1.3. Jenis-Jenis Motor DC**

Motor DC berdasarkan jenis penguatannya terbagi menjadi 2 yaitu: motor DC penguatan terpisah dan motor DC penguatan sendiri. Penguatan pada motor DC diberikan oleh belitan medan sehingga jenis penguatan motor DC berdasarkan pada cara pemberian catu tegangan pada belitan medan yang akan menimbulkan medan magnet.

Motor DC penguatan terpisah dicatu oleh dua sumber tegangan terpisah pada belitan medan dan belitan jangkarnya seperti pada *Gambar 2.3.* Motor DC penguatan sendiri dicatu oleh satu sumber pada belitan medan dan belitan jangkarnya. Motor DC penguatan sendiri berdasarkan cara menghubungkan belitan medan dan belitan jangkarnya terbagi menjadi tiga yaitu: motor DC *shunt*, seri dan kompon. Motor DC *shunt* belitan medan dan belitan jangkarnya dihubungkan paralel, motor DC seri belitan medan dan belitan jangkarnya dihubungkan seri, sedangkan motor DC kompon merupakan penggabungan dari

motor DC *shunt* dan motor DC seri yang terbagi menjadi dua macam yaitu: kompon panjang dan kompon pendek seperti pada *Gambar 2.4.* Motor DC penguatan terpisah dibandingkan motor DC penguatan sendiri memiliki kelebihan dalam pengaturan tegangan sumbernya yaitu pengaturan tegangan jangkar dan pengaturan tegangan medan sehingga memiliki jangkauan pengaturan yang lebih luas.




*Gambar 2-3. Rangkaian Jenis Motor DC Penguatan Terpisah*
*Sumber: Cathey, 2001:242*

### 2.1.4. Motor DC Penguatan Terpisah

Rangkaian motor DC penguatan terpisah seperti pada *Gambar 2.3.* terdiri atas belitan medan dan belitan jangkar yang modelnya dapat diwakili oleh unsur-unsur resistansi dan induktansi. Berdasarkan rangkaian tersebut didapatkan persamaan: *(Slemon, 1992:146)*

$$V_f = R_f.I_f + L_f \frac{dI_f}{dt} \tag{2-11}$$

$$V_a = E_a + I_a.R_a + L_a \frac{dI_a}{dt} \tag{2-12}$$

Jika persamaan (2-7) diberikan ke persamaan (2-12) didapatkan:

$$V_a = K.\phi.\omega_m + I_a.R_a + L_a \frac{dI_a}{dt} \quad (2\text{-}13)$$

Berdasarkan persamaan (2-6) untuk model mekanis motor DC penguatan terpisah adalah:

$$T = J\frac{d\omega}{dt} + B.\omega + T_w \quad (2\text{-}14)$$

dengan:

$V_f$ = Tegangan medan (V).

$L_f$ = Induktansi belitan medan (H).

$R_f$ = Resistansi belitan medan (Ω).

$I_f$ = Arus medan (A).

$V_a$ = Tegangan jangkar (V).

$L_a$ = Induktansi belitan jangkar (H).

$R_a$ = Resistansi belitan jangkar (Ω).

$I_a$ = Arus jangkar (A).

$J$ = Momen inersia (kg.m$^2$).

$B$ = Koefisien gesekan motor {Nm/(rad/s)}.

$T_w$ = Torsi beban (Nm).

Untuk operasi motor dalam keadaan mantap pada persamaan (2-11), (2-12), (2-13), dan (2-14) turunan terhadap waktunya adalah nol sehingga persamaannya berturut-turut menjadi:

$$V_f = R_f.I_f \quad (2\text{-}15)$$

$$V_a = E_a + I_a.R_a \quad (2\text{-}16)$$

$$V_a = K.\phi.\omega_m + I_a.R_a \quad (2\text{-}17)$$

$$T = B.\omega + T_w \quad (2\text{-}18)$$

Untuk menghitung kinerja dari motor DC maka digunakan keempat persamaan di atas, yang terdiri atas tiga persamaan sistem listrik pada jangkar dan medan yaitu: persamaan (2-15), (2-16), (2-17) dan satu persamaan sistem mekanik pada poros rotor yaitu persamaan (2-18).

Berdasarkan persamaan (2-17) didapatkan hubungan antara torsi dan kecepatan motor DC penguatan terpisah sebagai berikut:

$$\omega = \frac{Va - Ia.Ra}{(K\phi)} \quad (2\text{-}19)$$

**2.1.5. Pengaturan Kecepatan Motor DC**

Motor DC merupakan mesin penggerak yang banyak digunakan karena memiliki kelebihan pada pengaturan kecepatannya. Pada umumnya pengaturan kecepatan motor DC penguatan terpisah dan motor DC penguatan sendiri hampir sama. Pengaturan kecepatan pada motor DC penguatan terpisah lebih luas dibandingkan pengaturan kecepatan pada motor DC penguatan sendiri dikarenakan motor DC penguatan terpisah dicatu oleh dua sumber.

Untuk selanjutnya di sini hanya akan ditinjau pengaturan kecepatan motor DC penguatan terpisah.

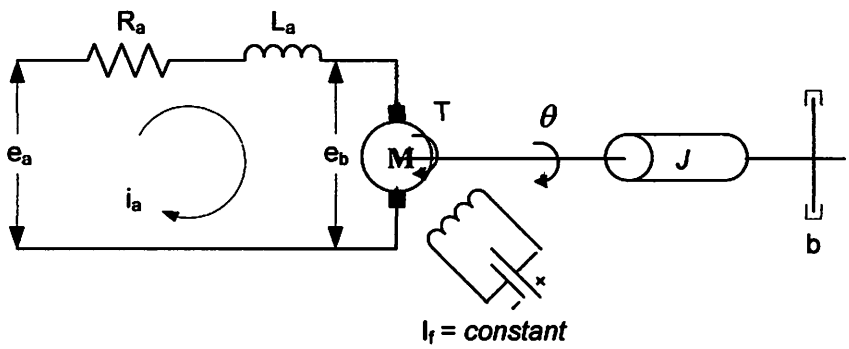
Kecepatan motor DC penguatan terpisah dapat diatur sesuai persamaan (2-19) dengan menggunakan tiga metode yaitu:

- Pengaturan fluksi/medan.
- Pengaturan resistansi jangkar.
- Pengaturan tegangan jangkar.

Untuk selanjutnya di sini hanya akan ditinjau motor DC penguatan terpisah dengan pengaturan jangkar.

### 2.1.6. Pengaturan Tegangan Jangkar

Pada metode ini resistansi jangkar dan sisi medannya dijaga konstan sehingga pengaturan hanya dilakukan pada tegangan jangkarnya. Pengaturan kecepatan dilakukan dengan mengatur tegangan jangkar seperti pada *Gambar 2.4.*




*Gambar 2-4. Diagram Skematik Jangkar-Magnet Terkontrol Motor DC*
*Sumber : Ogata, Jilid I, 1996 : 130*

Dimana :

- $R_a$ = tahanan kumparan jangkar *(ohm)*
- $L_a$ = induktansi kumparan jangkar *(Henry)*
- $e_a$ = tegangan yang dikenakan pada jangkar *(volt)*
- $e_b$ = gaya gerak listrik (ggl) lawan motor *(volt)*
- $i_a$ = arus kumparan jangkar *(ampere)*
- $i_f$ = arus medan *(ampere)*
- $\theta$ = perpindahan sudut dari poros motor *(radian)*
- T = torsi yang diberikan oleh motor *(radian)*
- $J$ = moment inersia ekivalen dari motor dan beban pada poros *(Kg-$m^2$)*

- b = koefisien gesek *(N-m/rad/det)*

### 2.1.7. Spesifikasi Motor DC

Adapun parameter motor DC yang akan digunakan dalam Skripsi ini adalah sebagai berikut :

Adapun spesifikasi dari motor DC yang digunakan adalah sebagai berikut :

Data motor DC penguatan terpisah yang dipergunakan berdasarkan *name plate* adalah sebagai berikut:

- Jenis Motor : Motor DC magnet permanen penguat terpisah

## 2.2 FUZZY LOGIC

### 2.2.1 UMUM

Secara leksikal fuzzy berarti kabur atau tidak jelas. Fuzzy logic merupakan metodologi untuk menyatakan hukum operasional sistem dengan ungkapan bahasa bukan dengan persamaan matematis dengan memberikan kualifikasi secara kualitatif seperti besar, kecil, tinggi, rendah, agak cukup, sangat dan sebagainya yang kesemuanya itu dikatakan sebagai variabel linguistik. Variabel linguistik inilah yang digunakan dalam konsep himpunan fuzzy.

Pada perancangan suatu sistem kontrol umumnya ditentukan model matematisnya terlebih dahulu. Dalam kenyataannya sistem asli ternyata tidak sepenuhnya dapat diwakili oleh model matematis yang dipakai untuk menyatakan sistem tersebut,sehingga sulit digunakan untuk mempresentasikanya. Dalam kasus seperti itu ungkapan bahasa yang digunakan dalam logika fuzzy dapat membantu mendefinisikannya melalui ungkapan bahasa yang dinyatakan dalam implikasi logika yaitu aturan jika-maka (if-then).

Logika Fuzzy Logic Controller (FLC ) memiliki kelebihan yaitu variabel-variabel yang digunakan adalah variabel linguistik sehingga logika FLC ini mengizinkan adanya unsur ketidakpastian, seperti halnya terdapat pada cara berfikir manusia. Logika FLC ini mampu membuat model matematis untuk melakukan pendekatan terhadap ketidakpastian pada cara berfikirnya manusia sehingga hal yang diinginkan untuk dikerjakan oleh komputer yang menggunakan logika Boolean (ekstrim 0 atau 1 ).

### 2.2.2 KONSEP DASAR LOGIKA FUZZY

Ketidakpastian merupakan dasar pemikiran dari logika fuzzy, dan bagaimana logika fuzzy mampu untuk mempresentasikan ketidakpastian yang ada pada suatu sistem merupakan tujuan digunakannya pengendali berbasis logika fuzzy.Dalam sistem pengendalian berbasis logika fuzzy tidak diperlukan adanya model matematis maupun fungsi alih tetapi didasarkan pada kecerdasan buatan/ AI (Artificial Intelegence).

### 2.2.3 HIMPUNAN KLASIK

Dalam mengontrol sistem atau proses seringkali menggunakan besaran pengukuran yang dinyatakan dengan ungkapan cepat, lambat, cukup, agak dan sebagainya. Untuk mempresentasikan yang tidak eksak ini digunakan suatu pendekatan yaitu dengan himpunan fuzzy.

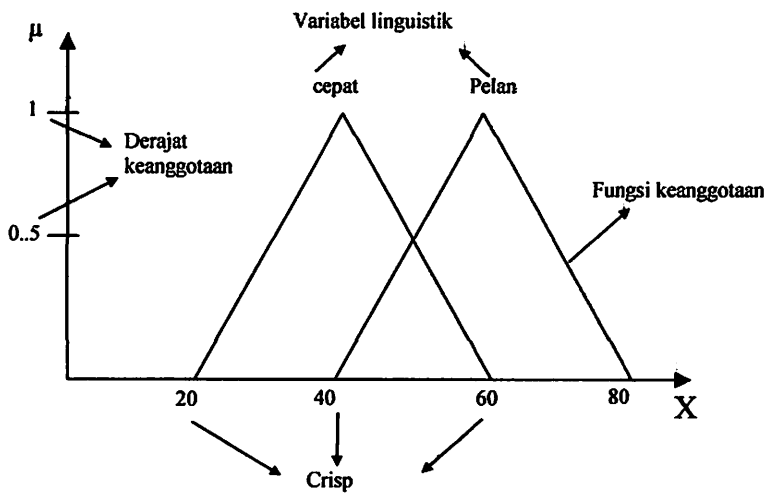
Pada himpunan klasik logika yang digunakan adalah logika Boolean (Boolean Logic).Nilai yang dimungkinkan antara 0 atau 1 yang menunjukkan bukan anggota atau anggota himpunan bagian. Harga 0 dan 1 diberikan oleh derajat keanggotaan (degree of membership) $\mu$. Suatu pernyataan yang menggunakan logika boolean dinamakan crisp atau pernyataan non fuzzy.

Harga kebenaran himpunan A dalam semesta pembicaraan X dapat dinyatakan dengan :

$$\mu_A(X): \begin{cases} 1 \text{ jika } \in A \\ 0 \text{ jika } \notin A \end{cases}$$

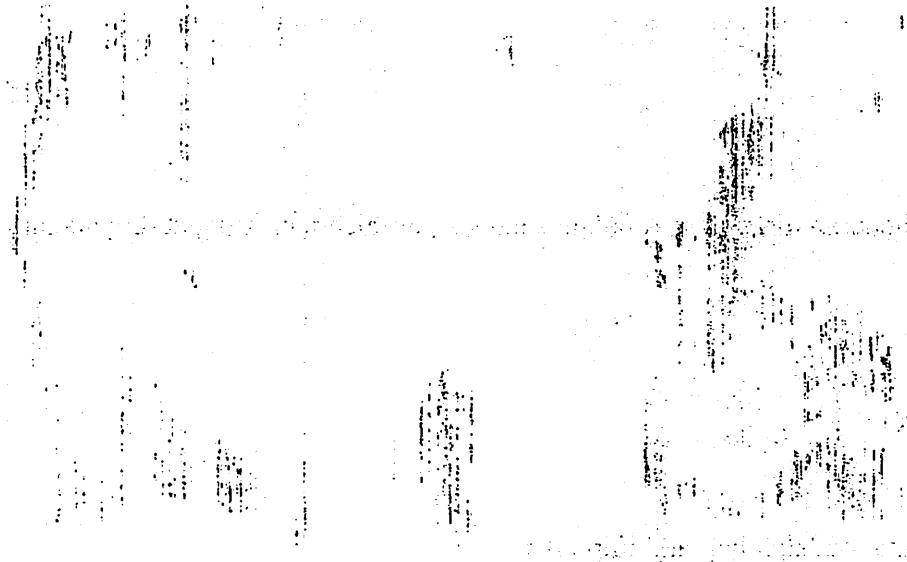
dimana $\in$ adalah bagian himpunan.

Pada fuzzy logic harga kebenaran ini diberikan oleh termonologi linguistik dengan menyatakan derajat kekaburannya (fuzziness). Agar lebih jelas gambar 2 - 9 memperlihatkan himpunan fuzzy beserta istilah-istilahnya



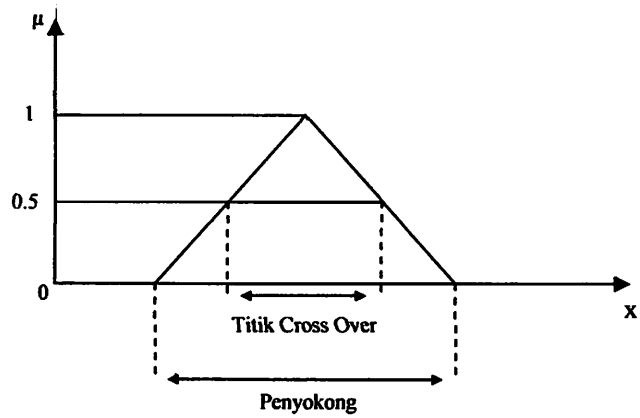

**Gambar 2 -5**
**Himpunan Fuzzy Dan Istilahnya**

Jika X mempunyai elemen-elemen atau titik-titik yang terdiri dari x maka suatu elemen x dalam himpunan bagian fuzzy A mempunyai derajat keanggotaan $\mu_A$ (x). Jika $\mu_A(x) = 1$ maka x adalah himpunan bagian A, begitu pula sebaliknya jika $\mu_A(x) = \mu$ dengan $0<\mu<1$ maka dikatakan anggota A mempunyai derajat keanggotaan $\mu$. Dalam himpunan fuzzy A, x disebut sebagai penyokong (support) A.Penyokong himpunan bagian fuzzy A adalah kumpulan semua titik mulai dari $x_1, x_2, \ldots\ldots, x_n$ yang mana $\mu_A(x)>0$ atau dapat ditulis sebagai :



Penyokong = $\{x \mid \mu_A(x)>0\}$

Jika $\mu_A(x) = 0.5$ dikatakan sebagai titik cross over di A. akan menjelaskan pernyataan tersebut.




**Gambar 2 - 6**
**Himpunan Fuzzy Dan Penyokongnya**

Himpunan fuzzy beserta unsur-unsur penyokongnya dapat dinyatakan sebagai berikut :

- $A = \{ \mu_A(xi), I = 1, 2, 3, \ldots\ldots, n \text{ dan } x \in X = \text{semesta pembicaraan} \}$ atau dapat pula dinyatakan dalam
- $A = \mu_1/x_1 + \mu_2/x_2 + \ldots\ldots + \mu_n/x_n$, dimana tanda + menyatakan gabungan

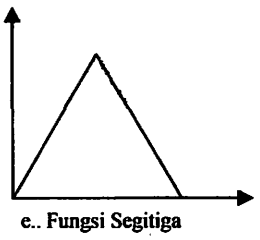
$$= \sum_{i=1}^{n} \mu_i/x_i$$

$$= \bigvee_{i}^{n} \mu_i/x_i$$

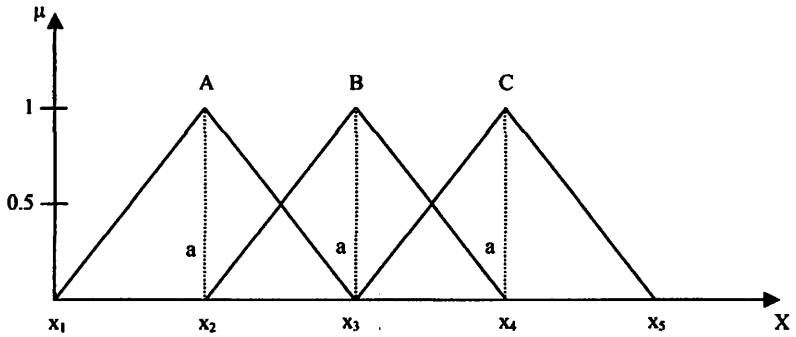
### 2.2.4 FUNGSI KEANGGOTAAN HIMPUNAN FUZZY

Untuk menyatakan tiap keanggotaan dari tiap penyokong dalam himpunan fuzzy digunakan fungsi keanggotaan (membership function). Fungsi keangotaan ini mengkarakteristikkan tiap penyokong dari himpunan fuzzy sedemikian rupa sehingga setiap penyokong mempunyai nilai keanggotaan dalam interval [0 : 1].

Fungsi keanggotaan dapat berbentuk fungsi segitiga, fungsi eksponen, trapesium, fungsi S atau fungsi Z. Gambar 2 - 11 memperlihatkan fungsi segi tiga karena mempunyai perhitungan yang lebih cepat.




**Gambar 2 - 7**
**Bentuk Fungsi Segi Tiga Keanggotaan**




**Gambar 2 - 8**
**Fungsi Keanggotaan Segitiga**

Derajat keanggotaan himpunan A adalah :

$\mu_A(x) = 0 \qquad \forall\ x \leq x_1$

$\mu_A(x) = (x-x_1)/x_2 \qquad \forall\ x_1 \leq x \leq x_2$

$=(x_3-x)/x_2 \qquad \forall\ x_2 \leq x \leq x$

dimana $\forall$ = "didefinisikan dengan" atau "dimana".

Derajat keanggotaan masing-masing himpunan bagian fuzzy dapat pula ditentukan dengan rumus :

$$\mu_F(x) = 1 - \sqrt{\frac{(x-a)^2}{b}}$$

dimana b adalah titik tengah semua himpunan bagian fuzzy. Karena yang digunakan adalah fungsi segitiga simetris (seragam) maka nilai b adalah sama untuk semua himpunan. Dalam hal ini $b = (x_3 - x_1)/2$ untuk himpunan A, $(x_4 - x_2)/2$ untuk himpunan B dan $(x_5 - x_3)/2$ untuk himpunan C. Sedangkan a adalah titik tengah masing-masing himpunan bagian fuzzy, yitu untuk himpunan A:$a = x_2$ untuk himpunan B:$a = x_3$ dan untuk himpunan C:$a = x_4$.

Pada umumnya fungsi keanggotaan yang digunakan didefinisikan dengan dua cara yaitu :

1) Pendefinisian secara numerik, digunakan untuk mendefinisikan fungsi keanggotaan pada himpunan fuzzy dengan bilangan.
2) Pendefenisian secara fungsional digunakan untuk mendefinisikan fungsi keanggotaan pada himpunan fuzzy dengan fungsi keanggotaan.

### 2.2.5 Operasi Himpunan Fuzzy

Operasi himpunan A dan B dari semesta pembicaraan (universe of discourse) X dengan fungsi keanggotaan masing-masing $\mu_A$ dan $\mu_B$ adalah :

a. A adalah komplemen B

$$\mu_A(x) = 1 - \mu_B(x)$$

b. Gabungan ( union ) A dengan b atau ( $A \cup B$ )

$$\mu_{A \cup B}(x) = M_{LX}[\mu_A, \mu_B](x)$$
$$= v[\mu_A, \mu_B]$$

c. Irisan ( Intersection ) A dengan B atau ( $A \cap B$ )

$$\mu_{A \cap B}(x) = \text{Min}[\mu_A, \mu_B]$$
$$= \wedge [\mu_A, \mu_B]$$

### 2.2.6 Variabel Linguistik

Dalam sistem kontrol variabel linguistik dapat dinyatakan dengan ungkapan linguistik NB ( Negatif Besar), NS ( Negatif Sedang), NK (Negatif Kecil), SN (Sekitar Nol), PK (Positif Kecil), PS (Positif Sedang), dan PB (Positif Besar) untuk variabel masukan dan keluaran.

Secara sederhana variabel linguistik dapat dinyatakan dengan pasangan ($\mu$, T($\mu$), X). $\mu$ menyatakan nama variabel dan T($\mu$) adalah istilah yang menyatakan seperangkat nama dari besaran linguistik untuk himpunan bagian fuzzy pada semesta pembicaraan X.

### 2.2.7 KONTOLLER DENGAN FUZZY LOGIC

Perancangan fuzzy logic kontroller menggabungkan aspek pendefinisian himpunan fuzzy dengan aspek fuzzy logic untuk memperoleh suatu sistem kontrol atau yang dikenal dengan sistem pengaturan cerdas. Sistem pengaturan cedas ini dapat dirancang berdasarkan empat pendekatan yaitu :

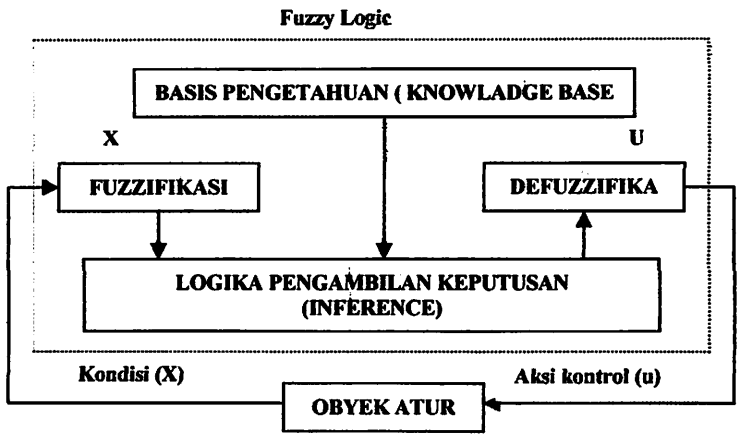
1. Pendekatan berdasarkan basis pengetahuan (knowladge Base) seorang operator ahli.
2. Pendekatan fuzzy logic dengan variabel linguistik.
3. Pendekatan dengan jaringan syaraf tiruan (artificial neural network).
4. Gabungan antara artificial neural network dan fuzzy logic yang dikenal sebagai Neuro fuzzy.

Fuzzy logic controller membuat keputusan dengan menggunakan aturan if-then (jika-maka) berdasarkan masukan dan keluaran.

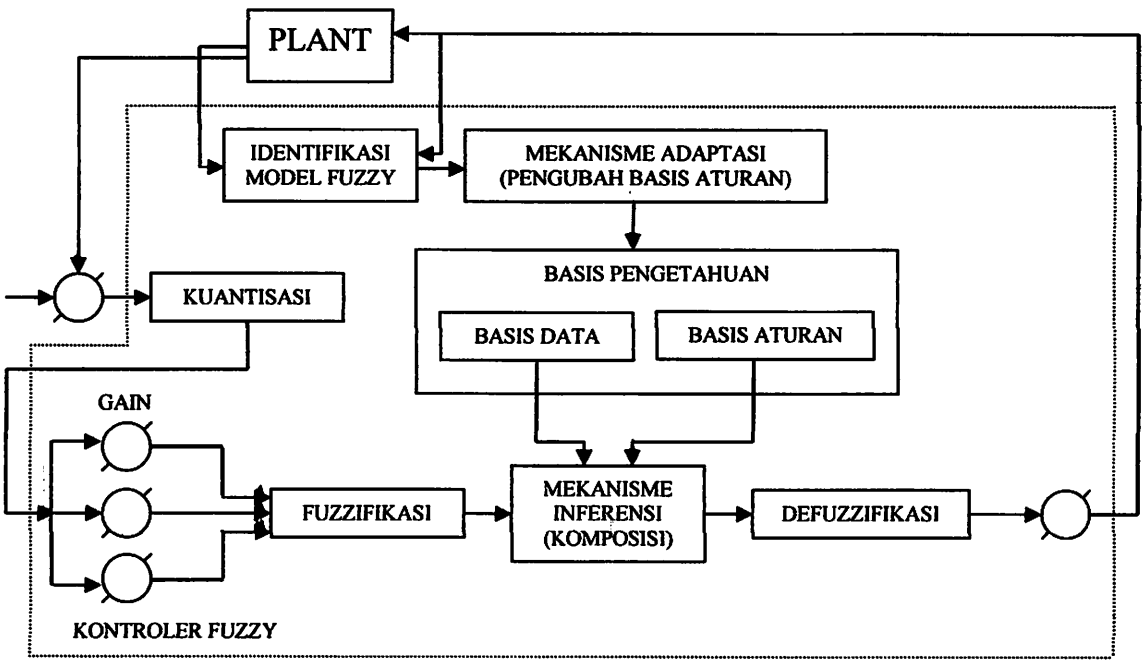
" jika e = NB dan de = PK maka u = NS "

Dimana e = error output dengan variabel linguistik NB, de = perubahan error dengan variabel linguistik PK, dan u = sinyal atur dengan variabel linguistik NS.

Secara umum fuzzy logic controller mempunyai empat bagian pokok yang dipresentasikan




**Gambar 2 - 9**
**Bagian Pokok Fuzzy Logic Controller**




**Gambar 2 - 10**
**Diagram Blok Sistem Pengaturan Dengan Logika Fuzzy**

Keempat bagian tersebut mempunyai fungsi sebagai berikut :

1) Fuzzifikasi, berfungsi untuk mentransformasikan sinyal masukan yang bersifat crisp (non fuzzy) ke himpunan fuzzy dengan menggunakan operator fuzzier.
2) Basis pengetahuan, berisi basis data dan basis aturan yang mendefinisikan himpunan fuzzy atas daerah-daerah masukan dan keluaran dan menyusunnya kedalam basis aturan.
3) Logika pengambilan keputusan, adalah inti dari fuzzy logic controller.
4) Defuzzifikasi, berfungsi untuk mentransformasikan kesimpulan tentang aksi kontrol yang bersifat fuzzy menjadi sinyal yang bersifat crisp melalui operator defuzzier.

**2.2.7.1 Fuzzifikasi**

Dalam aplikasi suatu pengaturan, besaran masukan yang diperoleh dari sistem melalui sensor akan selalu berupa crisp yang bersifat pasti dan kualitatif. Pemetaan dari masukan crisp kedalam himpunan fuzzy yang memakai variabel linguistik dinamakan pengkaburan atau fuzzyfikasi (fuzzyfication).

Fuzzifikasi merupakan proses awal untuk mengubah masukan yang berupa non fuzzy menjadi himpunan fuzzy menjadi himpunan fuzzy sehingga dalam tahap ini, mula-mula dikembangkan fungsi keanggotaan dan derajat keanggotan. Suatu masukan crisp mempunyai derajat keanggotaan dalam beberapa fungsi keanggotaan sekaligus dalam suatu saat.

Fuzzifikasi dinyatakan oleh pernyataan x = Fuzzifier dimana X adalah masukan crisp (non fuzzy), x adalah himpunan fuzzy yang disertai dengan derajat keanggotaan dan fuzzifier adalah operator fuzzifikasi.

**2.2.7.2 Basis Pengetahuan**

Basis pengetahuan (knowladge base) dalam fuzzy logic controller adalah bagian yang berisi basis data dan basis aturan. Basis pengetahuan harus dipersiapkan terlebih dahulu sebelum proses yang lain dimulai, sehingga pada saat melakukan proses mekanisme inferensi basis pengetahuan langsung dapat dipergunakan.

**1) Basis Data**

Basis data mendefinisikan himpunan fuzzy atas himpunan input-output dalam bentuk variabel linguistik. Pembuatan basis data harus memperhatikan hal-hal sebagai berikut :

1) Kuantisasi.

Proses kuantisasi yaitu pembagian semesta pembicaraan dalam segmen-segmen tertentu yang disebut dengan level kuantisasi. Prosedur ini biasanya dinyatakan dalam bentuk tabulasi penerapannya dalam bentuk look up table (tabel pandang ).

Pemilihan jumlah level kuantisasi mempengaruhi kepekaan fuzzy logic controller terhadap masukan dan kehalusan aksi kontrol pada keluaran. Semakin banyak level kuantisasinya adalah semakin baik. Tetapi karena look up table menggunakan beberapa memori pada komputer maka dalam menentukan jumlah level harus memperhatikan penggunaan memori

2) Pemilihan fungsi keanggotaan

Pemilihan fungsi keanggotaan dapat dipilih bebas termasuk jenis fungsinya beserta nilai keanggotaannya. Tetapi pemilihan ini harus dapat menggambarkan karakteristik masing-masing himpunan fuzzy.

**a. Basis Aturan**

Basis aturan adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikkan oleh sekumpulan variabel-variabel linguistik.Pernyatan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat. Dalam pembentukan basis aturan perlu diperhatikan aspek :

a) Variabel masukan dan keluaran error (e) dan perubahan error (de) dan hasil aksi kontrol (u) .

b) Penurunan aturan kontrol fuzzy

Salah satu cara menurunkan aturan kontrol fuzzy adalah dengan mengumpulkan aturan-aturan kontrol fuzzy yang dibentuk dari analisa perilaku obyek atur. Aturan kontrol diturunkan dengan jalan mengkoreksi simpangan keluaran sistem menuju kekeadaan yang diinginkan.

Agar diperoleh kinerja sistem kontrol yang lebih baik digunakan tujuh variabel linguistik yaitu NB, NS, NK, SN, PK, PS dan PB yang masing-masing berarti negatif besar, negatif sedang, negatif kecil, sekitar nol, positif kecil, positif sedang, positif besar untuk variabel e, de dan u Penurunan lengkap dari aturan kontrol ini ditunjukkan dalam tabel 2 - 1 yang disebut sebagai tabel Mac Vicar Whelan.

**Tabel 2 - 1**
**Aturan Kontrol Fuzzy Macvicar-Whelan**

| | | Δe | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NS | NK | SN | PK | PS | PB |
| error | NB | NB | NB | NB | NB | NS | NK | SN |
| | NS | NB | NB | NB | NS | NK | SN | PK |
| | NK | NB | NB | NS | NK | SN | PK | PS |
| | SN | NB | NS | NK | SN | NK | PS | PB |
| | PK | NS | NK | SN | PK | PS | PB | PB |
| | PS | NK | SN | PK | PS | PB | PB | PB |
| | PB | SN | PK | PS | PB | PB | PB | PB |

### 2.2.7.3 LOGIKA PENGAMBILAN KEPUTUSAN

Persoalan dari fuzzy logic controller terletak pada logika pengambilan keputusannya yang meniru pengambilan keputusan pada manusia. Untuk memahami proses pengambilan kesimpulandalam fuzzy logic, hal-hal berikut ini harus diperhatikan.

**a. Bentuk Aturan Kontrol Fuzzy**

Menggunakan aturan kontrol yang berbentuk if-then. Untuk sistem dengan satu masukan-satu keluaran (Single Input Single Output = SISO ) aturan kontrolnya adalah :

- Aturan ke-n : Jika A adalah $x_n$ maka B adalah $y_n$

Dimana A adalah variabel masukan dan B adalah variabel keluaran serta x dan y adalah variabel linguistiknya. Sedangkan untuk sistem banyak masukan satu keluaran (Multi Input Single Output) aturan kontrolnya adalah :

- Aturan ke-n : Jika A adalah $x_n$ dan B adalah $y_n$ maka C adalah $z_n$

  Dimana A dan B adalah variabel masukan (misalnya e dan de), x dan y adalah himpunan fuzzy dengan variabel linguistik yaitu, NB, NS, NK, SN, PK, PS, PB, C adalah variabel keluaran, z adalah himpunan fuzzy untuk z yaitu, NB, NS, NK, SN, PK, PS, PB.

**b. Fungsi Implikasi**

Aturan kontrol pada dasarnya adalah relasi fuzzy yang dinyatakan sebagai suatu hubungan sebab akibat yang disebut sebagai implikasi fuzzy (fuzzy implication). Pada suatu sistem kontrol dengan logika fuzzy hubungan ini menyatakan suatu pemetaan antara variabel fuzzy melalui pernyataan kondisional yaitu :

- Jika A adalah x maka B adalah y atau $A \rightarrow B$ dimana $x \in X$ dan $y \in Y$ atau

  $\mu_R = \mu (A \rightarrow B) (x,y)$ .....(2.14)

Tanda ($\rightarrow$) menyatakan fungsi implikasi yang menghubungkan suatu input (anteccendent) dari himpunan A dengan output B (consequent). Implikasi linguistik ini dinamakan relasi fuzzy.

Persamaan diatas digunakan untuk sistem satu input-satu output (SISO). Untuk sistem MISO berlaku :

- Jika A adalah X dan B adalah y maka C adalah z atau A x B $\rightarrow$C atau

  $\mu_R = \mu$ (A dan B$\rightarrow$C) (x,y,z) dimana $x \in X$, $y \in Y$ dan $z \in Z$

  $= [\mu_A (x)$ dan $\mu_B (y)] \rightarrow \mu_c (z)$ .....(2.15)

## c. MEKANISME INFERENSI

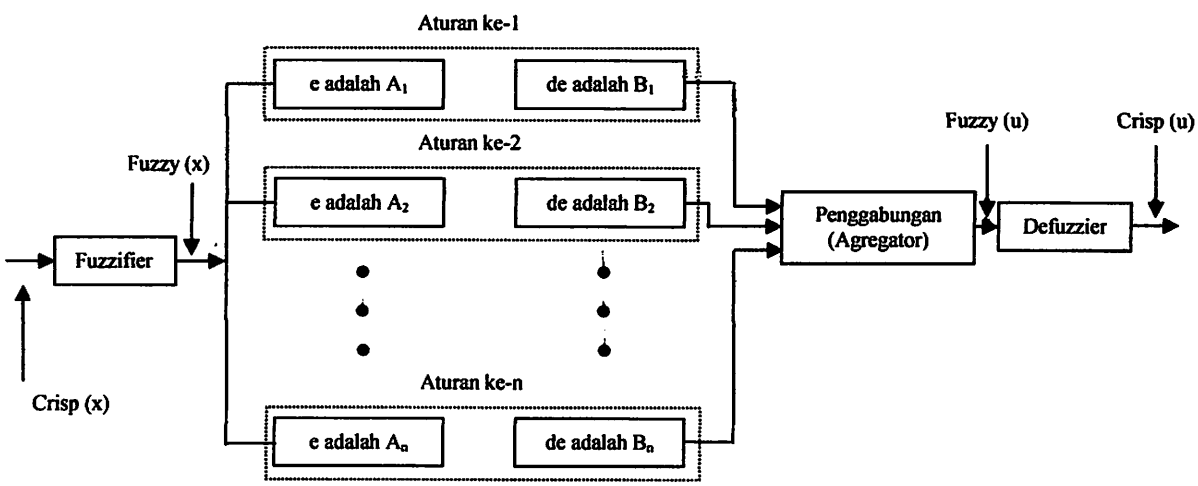
Untuk mengkombinasikan aturan-aturan kontrol yang digunakan kata hubung yang secara umum dinyatakan dengan :

- Rule 1 : Jika e adalah $A_1$ dan de adalah $B_1$ maka u adalah $C_1$
- Rule 2 : Jika e adalah $A_2$ dan de adalah $B_2$ maka u adalah $C_2$
- Rule n : Jika e adalah $A_n$ dan de adalah $B_n$ maka u adalah $C_n$

---

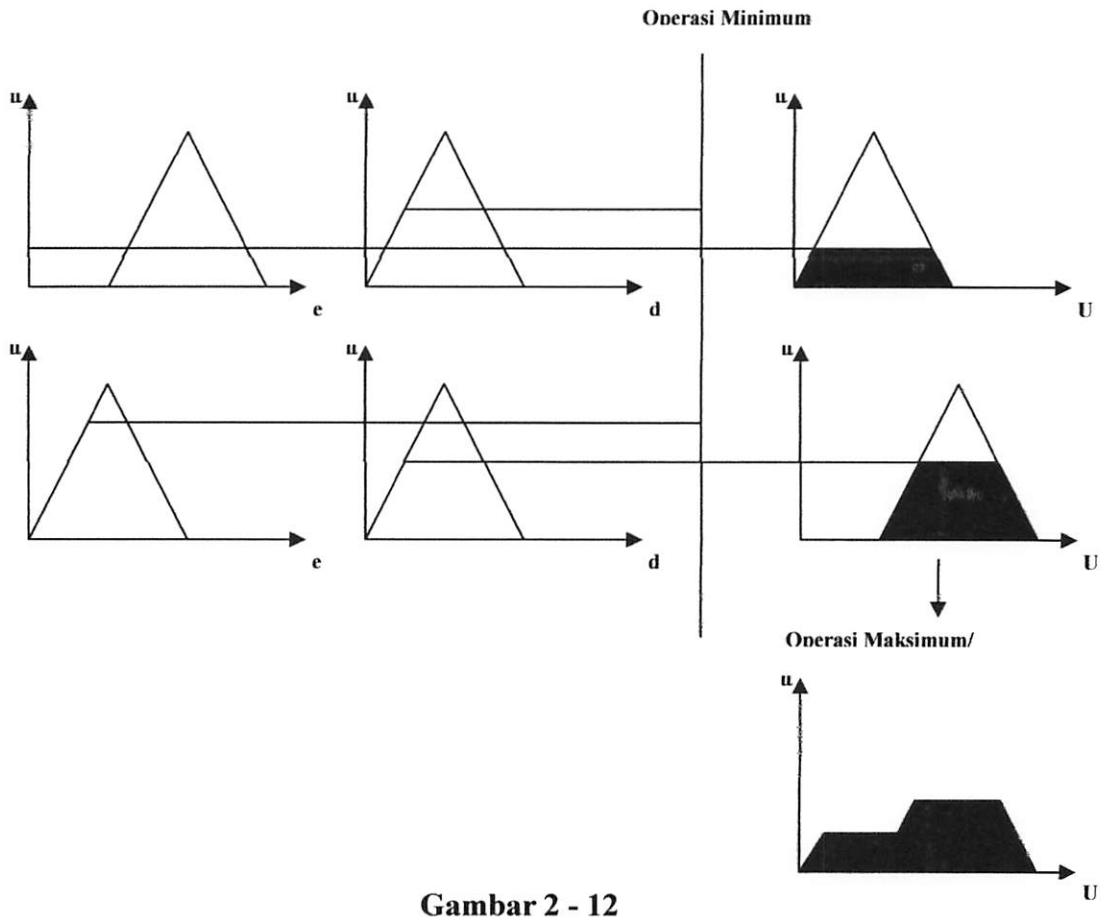
- Kesimpulan : u adalah C.

Maksud dari pernyataan diatas adalah variabel masukan e dan de masing-masing mempunyai linguistik A ( $A_1$.....$A_n$) dan B ( $B_1$.....$B_n$) dan variabel keluaran u mempunyai variabel linguistik C ( $C_1$.....$C_n$). Keseluruhan aturan kontrol ini harus dapat menghasilkan satu keluaran berupa kesimpulan. Proses pengambilan kesimpulan ini dinyatakan sebagai mekanisme inferensi.




**Gambar 2 - 11**
**Blok Diagram Mekanisme Inferensi**

Metode inferensi yang banyak digunakan adalah metode inferensi Mamdani.Metode inferensi untuk MISO dari Mamdani yang digunakan adalah metode maksimum-minimum (max-min method). Prinsip dasar operasi max-min terlhat dalam gambar 2 – 1



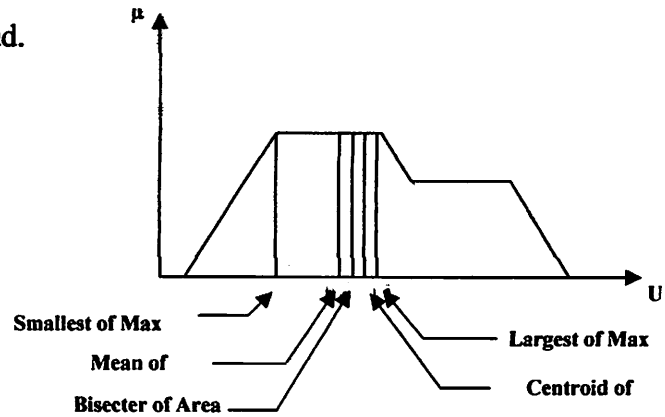

**Gambar 2 - 12**
**Mekanisme Inferensi Miso Dengan Operasi Max-Min**

### 2.2.7.4 DEFUZZIFIKASI

Proses defuzzifkasi merupakan kebalikan dari proses fuzzifikasi, yaitu mentransformasikan harga fuzzy ke harga bukan fuzzy (crisp) atau pemetaan dari ruang aksi kontrol fuzzy keruang aksi kontrol crisp. Secara garis besar strategi defuzzifikasi bertujuan menghasilkan aksi kontrol yang nyata yang dapat mempresentasikan aksi kontrol masing-masing basis aturan. Defuzzifikasi ini dinyatakan sebagai :

U = Defuzzifier (U) .....(2.16)

Ada beberapa metode defuzzifikasi yang terdapat dalam literatur, yaitu centroid of area, mean of max membership, bisevter of area, smallest of max dan largest of max. Gambar 2-19 mempresentasikan metode-metode defuzzifikasi yang dimaksud.




**Gambar 2 - 13**
**Metode-Metode Defuzzifikasi**

Dari lima metode defuzzifikasi tersebut, yang paling banyak digunakan adalah centroid of area method atau yang disebut juga sebagai center of area dan center of gravity. Metode ini didefinisikan sebagai nilai yang terletak dalam jangkauan variabel sinyal atur u dengan daerah yang terletak dibawah fungsi keanggotaan dibagi dua sub daerah yang sama.

Metode defuzzifikasi dengan centroid of area method mempunyai persamaan sederhana :

$$\mathbf{U} = \frac{\sum U.\mu}{\sum \mu} \quad \textbf{untuk SISO} \quad \text{.....(2.17)}$$

Sedangkan untuk MISO berlaku :

$$\mathbf{U} = \frac{\sum Ui.\mu i}{\sum \mu i} \quad \text{.....(2.18)}$$

Dimana:

i = 1, ....., 49 menyatakan jumlah aturan kontrol MISO (7 x 7)

$U_i$, adalah titik pusat masing-masing himpunan variabel keluaran ke-I

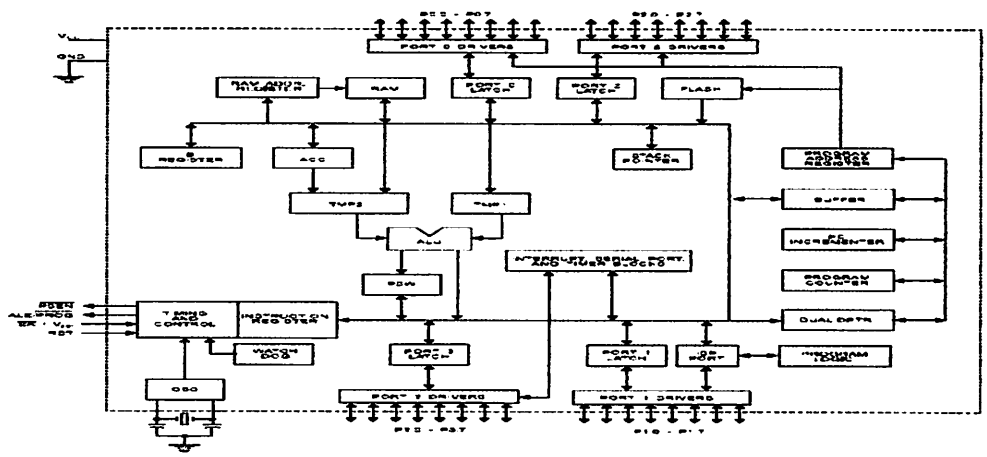
$\mu_I$, adalah variabel minimum untuk tiap variabel linguistik.

## 2.3. Mikrokontroler AT89S51

Secara umum Mikrokontroller AT89S51 memiliki :

- CPU 8 bit termasuk keluarga MCS-51
- 4 Kb Flash memory
- 128 byte Internal RAM
- 32 buah Port I/O, masing – masing terdiri atas 8 jalur I/O
- 2 Timer/ counter 16 bit
- 2 Serial Port Full Duplex
- Kecepatan pelaksanaan intruksi per siklus 1 us pada frekuensi clock 12 Mhz
- 2 DPTR (*Data Pointer*)
- Watchdog Timer
- Fleksibel ISP Programming

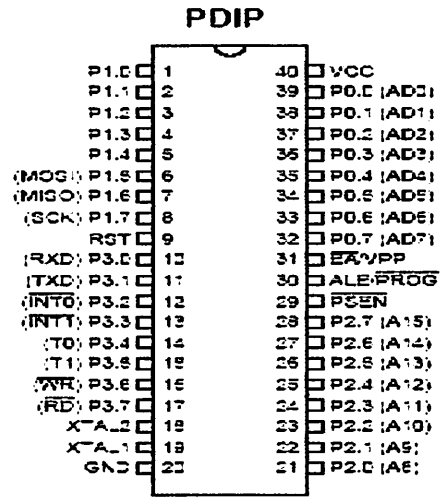
Dengan keistimewaan diatas pembuatan alat menggunakan AT89S51 menjadi lebih sederhana dan tidak memerlukan IC pendukung yang banyak. Adapun blok diagram dari Mikrokontroller AT89S51 adalah sebagai berikut:



**Gambar 2-2. Diagram Blok Mikrokontroller**

### 2.5.3. Konfigurasi Pena-Pena Mikrokontroller AT89S51

Mikrokontroller AT89S51 terdiri dari 40 pin dengan konfigurasi sebagai berikut:




**Gambar 2-14. Konfigurasi Pena-Pena AT89S51** [9]

Fungsi tiap pin-nya adalah sebagai berikut :

- VCC (Supply tegangan), pin 40
- GND (*Ground*), pin 20
- Port 0, pin 32 – 39

  Merupakan port input-output dua arah, tanpa internal pull-up dan konfigurasikan sebagai multipleks bus alamat rendah ($A_0$ – $A_7$) dan data selama pengaksesan program memory dan data memory eksternal
- Port 1, pin 1 – 8

  Merupakan port input-output dua arah dengan internal pull-up.

| Port Pin | Alternate Functions |
|---|---|
| P1.5 | MOSI (used for In-System Programming) |
| P1.6 | MISO (used for In-System Programming) |
| P1.7 | SCK (used for In-System Programming) |

## 1.2 Rumusan Masalah

Berdasarkan latar belakang diatas masalah yang dihadapi dapat dirumuskan sebagai berikut

1. Apa perbedaan perancangan alat kendali motor DC fuzzy logic dengan konvensional,
2. Bagaimana pengaruh setting kecepatan terhadap pengujian,
3. Bagaimana respon waktu untuk mencapai sertpoint,

## 1.3 Tujuan

Tujuan yang ingin dicapai dari skripsi ini adalah membuat alat kontrol kecepatan putaran motor DC berbasis fuzzy logic dengan menggunakan mikrokontroller AT89S51 dengan tampilan LCD.

## 1.4 Batasan Masalah

Permasalahan dalam motor DC dan Mikrokontroller adalah merupakan permasalahan yang luas luas sehingga dalam menganalisa suatu permasalahan perlu diadakan pembatasan-pembatasan yang sesuai dengan permasalahan tersebut.

Didalam penulisan skripsi ini pembatasan yang dilakukan adalah sebagai berikut:

1. Analisa dilakukan pada motor DC magnet permanen penguat pemanen
   - Type motor DGM-208-2A
   - V nom : 12V
2. Pembahasan hanya ditekankan pada analisis respon pengendalian motor DC oleh fuzzy contrller.

Merupakan Port yang digunakan sebagai ISP header.

- Port 2, pin 21 - 28

  Merupakan port input-output dengan internal pull-up. Mengeluarkan alamat tinggi selama pengambilan program memory external.

- Port 3, pin 10 – 17

  Merupakan port input-output dengan internal pull-up, dimana Port 3 juga memiliki fungsi khusus dan dapat dilihat pada tabel berikut ini;

**Tabel 2-15 Fungsi Khusus Pada Port 3**

| Nama Penyemat | Fungsi Khusus |
|---|---|
| *Port 3.0* | RxD (port masukan serial) |
| *Port 3.1* | TxD (port keluaran serial) |
| *Port 3.2* | /INTO (masukan interupsi eksternal 0) |
| *Port 3.3* | /INT1 (masukan interupsi eksternal 1) |
| *Port 3.4* | T0 (masukan pewaktu eksternal 0) |
| *Port 3.5* | T1 (masukan pewaktu eksternal 1) |
| *Port 3.6* | /WR (sinyal tulis memori data eksternal) |
| *Port 3.7* | /RD (sinyal baca memori data eksternal) |

- RST (*Reset*), pin 9

  Input Reset merupakan reset master untuk AT89S51.

- ALE/ Prog (*Address Latch Enable*), pin 30

  Digunakan untuk menahan alamat memori eksternal selama pelaksanaan intruksi.

- PSEN (*Program Store Enable*), pin 29

Merupakan sinyal pengontrol yang memperbolehkan program memori eksternal masuk ke dalam bus.

- EA / VPP (*External Access),* pin 31

  Dapat diberikan logika rendah (*Ground*) atau logika tinggi (+5V). Jika diberikan logika tinggi maka mikrokontroller akan mengakses program dari ROM internal (EEPROM/*Flash Memori*), dan jika diberikan logika rendah maka mikrokontroller akan mengakses program dari memori eksternal.

- X-TAL 1 dan X-TAL 2, pin 19, 18

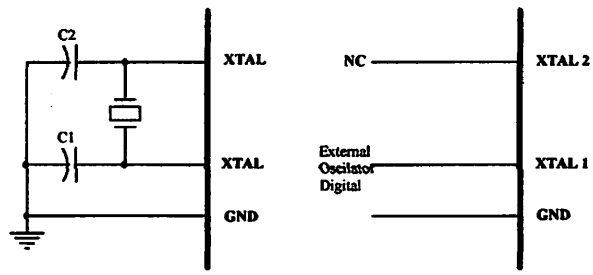
  Kaki ini dihubungkan dengan kristal bila menggunakan *osilator internal.* XTAL 1 merupakan *input inverting osilator amplifier* sedangkan XTAL 2 merupakan *out-put inverting osilator amplifier.*

### 2.5.4. Karakteristik *Oscillator Inverting.*

XTAL 1 dan XTAL 2 secara berurutan merupakan *input dan output* dari sebuah *inverting amplifier* yang dapat dikonfigurasikan penggunaannya sebagai *on chip oscillator* seperti yang ditunjukkan pada gambar 2-4a dibawah ini. XTAL1 dan XTAL 2 ini dapat menggunakan sebuah *kristal quartz* maupun *resonator* keramik.

Untuk memberikan AT89S51 dari sumber *clock external.* Maka pin XTAL 2 dibiarkan tidak berhubungan dan XTAL 1 dihubungkan dengan sumber *clock external* seperti pada gambar 2-4b. Rangkaian ini tidak melakukan *duty cycle* dari setiap sinyal *clock internal,* karena *input* bagi masukan rangkaian *clock*

*internal* dihubungkan ke *flip-flop* pembagi dua, tetapi spesifikasi nilai tegangan pada saat tinggi dan rendah, maksimum dan minimumnya harus diberikan.




a) *Oscilator Connector* b)*External Clock Drive Configuration*

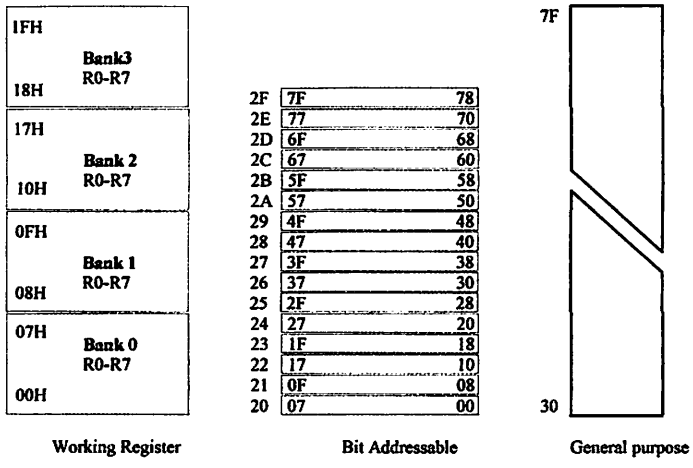
**Gambar 2-4. Karakteristik *Oscillator***

### 2.5.5. Organisasi *Memory.*

Mikrokontoller AT89S51 memiliki ruang alamat memori data dan memori program yang terpisah. Pemisahan memori program dan memori data tersebut membolehkan memori data diakses dengan alamat 8-bit, sehingga dapat dengan cepat dan mudah disimpan dan dimanipulasi oleh CPU 8-bit. Namun demikian, alamat memori data 16-bit bisa juga dihasilkan melalui register DPTR

#### 2.5.5.1. *Program Memory Internal*

AT89S51 memiliki program *memory internal* sebesar 4K*byte* dengan ruang alamat 0000H-0FFFH. Jika alamat-alamat program lebih tinggi daripada 0FFFH, yang melebihi kapasitas ROM*Fash memory internal* menyebabkan AT89S51 secara otomatis mengambil *Code Byte* dari program *memory external. Code Byte* juga dapat diambil hanya dari *external memory* dengan alamat 0000-FFFFH dengan cara menghubungkan Pin EA ke *Ground.*




**Gambar 2-16. Organisasi RAM *Internal*[9]**

### 2.5.6. SFR (*Special Function Register*)

Register Fungsi Khusus (*Special Function Register*) terletak pada 128 byte bagian atas memori data internal dan berisi register-register untuk pelayanan latch port, timer, program status words, control peripheral, dan sebagainya. Alamat register fungsi khusus ditunjukkan pada tabel 2.16

**Tabel 2-16 Special Function Register**

| Simbol | Nama Register | Alamat |
|---|---|---|
| ACC | Accumulator | $E0_H$ |
| B | Register B | $F0_H$ |
| PSW | Program Statut Word | $D0_H$ |
| SP | Stack Pointer | $81_H$ |
| DPTR | Data Pointer 2 Byte | |
| DPL | Bit rendah | $82_H$ |
| DPH | Bit Tinggi | $83_H$ |
| P0 | Port 0 | $80_H$ |
| P1 | Port 1 | $90_H$ |
| P2 | Port 2 | $A0_H$ |
| P3 | Port 3 | $B0_H$ |
| IP | Interupt Periority Control | $D8_H$ |
| IE | Interupt Enable Control | $A8_H$ |
| TMOD | Timer/Counter Mode Control | $89_H$ |
| TCON | Timer/Counter Control | $88_H$ |
| TH0 | Timer/Counter 0 High Control | $8C_H$ |
| TL0 | Timer/Counter 0 Low Control | $8A_H$ |
| TH1 | Timer/Counter 1 High Control | $8D_H$ |
| TL1 | Timer/Counter 1 Low Control | $8B_H$ |
| SCON | Serial Control | $98_H$ |
| SBUF | Serial Data Buffer | $99_H$ |
| PCON | Power Control | $87_H$ |

Beberapa macam register fungsi khusus yang sering digunakan adalah sebagai berikut ini :

- *Accumulator* (ACC) merupakan register untuk penambahan dan pengurangan. Perintah *mnemonic* untuk mengakses akumulator disederhanakan sebagai A.

- *Register B* merupakan register khusus yang berfungsi melayani operasi perkalian dan pembagian.
- *Stack Pointer* (SP) merupakan register 8 bit yang dapat diletakkan di alamat manapun pada RAM internal.
- *2 Data Pointer* (DPTR) terdiri dari dua register, yaitu untuk byte tinggi (*Data Pointer High*, DPH) dan byte rendah (*Data Pointer Low*, DPL) yang berfungsi untuk mengunci alamat 16 bit.
- *Port 0* sampai *Port 3* merupakan register yang berfungsi untuk membaca dan mengeluarkan data pada port 0, 1, 2, 3. Masing-masing register ini dapat dialamati per-byte maupun per-bit.
- *Control Register* terdiri dari register yang mempunyai fungsi kontrol. Untuk mengontrol sistem interupsi, terdapat dua register khusus, yaitu register IP (*Interupt Priority*) dan register IE (*Interupt Enable*). Untuk mengontrol pelayanan timer/counter terdapat register khusus, yaitu register TCON (*timer/counter control*) serta pelayanan port serial menggunakan register SCON (*Serial Port Control*).

### 2.5.7. Sistem Interupsi

Mikrokontroller AT89S51 mempunyai 5 buah sumber interupsi yang dapat membangkitkan permintaan interupsi, yaitu INT0, INT1, T1, T2 dan Port Serial. Saat terjadinya interupsi mikrokontroller secara otomatis akan menuju ke subrutin pada alamat tersebut. Setelah interupsi selesai dikerjakan, mikrokontroller akan mengerjakan program semula. Tiap-tiap sumber interupsi dapat *enable* atau *disable* secara software.

Tingkat prioritas semua sumber *interupt* dapat diprogram sendiri-sendiri dengan *set* atau *clear* bit pada (*Interupt Priorit*). Jika dua permintaan interupsi dengan tingkat prioritas yang berbeda diterima secara bersamaan, permintaan interupsi dengan prioritas tertinggi yang akan dilayani. Jika permintaan interupsi dengan prioritas yang sama diterima bersamaan, akan dilakukan polling untuk menentukan mana yang akan dilayani.

**2.4. LCD (Liquid Crystal Display)**

LCD (*Liquid Crystal Display*) adalah papan informasi / tampilan atau biasa disebut Display. Dengan konsumsi daya yang rendah yang disusun dari dot matrik dan dikontrol oleh internal ROM / RAM generator karakter dan RAM data display.

Semua fungsi display dikontrol dengan instruksi dan LCD dapat dengan mudah diinterfacekan dengan MCU / MPU.

Karakteristik dari LCD dot matrik yang digunakan adalah sebagai berikut :

- 16 x 2 karakter dengan 5 x 7 dot matrik + kursor.
- ROM generator karakter dengan 192 tipe karakter.
- RAM generator karakter – karakter dengan 8 tipe karakter (untuk program write).
- 80 x 8 bit RAM data display.
- Dapat di interfacekan dengan kemungkinan MPU 4 bit atau 8 bit.
- RAM data atau RAM generator karakter yang dapat dibaca oleh MPU.
- + 5 Volt single power supplay.
- Power On Reset.
- Rangkaian temperature operasi 0°c sampai dengan 50°c.

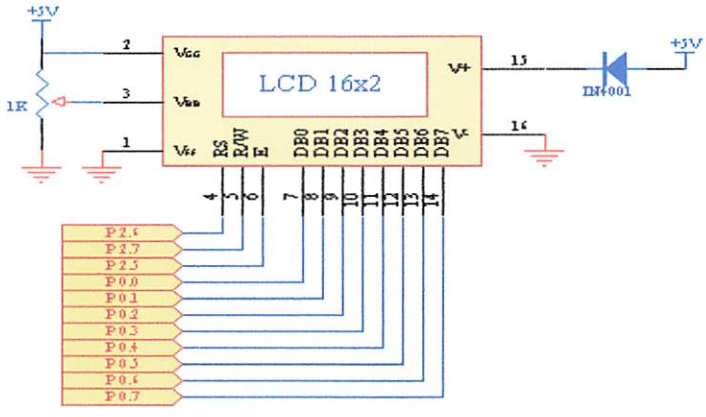
- Beberapa fungsi instruksi yaitu display clear, cursor home, display ON/OFF, cursor ON/OFF, display blink, cursor shift, dan display shift.

Display LCD mempunyai 16 terminal yang masing masing mempunyai fungsi sebagai berikut :

**Tabel 2-17**
**Display LCD**

| NO | SYMBOL | LEVEL | FUNCTION | |
|---|---|---|---|---|
| 1 | Vss | - | | 0 V (GND) |
| 2 | Vcc | - | | 5 V ±10% |
| 3 | Vee | - | | For LCD Drive |
| 4 | RC | H/L | H : Data Input | L : Instruction Input |
| 5 | R/W | H/L | H : Read | L : Write |
| 6 | E | H | Enable signal | |
| 7 | DB0 | H/L | Data bus | |
| 8 | DB1 | H/L | | |
| 9 | DB2 | H/L | | |
| 10 | DB3 | H/L | | |
| 11 | DB4 | H/L | | |
| 12 | DB5 | H/L | | |
| 13 | DB6 | H/L | | |
| 14 | DB7 | H/L | | |
| 15 | V+BL | - | Back light Supplay | 4 – 4.2 V<br>50 – 200 mA |
| 16 | V-BL | - | | 0 V (GND) |

LCD ini dapat menampilkan karakter yang ada pada ROM generator karakter, yang sudah berisi 190 jenis karakter, dengan cara memberikan kode generator. Untuk tiap – tiap karakter yang diinginkan pada bus data dan dengan menggunakan sinyal kontrol E, RS, R/W, dapat diatur berbagai program yang diberikan pada LCD dot matrik.




**Gambar 2-18. Rangkaian LCD 16 x 2**

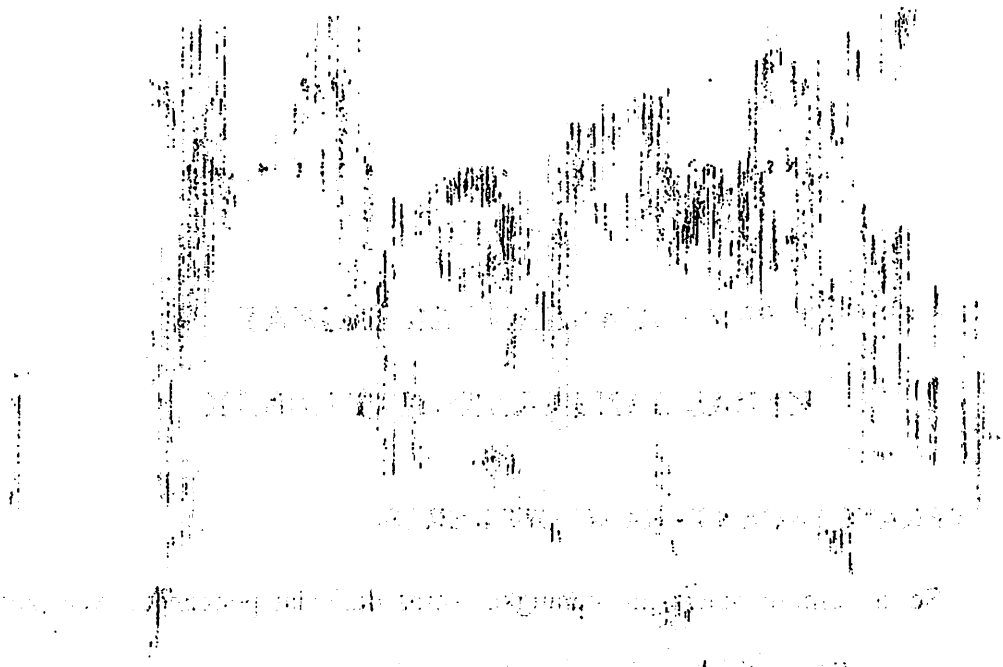
# BAB III

# PERANCANGAN PERANGKAT KERAS DAN PERANGKAT LUNAK

## 3.1 PERANCANGAN PERANGKAT KERAS

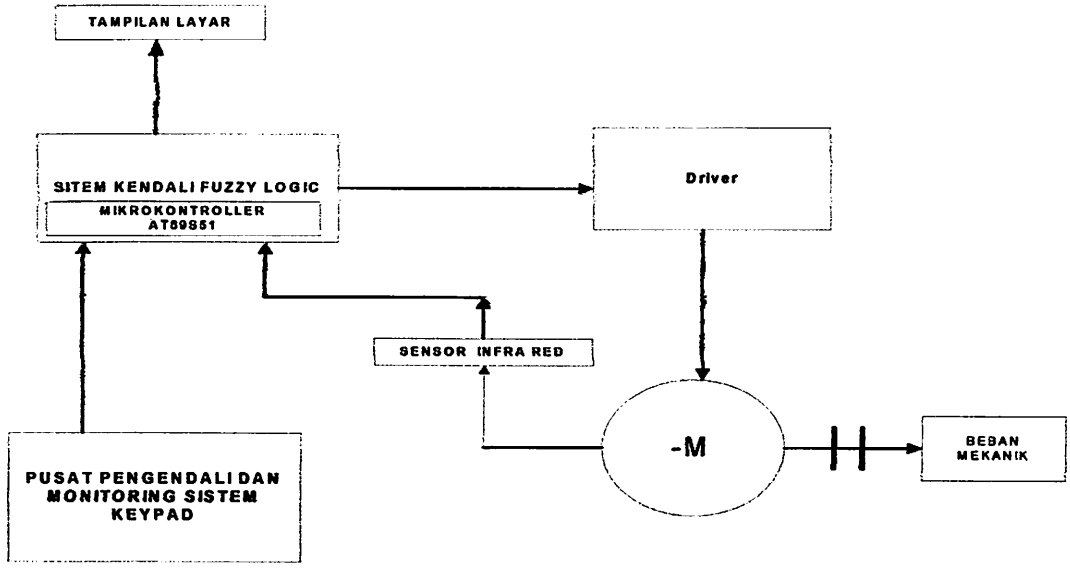
Secara umum rancangan perangkat keras dari alat pengendali kecepatan putaran motor DC terdiri dari 5 bagian utama, yaitu:

1) Motor DC magnit permanen
2) Catu Daya : sebagai sumber tenaga listrik bagi seluruh modu.
3) Mikrokontroller Utama: berfungsi sebagai interface antara keypad dengan inverter dan sebagai pengendali pensaklaran.
4) Tachometer : berfungsi sebagai umpan balik dari sistem kendali.
5) Pusat Pengendali (sistem kendali logika fuzzy) dan Monitoring Sistem : berfungsi untuk memberikan set-point kecepatan sekaligus memantau response sistem. Pusat pengendali dan monitoring sistem adalah sebuah keypad



### 3.1.1 Diagram Blok

Secara garis besar, rancangan keseluruhan dari sistem ini dapat digambarkan seperti gambar berikut ini.




**Gambar 3 – 1**
**Diagram Blok Keseluruhan Sistem**

Keterangan :

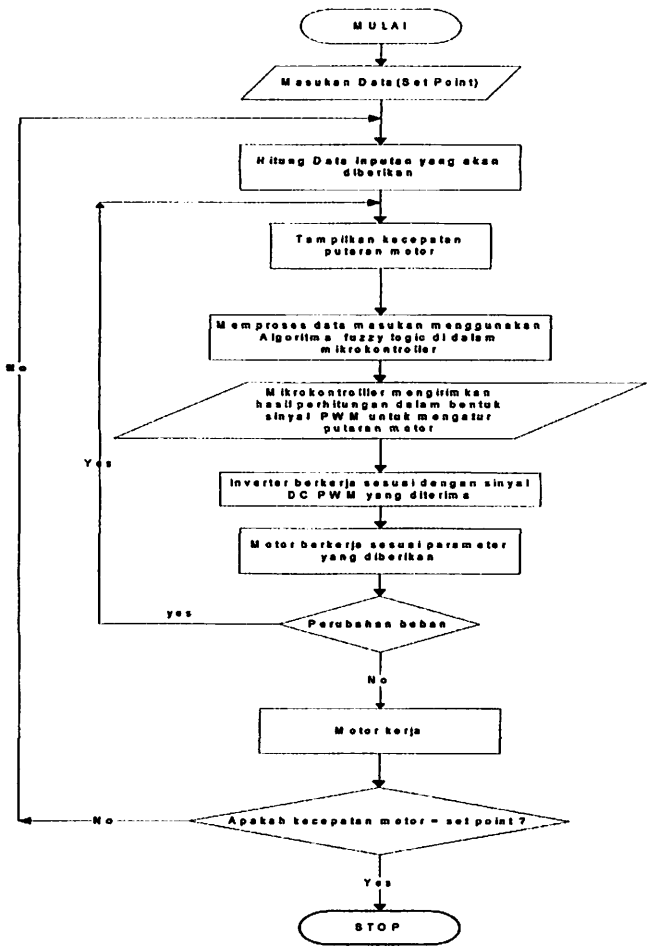
a. Catu Daya :
   - o 5 V : Mikrokontroller
   - o 12 V : Inverter

b. Mikrokontroller Utama : ATMEL AT89S8252 (Interface)

c. Tachometer : Sensor infra red

d. Keypad

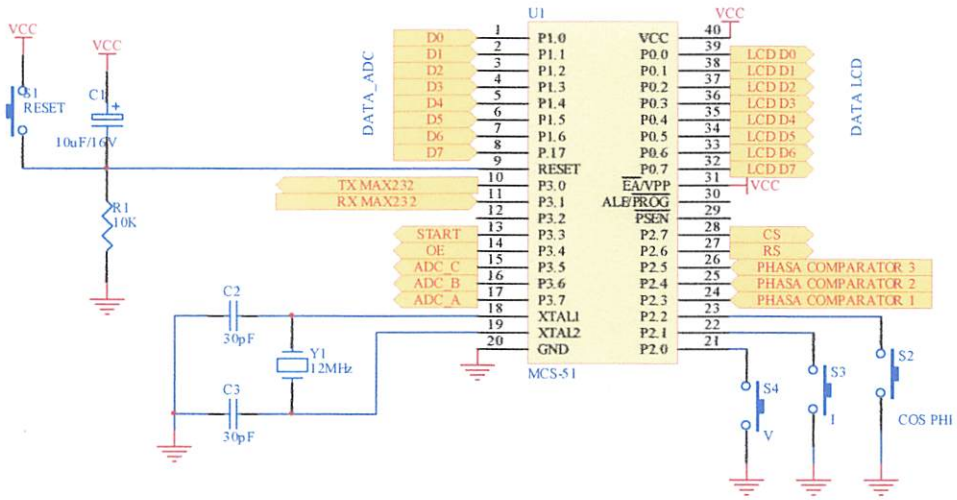
### 3.1.2 floechart




**gambar 3-2**
**Diagram alir cara kerja pengatur keccepatan motor**

### 3.2 Perancangan Minimum system Mikrokontroller AT89S51

Mikrokontroller disini menjadi komponen utama dalam perancangan alat ukur energi listrik ini,komponen ini mempunyai multi fungsi dalam berbagai bentuk aplikasi,termasuk dalam perancangan alat ini kami menggunakan mikrokontroller AT89S51yang termasuk dalam keluarga MCS-51.Mikrokontroller ini mempunyai banyak kelebihan seperti yang telah tertulis pada Bab II dalam penulisan skripsi ini.




**Gambar 3-3. Konfigurasi Kaki Mikrokontroller AT89S51**

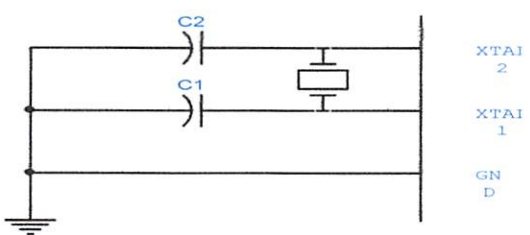
### 3.2.1 Rangkaian Clock Internal

Mikrokontroler AT89S51 ini memiliki *internal clock* generator yang berfungsi sebagai sumber *clock*, tetapi masih diperlukan rangkaian tambahan untuk membangkitkan *clock* tersebut. Rangkaian ini terdiri dari 2 buah kapasitor dan sebuah kristal dengan ketentuan:

C1 dan C2 = 20 pF – 40 pF untuk kristal

= 30 pF – 50 pF untuk keramik resonator

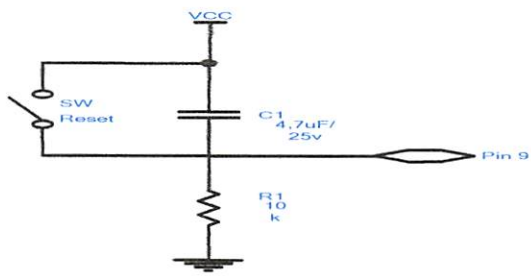
Dalam perencanaan rangkain mikrokontroller ini digunakan kapasitor sebesar 30 pF sampai dengan 10 pF.




**Gambar 3-4. Rangkaian Clock**

### 3.2.2 Rangkaian *Reset*

Rangkaian *Reset* digunakan untuk mereset atau mengembalikan ke keadaan awal dari mikrokontroller AT89S51. Rangkaian ini dipergunakan untuk mereset mikrokontroller pada keadaan pertama kali saat power diaktifkan atau disebut Power On Reset. Untuk menjalankan reset maka pin reset (pin no.9) pada mikrokontroller diberi sinyal *high* (1), rangkaian reset ini dapat ditunjukkan pada gambar berikut:




**Gambar 3-5. Rangkaian Reset**

## 3.3 Perancangan Rangkaian LCD

*LCD Display Module M1632* buatan Seiko Instrument Inc. terdiri dari dua bagian, yang pertama merupakan panel LCD sebagai media penampil informasi dalam bentuk huruf/angka dua baris, masing-masing baris bisa menampung 16 huruf/angka.

Bagian kedua merupakan sebuah sistem yang dibentuk dengan mikrokontroler yang ditempelkan dibalik pada panel LCD, berfungsi mengatur tampilan informasi serta berfungsi mengatur komunikasi L1632 dengan mikrokontroler yang memakai tampilan LCD itu. Dengan demikian pemakaian M1632 menjadi sederhana, sistem lain yang M1632 cukup mengirimkan kode-kode ASCII dari informasi yang ditampilkan seperti layaknya memakai sebuah printer.

Untuk berhubungan dengan mikrokontroler pemakai, M1632 dilengkapi dengan 8 jalur data (**DB0..DB7**) yang dipakai untuk menyalurkan kode ASCII maupun perintah pengatur kerjanya M1632. Selain itu dilengkapi pula dengan **E**, **R/W** dan **RS** seperti layaknya komponen yang kompatibel dengan mikroprosesor.

Kombinasi lainya **E** dan **R/W** merupakan sinyal standar pada komponen buatan Motorola. Sebaliknya sinyal-sinyal dari MCS51 merupakan sinyal khas Intel dengan kombinasi sinyal **WR** dan **RD.**

**RS**, singkatan dari Register Select, dipakai untuk membedakan jenis data yang dikirim ke M1632, kalau **RS=0** data yang dikirim adalah perintah untuk mengatur kerja M1632, sebaliknya kalau **RS=1** data yang dikirim adalah kode ASCII yang ditampilkan.

Demikian pula saat pengambilan data, saat **RS=0** data yang diambil dari M1632 merupakan data status yang mewakili aktivitas M1632, dan saat **RS=1** maka data yang diambil merupakan kode ASCII dari data yang ditampilkan.

Proses mengirim/mengambil data ke/dari M1632 bisa dijabarkan sebagai berikut :

1. **RS** harus dipersiapkan dulu, untuk menentukan jenis data seperti yang telah dibicarakan di atas.
2. **R/W** di-nol-kan untuk menandakan akan diadakan pengiriman data ke M1632. Data yang akan dikirim disiapkan di **DB0..DB7**, sesaat kemudian sinyal **E** di-satu-kan dan di-nol-kan kembali. Sinyal **E** merupakan sinyal sinkronisasi, saat **E** berubah dari 1 menjadi 0 data di **DB0 .. DB7** diterima oleh M1632.
3. Untuk mengambil data dari M1632 sinyal **R/W** di-satu-kan, menyusul sinyal **E** di-satu-kan. Pada saatu **E** menjadi 1, M1632

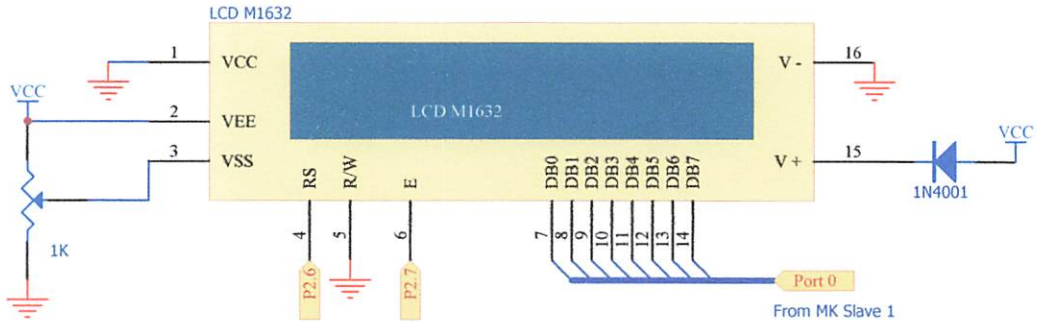
akan meletakkan datanya di **DB0 .. DB7**, data ini harus diambil sebelum sinyal **E** di-nol-kan kembali.

M1632 mempunyai seperangkat perintah untuk mengatur tata kerjanya, perangkat perintah tersebut meliputi perintah untuk menghapus tampilan, meletakkan kembali cursor pada baris huruf pertama baris pertama, menghidup/matikan tampilan dan lan sebagainya, semua itu dibahas secara terperinci dalam lembar data M1632.

Setelah diberi sumber daya, ada beberapa langkah persiapan yang harus dikerjakan dulu agar M1632 bisa dipakai, langkah-langkah tersebut antara lain adalah:

1. Tunggu dulu selama 15 mili-detik atau lebih.
2. Kirimkan perintah 30h, artinya trasfer data antar M1632 dan mikrokontroler dilakukan dengan mode 8 bit
3. Tunggu selama 4.1 mili-detik
4. Kirimkan sekali lagi perintah 30h
5. Tunggu lagi selama 100 mikro-detik

Setelah langkah-langkah tersebut di atas M1632 barulah bisa menerima data dan menampilkannya dengan baik. Pada awalnya tampilan akan nampak kacau, dengan demikian perlu segera dikirim perindah menghapus tampilan dan lain sebagainya, sesuai dengan petunjuk yang ada di lembar data.




**Gambar 3-6.**
**Perancangan Rangkaian *Liquid Crystal Display* (LCD)**
***Sumber :*** *Perancangan.*

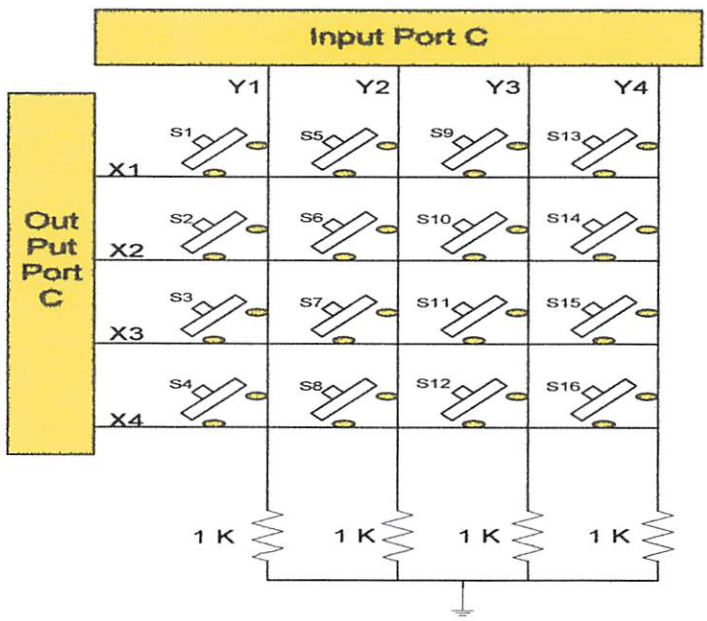
Untuk tampilan dipergunakan LCD Dot Matrik 2 x 16 karakter. Sinyal-sinyal yang diperlukan oleh LCD adalah RS dan Enable, sinyal RS dan Enable dipergunakan sebagai input yang outputnya dipakai untuk mengaktifkan LCD. LCD akan aktif apabila mikrokontroller memberikan instruksi tulis pada LCD. Saat kondisi RS don't care dan Enable 0 maka LCD tetap pada kondisi semula, pengiriman data ke LCD dilakukan saat RS berlogika 0 dan enable berlogika 1. Instruksi dikirim pada LCD bila keadaan RS 1 dan Enable 1. Pin LCD ini untuk data terkoneksi pada *Port 0* mikrokontoler Slave 1. Kemudian untuk RS dihubungkan pada *Port* 2.6, tulis/baca (*Read/Write)* diberikan logika *low* karena disini LCD bersifat menulis data, dan yang terakhir *Enable* (E) dikendalikan dengan *Port 2.7*.Gambar rangkaian LCD ditunjukkan pada gambar 3-6.

### 3.4. Papan Tombol (*Keypad*)

Papan tombol ini digunakan untuk memasukkan data referensi dan mengubah data bila diinginkan. Untuk menterjemahkan informasi yang diterima dari papan tombol, maka *keypad* dihubungkan dengan *port* C PPI 8255.

Papan tombol tersebut mempunyai matrik 4 baris dan 3 kolom. Deretan baris dan kolom dari papan tombol dihubungkan dengan *port* C PPI 8255 yang difungsikan sebagai masukan dan keluaran. Deretan kolom dihubungkan dengan *ground* (berlogika 0) dan *port* C (PC4-PC7) yang difungsikan sebagai *input*

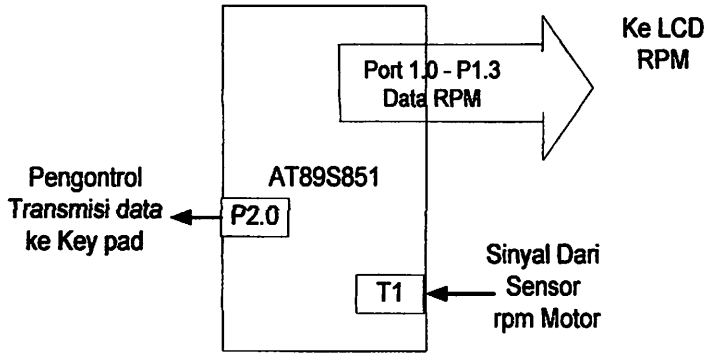
mikrokontroler. Sedangkan deretan baris dihubungkan ke *port* C (PC0-PC3) yang telah diberi data 0001 dan secara kontinyu data tersebut bergeser satu bit ke kiri. Pergeseran data satu bit ini dimaksudkan untuk menentukan posisi tombol yang ditekan dalam satu kolom. *Port* ini difungsikan sebagai *output* dari mikrokontroler. Dengan demikian kalau tombol tidak ditekan maka masukan *port* C (PC4-PC7) di pin yang terhubung tombol tersebut berlogika 0 dan bila tombol ditekan akan berlogika 1. Rangkaian papan tombol tersebut dapat dilihat dalam Gambar 3-7.




**Gambar 3-7**
**Rangkaian Keypad**

### 3.5 Rancangan Modul Tachometer

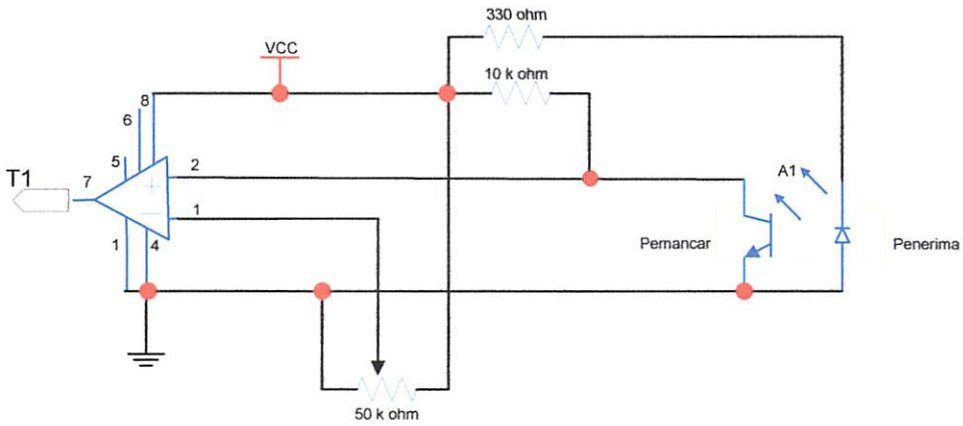
Tugas utama modul ini adalah mengirimkan data kecepatan putaran motor. Selain itu tachometer ini juga akan mengirimkan data hasil pengukuran tersebut kepada LCD untuk dapat melihat kinerja dari sistem kendali.




**Gambar 3.10**
**Modul Tachometer**

Tachometer akan mengukur kecepatan putaran motor dengan cara menghitung waktu yang dibutuhkan oleh motor untuk menyelesaikan satu putaran. Pendeteksian tersebut dilakukan dengan menggunakan sebuah sensor Optocoupler dan diletakkan pada sebuah piringan yang diberi lobang.Optocouler akan Off apabila bila cahaya lednya terhalang dan akan ON apabila cahaya led mengenai phototransistor saat piringan yang berlobang.Selanjutnya sinyal keluaran sensor diproses oleh mikrokontroler pada modul tachometer untuk mendapatkan hasil kecepatan putaran motor dalam rotasi per menit .

Optocoupler merupakan komponen penghubung (coupling) yang bekerja berdasarkan "picu"cahaya,yang bekerja bila ada cahaya sebagai tegangan bias.Optocoupler terdiri dari 2 buah bagian,yaitu bagian transmitter (pemancar) dan bagian receiver (penerima).Bagian transmitter dari optocoupler terdiri led infrarad untuk memperoleh keluaran yang baik dari sinyal atau cahaya yang tampak disbanding menggunakan led biasa.Sedangkan,bagian receiver terdiri dari phototransistor yang akan bekerja bila mendapat cahaya atau sinar ,cahaya menggatikan tegangan bias yang dibutuhkan transistor yang berasal dari led.




**Gambar 3 – 11**
**Rangkaian Sensor Menggunakan Infrared**

Ket : * A1 merupakan komponen Infrared sebagai sensor putaran.

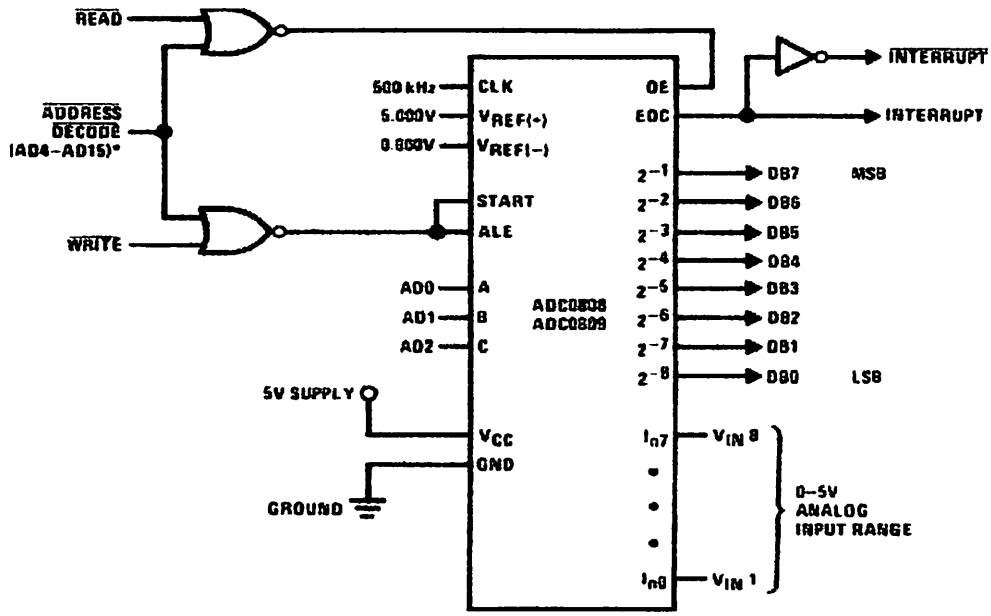
* LM 311 merupakan komponen Op-Ampdengan rangkaian Diffrensial yang terdiri dari dua terminal input dan satu terminal output.Terminal input terdiri dari terminal inverting (yang bertanda -) dan terminal non-inverting (yang bertanda +) berfungsi sebagai rangkaian pengurang antara dua terminal inputan karena dari sifat terminal inverting yang sifatnya membalik polaritas dari inputan.Fungsi LM311 ini adalah sebagai komparator,antara tegangan referensi (5V) dengan tegangan optocoupler (terbuka=0,4 V dan tertutup =3,2 V ). Untuk menghitung diameter lingkaran yaitu

**Rumus** 2 . r

2 x 5 =10 cm

### 3.6. Perancangan ADC dan DAC

Pada rangkaian ADC digunakan ADC 0809. Dimana rangkaian ADC ini berfungsi untuk mengkonversi sinyal analog menjadi data digital.

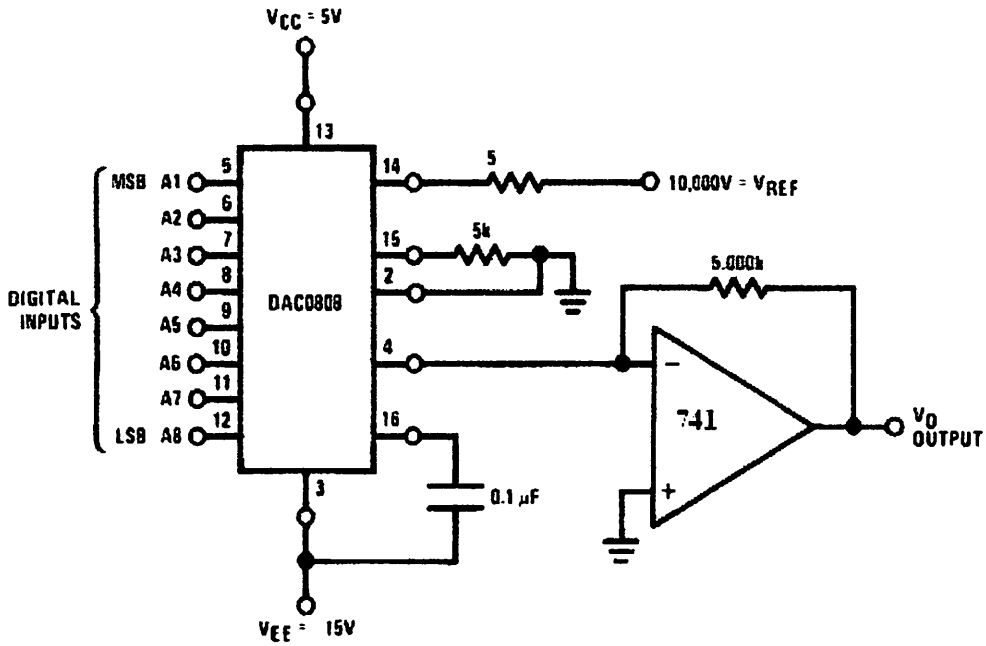



**Gambar 3-12 Rangkaian ADC 0809**

Input 0 (pin 26) merupakan masukan yang berupa sinyalanalog dari sensor suhu. Pin $2^{-1}$ sampai pin $2^{-8}$ merupakan output digital yang digunakan untuk mengirim data ke minimum system DT-51 petrafuz.

Kontrol ADC PORT C Pin no. 1 dari minimum system DT-51 Petrafuzz masuk ke control (ABC ADC 0809) pin yang A. EOC (End of Convertion) pin 7 akan high jika proses konversi telah selesai.

Sedangkan pada rangkaian DAC digunakan DAC 0808, yang berfungsi untuk mengkonversi data digital menjadi sinyal analog.




**Gambar 3-14 Rangkaian DAC 0808**

Pada pin 5 sampai pin 7 merupakan tempat inputan digital dari DT-51 petrafuz. Pada pin 6 (Vo) dari LM 741 merupakan tegangan output dari hasil konversi DAC 0808 yang akan digunakan sebagai inputan pada rangkaian pengubah frekwensi ke frekuensi. Dimana rangkaian pengubah frekwensi ke tegangan membutuhkan inputan analog antara 0-3 volt yang merupakan titik kerjanya.

## 3.7 Driver Motor

Driver motor pada alat yang dirancang berupa motor DC dengan power supply 12V. Besarnya kecepatan putar motor diatur melalui PC dimana prosesnya sebagai berikut melalui port pada mikro akan mengeluarkan perintah berupa data biner 8 bit atau dalam decimal 0 sampai 255. Perintah ini diubah oleh ADDA menjadi perubahan tegangan analog.

Dan dari perubahan tegangan ini akan diubah oleh trassistor TIP41 menjadi perubahan arus yang dilalui rangkaian darlington.

Dimana perubahan arus yang melalui basis ini akan mempengaruhi besarnya arus yang masuk dari motor DC ke kolektor dimana besarnya arus kolektor (IC1) sama dengan arus basis (IB1) dikalikan dengan hf1, dimana hfe adalah penguat trassistor.

Perubahan arus kolektor inilah yang menyebabkan kecepatan putar motor DC dapat berubah dimana kecepatan berbanding lurus dengan arus kolektor,dimana besarnya arus basis tergantung pada besarnya resistor dengan rumus:

$$R_{B'E'} = 2hfe\ VT / IC' \qquad (3.1)$$

Besarnya arus yang melalui basis akan mempengaruhi besarnya arus yang masuk dari motor DC ke kaki kolektor dimana besarnya dapat diketahui dengan rumus sebagai berikut:

$$IC_1 = IB_1.hfe_1$$

$$IC_2 = hfe_2 . IB_2 = hfe_2 . IC_1 = hfe_2 . hfe_1 . IB_1$$

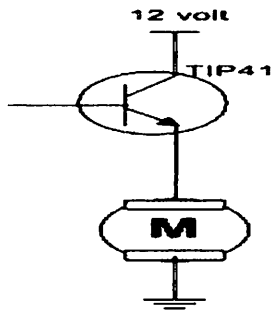
$$IC' = hfe_2 . hfe_1 . IB' \qquad (3.2)$$

Keterangan

Ib = Arus basis

Ic = Arus kolektor

Hfe = Faktor penguat transistor




Gambar 3.15 Rangkaian Untuk Menjalankan Motor

### 3.8 Rancangan Perangkat Lunak Sistem

Desain perangkat lunak keseluruhan sistem dapat dibagi menjadi 3 bagian utama, yaitu:

1. Perangkat lunak monitoring sistem berfungsi sebagai pemberi perintah berupa data Set Point yang harus dikerjakan oleh motor beserta pengendalinya, fungsi lainnya sebagai pencatat dan menggrafikan tanggapan sistem.
2. Perangkat lunak pendeteksi sinyal dari key pad dengan menggunakan listing program. Perangkat lunak ini ditempatkan pada mikrokontroller sebagai Interface ke inverter.
3. Perangkat lunak pendeteksi kecepatan putaran motor (tachometer), digunakan sebagai sarana umpan balik pengendali sehingga sistem kendali alat ini menjadi berjenis *closed loop*.

### 3.8.1 Rancangan Perangkat Lunak Pusat Pengendali Kecepatan Motor DC

#### A. Rancangan Sistem Logika Fuzzy

Perangkat lunak ini mengirimkan sinyal pada mikrokontroller dengan tugas utamanya untuk mengendalikan kecepatan putaran motor induksi agar sesuai dengan set point yang diberikan.

Fungsi keanggota yang direncanakan untuk imputan err dan Δ Err (keasalah dan perubahan) terdiri 5 label SN(sangat negatif),N(negatif),0,P(positif),SP(sangat positif) dimana:

Err(n) = SP(n) – PV(n).....(1)

ΔErr = Err(n) – Err(n-1).....(2)

Sedangkan fungsi keanggotaan Δoutput yang berupa garis vertikal (singleton)mempnyai label yang sama dengan Err dan Δ Err. Δ output FLC menuju driver motor

Untuk merepresentasikan fungsi keanggotaan tersebut diperlukan 4 nilai, yaitu: titik 1, titik 2, kemiringan 1. dan kemiringan 2. Hal ini dikarenakan derajat keanggotaan dari titik 1 dan titik 2 berturut-turut selalu $\mu = 0$ dan $\mu = 1$ (MU_MAX). Selain dari nilai basis titik 1 dan titik 2, nilai lainnya adalah kemiringan dari sisi fungsi keanggotaan yaitu kemiringan 1 dan kemiringan 2. Untuk itu fungsi keanggotaan (MBF) dikodekan pada persamaan 1

**MBF_DEFINITION : (Titik_1, Kemiringan_1, Titik_2,Kemiringan_2)**

**Persamaan 3 - 1**
**Definisi Fungsi keanggotaan**

Untuk masing-masing masukan akan digolongkan terlebih dahulu kepada daerah 1, daerah 2, atau daerah 3. Selanjutnya ditentukan nilai fuzzifikasinya menurut Persamaan 3-2

1. **Daerah 1 : Keanggotaan = 0**
2. **Daerah 2 : Keanggotaan = min{1,(Masukan-Titik_1)*Kemiringan_1}**
3. **Daerah 3 : Keanggotaan =max{0,1-(masukan-Titik_2)*kemiringan_2}**

**Persamaan 3 - 2.**

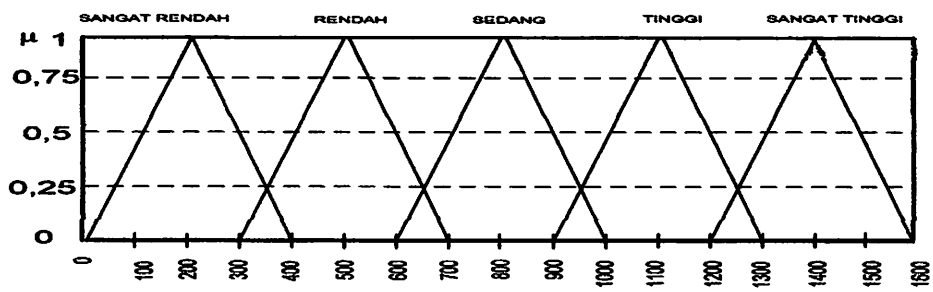
**Perhitungan Derajat Keanggotaan Untuk Masing-Masing Daerah**

- Variabel Linguistik:

T (kecepatan) = { sangat rendah, rendah, sedang, tinggi, sangat tinggi }

- Crisp (X) :
  - Kecepatan putaran sangat rendah : 0 – 400 rpm
  - Kecepatan putaran rendah : 300 – 700 rpm
  - Kecepatan putaran sedang : 600 – 1000 rpm
  - Kecepatan putaran tinggi : 900 – 1300 rpm
  - Kecepatan putaran sangat tinggi : 1200 – 1600 rpm

erajat keanggotaan dan fungsi keanggotaan untuk data diatas dapat digambarkan dalam gambar 3-13




**GAMBAR 3-16**
**Himpunan Fuzzy Untuk Kecepatan**

Perhitungan derajat keanggotaan dapat dicari sebagai berikut ( untuk fungsi segitiga ) :

- $\mu_{(0)} = (0\text{-}0)/200 \quad = 0$ ( untuk himpunan rendah )
- $\mu_{(360)} = (400\text{-}360)/200 \quad = 0.2$ ( untuk himpunan sangat rendah )

  $= (360\text{-}300)/200 \quad = 0.3$ ( untuk himpunan rendah )

### 3.8.2 Proses Inferensi

Adapun untuk proses inferensi logika fuzzy digunakan jenis Mamdani, yaitu metode MAX-MIN. Hal ini dikarenakan proses yang sederhana, tidak memerlukan perhitungan yang rumit, hanya membandingkan nilai-nilai yang ada. Pada tahap fuzzy inferensi ini, Software membandingakan nilai hasil fuzifukasi dari setiap rule.Setiap proses inferensi akan mempengaruhi nilai derajat fungsi keanggotaan keluaran. Hasil tersebut kemudian akan diproses pada tahap defuzifikasi.

### 3.8.3. Proses Defuzifikasi

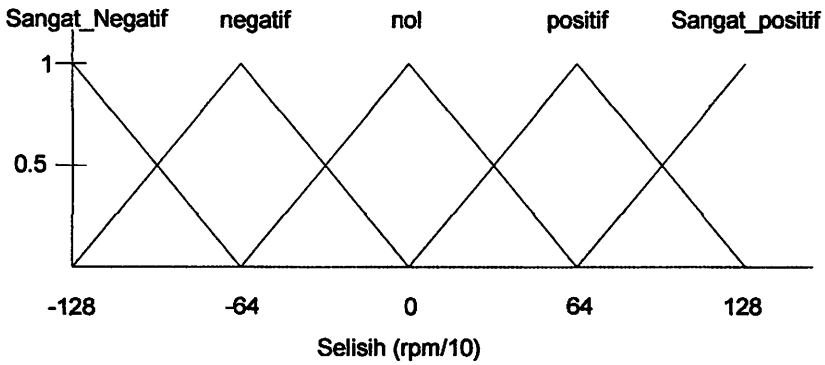
Metode defuzifikasi yang digunakan oleh penulis adalah metode centroid atau center of gravity yang telah disederhanakan, yaitu mengambil titik berat dari gabungan fungsi keanggotaan keluaran. Metode ini didasarkan pada Persamaan 3.

$$Keluaran = \frac{(a_1 \text{ x } b_1 + a_2 \text{ x } b_2 + \ldots + a_n \text{ x } b_n)}{b_1 + b_2 + b_n}$$

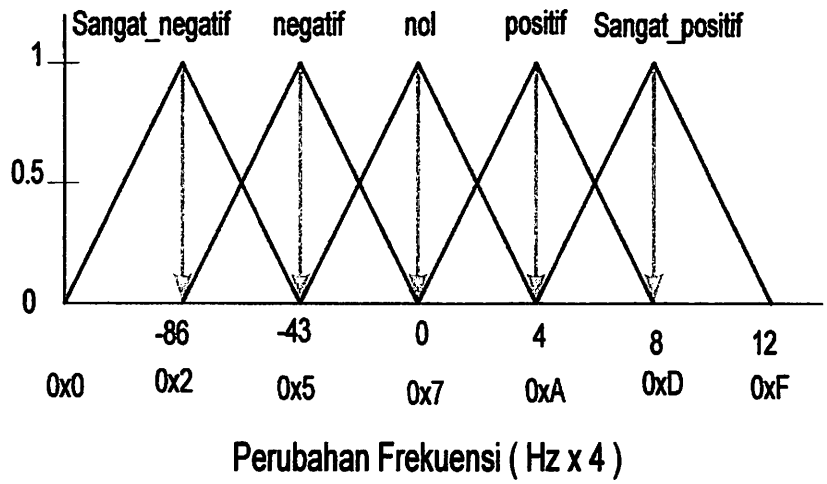
Dengan:
a: titik tengah fungsi keanggotaan keluaran
b: dearat fungsi kenggotaan keluaran
n: jumlah fungsi keanggotaan

**Persamaan 3 -3.**
**Perhitungan Dengan Metode Center Of Gravity.**

Dengan menggunakan metode intuisi dan trial and error maka dibuat fungsi keanggotaan untuk Selisih dan Frekuensi serta aturan logika fuzzy (fuzzy rules) sebagai berikut.




**Gambar 3- 17**
**Fungsi Keanggotaan Selisih**




**Gambar 3-18**
**Fungsi Keanggotaan Frekuensi**

### 3.8.4 Mekanisme Infersi

Untuk mengkombinasikan aturan-aturan kontrol yang digunakan kata hubung yang secara umum dinyatakan dengan:

Rule 1

"if input adalah 0 (0,25) AND output adalah 100 (0,5) Then kecepatan adalah sangat rendah

Rule 2

"if input adalah 100 (0,25) AND output adalah 200 (0,5) Then kecepatan adalah sangat rendah

Rule 3

"if input adalah 100 (0,25) AND output adalah 300 (0,5) Then kecepatan adalah sangat rendah

Rule 4

"if input adalah 100 (0,25) AND output adalah 400 (0,25) Then kecepatan adalah sangat rendah

Rule 5

"if input adalah 200 (0,25) AND output adalah 400 (0,25) Then kecepatan adalah sangat rendah

Rule 6

"if input adalah 300 (0,25) AND output adalah 400 (0,25) Then kecepatan adalah rendah

Rule 7

"if input adalah 300 (0,5) AND output adalah 500 (0,5) Then kecepatan adalah rendah

Rule 8

"if input adalah 300 (0,25) AND output adalah 600 (0,25) Then kecepatan adalah rendah

Rule 9

"if input adalah 300 (0,25) AND output adalah 700 (0,25) Then kecepatan adalah rendah

Rule 10

"if input adalah 400 (0,25) AND output adalah 600 (0,25) Then kecepatan adalah rendah

Rule 11

"if input adalah 600 (0,25) AND output adalah 700 (0,5) Then kecepatan adalah sedang

Rule 12

"if input adalah 700 (0,25) AND output adalah 800 (0,5) Then kecepatan adalah sedang

Rule 13

"if input adalah 700 (0,25) AND output adalah 800 (0,5) Then kecepatan adalah sedang

Rule 14

"if input adalah 800 (0,5) AND output adalah 900 (02,5) Then kecepatan adalah sedang

Rule 15

"if input adalah 900 (0,25) AND output adalah 1000 (0,5) Then kecepatan adalah tinggi

Rule 16

"if input adalah 1000 (0,5) AND output adalah 1100 (0,25) Then kecepatan adalah sedang

Rule 17

"if input adalah 1100 (0,75) AND output adalah 1200 (0,5) Then kecepatan adalah tinggi

Rule 18

"if input adalah 1200 (0,25) AND output adalah 1300 (0,5) Then kecepatan adalah tinggi

Rule 19

"if input adalah 1100 (0,75) AND output adalah 1300 (0,25) Then kecepatan adalah tinggi

Rule 20

"if input adalah 1000 (0,25) AND output adalah 1200 (0,25) Then kecepatan adalah tinggi

Rule 21

"if input adalah 1200 (0,25) AND output adalah 1400 (0,5) Then kecepatan adalah sangat tinggi

Rule 22

"if input adalah 1200 (0,25) AND output adalah 1500 (0,5) Then kecepatan adalah sangat tinggi

Rule 23

"if input adalah 1300 (0,25) AND output adalah 1400 (0,5) Then kecepatan adalah sangat tinggi

Rule 24

"if input adalah 1300 (0,25) AND output adalah 1500 (0,75) Then kecepatan adalah sangat tinggi

Rule 25

"if input adalah 1300 (0,25) AND output adalah 1600 (0,25) Then kecepatan adalah sangat tinggi

Tabel 3.1 Aturan Inferensi Logika Fuzzy

| | | Δe | | | | |
|---|---|---|---|---|---|---|
| | | NB | NK | SN | PK | PB |
| error | NB | NB | NB | NB | NS | SN |
| | NK | NB | NS | NK | SN | PS |
| | SN | NB | NK | SN | NK | PB |
| | PK | NB | SN | PK | PS | PB |
| | PB | SN | PS | PB | PB | PB |

# BAB IV

# PENGUJIAN ALAT DAN ANALISA DATA

## A. Alat yang dipergunakan :

a) Keypad

b) Rangkaian Kontrol : Mikrokontroller AT89S51

c) Rangkaian sensor : Infra red

d) Motor DC Magnit permanen, dengan data-data sebagai berikut:

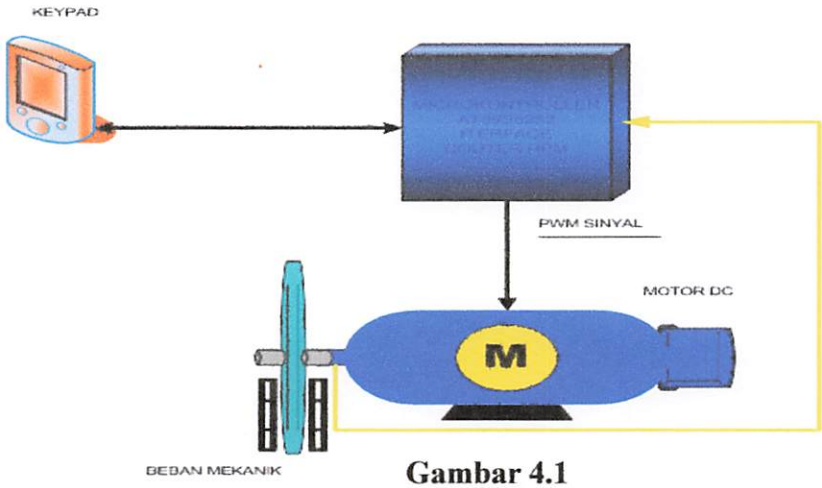
a. Tegangan : 12 Volt.

b. Arus : 0,9 Ampere.

c. Putaran : 1500 rpm

g) Digital Tachometer, DEUMO MP.

h) Alat Ukur (Voltmeter ).




**Gambar 4.1**
**Skema Rangkaian Pengujian**

## 4.1. Pengujian Alat Pengendali Kecepatan Berbeban

**Pelaksanaan Percobaan**

1. Menghubungkan KEYPAD dengan rangkaian kontrol (Mikrokontroler),
2. Motor DC yang akan diuji

3. Memasang sensor kecepatan (Infrared & Mikrokontroller) sebagai feedback ke KEYPAD.
4. Mengatur Tegangan,
5. Memantau kecepatan motor,besar tegangan, arus

**Tujuan Percobaan**

Ingin mengetahui daya inputan motor,arus,tegangan , dan kecepatan motor

**Rumus**

**Perancangan – pengujian / pengujiah**

**Data hasil pengujian**

Pengujian alat dengan acuan sebagai referensi set point yang akan

**Tabel 4-1**
**Hasil Pengujian**

| No | Set point (rpm) | kecepatan | | Error (%) |
|---|---|---|---|---|
| | | Min (rpm) | Max (rpm ) | |
| 1 | 100 | 60 | 120 | 0,6 |
| 2 | 500 | 480 | 500 | 0,12 |
| 3 | 1000 | 990 | 1060 | 0,06 |
| 4 | 1400 | 1380 | 1440 | 0,042 |
| 5 | 1500 | 1440 | 1500 | 0,04 |

**Perhitungan error RPM**

$$E = \frac{60}{100} \times 100\% = 0.6$$

$$E = \frac{60}{500} \times 100\% = 0.12$$

$$E = \frac{60}{1000} \times 100\% = 0,06$$

$$E = \frac{60}{1400} \times 100\% = 0,042$$

$$E = \frac{60}{1500} \times 100\% = 0,04$$

Hasil pengujian menunjukan bahwa ketika kecepatan diberi setting 100 rpm, kecepatan maximal mencapai 120 dan kecepatan minimum 60 rpm dalam pembebanan perlu dilakukan batasan beben yag diberikan ke motor, karena pada saat overload motor akan lebih lambat untuk mencapai kestabilan. Dari data diatas maka dapat kita simpulkan bahwa sangat perlu dilakukan pengaturan kecepatannya untuk kesetabilan putaran motor tersebut

## 1.2 pengujian optocopler(terbebani)

Tabel 4-2
Hasil Pengujian Optocopler(Terbebani)

| No | Set point RPM yang diinginkan | Beban | Hasil RPM |
|---|---|---|---|
| 1 | 500 | 0% | 500 |
| | | 50% | 300 |
| | | 75% | 180 |
| 2 | 1000 | 0% | 1000 |
| | | 50% | 840 |
| | | 75% | 420 |
| 3 | 1500 | 0% | 1500 |
| | | 50% | 300 |
| | | 75% | 180 |

**Rumus**

**Vton=Ton /Toff x VDC**

Pada1500rpmVDC= $\frac{10\ ms}{10ms}$ x 12 = 12 volt

Pada1400rpmVDC= $\frac{8,2ms}{10ms}$ x 12 = 9,89 volt

Pada1300rpmVDC= $\frac{7,8}{10}$ x 12 = 9,36 volt

Pada1200rpmVDC= $\frac{7,4}{10}$ x 12 = 8,88 volt

Pada1100rpmVDC= $\frac{7,2}{10}$ x 12 = 8,64 volt

Pada1000rpmVDC= $\frac{6,6}{10}$ x 12 = 792 volt

Pada900rpmVDC= $\frac{6,4}{10}$ x 12 = 7,68 volt

Pada800rpmVDC= $\frac{6,4}{10}$ x 12 = 7,44 volt

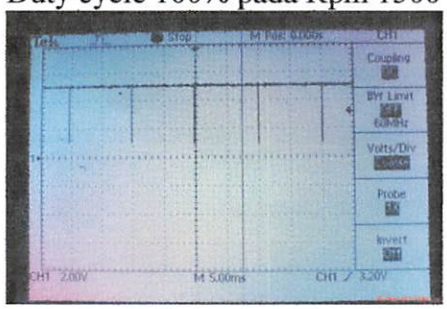
Pada700rpmVDC= = $\frac{5,4}{10}$ x 12 = 6,48 volt

**Tabel 4-3**
**Perhitungan dan pengujian VDC**

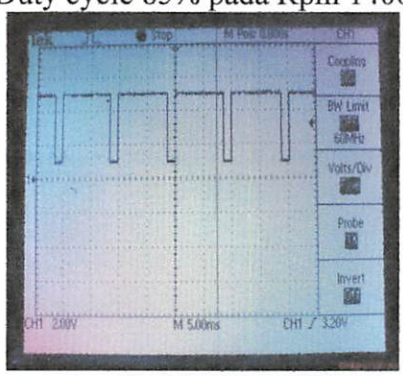
| NO | SETPOINT | TEGANGAN(v) | TEGANGAN(v) | DUTY CECLE % |
|---|---|---|---|---|
| 1 | 1500 | 12 | 11,6 | 100 |
| 2 | 1400 | 9,8 | 10,6 | 82 |
| 3 | 1300 | 9,36 | 10,1 | 78 |
| 4 | 1200 | 8,88 | 9,4 | 74 |
| 5 | 1100 | 8,64 | 8,7 | 72 |
| 6 | 1000 | 7,92 | 8,1 | 66 |
| 7 | 900 | 7,68 | 7,1 | 64 |
| 8 | 800 | 7,44 | 6,4 | 62 |
| 9 | 700 | 7,2 | 5,4 | 60 |
| 10 | 600 | 6,48 | 5,0 | 54 |

Hasil pengujian menunjukkan bahwa ketika kecepatan diberi setting 500 rpm, kecepatan stabil 500 rpm dan jika diberi beban maksimal 75% maka kecepatan motor 180 rpm mengalami perubahan, apabila permbebanan diatas 75% motor akan mengalami overload

Grafik 4-1
Duty cycle 100% pada Rpm 1500

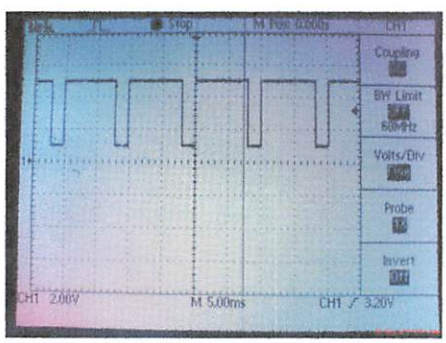



Grafik 4-2
Duty cycle 85% pada Rpm 1400

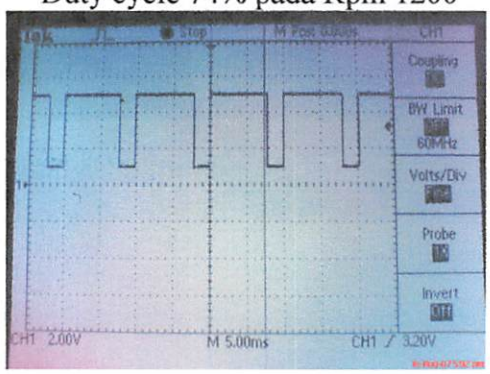



Grafik 4-3

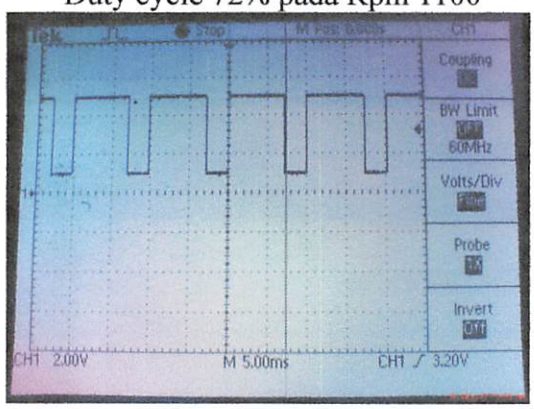
Duty cycle 78% pada Rpm 1300



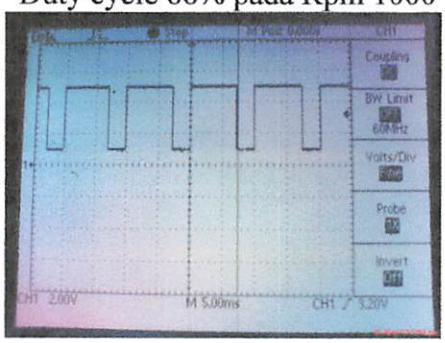
Grafik 4-4
Duty cycle 74% pada Rpm 1200



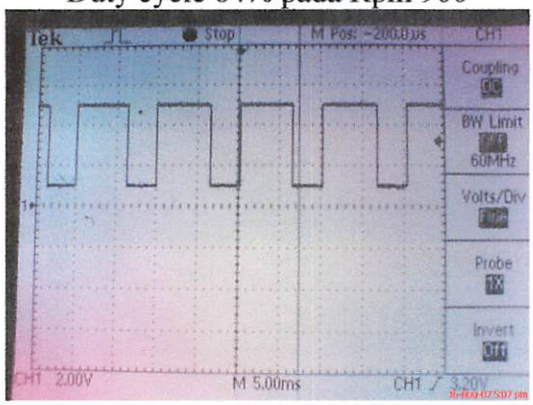

Grafik 4-5
Duty cycle 72% pada Rpm 1100



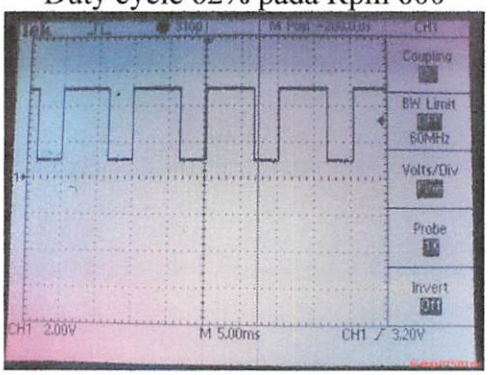

Grafik 4-6
Duty cycle 66% pada Rpm 1000



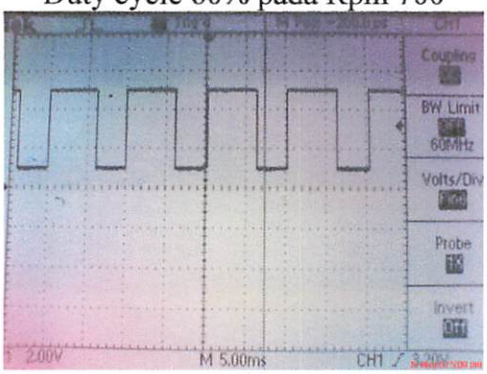
Grafik 4-7
Duty cycle 64% pada Rpm 900




Grafik 4-8
Duty cycle 62% pada Rpm 800




Grafik 4-9
Duty cycle 60% pada Rpm 700

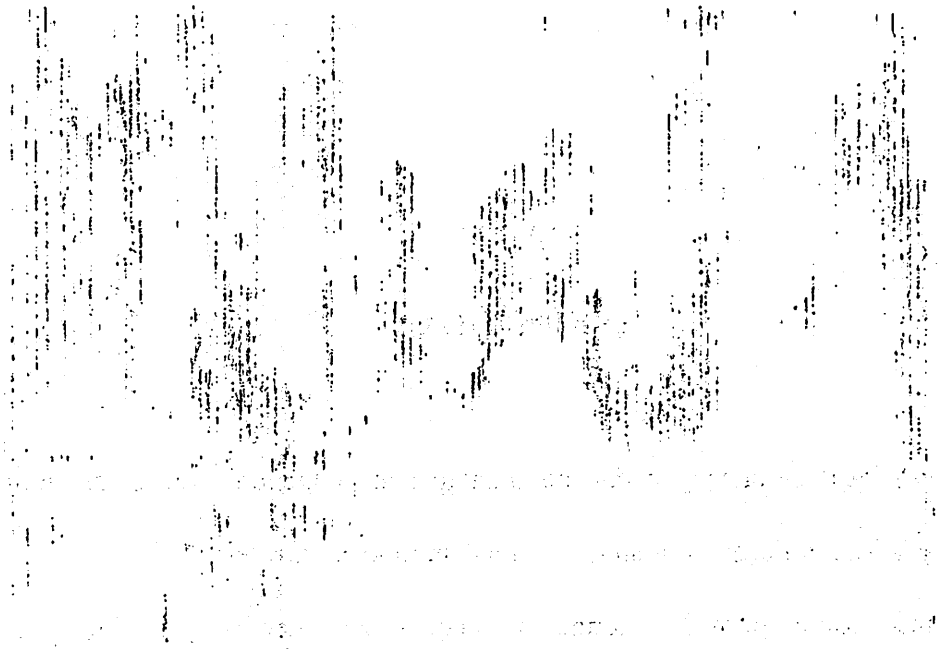


# BAB V

# KESIMPULAN

Dari hasil perancangan dan pembuatan alat pengendali kecepatan motor DC berbasis logika Fuzzy dengan menggunakan mikrokontroler :

1. Fuzzy logic controller merupakan suatu system kendali yang jauh lebih mudah penerapannya dibandingkan kendali konvensional,karena tidak perlu mencari model matematis dari system.
2. Hasil pengujian menunjukan bahwa ketika kecepatan diberi setting 100 rpm, kecepatan maximal mencapai 120 dan kecepatan minimum 60 rpm dalam pembebanan perlu dilakukan batasan beban yag diberikan ke motor, karena pada saat overload motor akan lebih lambat untuk mencapai kestabilan. Dari data diatas maka dapat kita simpulkan bahwa sangat perlu dilakukan pengaturan kecepatannya untuk kesetabilan putaran motor tersebut.
3. Mempunyai waktu yang ditempuh dari mengalami perubahan ke set point yang telah ditentukan 2,5 detik.



# DAFTAR PUSTAKA


[1] Ogata, Katsuhiko, "*Teknik Kontrol Automatik Jilid 1*", University Of Minnesota, Edisi kedua, Penerbit Erlangga, Jakarta, 1997.

[2] Zuhal,"*Dasar Teknik Tenaga Listrik*", Penerbit ITB Bandung, 1991.

[4] *http:// www.*HyperPhysics.com

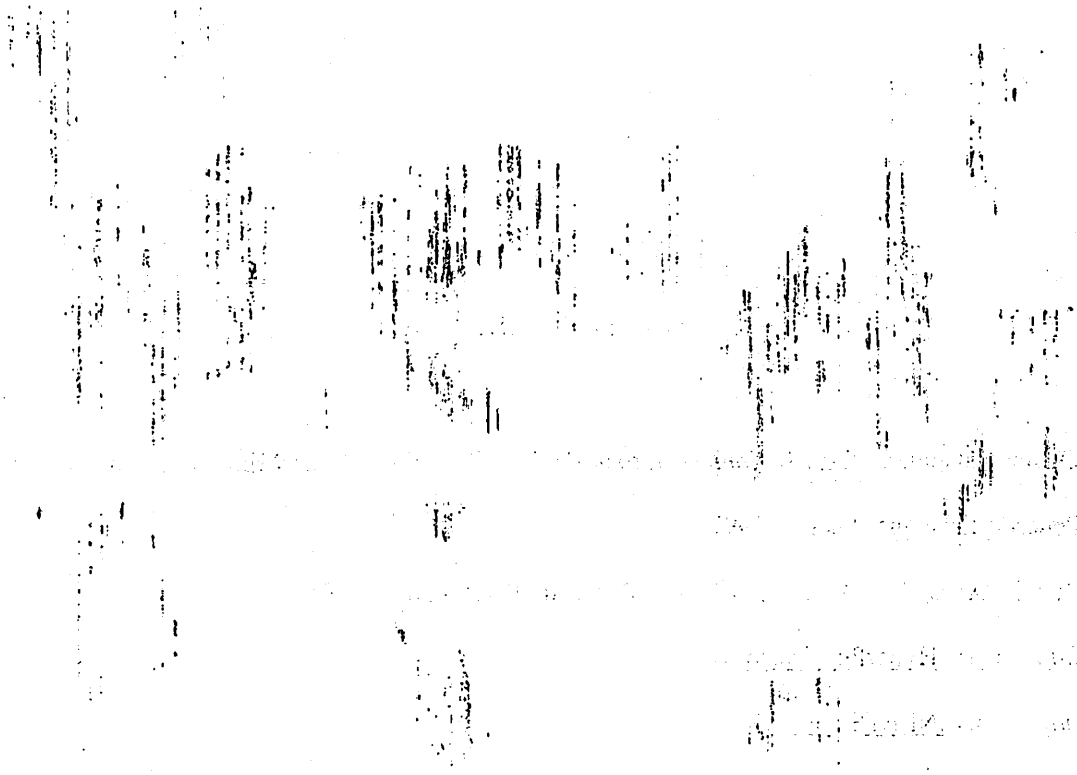
[5] *http:// www.*NEETS.com

[6] *http:// www.*ATMEL.com

[7] Ridwan Gunawan & Taufan H," **KENDALI KECEPATAN PUTAR MOTOR INDUKSI TIGA-PHASA BERBASIS LOGIKA FUZZY DENGAN MK AT89C51**," ITS IES 2004.

[8] Hani,Agus,Resman,"**IMPLEMENTASI FUZZY LOGIC PADA MIKROKONTROLLER UNTUK KENDALI MOTOR DC**," Industrial Elektrinic Seminar 1999(SSTE 1999).

[9] J. S. R. Jang, C. T. Sun & E Mizutami," **NEURO FUZZY AND SOFT COMPUTING**". Acomputational Aprouch To Learning And Machine Intelegence. **erfacing Komputer dan Mikrokontroler**", Yayasan PUIL, Jakarta 2004.




INSTITUT TEKNOLOGI NASIONAL
ITN
MALANG

# LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

# PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Senin
Tanggal : 03 September 2007
Telah dilakukan perbaikan skripsi oleh :

1. Nama : Tulus Junaedhi
2. NIM : 01.12.123
3. Jurusan : Teknik Elektro
4. Konsentrasi : Teknik Energi Listrik
5. Judul Skripsi : Perencanaan dan Pembuatan Alat Pengendali Motor DC Berbasis Fuzzy Logic dengan Menggunakan MK AT89S51 Tampilan LCD.

Perbaikan meliputi :

| No | Materi Perbaikan | Ket |
|---|---|---|
| 1. | Abstraksi dibetulkan | |
| 2. | Latar belakang | |
| 3. | Rumusan masalah dibetulkan | |
| 4. | Keterangan gambar 1 spasi | |
| 5 | Kesimpulan diganti | |
| 6 | Cara perhitung,Hitung diameter putaran | |

**Anggota Penguji**
**Penguji Pertama**

**Bambang Prio Hartono, ST, MT**

**Dosen Pembimbing I**

**Ir. Widodo Pujdi M., MT**

**Dosen Pembimbing II**

**I Komang Somawirata, ST, MT**

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

# PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Senin
Tanggal : 03 September 2007
Telah dilakukan perbaikan skripsi oleh :

1. Nama : Tulus Junaedhi
2. NIM : 01.12.123
3. Jurusan : Teknik Elektro
4. Konsentrasi : Teknik Energi Listrik
5. Judul Skripsi : Perencanaan Pembuatan Alat Pengendali Motor DC Berbasis Fuzzy Logic Dengan Menggunakan MK AT89S51 Dengan Tampilan LCD.

Perbaikan meliputi :

| No | Materi Perbaikan | Ket |
|---|---|---|
| 1. | Kesimpulan. | |
| 2. | Hasil penulisan uji peralatan. | |
| 3. | Tabel 4.3 set point tentukan, Tegangan diukur dengan volt meter. | |
| 4. | Tabel 4.2 cara mengukur. | |

**Anggota Penguji**
**Penguji Kedua**

**Ir. Taufik Hidayat, MT**

**Dosen Pembimbing I**

**Ir. Widodo Pujdi M., MT**

**Dosen Pembimbing II**

**I komang Samawirata, ST., MT**

**INSTITUT TEKNOLOGI NASIONAL MALANG**
**FAKULTAS TEKNOLOGI INDUSTRI**
**JURUSAN TEKNIK ELEKTRO S-1**
**KONSENTRASI TEKNIK ENERGI LISTRIK**

# BERITA ACARA UJIAN SKRIPSI
# FAKULTAS TEKNOLOGI INDUSTRI

1. **Nama** : Tulus Junaedhi
2. **NIM** : 01.12.123
3. **Jurusan** : Teknik Elektro S-1
4. **Konsentrasi** : Teknik Energi Listrik
5. **Judul Skripsi** : **PERENCANAAN DAN PEMBUATAN ALAT KONTROL MOTOR DC BERBASIS FUZZY LOGIC DENGAN MENGGUNAKAN MIKROCONTRLLER AT89S51 DENGAN TAMPILAN LCD**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

**Hari** : Senin
**Tanggal** : 3 September 2007
**Dengan Nilai** : 70,15 ( B )

**Panitia Ujian Skripsi**

| | |
|---|---|
| **Ir. Mochtar Asroni, MSME** | **Ir. F. Yudi Limpraptono, MT** |
| **Ketua** | **Sekretaris** |

**Anggota Penguji**

| | |
|---|---|
| **Bambang Prio Hartono,ST,MT** | **Ir. Taufik Hidayat, MT** |
| **Penguji Pertama** | **Penguji Kedua** |

Institut Teknologi Nasional Malang
**Fakultas Teknologi Industri**
**Jurusan Teknik Elektro**

# FORMULIR BIMBINGAN SKRIPSI

**Nama** : **TULUS JUNAEDHI**

**Nim** : **01.12.123**

**Masa Bimbingan** : **18 Juni 2007 sd 18 November 2007**

**Judul** : **PERENCANAAN PEMBUATAN ALAT PENGENDALI KECEPATAN MOTOR DC BERBASIS FUZZY LOGIC DENGAN MENGGUNAKAN MIKROKONTROLLER AT89S51 DENGAN TAMPILAN LCD**

| NO | Tanggal | Uraian | Paraf Pembimbing |
|---|---|---|---|
| 1 | 20/07 07 | konsultasi makalah/proposal | |
| 2 | 2/08 07 | Bab I dan II | |
| 3 | 6/08 07 | Bab III | |
| 4 | 7/08 07 | refisi bab III | |
| 5 | 13/08 07 | Teori kurang di bab III | |
| 6 | 28/08 07 | Bab IV | |
| 7 | 31/08 07 | Acc skripsi | |
| 8 | | | |
| 9 | | | |
| 10 | | | |

**Malang,** **2007**

**Dosen Pembimbing,**

Ir. WIDODO PUDJI M, MT
**NIP. Y. 102 870 0171**

Form S-4b

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro

# FORMULIR BIMBINGAN SKRIPSI

**Nama** : TULUS JUNAEDHI

**Nim** : 01.12.123

**Masa Bimbingan** : 18 Juni 2007 sd 18 November 2007

**Judul** : PERENCANAAN PEMBUATAN ALAT PENGENDALI KECEPATAN MOTOR DC BERBASIS FUZZY LOGIC DENGAN MENGGUNAKAN MIKROKONTROLLER AT89S51 DENGAN TAMPILAN LCD

| NO | Tanggal | Uraian | Paraf Pembimbing |
|---|---|---|---|
| 1 | 22/07 07 | Konsultasi judul/proposal | |
| 2 | 5/08 07 | Bab. I & II | |
| 3 | 29/08 07 | Bab II & Bab IV | |
| 4 | 31/08 07 | Acc Skripsi | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |

Malang, 2007

Dosen Pembimbing,

I KOMANG SOMAWIRATA.ST, MT
NIP. P. 103 010 0361

Form S-4b

```
;--------------------------------------------------------
; File Created by SDCC : free open source ANSI-C Compiler
; Version 2.7.0 #4818 (May 31 2007)
; This file generated Wed Aug 08 02:17:36 2007
;--------------------------------------------------------
        .module pwm
        .optsdcc -mmcs51 --model-small

;--------------------------------------------------------
; Public variables in this module
;--------------------------------------------------------
        .globl _main
        .globl _set_pwm
        .globl _hitung_pwm_awal
        .globl _cetak_label
        .globl _delay
        .globl _timer_init
        .globl _timer_interupt
        .globl _CY
        .globl _AC
        .globl _F0
        .globl _RS1
        .globl _RS0
        .globl _OV
        .globl _FL
        .globl _P
        .globl _PS
        .globl _PT1
        .globl _PX1
        .globl _PT0
        .globl _PX0
        .globl _RD
        .globl _WR
        .globl _T1
        .globl _T0
        .globl _INT1
        .globl _INT0
        .globl _TXD
        .globl _RXD
        .globl _P3_7
        .globl _P3_6
        .globl _P3_5
        .globl _P3_4
        .globl _P3_3
        .globl _P3_2
        .globl _P3_1
        .globl _P3_0
        .globl _EA
        .globl _ES
        .globl _ET1
        .globl _EX1
        .globl _ET0
```

```
.globl _EX0
.globl _P2_7
.globl _P2_6
.globl _P2_5
.globl _P2_4
.globl _P2_3
.globl _P2_2
.globl _P2_1
.globl _P2_0
.globl _SM0
.globl _SM1
.globl _SM2
.globl _REN
.globl _TB8
.globl _RB8
.globl _TI
.globl _RI
.globl _P1_7
.globl _P1_6
.globl _P1_5
.globl _P1_4
.globl _P1_3
.globl _P1_2
.globl _P1_1
.globl _P1_0
.globl _TF1
.globl _TR1
.globl _TF0
.globl _TR0
.globl _IE1
.globl _IT1
.globl _IE0
.globl _IT0
.globl _P0_7
.globl _P0_6
.globl _P0_5
.globl _P0_4
.globl _P0_3
.globl _P0_2
.globl _P0_1
.globl _P0_0
.globl _B
.globl _A
.globl _ACC
.globl _PSW
.globl _IP
.globl _P3
.globl _IE
.globl _P2
.globl _SBUF
.globl _SCON
.globl _P1
```

```
        .globl _TH1
        .globl _TH0
        .globl _TL1
        .globl _TL0
        .globl _TMOD
        .globl _TCON
        .globl _PCON
        .globl _DPH
        .globl _DPL
        .globl _SP
        .globl _P0
        .globl _rpm_is_update
        .globl _rpm_opto_old
        .globl _rpm_div
        .globl _rpm_value
        .globl _rpm_counter
        .globl _pwm_value
        .globl _pwm_step
;--------------------------------------------------------
; special function registers
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0     =       0x0080
_SP     =       0x0081
_DPL    =       0x0082
_DPH    =       0x0083
_PCON   =       0x0087
_TCON   =       0x0088
_TMOD   =       0x0089
_TL0    =       0x008a
_TL1    =       0x008b
_TH0    =       0x008c
_TH1    =       0x008d
_P1     =       0x0090
_SCON   =       0x0098
_SBUF   =       0x0099
_P2     =       0x00a0
_IE     =       0x00a8
_P3     =       0x00b0
_IP     =       0x00b8
_PSW    =       0x00d0
_ACC    =       0x00e0
_A      =       0x00e0
_B      =       0x00f0
;--------------------------------------------------------
; special function bits
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0_0   =       0x0080
_P0_1   =       0x0081
_P0_2   =       0x0082
_P0_3   =       0x0083
```

```
_P0_4   =       0x0084
_P0_5   =       0x0085
_P0_6   =       0x0086
_P0_7   =       0x0087
_IT0    =       0x0088
_IE0    =       0x0089
_IT1    =       0x008a
_IE1    =       0x008b
_TR0    =       0x008c
_TF0    =       0x008d
_TR1    =       0x008e
_TF1    =       0x008f
_P1_0   =       0x0090
_P1_1   =       0x0091
_P1_2   =       0x0092
_P1_3   =       0x0093
_P1_4   =       0x0094
_P1_5   =       0x0095
_P1_6   =       0x0096
_P1_7   =       0x0097
_RI     =       0x0098
_TI     =       0x0099
_RB8    =       0x009a
_TB8    =       0x009b
_REN    =       0x009c
_SM2    =       0x009d
_SM1    =       0x009e
_SM0    =       0x009f
_P2_0   =       0x00a0
_P2_1   =       0x00a1
_P2_2   =       0x00a2
_P2_3   =       0x00a3
_P2_4   =       0x00a4
_P2_5   =       0x00a5
_P2_6   =       0x00a6
_P2_7   =       0x00a7
_EX0    =       0x00a8
_ET0    =       0x00a9
_EX1    =       0x00aa
_ET1    =       0x00ab
_ES     =       0x00ac
_EA     =       0x00af
_P3_0   =       0x00b0
_P3_1   =       0x00b1
_P3_2   =       0x00b2
_P3_3   =       0x00b3
_P3_4   =       0x00b4
_P3_5   =       0x00b5
_P3_6   =       0x00b6
_P3_7   =       0x00b7
_RXD    =       0x00b0
_TXD    =       0x00b1
```

```
_INT0    =       0x00b2
_INT1    =       0x00b3
_T0      =       0x00b4
_T1      =       0x00b5
_WR      =       0x00b6
_RD      =       0x00b7
_PX0     =       0x00b8
_PT0     =       0x00b9
_PX1     =       0x00ba
_PT1     =       0x00bb
_PS      =       0x00bc
_P       =       0x00d0
_FL      =       0x00d1
_OV      =       0x00d2
_RS0     =       0x00d3
_RS1     =       0x00d4
_F0      =       0x00d5
_AC      =       0x00d6
_CY      =       0x00d7
;--------------------------------------------------------
; overlayable register banks
;--------------------------------------------------------
        .area REG_BANK_0        (REL,OVR,DATA)
        .ds 8
;--------------------------------------------------------
; internal ram data
;--------------------------------------------------------
        .area DSEG    (DATA)
_pwm_step::
        .ds 1
_pwm_value::
        .ds 1
_rpm_counter::
        .ds 1
_rpm_value::
        .ds 1
_rpm_div::
        .ds 1
;--------------------------------------------------------
; overlayable items in internal ram
;--------------------------------------------------------
        .area   OSEG    (OVR,DATA)
        .area   OSEG    (OVR,DATA)
;--------------------------------------------------------
; Stack segment in internal ram
;--------------------------------------------------------
        .area   SSEG    (DATA)
__start__stack:
        .ds     1

;--------------------------------------------------------
; indirectly addressable internal ram data
```

```
;--------------------------------------------------------
        .area ISEG    (DATA)
;--------------------------------------------------------
; absolute internal ram data
;--------------------------------------------------------
        .area IABS    (ABS,DATA)
        .area IABS    (ABS,DATA)
;--------------------------------------------------------
; bit data
;--------------------------------------------------------
        .area BSEG    (BIT)
_rpm_opto_old::
        .ds 1
_rpm_is_update::
        .ds 1
;--------------------------------------------------------
; paged external ram data
;--------------------------------------------------------
        .area PSEG    (PAG,XDATA)
;--------------------------------------------------------
; external ram data
;--------------------------------------------------------
        .area XSEG    (XDATA)
;--------------------------------------------------------
; absolute external ram data
;--------------------------------------------------------
        .area XABS    (ABS,XDATA)
;--------------------------------------------------------
; external initialized ram data
;--------------------------------------------------------
        .area XISEG   (XDATA)
        .area HOME    (CODE)
        .area GSINIT0 (CODE)
        .area GSINIT1 (CODE)
        .area GSINIT2 (CODE)
        .area GSINIT3 (CODE)
        .area GSINIT4 (CODE)
        .area GSINIT5 (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area CSEG    (CODE)
;--------------------------------------------------------
; interrupt vector
;--------------------------------------------------------
        .area HOME    (CODE)
__interrupt_vect:
        ljmp    __sdcc_gsinit_startup
        reti
        .ds     7
        ljmp    _timer_interupt
;--------------------------------------------------------
; global & static initialisations
```

```
;--------------------------------------------------------
        .area HOME    (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area GSINIT  (CODE)
        .globl __sdcc_gsinit_startup
        .globl __sdcc_program_startup
        .globl __start__stack
        .globl __mcs51_genXINIT
        .globl __mcs51_genXRAMCLEAR
        .globl __mcs51_genRAMCLEAR
;       pwm.c:9: volatile unsigned char pwm_step = 100, pwm_value =
0;
        mov     _pwm_step,#0x64
;       pwm.c:9: }

        mov     _pwm_value,#0x00
;       pwm.c:10: volatile unsigned char rpm_counter = 0, rpm_value =
0, rpm_div = 50;
        mov     _rpm_counter,#0x00
;       pwm.c:10: }

        mov     _rpm_value,#0x00
;       pwm.c:10: volatile unsigned char rpm_counter = 0, rpm_value =
0, rpm_div = 50;
        mov     _rpm_div,#0x32
        .area GSFINAL (CODE)
        ljmp    __sdcc_program_startup
;--------------------------------------------------------
; Home
;--------------------------------------------------------
        .area HOME    (CODE)
        .area HOME    (CODE)
__sdcc_program_startup:
        lcall   _main
;       return from main will lock up
        sjmp .
;--------------------------------------------------------
; code
;--------------------------------------------------------
        .area CSEG    (CODE)
;------------------------------------------------------------
;Allocation info for local variables in function 'timer_interupt'
;------------------------------------------------------------
;------------------------------------------------------------
;       pwm.c:13: void timer_interupt() __interrupt (1) {
;       -----------------------------------------
;        function timer_interupt
;       -----------------------------------------
_timer_interupt:
        ar2 = 0x02
        ar3 = 0x03
```

```
                                ar4 = 0x04
                                ar5 = 0x05
                                ar6 = 0x06
                                ar7 = 0x07
                                ar0 = 0x00
                                ar1 = 0x01
                                push    acc
                                push    ar2
                                push    psw
                                mov     psw,#0x00
                        ;       pwm.c:14: if (!pwm_step--) {
                                mov     r2,_pwm_step
                                dec     _pwm_step
                                mov     a,r2
                                jnz     00104$
                        ;       pwm.c:15: if (!rpm_div--) {
                                mov     r2,_rpm_div
                                dec     _rpm_div
                                mov     a,r2
                        ;       pwm.c:16: rpm_value = rpm_counter;
                        ;       pwm.c:17: rpm_counter = 0;
                                jnz     00102$
                                mov     _rpm_value,_rpm_counter
                                mov     _rpm_counter,a
                        ;       pwm.c:18: rpm_div = 50;
                                mov     _rpm_div,#0x32
                        ;       pwm.c:19: rpm_is_update = 1;
                                setb    _rpm_is_update
                        00102$:
                        ;       pwm.c:21: pwm_step = 100;
                                mov     _pwm_step,#0x64
                        ;       pwm.c:22: PWM_OUT_PIN = 0;
                                clr     _P3_2
                        00104$:
                        ;       pwm.c:25: if (RPM_OPTO_PIN != rpm_opto_old) {
                                mov     c,_P3_1
                                jb      _rpm_opto_old,00117$
                                cpl     c
                        00117$:
                                jc      00106$
                        ;       pwm.c:26: rpm_counter++;
                                inc     _rpm_counter
                        ;       pwm.c:27: rpm_opto_old = RPM_OPTO_PIN;
                                mov     c,_P3_1
                                mov     _rpm_opto_old,c
                        00106$:
                        ;       pwm.c:30: if (pwm_step == pwm_value)
                                mov     a,_pwm_value
                                cjne    a,_pwm_step,00109$
                        ;       pwm.c:31: PWM_OUT_PIN = 1;
                                setb    _P3_2
                        00109$:
```

```
        pop     psw
        pop     ar2
        pop     acc
        reti
;       eliminated unneeded push/pop dpl
;       eliminated unneeded push/pop dph
;       eliminated unneeded push/pop b
;------------------------------------------------------------
;Allocation info for local variables in function 'timer_init'
;------------------------------------------------------------
;------------------------------------------------------------
;       pwm.c:35: void timer_init() {
;       ----------------------------------------
;        function timer_init
;       ----------------------------------------
_timer_init:
;       pwm.c:36: TH0 = 256-100;
        mov     _TH0,#0x9C
;       pwm.c:37: TMOD = 0x02;
        mov     _TMOD,#0x02
;       pwm.c:38: ET0 = 1;
        setb    _ET0
;       pwm.c:39: EA = 1;
        setb    _EA
;       pwm.c:40: TR0 = 1;
        setb    _TR0
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'delay'
;------------------------------------------------------------
;i                         Allocated to registers r2 r3
;------------------------------------------------------------
;       pwm.c:43: void delay(unsigned int i) {
;       ----------------------------------------
;        function delay
;       ----------------------------------------
_delay:
        mov     r2,dpl
        mov     r3,dph
;       pwm.c:44: while (i--)
00101$:
        mov     ar4,r2
        mov     ar5,r3
        dec     r2
        cjne    r2,#0xff,00109$
        dec     r3
00109$:
        mov     a,r4
        orl     a,r5
        jz      00104$
;       pwm.c:45: lcd_delay_long();
        push    ar2
```

```
        push    ar3
        lcall   _lcd_delay_long
        pop     ar3
        pop     ar2
        sjmp    00101$
00104$:
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'cetak_label'
;------------------------------------------------------------
;------------------------------------------------------------
;       pwm.c:48: void cetak_label() {
;       ----------------------------------------
;        function cetak_label
;       ----------------------------------------
_cetak_label:
;       pwm.c:49: lcd_clear();
        lcall   _lcd_clear
;       pwm.c:50: printf_fast("Kontrol Kec.");
        mov     a,#__str_0
        push    acc
        mov     a,#(__str_0 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:51: lcd_goto(0x40);
        mov     dpl,#0x40
        lcall   _lcd_goto
;       pwm.c:52: printf_fast("Motor DC");
        mov     a,#__str_1
        push    acc
        mov     a,#(__str_1 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:53: delay(3);
        mov     dptr,#0x0003
        lcall   _delay
;       pwm.c:54: lcd_clear();
        lcall   _lcd_clear
;       pwm.c:55: printf_fast("TULUS JUNAIDHY");
        mov     a,#__str_2
        push    acc
        mov     a,#(__str_2 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:56: lcd_goto(0x40);
        mov     dpl,#0x40
```

```
        lcall   _lcd_goto
;       pwm.c:57: printf_fast("01 12 123");
        mov     a,#__str_3
        push    acc
        mov     a,#(__str_3 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:58: delay(3);
        mov     dptr,#0x0003
        lcall   _delay
;       pwm.c:59: lcd_clear();
        lcall   _lcd_clear
;       pwm.c:60: printf_fast("Seting Kec.");
        mov     a,#__str_4
        push    acc
        mov     a,#(__str_4 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:61: lcd_goto(0x40);
        mov     dpl,#0x40
        ljmp    _lcd_goto
;------------------------------------------------------------
;Allocation info for local variables in function 'hitung_pwm_awal'
;------------------------------------------------------------
;pwm                       Allocated to registers r2
;------------------------------------------------------------
;       pwm.c:64: unsigned char hitung_pwm_awal(unsigned char pwm) {
;       -----------------------------------------
;        function hitung_pwm_awal
;       -----------------------------------------
_hitung_pwm_awal:
        mov     r2,dpl
;       pwm.c:65: pwm = pwm + 50;
        mov     a,#0x32
        add     a,r2
;       pwm.c:66: return pwm;
        mov     dpl,a
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'set_pwm'
;------------------------------------------------------------
;delta                     Allocated to registers r2
;temp                      Allocated to registers r3 r4
;------------------------------------------------------------
;       pwm.c:69: void set_pwm(char delta) {
;       -----------------------------------------
;        function set_pwm
;       -----------------------------------------
```

```
_set_pwm:
        mov     r2,dpl
;       pwm.c:70: int temp = pwm_value;
        mov     r3,_pwm_value
        mov     r4,#0x00
;       pwm.c:71: temp += delta;
        mov     a,r2
        rlc     a
        subb    a,acc
        mov     r5,a
        mov     a,r2
        add     a,r3
        mov     r3,a
        mov     a,r5
        addc    a,r4
;       pwm.c:72: if (temp<0)
        mov     r4,a
        jnb     acc.7,00102$
;       pwm.c:73: temp = 0;
        mov     r3,#0x00
        mov     r4,#0x00
00102$:
;       pwm.c:74: if (temp>100)
        clr     c
        mov     a,#0x64
        subb    a,r3
        mov     a,#(0x00 ^ 0x80)
        mov     b,r4
        xrl     b,#0x80
        subb    a,b
        jnc     00104$
;       pwm.c:75: temp = 100;
        mov     r3,#0x64
        mov     r4,#0x00
00104$:
;       pwm.c:76: pwm_value = temp;
        mov     _pwm_value,r3
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'main'
;------------------------------------------------------------
;rpm                       Allocated to registers r2
;rpm_target                Allocated to registers r4
;rpm_input                 Allocated to registers r2 r3
;------------------------------------------------------------
;       pwm.c:79: void main() {
;       -----------------------------------------
;        function main
;       -----------------------------------------
_main:
;       pwm.c:81: int rpm_input = -1;
        mov     r2,#0xFF
```

```
        mov     r3,#0xFF
;       pwm.c:83: timer_init();
        push    ar2
        push    ar3
        lcall   _timer_init
;       pwm.c:84: lcd_init();
        lcall   _lcd_init
;       pwm.c:85: cetak_label();
        lcall   _cetak_label
        pop     ar3
        pop     ar2
;       pwm.c:87: while (rpm_input < 0) {
00101$:
        mov     a,r3
        jnb     acc.7,00103$
;       pwm.c:88: lcd_goto(0x40);
        mov     dpl,#0x40
        lcall   _lcd_goto
;       pwm.c:89: printf_fast("    ");
        mov     a,#__str_5
        push    acc
        mov     a,#(__str_5 >> 8)
        push    acc
        lcall   _printf_fast
        dec     sp
        dec     sp
;       pwm.c:90: lcd_goto(0x40);
        mov     dpl,#0x40
        lcall   _lcd_goto
;       pwm.c:91: rpm_input = getint();
        lcall   _getint
        mov     r2,dpl
        mov     r3,dph
        sjmp    00101$
00103$:
;       pwm.c:94: lcd_clear();
        push    ar2
        push    ar3
        lcall   _lcd_clear
        pop     ar3
        pop     ar2
;       pwm.c:95: rpm_target = (char)(rpm_input / 30);
        mov     __divsint_PARM_2,#0x1E
        clr     a
        mov     (__divsint_PARM_2 + 1),a
        mov     dpl,r2
        mov     dph,r3
        push    ar2
        push    ar3
        lcall   __divsint
        mov     r4,dpl
        pop     ar3
```

```
        pop     ar2
;       pwm.c:96: printf_fast("Seting %drpm",rpm_input);
        push    ar4
        push    ar2
        push    ar3
        mov     a,#__str_6
        push    acc
        mov     a,#(__str_6 >> 8)
        push    acc
        lcall   _printf_fast
        mov     a,sp
        add     a,#0xfc
        mov     sp,a
        pop     ar4
;       pwm.c:98: pwm_value = 50;
        mov     _pwm_value,#0x32
;       pwm.c:99: while(1) {
00107$:
;       pwm.c:100: if (rpm_is_update) {
        jnb     _rpm_is_update,00107$
;       pwm.c:101: rpm = rpm_value;
        mov     r2,_rpm_value
;       pwm.c:102: rpm_is_update = 0;
        clr     _rpm_is_update
;       pwm.c:103: set_pwm((rpm_target - rpm) / 2);
        mov     a,r4
        mov     r3,a
        rlc     a
        subb    a,acc
        mov     r5,a
        mov     a,r2
        mov     r6,a
        rlc     a
        subb    a,acc
        mov     r7,a
        mov     a,r3
        clr     c
        subb    a,r6
        mov     dpl,a
        mov     a,r5
        subb    a,r7
        mov     dph,a
        mov     __divsint_PARM_2,#0x02
        clr     a
        mov     (__divsint_PARM_2 + 1),a
        push    ar2
        push    ar4
        lcall   __divsint
        lcall   _set_pwm
;       pwm.c:104: lcd_goto(0x40);
        mov     dpl,#0x40
        lcall   _lcd_goto
```

```
        pop     ar4
        pop     ar2
;       pwm.c:105: printf_fast("Nilai %drpm   ", rpm * 30);
        clr     F0
        mov     b,#0x1e
        mov     a,r2
        jnb     acc.7,00117$
        cpl     F0
        cpl     a
        inc     a
00117$:
        mul     ab
        jnb     F0,00118$
        cpl     a
        add     a,#1
        xch     a,b
        cpl     a
        addc    a,#0
        xch     a,b
00118$:
        mov     r2,a
        mov     r3,b
        push    ar4
        push    ar2
        push    ar3
        mov     a,#__str_7
        push    acc
        mov     a,#(__str_7 >> 8)
        push    acc
        lcall   _printf_fast
        mov     a,sp
        add     a,#0xfc
        mov     sp,a
        pop     ar4
        ljmp    00107$
        .area CSEG    (CODE)
        .area CONST   (CODE)
__str_0:
        .ascii "Kontrol Kec."
        .db 0x00
__str_1:
        .ascii "Motor DC"
        .db 0x00
__str_2:
        .ascii "TULUS JUNAIDHY"
        .db 0x00
__str_3:
        .ascii "01 12 123"
        .db 0x00
__str_4:
        .ascii "Seting Kec."
        .db 0x00
```

```
__str_5:
        .ascii "     "
        .db 0x00
__str_6:
        .ascii "Seting %drpm"
        .db 0x00
__str_7:
        .ascii "Nilai %drpm    "
        .db 0x00
        .area XINIT   (CODE)
        .area CABS    (ABS,CODE)
```

```
;--------------------------------------------------------
; File Created by SDCC : free open source ANSI-C Compiler
; Version 2.7.0 #4818 (May 31 2007)
; This file generated Wed Aug 08 02:17:36 2007
;--------------------------------------------------------
        .module lcd
        .optsdcc -mmcs51 --model-small

;--------------------------------------------------------
; Public variables in this module
;--------------------------------------------------------
        .globl _lcd_instr
        .globl _CY
        .globl _AC
        .globl _F0
        .globl _RS1
        .globl _RS0
        .globl _OV
        .globl _FL
        .globl _P
        .globl _PS
        .globl _PT1
        .globl _PX1
        .globl _PT0
        .globl _PX0
        .globl _RD
        .globl _WR
        .globl _T1
        .globl _T0
        .globl _INT1
        .globl _INT0
        .globl _TXD
        .globl _RXD
        .globl _P3_7
        .globl _P3_6
        .globl _P3_5
        .globl _P3_4
        .globl _P3_3
        .globl _P3_2
        .globl _P3_1
        .globl _P3_0
        .globl _EA
        .globl _ES
        .globl _ET1
        .globl _EX1
        .globl _ET0
        .globl _EX0
        .globl _P2_7
        .globl _P2_6
        .globl _P2_5
        .globl _P2_4
        .globl _P2_3
        .globl _P2_2
        .globl _P2_1
        .globl _P2_0
        .globl _SM0
        .globl _SM1
        .globl _SM2
        .globl _REN
        .globl _TB8
        .globl _RB8
        .globl _TI
```

```
        .globl _RI
        .globl _P1_7
        .globl _P1_6
        .globl _P1_5
        .globl _P1_4
        .globl _P1_3
        .globl _P1_2
        .globl _P1_1
        .globl _P1_0
        .globl _TF1
        .globl _TR1
        .globl _TF0
        .globl _TR0
        .globl _IE1
        .globl _IT1
        .globl _IE0
        .globl _IT0
        .globl _P0_7
        .globl _P0_6
        .globl _P0_5
        .globl _P0_4
        .globl _P0_3
        .globl _P0_2
        .globl _P0_1
        .globl _P0_0
        .globl _B
        .globl _A
        .globl _ACC
        .globl _PSW
        .globl _IP
        .globl _P3
        .globl _IE
        .globl _P2
        .globl _SBUF
        .globl _SCON
        .globl _P1
        .globl _TH1
        .globl _TH0
        .globl _TL1
        .globl _TL0
        .globl _TMOD
        .globl _TCON
        .globl _PCON
        .globl _DPH
        .globl _DPL
        .globl _SP
        .globl _P0
        .globl _lcd_delay
        .globl _lcd_delay_long
        .globl _lcd_goto
        .globl _lcd_cursor_on
        .globl _lcd_cursor_off
        .globl _lcd_clear
        .globl _lcd_init
        .globl _putchar
;--------------------------------------------------------
; special function registers
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0     =       0x0080
_SP     =       0x0081
_DPL    =       0x0082
```

```
_DPH    =       0x0083
_PCON   =       0x0087
_TCON   =       0x0088
_TMOD   =       0x0089
_TL0    =       0x008a
_TL1    =       0x008b
_TH0    =       0x008c
_TH1    =       0x008d
_P1     =       0x0090
_SCON   =       0x0098
_SBUF   =       0x0099
_P2     =       0x00a0
_IE     =       0x00a8
_P3     =       0x00b0
_IP     =       0x00b8
_PSW    =       0x00d0
_ACC    =       0x00e0
_A      =       0x00e0
_B      =       0x00f0
;--------------------------------------------------------
; special function bits
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0_0   =       0x0080
_P0_1   =       0x0081
_P0_2   =       0x0082
_P0_3   =       0x0083
_P0_4   =       0x0084
_P0_5   =       0x0085
_P0_6   =       0x0086
_P0_7   =       0x0087
_IT0    =       0x0088
_IE0    =       0x0089
_IT1    =       0x008a
_IE1    =       0x008b
_TR0    =       0x008c
_TF0    =       0x008d
_TR1    =       0x008e
_TF1    =       0x008f
_P1_0   =       0x0090
_P1_1   =       0x0091
_P1_2   =       0x0092
_P1_3   =       0x0093
_P1_4   =       0x0094
_P1_5   =       0x0095
_P1_6   =       0x0096
_P1_7   =       0x0097
_RI     =       0x0098
_TI     =       0x0099
_RB8    =       0x009a
_TB8    =       0x009b
_REN    =       0x009c
_SM2    =       0x009d
_SM1    =       0x009e
_SM0    =       0x009f
_P2_0   =       0x00a0
_P2_1   =       0x00a1
_P2_2   =       0x00a2
_P2_3   =       0x00a3
_P2_4   =       0x00a4
_P2_5   =       0x00a5
_P2_6   =       0x00a6
```

```
_P2_7   =       0x00a7
_EX0    =       0x00a8
_ET0    =       0x00a9
_EX1    =       0x00aa
_ET1    =       0x00ab
_ES     =       0x00ac
_EA     =       0x00af
_P3_0   =       0x00b0
_P3_1   =       0x00b1
_P3_2   =       0x00b2
_P3_3   =       0x00b3
_P3_4   =       0x00b4
_P3_5   =       0x00b5
_P3_6   =       0x00b6
_P3_7   =       0x00b7
_RXD    =       0x00b0
_TXD    =       0x00b1
_INT0   =       0x00b2
_INT1   =       0x00b3
_T0     =       0x00b4
_T1     =       0x00b5
_WR     =       0x00b6
_RD     =       0x00b7
_PX0    =       0x00b8
_PT0    =       0x00b9
_PX1    =       0x00ba
_PT1    =       0x00bb
_PS     =       0x00bc
_P      =       0x00d0
_FL     =       0x00d1
_OV     =       0x00d2
_RS0    =       0x00d3
_RS1    =       0x00d4
_F0     =       0x00d5
_AC     =       0x00d6
_CY     =       0x00d7
;--------------------------------------------------------
; overlayable register banks
;--------------------------------------------------------
        .area REG_BANK_0        (REL,OVR,DATA)
        .ds 8
;--------------------------------------------------------
; internal ram data
;--------------------------------------------------------
        .area DSEG    (DATA)
;--------------------------------------------------------
; overlayable items in internal ram
;--------------------------------------------------------
        .area   OSEG    (OVR,DATA)
;--------------------------------------------------------
; indirectly addressable internal ram data
;--------------------------------------------------------
        .area ISEG    (DATA)
;--------------------------------------------------------
; absolute internal ram data
;--------------------------------------------------------
        .area IABS    (ABS,DATA)
        .area IABS    (ABS,DATA)
;--------------------------------------------------------
; bit data
;--------------------------------------------------------
        .area BSEG    (BIT)
```

```
;--------------------------------------------------------
; paged external ram data
;--------------------------------------------------------
        .area PSEG    (PAG,XDATA)
;--------------------------------------------------------
; external ram data
;--------------------------------------------------------
        .area XSEG    (XDATA)
;--------------------------------------------------------
; absolute external ram data
;--------------------------------------------------------
        .area XABS    (ABS,XDATA)
;--------------------------------------------------------
; external initialized ram data
;--------------------------------------------------------
        .area XISEG   (XDATA)
        .area HOME    (CODE)
        .area GSINIT0 (CODE)
        .area GSINIT1 (CODE)
        .area GSINIT2 (CODE)
        .area GSINIT3 (CODE)
        .area GSINIT4 (CODE)
        .area GSINIT5 (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area CSEG    (CODE)
;--------------------------------------------------------
; global & static initialisations
;--------------------------------------------------------
        .area HOME    (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area GSINIT  (CODE)
;--------------------------------------------------------
; Home
;--------------------------------------------------------
        .area HOME    (CODE)
        .area HOME    (CODE)
;--------------------------------------------------------
; code
;--------------------------------------------------------
        .area CSEG    (CODE)
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_delay'
;------------------------------------------------------------
;i                         Allocated to registers r2
;j                         Allocated to registers r3
;------------------------------------------------------------
;       lcd.c:4: void lcd_delay(void)
;       -----------------------------------------
;        function lcd_delay
;       -----------------------------------------
_lcd_delay:
        ar2 = 0x02
        ar3 = 0x03
        ar4 = 0x04
        ar5 = 0x05
        ar6 = 0x06
        ar7 = 0x07
        ar0 = 0x00
        ar1 = 0x01
;       lcd.c:7: for (i=0;i<0x10;i++)
```

```
        mov     r2,#0x00
00104$:
        cjne    r2,#0x10,00116$
00116$:
        jnc     00108$
;       lcd.c:8: for (j=0;j<0xFF;j++);
        mov     r3,#0xFF
00103$:
        djnz    r3,00103$
;       lcd.c:7: for (i=0;i<0x10;i++)
        inc     r2
        sjmp    00104$
00108$:
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_delay_long'
;------------------------------------------------------------
;i                          Allocated to registers r2
;------------------------------------------------------------
;       lcd.c:11: void lcd_delay_long(void)
;       ---------------------------------------
;        function lcd_delay_long
;       ---------------------------------------
_lcd_delay_long:
;       lcd.c:14: for (i=0;i<0x30;i++)
        mov     r2,#0x30
00103$:
;       lcd.c:15: lcd_delay();
        push    ar2
        lcall   _lcd_delay
        pop     ar2
        djnz    r2,00103$
;       lcd.c:14: for (i=0;i<0x30;i++)
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_instr'
;------------------------------------------------------------
;instr                      Allocated to registers r2
;------------------------------------------------------------
;       lcd.c:18: void lcd_instr(char instr)
;       ---------------------------------------
;        function lcd_instr
;       ---------------------------------------
_lcd_instr:
        mov     r2,dpl
;       lcd.c:20: LCD_PORT_RS = 0;
        clr     _P2_7
;       lcd.c:21: LCD_PORT_DATA = instr;
        mov     _P0,r2
;       lcd.c:22: LCD_PORT_EN = 1;
        setb    _P2_6
;       lcd.c:23: LCD_PORT_EN = 0;
        clr     _P2_6
;       lcd.c:24: lcd_delay();
        ljmp    _lcd_delay
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_goto'
;------------------------------------------------------------
;addr                       Allocated to registers r2
;------------------------------------------------------------
;       lcd.c:27: void lcd_goto(char addr)
;       ---------------------------------------
```

```
;        function lcd_goto
;        ---------------------------------------
_lcd_goto:
        mov     r2,dpl
;       lcd.c:29: lcd_instr(addr | 0x80);
        mov     a,#0x80
        orl     a,r2
        mov     dpl,a
        ljmp    _lcd_instr
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_cursor_on'
;------------------------------------------------------------
;------------------------------------------------------------
;       lcd.c:32: void lcd_cursor_on(void)
;       ---------------------------------------
;        function lcd_cursor_on
;       ---------------------------------------
_lcd_cursor_on:
;       lcd.c:34: lcd_instr(0x0F);
        mov     dpl,#0x0F
        ljmp    _lcd_instr
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_cursor_off'
;------------------------------------------------------------
;------------------------------------------------------------
;       lcd.c:37: void lcd_cursor_off(void)
;       ---------------------------------------
;        function lcd_cursor_off
;       ---------------------------------------
_lcd_cursor_off:
;       lcd.c:39: lcd_instr(0x0C);
        mov     dpl,#0x0C
        ljmp    _lcd_instr
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_clear'
;------------------------------------------------------------
;------------------------------------------------------------
;       lcd.c:42: void lcd_clear(void)
;       ---------------------------------------
;        function lcd_clear
;       ---------------------------------------
_lcd_clear:
;       lcd.c:44: lcd_instr(0x01);
        mov     dpl,#0x01
        lcall   _lcd_instr
;       lcd.c:45: lcd_instr(0x02);
        mov     dpl,#0x02
        lcall   _lcd_instr
;       lcd.c:46: lcd_instr(0x02);
        mov     dpl,#0x02
        ljmp    _lcd_instr
;------------------------------------------------------------
;Allocation info for local variables in function 'lcd_init'
;------------------------------------------------------------
;------------------------------------------------------------
;       lcd.c:49: void lcd_init(void)
;       ---------------------------------------
;        function lcd_init
;       ---------------------------------------
_lcd_init:
;       lcd.c:51: lcd_delay_long();
        lcall   _lcd_delay_long
```

```
;       lcd.c:52: lcd_instr(0x3F);
        mov     dpl,#0x3F
        lcall   _lcd_instr
;       lcd.c:53: lcd_delay_long();
        lcall   _lcd_delay_long
;       lcd.c:54: lcd_instr(0x0D);
        mov     dpl,#0x0D
        lcall   _lcd_instr
;       lcd.c:55: lcd_delay_long();
        lcall   _lcd_delay_long
;       lcd.c:56: lcd_instr(0x06);
        mov     dpl,#0x06
        lcall   _lcd_instr
;       lcd.c:57: lcd_delay_long();
        lcall   _lcd_delay_long
;       lcd.c:58: lcd_instr(0x01);
        mov     dpl,#0x01
        lcall   _lcd_instr
;       lcd.c:59: lcd_delay_long();
        lcall   _lcd_delay_long
;       lcd.c:60: lcd_instr(0x0C);
        mov     dpl,#0x0C
        lcall   _lcd_instr
;       lcd.c:61: lcd_delay_long();
        lcall   _lcd_delay_long
;       lcd.c:62: lcd_clear();
        ljmp    _lcd_clear
;------------------------------------------------------------
;Allocation info for local variables in function 'putchar'
;------------------------------------------------------------
;_data                      Allocated to registers r2
;------------------------------------------------------------
;       lcd.c:65: void putchar(char _data)
;       -----------------------------------------
;        function putchar
;       -----------------------------------------
_putchar:
        mov     r2,dpl
;       lcd.c:67: LCD_PORT_RS = 1;
        setb    _P2_7
;       lcd.c:68: LCD_PORT_DATA = _data;
        mov     _P0,r2
;       lcd.c:69: LCD_PORT_EN = 1;
        setb    _P2_6
;       lcd.c:70: LCD_PORT_EN = 0;
        clr     _P2_6
;       lcd.c:71: lcd_delay();
        ljmp    _lcd_delay
        .area CSEG    (CODE)
        .area CONST   (CODE)
        .area XINIT   (CODE)
        .area CABS    (ABS,CODE)
```

```
;--------------------------------------------------------
; File Created by SDCC : free open source ANSI-C Compiler
; Version 2.7.0 #4818 (May 31 2007)
; This file generated Wed Aug 08 02:17:35 2007
;--------------------------------------------------------
        .module keypad
        .optsdcc -mmcs51 --model-small

;--------------------------------------------------------
; Public variables in this module
;--------------------------------------------------------
        .globl _char_table
        .globl _ln2
        .globl _CY
        .globl _AC
        .globl _F0
        .globl _RS1
        .globl _RS0
        .globl _OV
        .globl _FL
        .globl _P
        .globl _PS
        .globl _PT1
        .globl _PX1
        .globl _PT0
        .globl _PX0
        .globl _RD
        .globl _WR
        .globl _T1
        .globl _T0
        .globl _INT1
        .globl _INT0
        .globl _TXD
        .globl _RXD
        .globl _P3_7
        .globl _P3_6
        .globl _P3_5
        .globl _P3_4
        .globl _P3_3
        .globl _P3_2
        .globl _P3_1
        .globl _P3_0
        .globl _EA
        .globl _ES
        .globl _ET1
        .globl _EX1
        .globl _ET0
        .globl _EX0
        .globl _P2_7
        .globl _P2_6
        .globl _P2_5
        .globl _P2_4
        .globl _P2_3
        .globl _P2_2
        .globl _P2_1
        .globl _P2_0
        .globl _SM0
        .globl _SM1
        .globl _SM2
        .globl _REN
        .globl _TB8
        .globl _RB8
```

```
        .globl _TI
        .globl _RI
        .globl _P1_7
        .globl _P1_6
        .globl _P1_5
        .globl _P1_4
        .globl _P1_3
        .globl _P1_2
        .globl _P1_1
        .globl _P1_0
        .globl _TF1
        .globl _TR1
        .globl _TF0
        .globl _TR0
        .globl _IE1
        .globl _IT1
        .globl _IE0
        .globl _IT0
        .globl _P0_7
        .globl _P0_6
        .globl _P0_5
        .globl _P0_4
        .globl _P0_3
        .globl _P0_2
        .globl _P0_1
        .globl _P0_0
        .globl _B
        .globl _A
        .globl _ACC
        .globl _PSW
        .globl _IP
        .globl _P3
        .globl _IE
        .globl _P2
        .globl _SBUF
        .globl _SCON
        .globl _P1
        .globl _TH1
        .globl _TH0
        .globl _TL1
        .globl _TL0
        .globl _TMOD
        .globl _TCON
        .globl _PCON
        .globl _DPH
        .globl _DPL
        .globl _SP
        .globl _P0
        .globl _getchar
        .globl _getint
;--------------------------------------------------------
; special function registers
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0     =       0x0080
_SP     =       0x0081
_DPL    =       0x0082
_DPH    =       0x0083
_PCON   =       0x0087
_TCON   =       0x0088
_TMOD   =       0x0089
_TL0    =       0x008a
```

```
_TL1    =       0x008b
_TH0    =       0x008c
_TH1    =       0x008d
_P1     =       0x0090
_SCON   =       0x0098
_SBUF   =       0x0099
_P2     =       0x00a0
_IE     =       0x00a8
_P3     =       0x00b0
_IP     =       0x00b8
_PSW    =       0x00d0
_ACC    =       0x00e0
_A      =       0x00e0
_B      =       0x00f0
;--------------------------------------------------------
; special function bits
;--------------------------------------------------------
        .area RSEG    (DATA)
_P0_0   =       0x0080
_P0_1   =       0x0081
_P0_2   =       0x0082
_P0_3   =       0x0083
_P0_4   =       0x0084
_P0_5   =       0x0085
_P0_6   =       0x0086
_P0_7   =       0x0087
_IT0    =       0x0088
_IE0    =       0x0089
_IT1    =       0x008a
_IE1    =       0x008b
_TR0    =       0x008c
_TF0    =       0x008d
_TR1    =       0x008e
_TF1    =       0x008f
_P1_0   =       0x0090
_P1_1   =       0x0091
_P1_2   =       0x0092
_P1_3   =       0x0093
_P1_4   =       0x0094
_P1_5   =       0x0095
_P1_6   =       0x0096
_P1_7   =       0x0097
_RI     =       0x0098
_TI     =       0x0099
_RB8    =       0x009a
_TB8    =       0x009b
_REN    =       0x009c
_SM2    =       0x009d
_SM1    =       0x009e
_SM0    =       0x009f
_P2_0   =       0x00a0
_P2_1   =       0x00a1
_P2_2   =       0x00a2
_P2_3   =       0x00a3
_P2_4   =       0x00a4
_P2_5   =       0x00a5
_P2_6   =       0x00a6
_P2_7   =       0x00a7
_EX0    =       0x00a8
_ET0    =       0x00a9
_EX1    =       0x00aa
_ET1    =       0x00ab
```

```
_ES     =       0x00ac
_EA     =       0x00af
_P3_0   =       0x00b0
_P3_1   =       0x00b1
_P3_2   =       0x00b2
_P3_3   =       0x00b3
_P3_4   =       0x00b4
_P3_5   =       0x00b5
_P3_6   =       0x00b6
_P3_7   =       0x00b7
_RXD    =       0x00b0
_TXD    =       0x00b1
_INT0   =       0x00b2
_INT1   =       0x00b3
_T0     =       0x00b4
_T1     =       0x00b5
_WR     =       0x00b6
_RD     =       0x00b7
_PX0    =       0x00b8
_PT0    =       0x00b9
_PX1    =       0x00ba
_PT1    =       0x00bb
_PS     =       0x00bc
_P      =       0x00d0
_FL     =       0x00d1
_OV     =       0x00d2
_RS0    =       0x00d3
_RS1    =       0x00d4
_F0     =       0x00d5
_AC     =       0x00d6
_CY     =       0x00d7
;--------------------------------------------------------
; overlayable register banks
;--------------------------------------------------------
        .area REG_BANK_0        (REL,OVR,DATA)
        .ds 8
;--------------------------------------------------------
; internal ram data
;--------------------------------------------------------
        .area DSEG    (DATA)
_getint_buffer_1_1:
        .ds 4
_getint_key_1_1:
        .ds 1
;--------------------------------------------------------
; overlayable items in internal ram
;--------------------------------------------------------
        .area   OSEG    (OVR,DATA)
;--------------------------------------------------------
; indirectly addressable internal ram data
;--------------------------------------------------------
        .area ISEG    (DATA)
;--------------------------------------------------------
; absolute internal ram data
;--------------------------------------------------------
        .area IABS    (ABS,DATA)
        .area IABS    (ABS,DATA)
;--------------------------------------------------------
; bit data
;--------------------------------------------------------
        .area BSEG    (BIT)
;--------------------------------------------------------
```

```
; paged external ram data
;--------------------------------------------------------
        .area PSEG    (PAG,XDATA)
;--------------------------------------------------------
; external ram data
;--------------------------------------------------------
        .area XSEG    (XDATA)
;--------------------------------------------------------
; absolute external ram data
;--------------------------------------------------------
        .area XABS    (ABS,XDATA)
;--------------------------------------------------------
; external initialized ram data
;--------------------------------------------------------
        .area XISEG   (XDATA)
        .area HOME    (CODE)
        .area GSINIT0 (CODE)
        .area GSINIT1 (CODE)
        .area GSINIT2 (CODE)
        .area GSINIT3 (CODE)
        .area GSINIT4 (CODE)
        .area GSINIT5 (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area CSEG    (CODE)
;--------------------------------------------------------
; global & static initialisations
;--------------------------------------------------------
        .area HOME    (CODE)
        .area GSINIT  (CODE)
        .area GSFINAL (CODE)
        .area GSINIT  (CODE)
;--------------------------------------------------------
; Home
;--------------------------------------------------------
        .area HOME    (CODE)
        .area HOME    (CODE)
;--------------------------------------------------------
; code
;--------------------------------------------------------
        .area CSEG    (CODE)
;------------------------------------------------------------
;Allocation info for local variables in function 'ln2'
;------------------------------------------------------------
;v                         Allocated to registers r2
;i                         Allocated to registers r3
;------------------------------------------------------------
;       keypad.c:10: unsigned char ln2(unsigned char v) {
;       -----------------------------------------
;        function ln2
;       -----------------------------------------
_ln2:
        ar2 = 0x02
        ar3 = 0x03
        ar4 = 0x04
        ar5 = 0x05
        ar6 = 0x06
        ar7 = 0x07
        ar0 = 0x00
        ar1 = 0x01
        mov     r2,dpl
;       keypad.c:12: while(v) {
```

```
        mov     r3,#0x00
00101$:
        mov     a,r2
        jz      00103$
;       keypad.c:13: i++;
        inc     r3
;       keypad.c:14: v >>= 1;
        mov     a,r2
        clr     c
        rrc     a
        mov     r2,a
        sjmp    00101$
00103$:
;       keypad.c:16: return i-1;
        mov     a,r3
        dec     a
        mov     dpl,a
        ret
;------------------------------------------------------------
;Allocation info for local variables in function 'getchar'
;------------------------------------------------------------
;row                       Allocated to registers r2
;col                       Allocated to registers r3
;result                    Allocated to registers r3
;------------------------------------------------------------
;       keypad.c:19: char getchar() {
;       -----------------------------------------
;        function getchar
;       -----------------------------------------
_getchar:
;       keypad.c:20: unsigned char row = 1,col;
        mov     r2,#0x01
;       keypad.c:22: while(1) {
00110$:
;       keypad.c:23: KEY_PORT = ~row;
        mov     a,r2
        cpl     a
        mov     _P1,a
;       keypad.c:24: col = ~(KEY_PORT | 0x0F);
        mov     a,#0x0F
        orl     a,_P1
;       keypad.c:25: if (col) {
        cpl     a
        mov     r3,a
        jz      00105$
;       keypad.c:26: col = (col << 4) | (col >> 4);
        mov     a,r3
        swap    a
        mov     r3,a
;       keypad.c:27: result = char_table[(ln2(row)<<2)+ln2(col)];
        mov     dpl,r2
        push    ar3
        lcall   _ln2
        mov     r4,dpl
        pop     ar3
        mov     a,r4
        add     a,r4
        add     a,acc
        mov     r4,a
        mov     dpl,r3
        push    ar4
        lcall   _ln2
```

```
        mov     r3,dpl
        pop     ar4
        mov     a,r3
        add     a,r4
        mov     dptr,#_char_table
        movc    a,@a+dptr
        mov     r3,a
;       keypad.c:28: while ((~KEY_PORT & 0xF0));
00101$:
        mov     r4,_P1
        mov     r5,#0x00
        mov     a,r4
        cpl     a
        mov     r4,a
        mov     a,r5
        cpl     a
        mov     r5,a
        mov     a,r4
        anl     a,#0xF0
        jnz     00101$
;       keypad.c:29: return result;
        mov     dpl,r3
        ret
00105$:
;       keypad.c:31: if (row==8)
        cjne    r2,#0x08,00107$
;       keypad.c:32: row = 1;
        mov     r2,#0x01
        sjmp    00110$
00107$:
;       keypad.c:34: row *= 2;
        mov     a,r2
        add     a,r2
        mov     r2,a
        sjmp    00110$
;------------------------------------------------------------
;Allocation info for local variables in function 'getint'
;------------------------------------------------------------
;buffer                    Allocated with name '_getint_buffer_1_1'
;size                      Allocated to registers r2
;key                       Allocated with name '_getint_key_1_1'
;result                    Allocated to registers r3 r4
;------------------------------------------------------------
;       keypad.c:38: int getint() {
;       -----------------------------------------
;        function getint
;       -----------------------------------------
_getint:
;       keypad.c:39: char buffer[4], size = 0, key;
        mov     r2,#0x00
;       keypad.c:40: int result = 0;
        mov     r3,#0x00
        mov     r4,#0x00
;       keypad.c:41: do {
        mov     r5,#0x00
00107$:
;       keypad.c:42: key = getchar();
        push    ar2
        push    ar3
        push    ar4
        push    ar5
        lcall   _getchar
```

```
        mov     r6,dpl
        pop     ar5
        pop     ar4
        pop     ar3
        pop     ar2
;       keypad.c:43: if (key == 28)
        cjne    r6,#0x1C,00102$
;       keypad.c:44: return -1;
        mov     dptr,#0xFFFF
        ret
00102$:
;       keypad.c:45: if ((key >= '0') && (key <= '9') && (size<4)) {
        clr     c
        mov     a,r6
        xrl     a,#0x80
        subb    a,#0xb0
        jc      00108$
        mov     a,#(0x39 ^ 0x80)
        mov     b,r6
        xrl     b,#0x80
        subb    a,b
        jc      00108$
        mov     a,r5
        xrl     a,#0x80
        subb    a,#0x84
        jnc     00108$
;       keypad.c:46: buffer[size] = key - '0';
        mov     a,r5
        add     a,#_getint_buffer_1_1
        mov     r0,a
        mov     a,r6
        add     a,#0xd0
        mov     @r0,a
;       keypad.c:47: size++;
        inc     r5
        mov     ar2,r5
;       keypad.c:48: putchar(key);
        mov     dpl,r6
        push    ar2
        push    ar3
        push    ar4
        push    ar5
        push    ar6
        lcall   _putchar
        pop     ar6
        pop     ar5
        pop     ar4
        pop     ar3
        pop     ar2
00108$:
;       keypad.c:50: } while(key != 13);
        cjne    r6,#0x0D,00107$
;       keypad.c:51: for (key=0; key < size; key++)
        mov     _getint_key_1_1,#0x00
00110$:
        clr     c
        mov     a,_getint_key_1_1
        xrl     a,#0x80
        mov     b,r2
        xrl     b,#0x80
        subb    a,b
        jnc     00113$
```

```
;       keypad.c:52: result = result*10 + buffer[key];
        push    ar2
        mov     __mulint_PARM_2,r3
        mov     (__mulint_PARM_2 + 1),r4
        mov     dptr,#0x000A
        push    ar2
        lcall   __mulint
        mov     r6,dpl
        mov     r7,dph
        pop     ar2
        mov     a,_getint_key_1_1
        add     a,#_getint_buffer_1_1
        mov     r0,a
        mov     a,@r0
        mov     r5,a
        rlc     a
        subb    a,acc
        mov     r2,a
        mov     a,r5
        add     a,r6
        mov     r3,a
        mov     a,r2
        addc    a,r7
        mov     r4,a
;       keypad.c:51: for (key=0; key < size; key++)
        inc     _getint_key_1_1
        pop     ar2
        sjmp    00110$
00113$:
;       keypad.c:53: return result;
        mov     dpl,r3
        mov     dph,r4
        ret
        .area CSEG    (CODE)
        .area CONST   (CODE)
_char_table:
        .db #0x31
        .db #0x32
        .db #0x33
        .db #0x43
        .db #0x34
        .db #0x35
        .db #0x36
        .db #0x44
        .db #0x37
        .db #0x38
        .db #0x39
        .db #0x45
        .db #0x1C
        .db #0x30
        .db #0x0D
        .db #0x46
        .area XINIT   (CODE)
        .area CABS    (ABS,CODE)
```

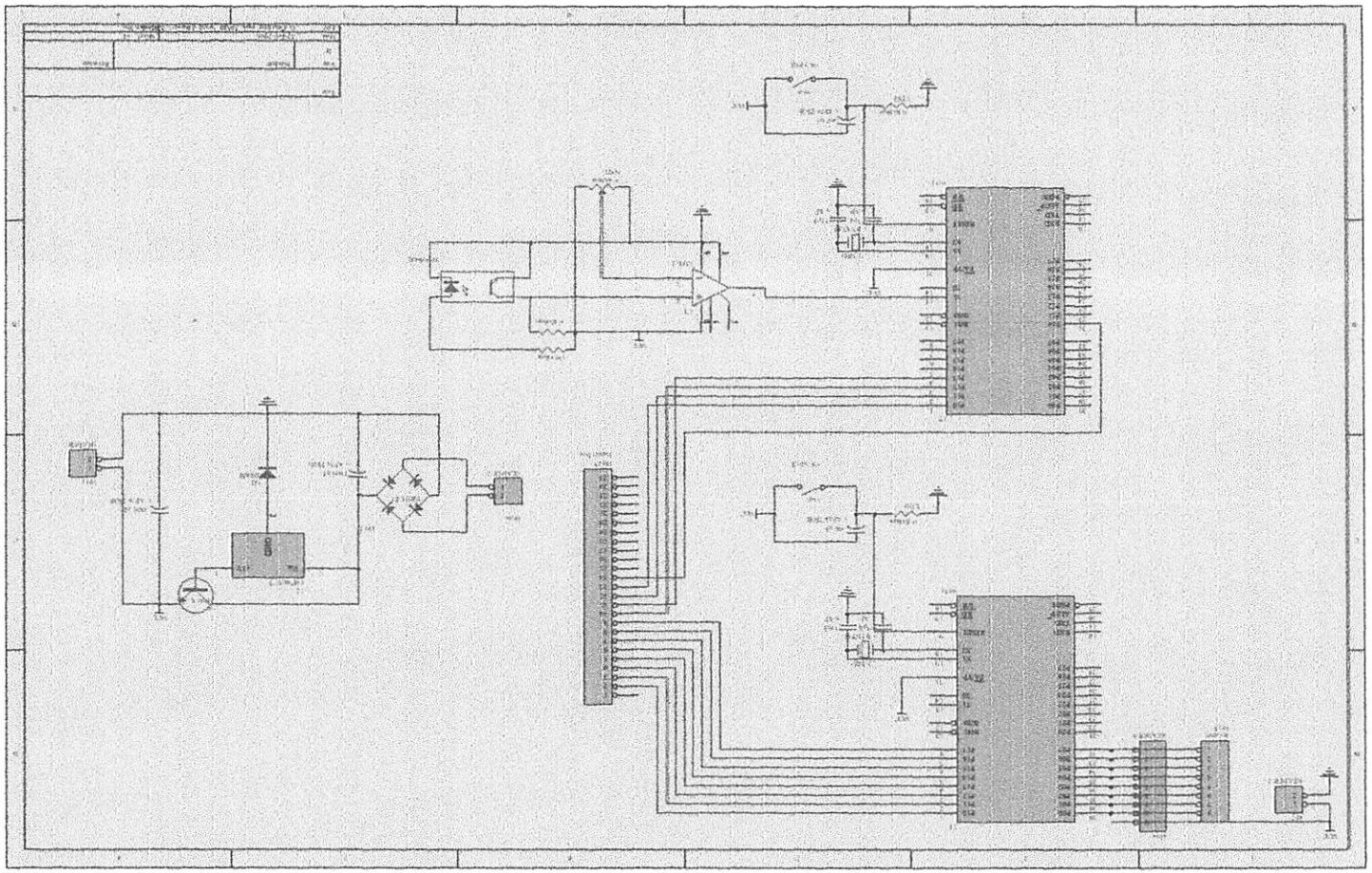
LISTING PROGRAM FUZZY

```
Unit Unit 1;

interface

uses
Window,Messages,SYsUntils,Variants,Classses,Graphics,Controls,Forms,
Dialogs,StdCtrls,TeEngine,Series,ExtCntrl,Teeprocs,Chart,ComCtrls;

type
TForm1 =class(TForm)
Chart1: Tchart;
Series1 : TLineSeries;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Botton1:TBotton;
Botton2:TBotton;
Timer1: TBotton;
SEries2: TBotton;
Timer1:TTimer;
Series2:TLineSeries;
Updown1: TUpDown;
GroupBox1TGroupBox;
Timer2:TTimer;
Edit4:TEdit;
Lbel6:TLabel;
Edit5:Tedit;
Label5:TLabel;
Label4:4TLabel;
Label7:TLabel;
Edit6:TEdit;
Edit7:TEdit;
Label8: TLabel;
CheckBox1;: TCheckBox;
procedure Timer1(sender: TOject);
procedure Botton1(sender: TOject);
procedure Botton2(sender: TOject);
procedure Updown1click(sender: TOject;Botton:TUDBtnType);
procedure Timer2Timer(sender: TOject);
procedure editChange(sender: TOject);
private
 {Private eclaration }
public
 { pulic declaratio }
end;

var
Form1: TForm1;
frek, rpm, i,T : integer;
rpm1 : integer;
rpm2 : array [1..200] of integer;
sem, sem1 : integer;


implementation
uses unit2;
{$R*.dfm}
function Out32(wAddr:word:bOut:byte):byte;stdcall;external'inpout32dll';
functin np32(wAddr:word):byte;stdcall;externa'inpout32.dll';

procedure TForm1.Timer1Timer(sender:TObjec);
```

```
var
 dfrek,error,j:integer;
 ESN,EN,EZ,EP,ESP : real;
 begin
 timer1.Enabled:=false;
 if rpm=0 then rpm:=100;
 //sem2:=rpm1;
 //rpm1:=rpm;
 if rpm>(frek*60+400)then rpm:=frek*60;
 ifrpm<=(frek*60-400)then rpm:=frek*60;
 rpm1=rpm;
 error:=strtoint(edit1.Tex)-rpm;
 if error<=-127 then error:=-127;
 if error>-64 then
 begin
  ESN:=1-((error+127)/63);
  EN:=((error+127)/63);
  EZ:=0;
  EP:=0:
  ESP:=0;
 end
 else if error<=0 then
 begin
  ESN:=0;
  EN:=1-((error+64)/64);
  EZ:=((error+64)/64);
  EP:=0;
  ESP:=0;
end
else if error<=64then
begin
 ESN:=0;
 EN:=0;
 EZ:=1-((errr)/64);
 EP:((error)/64);
 ESP:0;
end
else if error<=128 then
begin
  ESN:=0;
  EN:=0;
  EZ:=0;
  EP:=1-((error-64)/64);
  ESP:=((error-64)/64);
end
else if error>=128then
begin
 ESN:=0;
 EN:=0;
 EZ:=0;
 EP:=0;
 ESP:=1;
end;
dfrek:=round(ESN*2+EN*1+EZ*0-EP*1-ESP*2);
frek:=frek-dfrek;
if frek<=29 then frek:=29;
//if frek>=50 then frek:=50;
T:=round(1000000/(frek*6*23));
out32($378,T);
//rpm :=frek*30;
i:i+1;

series1.AddXY(i*0.05,rpm,",clRed);
series2.AddXY(i*0.05,strtoint(edit1.Tex),",clgreen);
if i<=200 then
begin
form2.ADOTable1.Edit;
form2.ADOTable1.RecNo:=i;
```

```
form2.ADOTable1.FiledByName('No').AsInteger:=1;
form2.ADOTable1.FiledByName('setpouint').AsInteger:=strtoint(edit1.Text);;
form2.ADOTable1.FiledByName('frek').AsInteger:=frek;
form2.ADOTable1.FiledByName('kec').AsInteger:=rpm1;
form2.ADOTable1.FiledByName('ns').AsInteger:=round(rpm/(1-0.06));
form2.ADOTable1.FiledByName('fs').AsInteger:=round(rpm/((1-0.06)*60));
form2.ADOTable1.Post;

end;
edit2.Text:=inttostr(frek);
edit3.Text:=inttostr(rpm);
edit6.Text:=inttostr(ound(rpm/((1-0.06)));
edit7.Text:=inttostr(ound(rpm/((1-0.06)*60)));
Timer1.Enabled:=true;

procedure TForm1.Button1Clik(Sender:TObject);
begin
 i:=0;
 rpm:=0;
 frek:=29;
 sem:=1;
 sem1:=1;
 series1.Clear;
 series2.Clear;
 timer1.Enabled:true;
 timer2.Enabled:true;
 from2.ADOTable1.Open;

end;

procedure TForm1.Button2Click(Seder:TObject);
var e1,e2,e3: integer;
begin
timer1.Enablcd:= false;
form2.show;
for i:=1 to 200 do
begin
if rpm2[i]=0 then rpm2[i+1]:=100;
if rpm2[i]=0 then rpm2[i+2]:100;
e1:=round (abs(rpm2[i]-strtoint(edit1.Text))*100/rpm2[i];
e2:=round (abs(rpm2[i+1]-strtoint(edit1.Text))*100/rpm2[i+1];
e1:=round (abs(rpm2[i+2]-strtoint(edit1.Text))*100/rpm2[i+2];
if(e 1<=5) and (e2<=5) and (e3<=5) then
begin
edit4.text:=inttostr(rpm2[i];
edit5.Text:=inttostr(i*50);
break;
end;
end;
end;


prosedure Tform1.UpDwn1clik(Sender:TObjct;;Botten:TUDBtnType);
pegin
  timer2.Enable:=false;
  if (inp32($379)and$08)=8 then
pegin
sem:=trunc((inp32($379) xor $80)/16);//16);
end;
if(inp32($379)and $80)=0 then
pegin sem 1:runc(inp32($379) xor $80) and $0F0);//and $0F0;
end;
sem:=sem+sem1;
if sem=0 then sem:=1
if checkbox1.Checked=true then tpm:=frek*60;
timer2.Enabled:true;
end;
prosedure Tform1.EDit1Change(sender: T object);
```

```
                                        program
egin
pdown1.position:strtoint(edi1.text);
nd;

nd.
```



# de KITS Application Note

## AN13 - Automatic Transmission with Encoder Meter and Display

Oleh: Tim IE

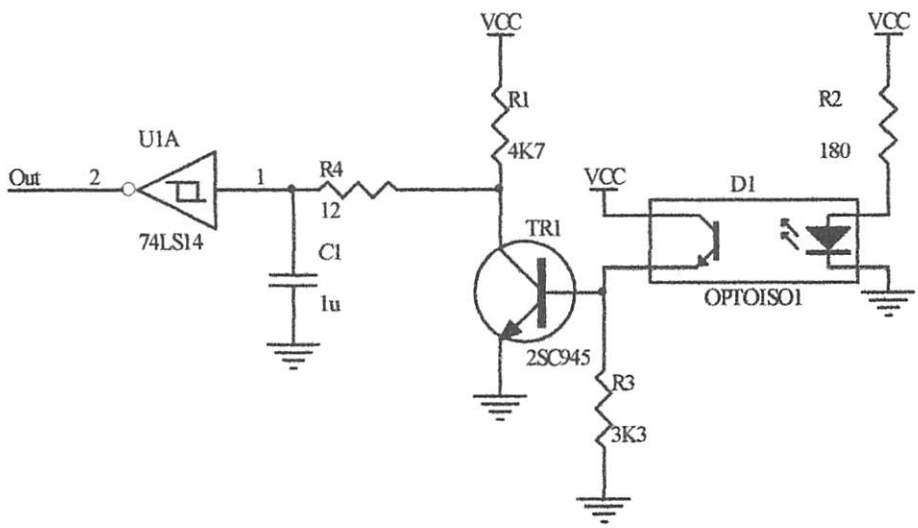
Aplikasi 'Automatic Transmission with Encoder Meter and Display' ini bertugas menghitung pulsa kecepatan motor dan menampilkannya di 7 segment. Modul yang digunakan adalah DT-51 MinSys Ver 3.0, DT-51 KND, de KITS SPC DC Motor (K6), Motor DC, dan rangkaian Sensor Kecepatan.

Hubungan antara DT-51 MinSys Ver 3.0 dan de KITS SPC DC Motor adalah sebagai berikut:

| de KITS SPC DC Motor | DT-51 MinSys Ver 3.0 Port C & Port 1 |
|---|---|
| SCL / J7 Pin 15 | Pin 15 (Port 1.6) |
| SDA / J7 Pin 16 | Pin 16 (Port 1.7) |

**Tabel 1**
**Hubungan DT-51 MinSys dengan de KITS SPC DC Motor**

Cara menghubungkan DT-51 KND dengan DT-51 MinSys dapat dilihat pada DT-51 KND User's Guide.
Cara menghubungkan DC motor dengan de KITS SPC DC Motor dapat dilihat pada Manual de KITS SPC DC Motor.
Rangkaian Sensor Kecepatan menggunakan OptoIsolator/OptoCoupler model "U" (860D) dapat dilihat pada gambar berikut:




**Gambar 1**
**Rangkaian Sensor Kecepatan**

Dengan dibantu lempeng lingkaran yang dilubangi, sensor kecepatan akan menghasilkan pulsa high ( ⎍⎍ ) jika terdapat lubang. Posisi sensor secara mekanis dapat dilihat pada gambar 2. Perlu diingat bahwa jumlah lubang yang dibuat akan mempengaruhi hasil tampilan di DT-51 KND. Makin banyak lubang maka pembacaan akan makin sering dan jika dikonversi ke RPM akan didapat hasil yang makin mendekati kondisi aslinya.

Dengan dipasang lampeng lingkaran yang dipasang sensor kecepatan akan mendapatkan bentuk pin ( "L" ) like tergabet lapang. Posisi sensor secara maksimal cobat dahal bagi dengan 5. Pasti dipasang pasma lapang lapang yang dipasi akan memengaruhi pasil tembasan di DT-51 KMD. Maksi penyak lapang maka pengaruhnya akan maksi sebuh dan like dikonversi ke PAM akan dipabet pasi yang maksi mendekati kondisi sebuh.



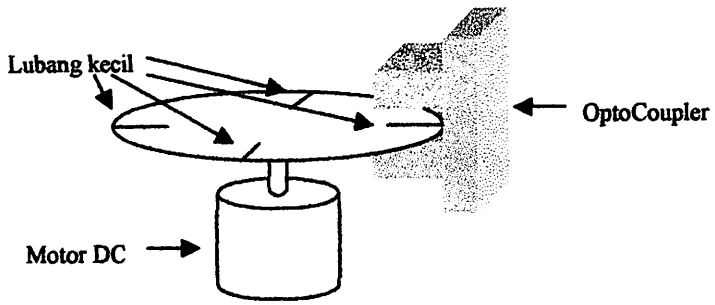
Aplikasi Analogic Transmission with Encoder Meter and Display, ini peruntuk mendapatkan bentuk kecepatan motor dan mentransmisikannya di 7 segment. Modul yang digunakan adalah DT-51 Minsys Ver 3.0, DT-51 KMD, dan KITS 89C DC Motor (kit)," Motor DC" dan rangkaian sensor kecepatan.

Hubungan antara DT-51 Minsys Ver 3.0 dan KITS 89C DC Motor adalah sebagai berikut:

| DT-51 Minsys Ver 3.0 | Port C & Port 1 |
|---|---|
| Motor | Port 1 (Port 1.6) |
| KITS 89C DC | P0 to P7 (Port 1.7) |
| SDA \ TX P0 to | SCL \ TX P0 to |

Tabel 1
Hubungan DT-51 Minsys dengan KITS 89C DC Motor

Cara mengunduhkan DT-51 KMD dengan DT-51 Minsys cobat dihasi bagi DT-51 KMD hasil Gulga.
Cara mengunduhkan DC motor dengan KITS 89C DC Motor cobat dihasi bagi Mainsi dengan KITS 89C DC Motor:
Rangkaian Sensor kecepatan menggunakan Optoisolator/Optocoupler model "U" (4600) cobat dihasi bagi dengan pelikat:




**Gambar 2**
**Posisi Mekanis**

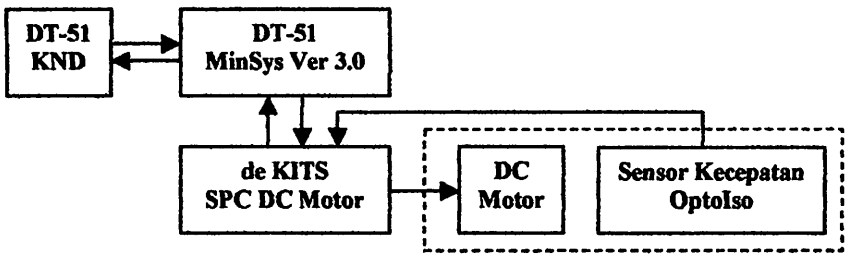
Hubungan antara de KITS SPC DC Motor dengan rangkaian sensor ini adalah sebagai berikut:

| Rangkaian Sensor | de KITS SPC DC Motor |
|---|---|
| Out | IN1 |

**Tabel 2**
**Hubungan de KITS SPC DC Motor dengan Rangkaian Sensor**

Setelah menghubungkan rangkaian dan menghubungkan supply tegangan yang tepat, download-lah program RPMDISP.HEX ke DT-51 MinSys Ver3.0.

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:




**Gambar 3**
**Blok Diagram AN13**

Rangkaian Sensor Kecepatan berfungsi untuk mendeteksi jumlah putaran per satuan waktu. Hal ini telah diakomodasi oleh de KITS SPC DC Motor pada IN1 untuk mendeteksi pulsa kecepatan motor.
DT-51 KND berfungsi sebagai input transmisi (Top, Brake, Up, Down, Neutral) pada keypad dan output display pada 7 segment. Penekanan TOP akan memutar motor pada kecepatan maksimum (tanpa PWM). Penekanan Up atau Down akan mengurangi atau menambah nilai PWM sebesar 10d. Penekanan N akan menghentikan putaran motor. Penekanan Brake akan menghentikan putaran motor secara cepat (layout dapat dilihat pada gambar 4).
Selain itu, DT-51 KND juga berfungsi untuk menampilkan hasil penghitungan pulsa kecepatan motor (dalam desimal). Setiap 1 detik tampilan pada 7 segment akan di-update. Tampilan hanya menyatakan jumlah pulsa bukan menyatakan kecepatan motor dalam RPM.

Jika ingin menghitung RPM, maka bisa dihitung dengan rumus:

$$RPM = \frac{\text{Hasil Tampilan}}{\text{Jumlah Lubang}} \times \frac{60 \text{ detik}}{\text{GateTime}}$$

Gambar 2
Posisi Mekanis

Hubungan antara de KITS SPC DC Motor dengan rangkaian sensor in adalah sebagai berikut:

| Rangkaian Sensor | de KITS SPC DC Motor |
|---|---|
| Out | IN1 |

Tabel 2
Hubungan da KITS SPC DC Motor dengan Rangkaian Sensor

Selain menghubungkan rangkaian dan menghubungkan supply tegangan yang lepat, downloadkan program RPMDISP.HEX ke DT-51 MinSys Ver3.0

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:




Gambar 3
Blok Diagram AN13

Rangkaian Sensor Kecepatan berfungsi untuk mendeteksi jumlah putaran per satuan waktu. Hal ini telah diskomodasi oleh de KITS SPC DC Motor pada IN1 untuk mendeteksi pulsa kecepatan motor.
DT-51 KND berfungsi sebagai input transmisi (Top, Brake, Up, Down, Neutral) pada keypad dan output display pada 7 segment. Penekanan TOP akan memutar motor pada kecepatan maksimum (tanpa PWM). Penekanan Up atau Down akan mengurangi atau menambah nilai PWM sebesar 10d. Penekanan N akan mengembalikan putaran motor. Penekanan Brake akan mengembalikan putaran motor secara cepat (layout dapat dilihat pada gambar 4).
Selain itu, DT-51 KND juga berfungsi untuk menampilkan hasil pengitungan pulsa kecepatan motor (dalam desimal). Setiap 1 detik tampilan pada 7 segment akan di-update. Tampilan hanya menyatakan jumlah pulsa bukan menyatakan kecepatan motor dalam RPM.

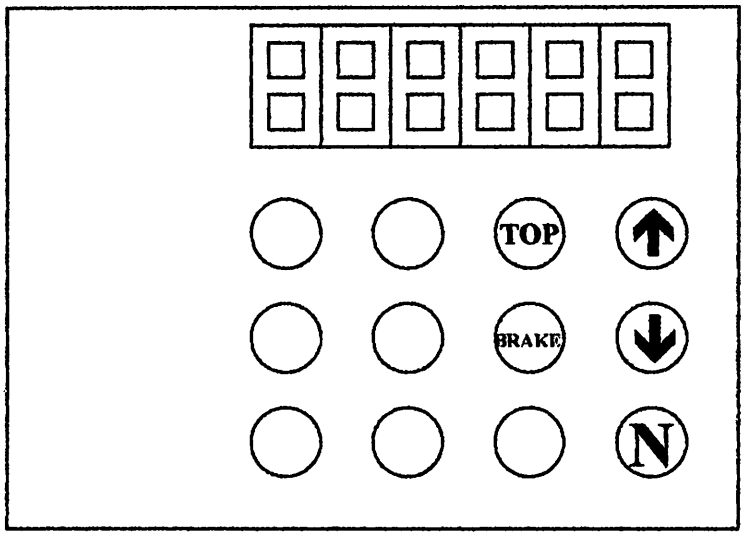
Jika ingin mengitung RPM, maka bisa dihitung dengan rumus:

$$RPM = \frac{\text{Hasil Tampilan}}{\text{Jumlah Lubang}} \times \frac{\text{60 detik}}{\text{DateTime}}$$

Pada RPMDISP.ASM digunakan GateTime 1 detik sehingga rumus menjadi:

$$RPM = \frac{\text{Hasil Tampilan}}{\text{Jumlah Lubang}} \times 60$$

Jika hasil tampilan sebesar 135 pulsa dengan jumlah lubang 16, berarti kecepatan putaran motor sekitar 506 RPM.




**Gambar 4**
**Alokasi Keypad DT-51 KND**

Pada RPM0ISP.ASM digunakan GateTime 1 detik sehingga rumus menjadi:

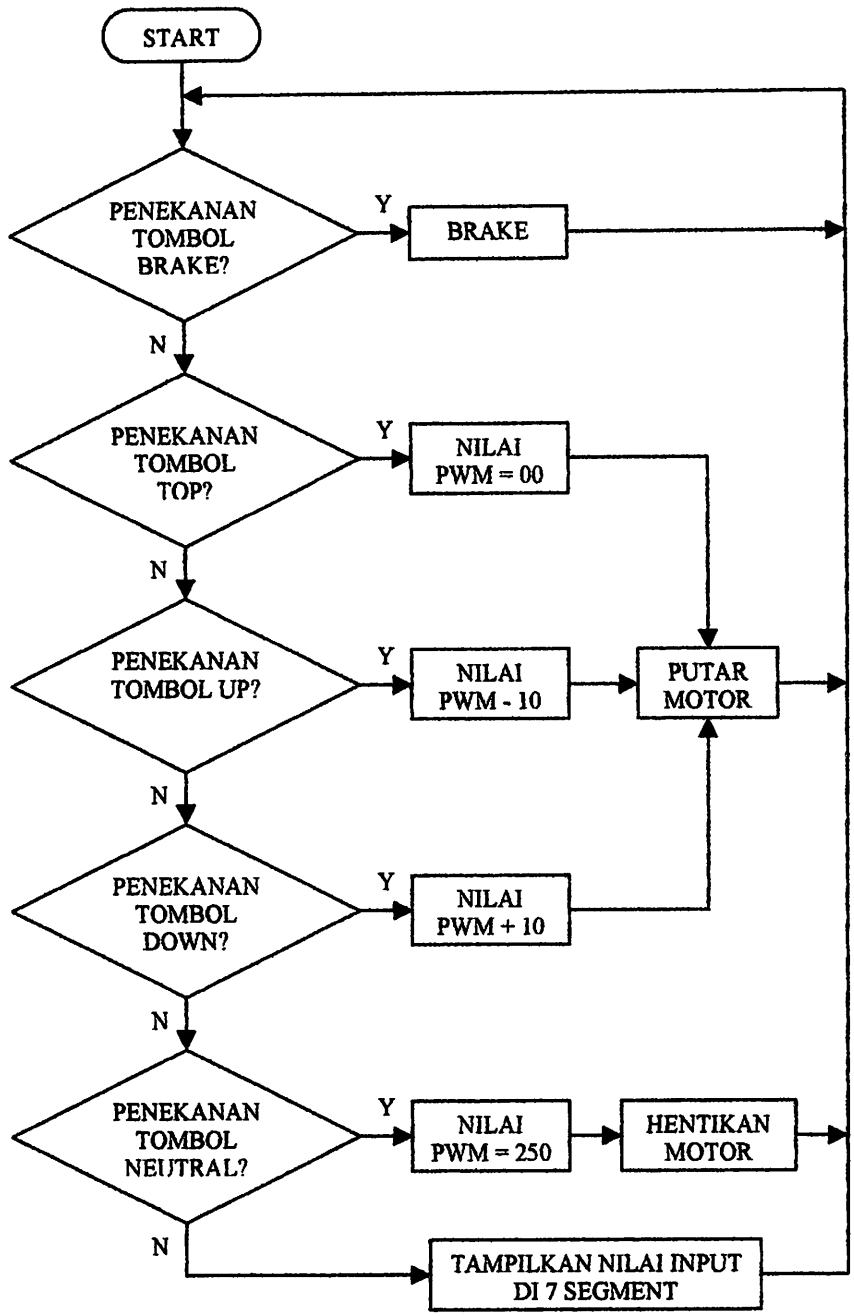
$$RPM = \frac{\text{Hasil Tampilan}}{\text{Jumlah Lubang}} \times 60$$

Jika hasil tampilan sebesar 135 pulsa dengan jumlah lubang 15, berarti kecepatan putaran motor sekitar 505 RPM.




Gambar 4
Aplikasi Keypad DT-61 KNB

Flowchart dari sistem ini adalah sebagai berikut:




**Gambar 5**
**Flowchart Program**

Listing program **RPMDISP.ASM** dapat dilihat pada AN13.Zip.

Berikut adalah sistem ini berjalan secara:




**Gambar 5**
**Flowchart Program**

Listing program RPMDISP.A5M dapat dilihat pada AN13.2.lp

## Features

- Compatible with MCS-51® Products
- 4K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)

## Description

he AT89S51 is a low-power, high-performance CMOS 8-bit microcontroller with 4K ytes of in-system programmable Flash memory. The device is manufactured using tmel's high-density nonvolatile memory technology and is compatible with the indus-y-standard 80C51 instruction set and pinout. The on-chip Flash allows the program emory to be reprogrammed in-system or by a conventional nonvolatile memory pro-rammer. By combining a versatile 8-bit CPU with in-system programmable Flash on a onolithic chip, the Atmel AT89S51 is a powerful microcontroller which provides a ighly-flexible and cost-effective solution to many embedded control applications.

he AT89S51 provides the following standard features: 4K bytes of Flash, 128 bytes of AM, 32 I/O lines, Watchdog timer, two data pointers, two 16-bit timer/counters, a five-ector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and ock circuitry. In addition, the AT89S51 is designed with static logic for operation own to zero frequency and supports two software selectable power saving modes. he Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and terrupt system to continue functioning. The Power-down mode saves the RAM con-nts but freezes the oscillator, disabling all other chip functions until the next external terrupt or hardware reset.

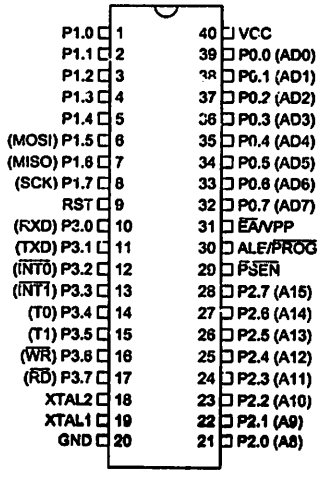
# 8-bit Microcontroller with 4K Bytes In-System Programmable Flash

## AT89S51

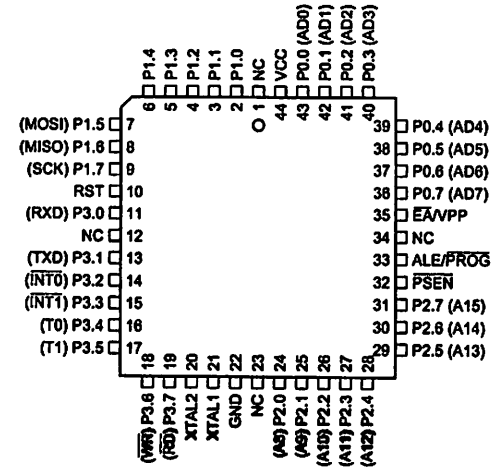
Rev. 2487A–10/01

## Pin Configurations

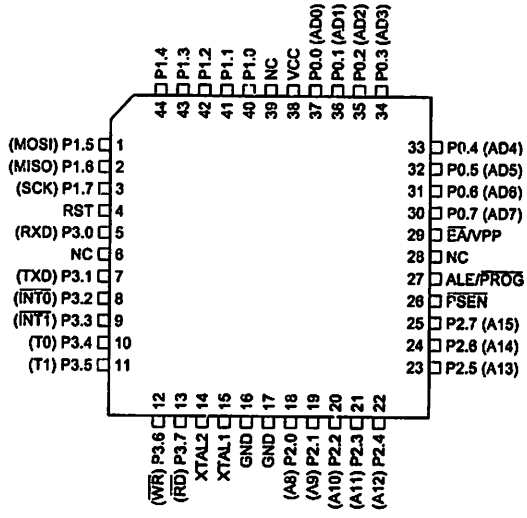
### PDIP



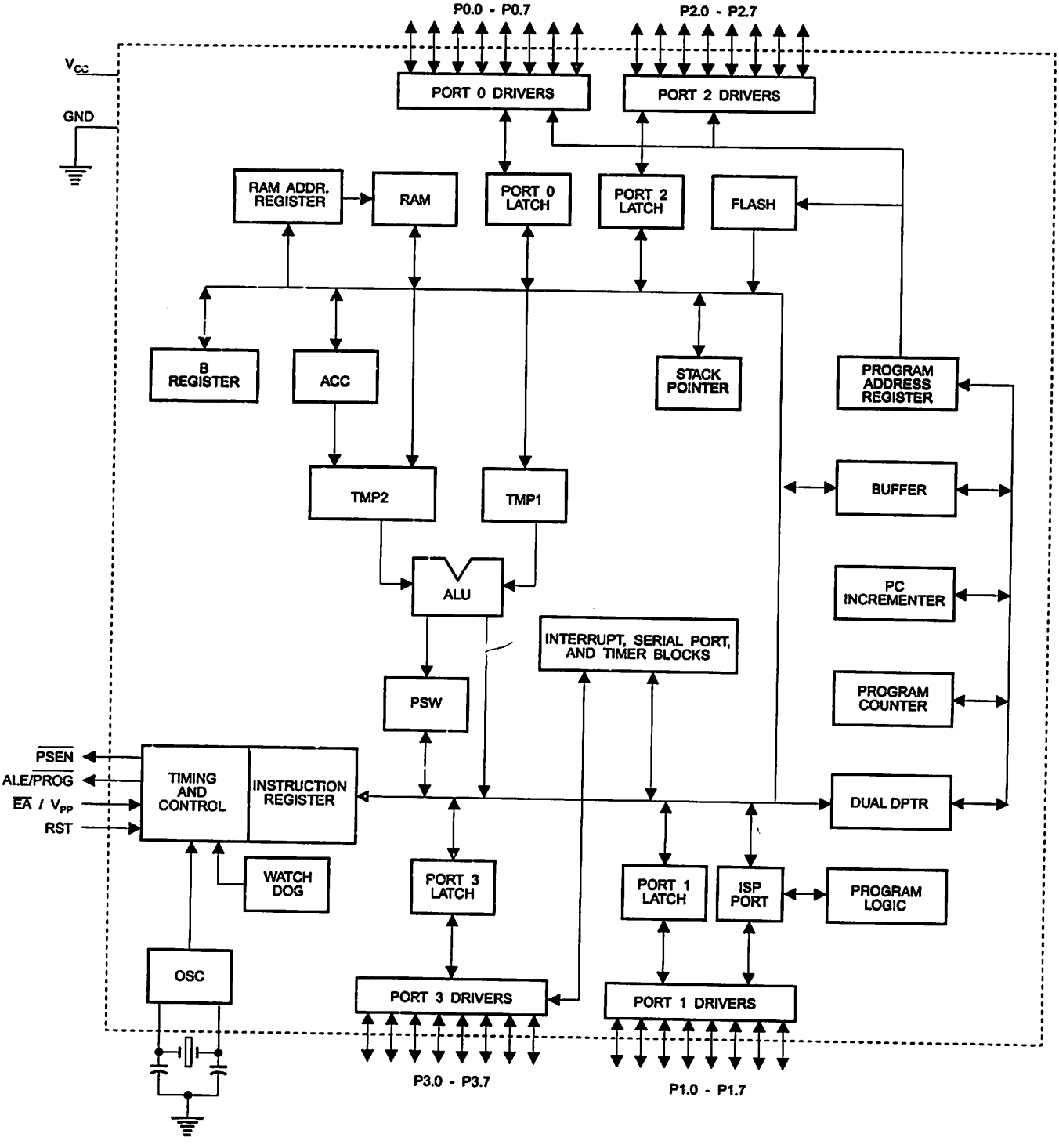

### PLCC

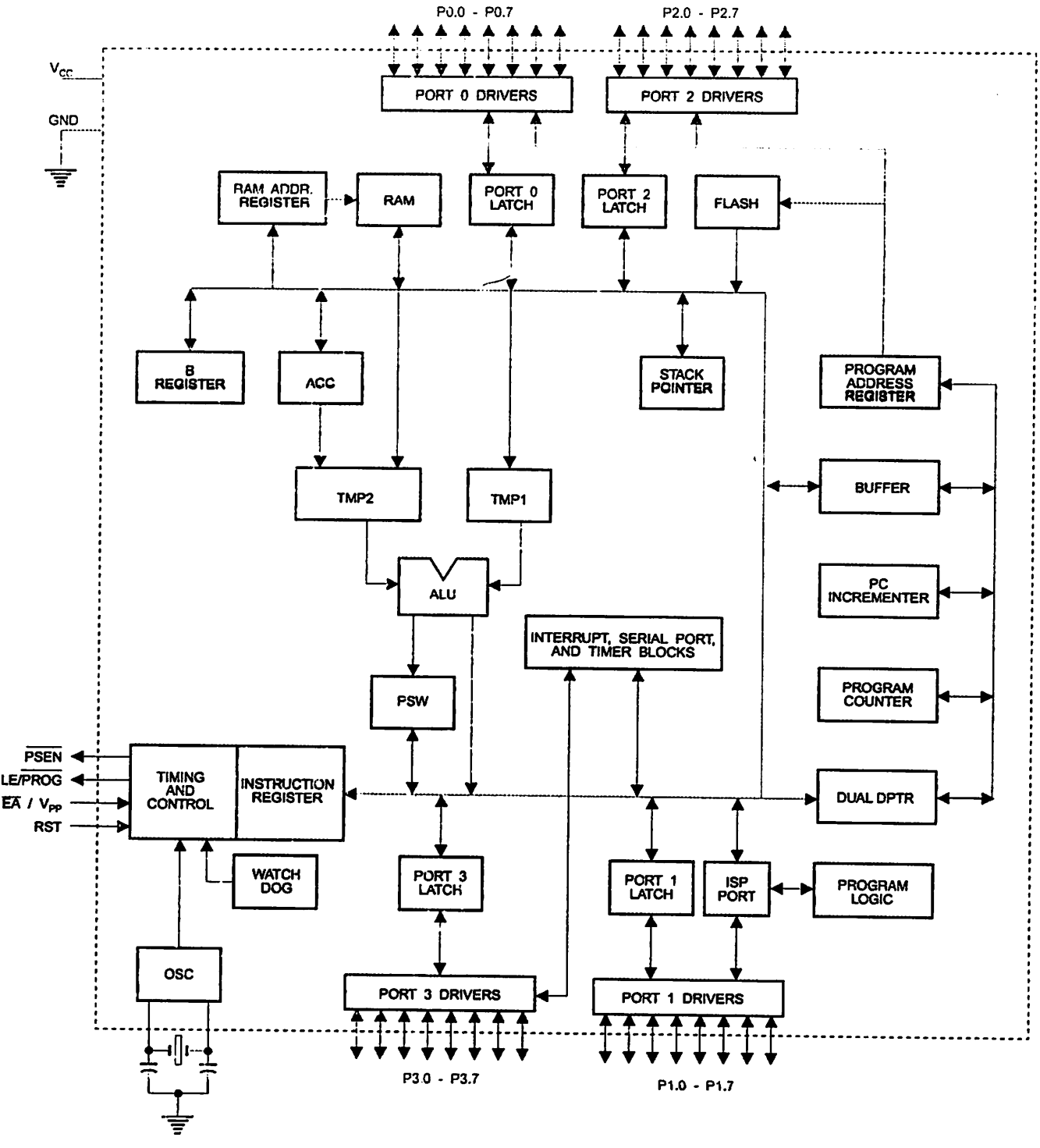



### TQFP



## Block Diagram




P0.0 - P0.7
P2.0 - P2.7
VCC
GND
PORT 0 DRIVERS
PORT 2 DRIVERS
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
FLASH
B REGISTER
ACC
STACK POINTER
PROGRAM ADDRESS REGISTER
TMP2
TMP1
BUFFER
ALU
PC INCREMENTER
INTERRUPT, SERIAL PORT, AND TIMER BLOCKS
PSW
PROGRAM COUNTER
PSEN
ALE/PROG
EA / VPP
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DUAL DPTR
WATCH DOG
PORT 3 LATCH
PORT 1 LATCH
ISP PORT
PROGRAM LOGIC
OSC
PORT 3 DRIVERS
PORT 1 DRIVERS
P3.0 - P3.7
P1.0 - P1.7

## in Description

### ...

...

### ort 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL [illegible]

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

### ort 1

Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 1 output buffers can [illegible] internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being

| Port Pin | Alternate Functions |
|---|---|
| P1.5 | MOSI (used for In-System Programming) |
| P1.6 | MISO (used for In-System Programming) |
| P1.7 | SCK (used for In-System Programming) |

### rt 2

Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 2 output buffers can [illegible] internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being

Port 2 emits the high-order address byte during fetches from external program memory and [illegible] application. Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external

Port 2 also receives the high-order address bits and some control signals during Flash pro-

### rt 3

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 3 output buffers can [illegible] internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being

Port 3 also serves the functions of various special features of the AT89S51, as shown in the

| Port Pin | Alternate Functions |
|---|---|
| P3.0 | RXD (serial input port) |
| P3.1 | TXD (serial output port) |
| P3.2 | INT0 (external interrupt 0) |
| P3.3 | INT1 (external interrupt 1) |
| P3.4 | T0 (timer 0 external input) |
| P3.5 | T1 (timer 1 external input) |
| P3.6 | WR (external data memory write strobe) |
| P3.7 | RD (external data memory read strobe) |

**S1**

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 98 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

**LE/PROG**

Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is inactive during MOVX and MOVC instructions. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution.

**SEN**

Program Store Enable (PSEN) is the read strobe to external program memory.

When the AT89S51 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

**A/VPP**

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset.

EA should be strapped to V

This pin also receives the 12 volt programming enable voltage (V during Flash

## pecial
## egisters

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

Table 1. AT89C51 SFR Map and Reset Values

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0F8H | | | | | | | | 0FFH |
| 0F0H | B<br>00000000 | | | | | | | 0F7H |
| 0E8H | | | | | | | | 0EFH |
| 0E0H | ACC<br>00000000 | | | | | | | 0E7H |
| 0D8H | | | | | | | | 0DFH |
| 0D0H | PSW<br>00000000 | | | | | | | 0D7H |
| 0C8H | | | | | | | | 0CFH |
| 0C0H | | | | | | | | 0C7H |
| 0B8H | IP<br>XX000000 | | | | | | | 0BFH |
| 0B0H | P3<br>11111111 | | | | | | | 0B7H |
| 0A8H | IE<br>0X000000 | | | | | | | 0AFH |
| 0A0H | P2<br>11111111 | | AUXR1<br>XXXXXXX0 | | | | WDTRST<br>XXXXXXXX | | 0A7H |
| 98H | SCON<br>00000000 | SBUF<br>XXXXXXXX | | | | | | | 9FH |
| 90H | P1<br>11111111 | | | | | | | | 97H |
| 88H | TCON<br>00000000 | TMOD<br>00000000 | TL0<br>00000000 | TL1<br>00000000 | TH0<br>00000000 | TH1<br>00000000 | AUXR<br>XXX00XX0 | | 8FH |
| 80H | P0<br>11111111 | SP<br>00000111 | DP0L<br>00000000 | DP0H<br>00000000 | DP1L<br>00000000 | DP1H<br>00000000 | | PCON<br>0XXX0000 | 87H |

User software should not write 1s to these unlisted locations, since they may be used in future

**Interrupt Registers:** The individual interrupt enable bits are in the IE register. Two priorities

Table 3. AUXR: Auxiliary Register

| AUXR | Address = 8EH | | | | | | | Reset Value = XXX00XX0B |
|---|---|---|---|---|---|---|---|---|
| Not Bit Addressable | | | | | | | | |
| Bit | – | – | – | WDIDLE | DISRTO | – | – | DISALE |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | |
|---|---|
| – | Reserved for future expansion |
| DISALE | Disable/Enable ALE |
| | DISALE<br>Operating Mode |
| | 0 ALE is emitted at a constant rate of 1/6 the oscillator frequency |
| | 1 ALE is active only during a MOVX or MOVC instruction |
| DISRTO | Disable/Enable Reset out |
| | DISRTO |
| | 0 Reset pin is driven High after WDT times out |
| | 1 Reset pin is input only |
| WDIDLE | Disable/Enable WDT in IDLE mode |
| | WDIDLE |
| | 0 WDT continues to count in IDLE mode |
| | 1 WDT halts counting in IDLE mode |

**Dual Data Pointer Registers:** To facilitate accessing both internal and external data memory, two banks of 16-bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR AUXR1 selects DP0 and DPS = 1 selects DP1. The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

**Power Off Flag:** The Power Off Flag (POF) is located at bit 4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and rest under software control and is not affected by reset.

**Table 3.** AUXR1: Auxiliary Register 1

AUXR1

Address = A2H

Reset Value = XXXXXXX0B

Not Bit Addressable

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | DPS |

– Reserved for future expansion

DPS Data Pointer Register Select

| DPS | |
|---|---|
| 0 | Selects DPTR Registers DP0L, DP0H |
| 1 | Selects DPTR Registers DP1L, DP1H |

## emory rganization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

## ogram Memory

If the $\overline{EA}$ pin is connected to GND, all program fetches are directed to external memory.

On the AT89S51, if $\overline{EA}$ is connected to $V_{CC}$, program fetches to addresses 0000H through FFFH are directed to internal memory and fetches to addresses 1000H through FFFFH are directed to external memory.

## ata Memory

The AT89S51 implements 128 bytes of on-chip RAM. The 128 bytes are accessible via direct and indirect addressing modes. Stack operations are examples of indirect addressing, so the 128 bytes of data RAM are available as stack space.

## atchdog mer ne-time ...ram with ...eset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 14-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

## sing the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT, the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC=1/FOSC. To make the best use of the WDT, it

should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

## WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt, which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S51 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode.

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S51 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and re-enter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count when it is back from IDLE.

## UART

The UART in the AT89S51 operates the same way as the UART in the AT89C51. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then '8051 Products'.

## Timer 0 and 1

Timer 0 and Timer 1 in the AT89S51 operate the same way as Timer 0 and Timer 1 in the AT89C51. For further information on the Timers, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then '8051 Product Overview'.

## Interrupts

The AT89S51 has a total of five interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), two timer interrupts (Timers 0 and 1), and the serial port interrupt. These interrupts are all shown in Figure 1.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 4 shows that bit position IE.6 is unimplemented. In the AT89S51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle.

**Table 4.** Interrupt Enable (IE) Register

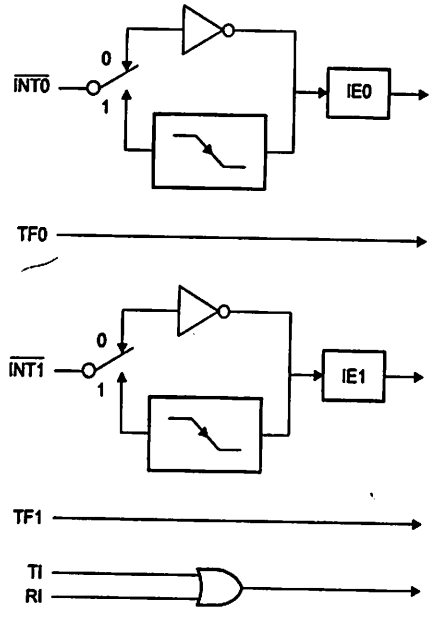
| (MSB) | | | | | | | (LSB) |
|---|---|---|---|---|---|---|---|
| EA | – | – | ES | ET1 | EX1 | ET0 | EX0 |

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

| Symbol | Position | Function |
|---|---|---|
| EA | IE.7 | Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| – | IE.6 | Reserved |
| – | IE.5 | Reserved |
| ES | IE.4 | Serial Port interrupt enable bit |
| ET1 | IE.3 | Timer 1 interrupt enable bit |
| EX1 | IE.2 | External interrupt 1 enable bit |
| ET0 | IE.1 | Timer 0 interrupt enable bit |
| EX0 | IE.0 | External interrupt 0 enable bit |
| User software should never write 1s to reserved bits, because they may be used in future AT89 products. | | |

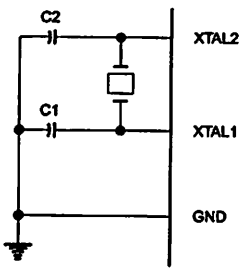
**Figure 1.** Interrupt Sources



## Oscillator Characteristics

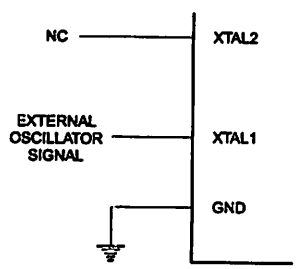
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 3. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

**Figure 2.** Oscillator Connections




Note: C1, C2 = 30 pF ± 10 pF for Crystals = 40 pF ± 10 pF for Ceramic Resonators

**Figure 3.** External Clock Drive Configuration




## Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special function registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

## Power-down Mode

In the Power-down mode, the oscillator is stopped, and the instruction that invokes Power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power-down mode is terminated. Exit from Power-down mode can be initiated either by a hardware reset or by activation of an enabled external interrupt into $\overline{INT0}$ or $\overline{INT1}$. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before $V_{CC}$ is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Table 5. Status of External Pins During Idle and Power-down Modes

| Mode | Program Memory | ALE | $\overline{PSEN}$ | PORT0 | PORT1 | PORT2 | PORT3 |
|---|---|---|---|---|---|---|---|
| Idle | Internal | 1 | 1 | Data | Data | Data | Data |
| Idle | External | 1 | 1 | Float | Data | Address | Data |
| Power-down | Internal | 0 | 0 | Data | Data | Data | Data |
| Power-down | External | 0 | 0 | Float | Data | Data | Data |

## Program Memory Lock Bits

The AT89S51 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

Table 6. Lock Bit Protection Modes

| Program Lock Bits | | | | |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | Protection Type |
| 1 | U | U | U | No program lock features |
| 2 | P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash memory is disabled |
| 3 | P | P | U | Same as mode 2, but verify is also disabled |
| 4 | P | P | P | Same as mode 3, but external execution is also disabled |

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

## Programming the Flash – Parallel Mode

The AT89S51 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S51 code memory array is programmed byte-by-byte.

**Programming Algorithm:** Before programming the AT89S51, the address, data, and control signals should be set up according to the Flash programming mode table and Figures 13 and 14. To program the AT89S51, take the following steps:

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 50 µs. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

**$\overline{Data}$ Polling:** The AT89S51 features $\overline{Data}$ Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

**Ready/$\overline{\text{Busy}}$:** The progress of byte programming can also be monitored by the RDY/$\overline{\text{BSY}}$ output signal. P3.0 is pulled low after ALE goes high during programming to indicate $\overline{\text{BUSY}}$. P3.0 is pulled high again when programming is done to indicate READY.

**Program Verify:** If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The status of the individual lock bits can be verified directly by reading them back.

**Reading the Signature Bytes:** The signature bytes are read by the same procedure as a normal verification of locations 000H, 100H, and 200H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(100H) = 51H indicates 89S51
(200H) = 06H

**Chip Erase:** In the parallel programming mode, a chip erase operation is initiated by using the proper combination of control signals and by pulsing ALE/$\overline{\text{PROG}}$ low for a duration of 200 ns - 500 ns.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 500 ms.

During chip erase, a serial read from any address location will return 00H at the data output.

## Programming the Flash – Serial Mode

The Code memory array can be programmed using the serial ISP interface while RST is pulled to $V_{CC}$. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before other operations can be executed. Before a reprogramming sequence can occur, a Chip Erase operation is required.

The Chip Erase operation turns the content of every memory location in the Code array into FFH.

Either an external system clock can be supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/16 of the crystal frequency. With a 33 MHz oscillator clock, the maximum SCK frequency is 2 MHz.

## Serial Programming Algorithm

To program and verify the AT89S51 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

   If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 33 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 16.
3. The Code array is programmed one byte at a time in either the Byte or Page mode. The write cycle is self-timed and typically takes less than 0.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction that returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal device operation.

Power-off sequence (if needed):

Set XTAL1 to "L" (if a crystal is not used).

Set RST to "L".

Turn $V_{CC}$ power off.

**$\overline{\text{Data}}$ Polling:** The $\overline{\text{Data}}$ Polling feature is also available in the serial mode. In this mode, during a write cycle an attempted read of the last byte written will result in the complement of the MSB of the serial output byte on MISO.

## erial ogramming struction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 8 on page 18.

## rogramming terface – arallel Mode

Every code byte in the Flash array can be programmed by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

**All major programming vendors offer worldwide support for the Atmel microcontroller series.** Please contact your local programming vendor for the appropriate software revision.

ble 7. Flash Programming Modes

| lode | $V_{CC}$ | RST | $\overline{\text{PSEN}}$ | ALE/ $\overline{\text{PROG}}$ | $\overline{\text{EA}}$/ $V_{PP}$ | P2.6 | P2.7 | P3.3 | P3.6 | P3.7 | P0.7-0 Data | P2.3-0 Address | P1.7-0 Address |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vrite Code Data | 5V | H | L | (2) ⌐‾⌐ | 12V | L | H | H | H | H | $D_{IN}$ | A11-8 | A7-0 |
| ead Code Data | 5V | H | L | H | H | L | L | L | H | H | $D_{OUT}$ | A11-8 | A7-0 |
| Vrite Lock Bit 1 | 5V | H | L | (3) ⌐‾⌐ | 12V | H | H | H | H | H | X | X | X |
| Vrite Lock Bit 2 | 5V | H | L | (3) ⌐‾⌐ | 12V | H | H | H | L | L | X | X | X |
| Vrite Lock Bit 3 | 5V | H | L | (3) ⌐‾⌐ | 12V | H | L | H | H | L | X | X | X |
| Read Lock Bits , 2, 3 | 5V | H | L | H | H | H | H | L | H | L | P0.2, P0.3, P0.4 | X | X |
| Chip Erase | 5V | H | L | (1) ⌐‾⌐ | 12V | H | L | H | L | L | X | X | X |
| Read Atmel ID | 5V | H | L | H | H | L | L | L | L | L | 1EH | 0000 | 00H |
| Read Device ID | 5V | H | L | H | H | L | L | L | L | L | 51H | 0001 | 00H |
| Read Device ID | 5V | H | L | H | H | L | L | L | L | L | 06H | 0010 | 00H |

otes:
1. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Chip Erase.
2. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Write Code Data.
3. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Write Lock Bits.
4. RDY/$\overline{\text{BSY}}$ signal is output on P3.0 during programming.
5. X = don't care.

**Figure 4.** Programming the Flash Memory (Parallel Mode)

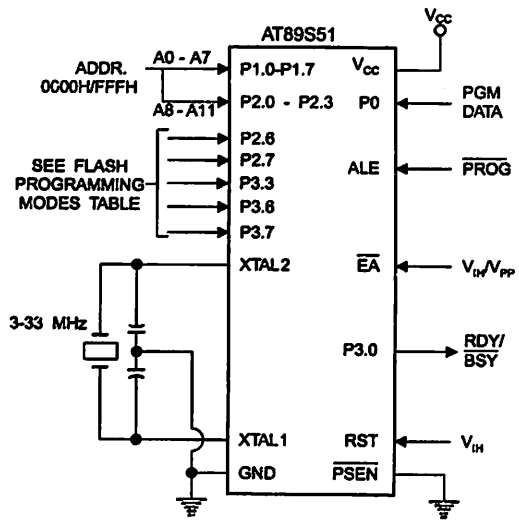
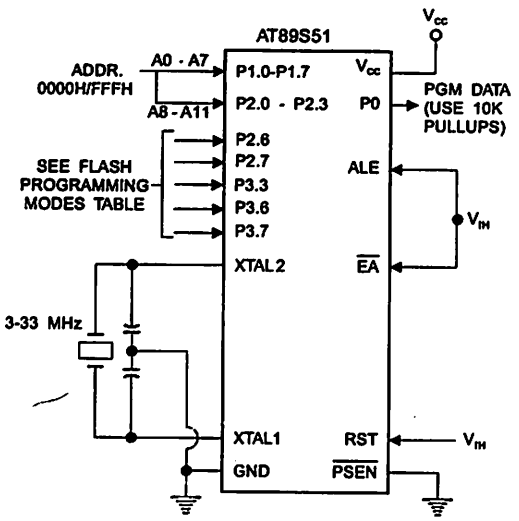



**Figure 5.** Verifying the Flash Memory (Parallel Mode)

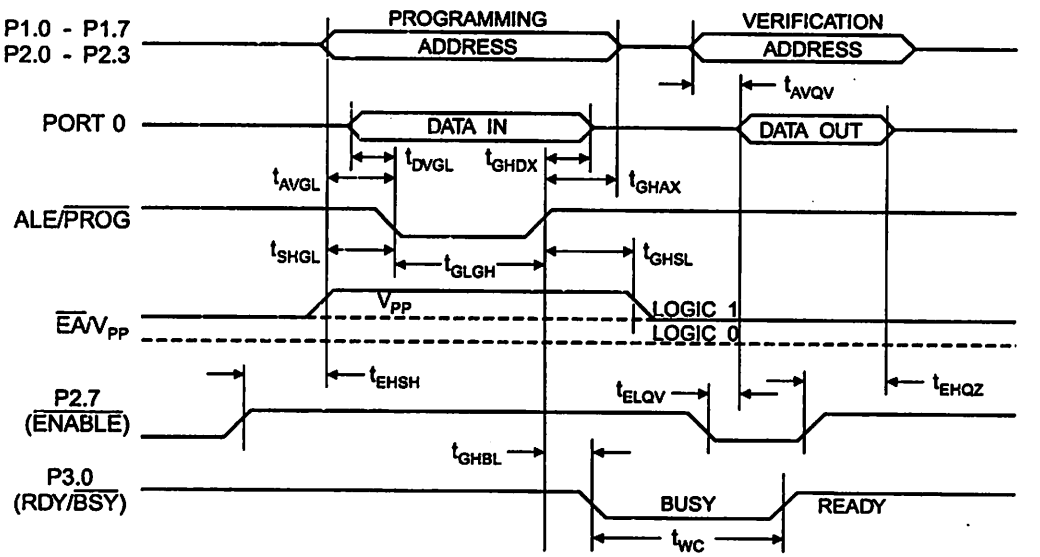


## ash Programming and Verification Characteristics (Parallel Mode)

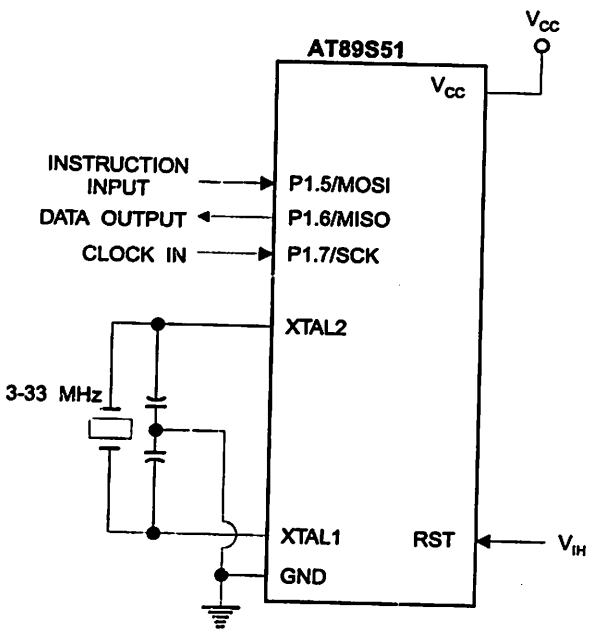
= 20°C to 30°C, $V_{CC}$ = 4.5 to 5.5V

| mbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| PP | Programming Supply Voltage | 11.5 | 12.5 | V |
| P | Programming Supply Current | | 10 | mA |
| C | $V_{CC}$ Supply Current | | 30 | mA |
| $t_{CLCL}$ | Oscillator Frequency | 3 | 33 | MHz |
| AVGL | Address Setup to $\overline{PROG}$ Low | $48t_{CLCL}$ | | |
| GHAX | Address Hold After $\overline{PROG}$ | $48t_{CLCL}$ | | |
| DVGL | Data Setup to $\overline{PROG}$ Low | $48t_{CLCL}$ | | |
| GHDX | Data Hold After $\overline{PROG}$ | $48t_{CLCL}$ | | |
| EHSH | P2.7 ($\overline{ENABLE}$) High to $V_{PP}$ | $48t_{CLCL}$ | | |
| SHGL | $V_{PP}$ Setup to $\overline{PROG}$ Low | 10 | | µs |
| GHSL | $V_{PP}$ Hold After $\overline{PROG}$ | 10 | | µs |
| GLGH | $\overline{PROG}$ Width | 0.2 | 1 | µs |
| AVQV | Address to Data Valid | | $48t_{CLCL}$ | |
| ELQV | $\overline{ENABLE}$ Low to Data Valid | | $48t_{CLCL}$ | |
| EHQZ | Data Float After $\overline{ENABLE}$ | 0 | $48t_{CLCL}$ | |
| GHBL | $\overline{PROG}$ High to $\overline{BUSY}$ Low | | 1.0 | µs |
| WC | Byte Write Cycle Time | | 50 | µs |

ure 6. Flash Programming and Verification Waveforms – Parallel Mode



gure 7. Flash Memory Serial Downloading




## ash Programming and Verification Waveforms – Serial Mode

ure 8. Serial Programming Waveforms

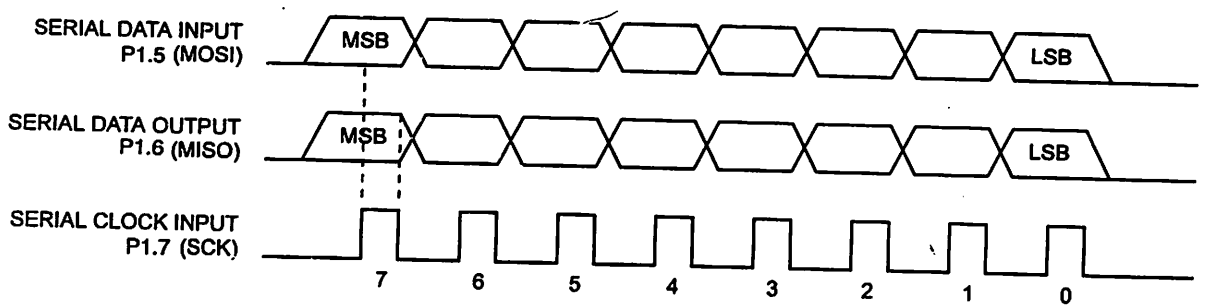


**Table 8.** Serial Programming Instruction Set

| Instruction | Instruction Format Byte 1 | Byte 2 | Byte 3 | Byte 4 | Operation |
|---|---|---|---|---|---|
| Programming Enable | 1010 1100 | 0101 0011 | xxxx xxxx | xxxx xxxx<br>0110 1001 (Output) | Enable Serial Programming while RST is high |
| Chip Erase | 1010 1100 | 100x xxxx | xxxx xxxx | xxxx xxxx | Chip Erase Flash memory array |
| Read Program Memory (Byte Mode) | 0010 0000 | xxxx A11 A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0 | Read data from Program memory in the byte mode |
| Write Program Memory (Byte Mode) | 0100 0000 | xxxx A11 A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0 | Write data to Program memory in the byte mode |
| Write Lock Bits[2] | 1010 1100 | 1110 00 B1 B2 | xxxx xxxx | xxxx xxxx | Write Lock bits. See Note (2). |
| Read Lock Bits | 0010 0100 | xxxx xxxx | xxxx xxxx | xx LB3 LB2 LB1 xx | Read back current status of the lock bits (a programmed lock bit reads back as a "1") |
| Read Signature Bytes[1] | 0010 1000 | xxx A5 A4 A3 A2 A1 | A0 xxx xxxx | Signature Byte | Read Signature Byte |
| Read Program Memory (Page Mode) | 0011 0000 | xxxx A11 A10 A9 A8 | Byte 0 | Byte 1...<br>Byte 255 | Read data from Program memory in the Page Mode (256 bytes) |
| Write Program Memory (Page Mode) | 0101 0000 | xxxx A11 A10 A9 A8 | Byte 0 | Byte 1...<br>Byte 255 | Write data to Program memory in the Page Mode (256 bytes) |

Notes: 1. The signature bytes are not readable in Lock Bit Modes 3 and 4.

2. B1 = 0, B2 = 0 → Mode 1, no lock protection
   B1 = 0, B2 = 1 → Mode 2, lock bit 1 activated
   B1 = 1, B2 = 0 → Mode 3, lock bit 2 activated
   B1 = 1, B1 = 1 → Mode 4, lock bit 3 activated

   } Each of the lock bits needs to be activated sequentially before Mode 4 can be executed.

After Reset signal is high, SCK should be low for at least 64 system clocks before it goes high to clock in the enable data bytes. No pulsing of Reset signal is necessary. SCK should be no faster than 1/16 of the system clock at XTAL1.

For Page Read/Write, the data always starts from byte 0 to 255. After the command byte and upper address byte are latched, each byte thereafter is treated as data until all 256 bytes are shifted in/out. Then the next instruction will be ready to be decoded.

## Serial Programming Characteristics

**Figure 9.** Serial Programming Timing

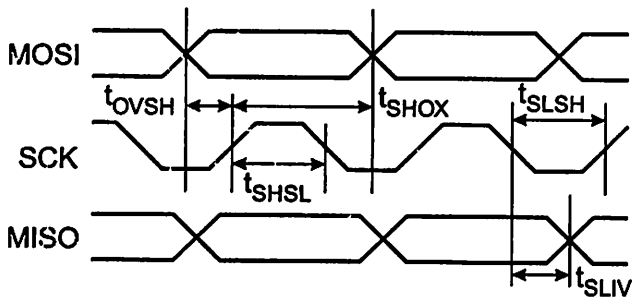



**Table 9.** Serial Programming Characteristics, $T_A$ = -40° C to 85° C, $V_{CC}$ = 4.0 - 5.5V (Unless Otherwise Noted)

| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | | 33 | MHz |
| $t_{CLCL}$ | Oscillator Period | 30 | | | ns |
| $t_{SHSL}$ | SCK Pulse Width High | 8 $t_{CLCL}$ | | | ns |
| $t_{SLSH}$ | SCK Pulse Width Low | 8 $t_{CLCL}$ | | | ns |
| $t_{OVSH}$ | MOSI Setup to SCK High | $t_{CLCL}$ | | | ns |
| $t_{SHOX}$ | MOSI Hold after SCK High | 2 $t_{CLCL}$ | | | ns |
| $t_{SLIV}$ | SCK Low to MISO Valid | 10 | 16 | 32 | ns |
| $t_{ERASE}$ | Chip Erase Instruction Cycle Time | | | 500 | ms |
| $t_{SWC}$ | Serial Byte Write Cycle Time | | | 64 $t_{CLCL}$ + 400 | µs |

## Absolute Maximum Ratings*

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage on Any Pin with Respect to Ground | -1.0V to +7.0V |
| Maximum Operating Voltage | 6.6V |
| DC Output Current | 15.0 mA |

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## DC Characteristics

The values shown in this table are valid for $T_A$ = -40°C to 85°C and $V_{CC}$ = 4.0V to 5.5V, unless otherwise noted.

| Symbol | Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | (Except $\overline{EA}$) | -0.5 | 0.2 $V_{CC}$-0.1 | V |
| $V_{IL1}$ | Input Low Voltage ($\overline{EA}$) | | -0.5 | 0.2 $V_{CC}$-0.3 | V |
| $V_{IH}$ | Input High Voltage | (Except XTAL1, RST) | 0.2 $V_{CC}$+0.9 | $V_{CC}$+0.5 | V |
| $V_{IH1}$ | Input High Voltage | (XTAL1, RST) | 0.7 $V_{CC}$ | $V_{CC}$+0.5 | V |
| $V_{OL}$ | Output Low Voltage[1] (Ports 1,2,3) | $I_{OL}$ = 1.6 mA | | 0.45 | V |
| $V_{OL1}$ | Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$) | $I_{OL}$ = 3.2 mA | | 0.45 | V |
| $V_{OH}$ | Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$) | $I_{OH}$ = -60 µA, $V_{CC}$ = 5V ±10% | 2.4 | | V |
| | | $I_{OH}$ = -25 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -10 µA | 0.9 $V_{CC}$ | | V |
| $V_{OH1}$ | Output High Voltage (Port 0 in External Bus Mode) | $I_{OH}$ = -800 µA, $V_{CC}$ = 5V ±10% | 2.4 | | V |
| | | $I_{OH}$ = -300 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -80 µA | 0.9 $V_{CC}$ | | V |
| $I_{IL}$ | Logical 0 Input Current (Ports 1,2,3) | $V_{IN}$ = 0.45V | | -50 | µA |
| $I_{TL}$ | Logical 1 to 0 Transition Current (Ports 1,2,3) | $V_{IN}$ = 2V, $V_{CC}$ = 5V ±10% | | -650 | µA |
| $I_{LI}$ | Input Leakage Current (Port 0, $\overline{EA}$) | 0.45 < $V_{IN}$ < $V_{CC}$ | | ±10 | µA |
| RRST | Reset Pulldown Resistor | | 50 | 300 | KΩ |
| $C_{IO}$ | Pin Capacitance | Test Freq. = 1 MHz, $T_A$ = 25°C | | 10 | pF |
| $I_{CC}$ | Power Supply Current | Active Mode, 12 MHz | | 25 | mA |
| | | Idle Mode, 12 MHz | | 6.5 | mA |
| | Power-down Mode[2] | $V_{CC}$ = 5.5V | | 50 | µA |

Notes:
1. Under steady state (non-transient) conditions, $I_{OL}$ must be externally limited as follows:
   Maximum $I_{OL}$ per port pin: 10 mA
   Maximum $I_{OL}$ per 8-bit port:
   Port 0: 26 mA  Ports 1, 2, 3: 15 mA
   Maximum total $I_{OL}$ for all output pins: 71 mA
   If $I_{OL}$ exceeds the test condition. $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum $V_{CC}$ for Power-down is 2V.

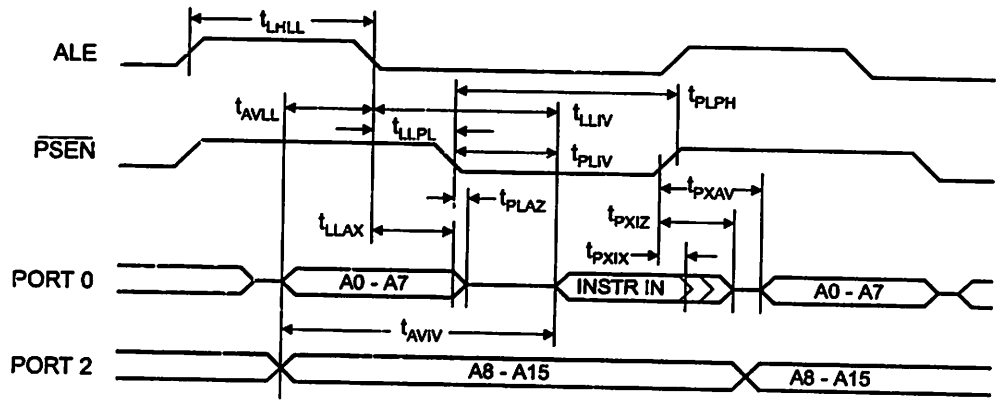
## C Characteristics

der operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other tputs = 80 pF.

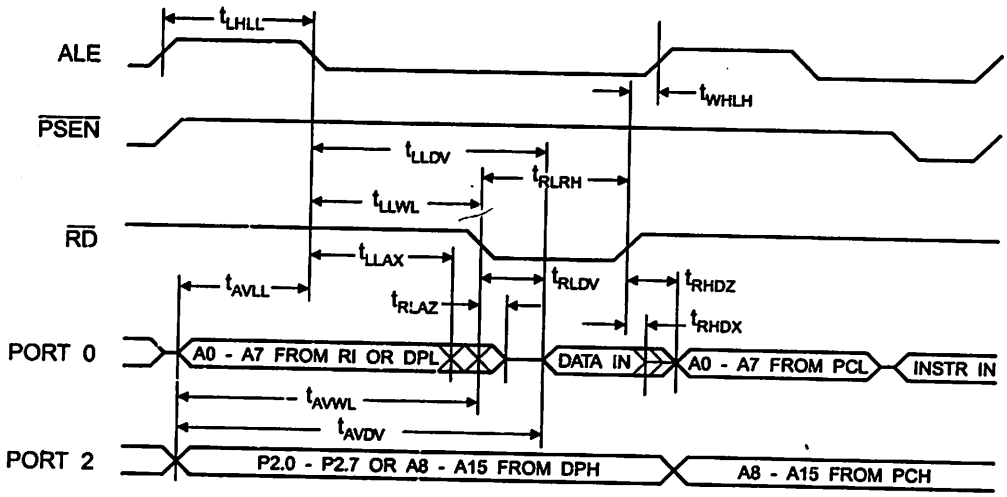
## xternal Program and Data Memory Characteristics

| ymbol | Parameter | 12 MHz Oscillator | | Variable Oscillator | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | | | 0 | 33 | MHz |
| HLL | ALE Pulse Width | 127 | | $2t_{CLCL}$-40 | | ns |
| AVLL | Address Valid to ALE Low | 43 | | $t_{CLCL}$-25 | | ns |
| LAX | Address Hold After ALE Low | 48 | | $t_{CLCL}$-25 | | ns |
| LIV | ALE Low to Valid Instruction In | | 233 | | $4t_{CLCL}$-65 | ns |
| LPL | ALE Low to $\overline{PSEN}$ Low | 43 | | $t_{CLCL}$-25 | | ns |
| PLPH | $\overline{PSEN}$ Pulse Width | 205 | | $3t_{CLCL}$-45 | | ns |
| PLIV | $\overline{PSEN}$ Low to Valid Instruction In | | 145 | | $3t_{CLCL}$-60 | ns |
| PXIX | Input Instruction Hold After $\overline{PSEN}$ | 0 | | 0 | | ns |
| PXIZ | Input Instruction Float After $\overline{PSEN}$ | | 59 | | $t_{CLCL}$-25 | ns |
| PXAV | $\overline{PSEN}$ to Address Valid | 75 | | $t_{CLCL}$-8 | | ns |
| AVIV | Address to Valid Instruction In | | 312 | | $5t_{CLCL}$-80 | ns |
| PLAZ | $\overline{PSEN}$ Low to Address Float | | 10 | | 10 | ns |
| RLRH | $\overline{RD}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| WLWH | $\overline{WR}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| RLDV | $\overline{RD}$ Low to Valid Data In | | 252 | | $5t_{CLCL}$-90 | ns |
| RHDX | Data Hold After $\overline{RD}$ | 0 | | 0 | | ns |
| RHDZ | Data Float After $\overline{RD}$ | | 97 | | $2t_{CLCL}$-28 | ns |
| LLDV | ALE Low to Valid Data In | | 517 | | $8t_{CLCL}$-150 | ns |
| AVDV | Address to Valid Data In | | 585 | | $9t_{CLCL}$-165 | ns |
| LLWL | ALE Low to $\overline{RD}$ or $\overline{WR}$ Low | 200 | 300 | $3t_{CLCL}$-50 | $3t_{CLCL}$+50 | ns |
| AVWL | Address to $\overline{RD}$ or $\overline{WR}$ Low | 203 | | $4t_{CLCL}$-75 | | ns |
| QVWX | Data Valid to $\overline{WR}$ Transition | 23 | | $t_{CLCL}$-30 | | ns |
| QVWH | Data Valid to $\overline{WR}$ High | 433 | | $7t_{CLCL}$-130 | | ns |
| WHQX | Data Hold After $\overline{WR}$ | 33 | | $t_{CLCL}$-25 | | ns |
| RLAZ | $\overline{RD}$ Low to Address Float | | 0 | | 0 | ns |
| WHLH | $\overline{RD}$ or $\overline{WR}$ High to ALE High | 43 | 123 | $t_{CLCL}$-25 | $t_{CLCL}$+25 | ns |

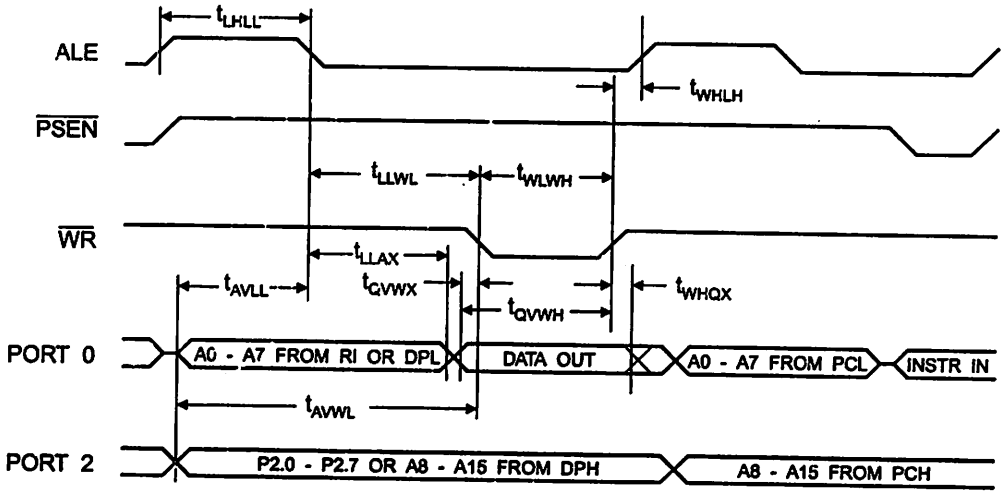
## External Program Memory Read Cycle



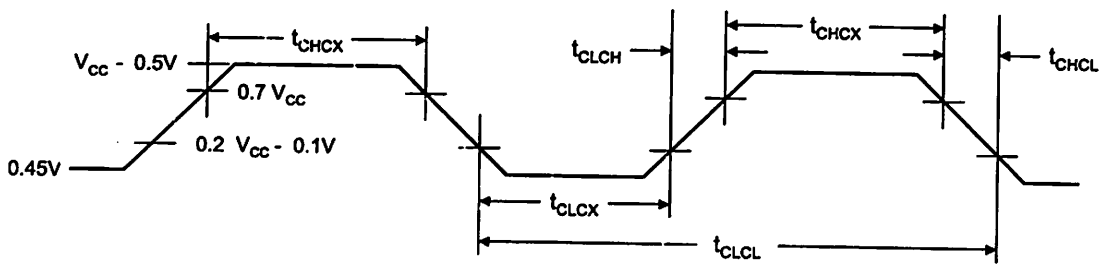

## External Data Memory Read Cycle



## External Data Memory Write Cycle




## External Clock Drive Waveforms




## External Clock Drive

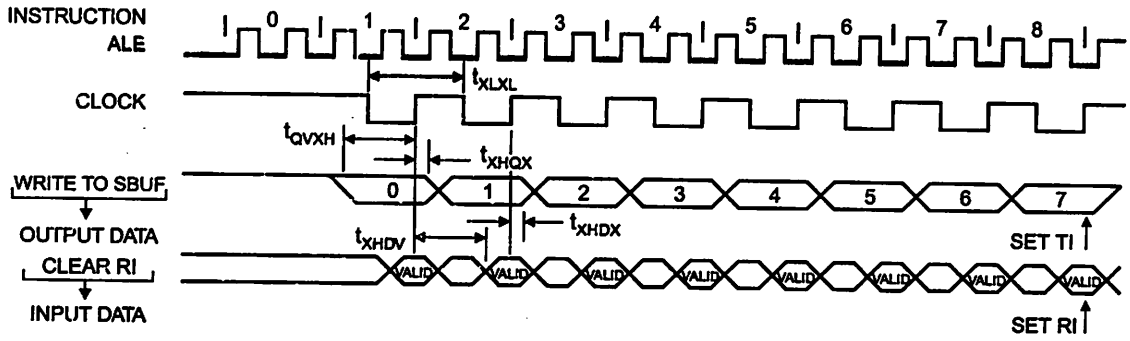
| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | 33 | MHz |
| $t_{CLCL}$ | Clock Period | 30 | | ns |
| $t_{CHCX}$ | High Time | 12 | | ns |
| $t_{CLCX}$ | Low Time | 12 | | ns |
| $t_{CLCH}$ | Rise Time | | 5 | ns |
| $t_{CHCL}$ | Fall Time | | 5 | ns |

## Serial Port Timing: Shift Register Mode Test Conditions

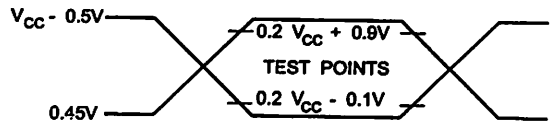
The values in this table are valid for $V_{CC}$ = 4.0V to 5.5V and Load Capacitance = 80 pF.

| Symbol | Parameter | 12 MHz Osc | | Variable Oscillator | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{XLXL}$ | Serial Port Clock Cycle Time | 1.0 | | $12t_{CLCL}$ | | µs |
| $t_{QVXH}$ | Output Data Setup to Clock Rising Edge | 700 | | $10t_{CLCL}$-133 | | ns |
| $t_{XHQX}$ | Output Data Hold After Clock Rising Edge | 50 | | $2t_{CLCL}$-80 | | ns |
| $t_{XHDX}$ | Input Data Hold After Clock Rising Edge | 0 | | 0 | | ns |
| $t_{XHDV}$ | Clock Rising Edge to Input Data Valid | | 700 | | $10t_{CLCL}$-133 | ns |

## Shift Register Mode Timing Waveforms

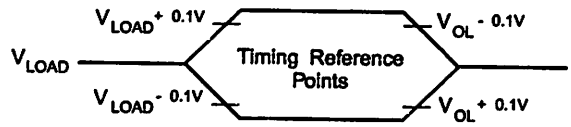



## AC Testing Input/Output Waveforms(1)




Note: 1. AC Inputs during testing are driven at $V_{CC}$ - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at $V_{IH}$ min. for a logic 1 and $V_{IL}$ max. for a logic 0.

## Float Waveforms(1)




Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded $V_{OH}/V_{OL}$ level occurs.

## Ordering Information

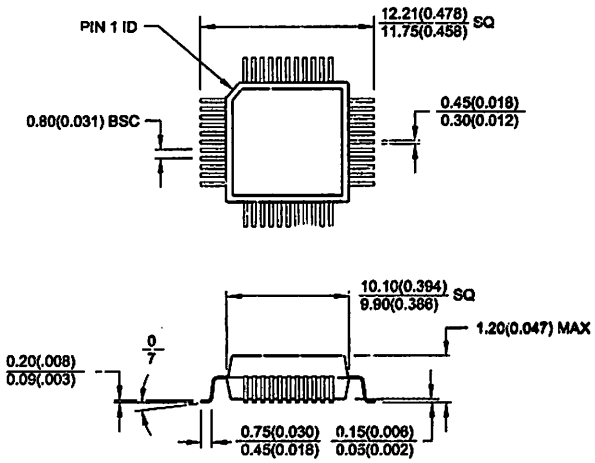
| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 24 | 4.0V to 5.5V | AT89S51-24AC<br>AT89S51-24JC<br>AT89S51-24PC | 44A<br>44J<br>40P6 | Commercial (0°C to 70°C) |
| | | AT89S51-24AI<br>AT89S51-24JI<br>AT89S51-24PI | 44A<br>44J<br>40P6 | Industrial (-40°C to 85°C) |

= Preliminary Availability

| | Package Type |
|---|---|
| A | 44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP) |
| J | 44-lead, Plastic J-leaded Chip Carrier (PLCC) |
| P6 | 40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP) |

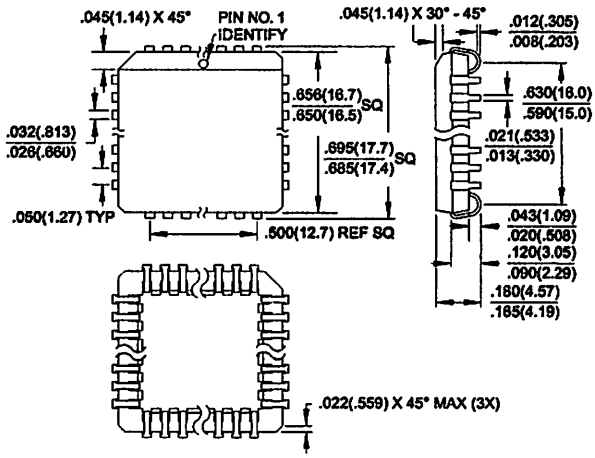
# Packaging Information

**44A**, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
Dimensions in Millimeters and (Inches)*

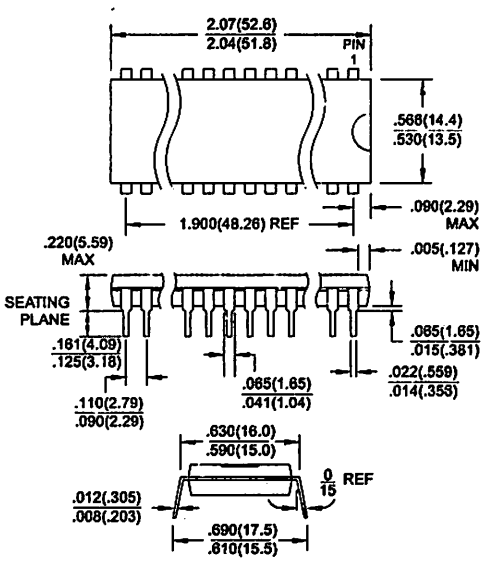



*Controlling dimension: millimeters

**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)




**40P6**, 40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AC



## Atmel Headquarters

*Corporate Headquarters*
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

*Europe*
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

*Asia*
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

*Japan*
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

## Atmel Product Operations

*Atmel Colorado Springs*
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

*Atmel Grenoble*
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

*Atmel Heilbronn*
Theresienstrasse 2
POB 3535
D-74025 Heilbronn, Germany
TEL (49) 71 31 67 25 94
FAX (49) 71 31 67 24 23

*Atmel Nantes*
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 0 2 40 18 18 18
FAX (33) 0 2 40 18 19 60

*Atmel Rousset*
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

*Atmel Smart Card ICs*
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

---

*e-mail*
literature@atmel.com

*Web Site*
http://www.atmel.com



2487A–10/01/xM