SISTEM PENGATURAN ADAPTIF PADA MOTOR ASINKRON 3 PHASA MENGGUNAKAN KOMPUTER SEBAGAI UNIT PENGONTROL

TUGAS AKHIR

Disusun Oleh :

RIO SUPRIAGA

292 220 2006

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
1996

SISTEM PENGATURAN ADAPTIF PADA MOTOR ASINKRON 3 PHASA MENGGUNAKAN KOMPUTER SEBAGAI UNIT PENGONTROL

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar
Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. HARMANI SUHARDJO)	(Ir. HENDRA KUSUMA)
NIP. 130 368 611	NIP. 131 846 104

SURABAYA
MARET, 1996

Sesungguhnya kami menurunkan kepada engkau, kitab (Al Qur'an) yang pasti benarnya siapa yang dapat petunjuk, maka itu untuk dirinya sendiri dan siapa yang sesat, maka dia merugi sendiri. Dan engkau bukan penjaga mereka. (Q.S. Az-Zumar : 41)

ABSTRAK

ABSTRAK

Dewasa ini perkembangan teknologi digital sangat pesat, yang meliputi seluruh bidang dalam kehidupan manusia. Di dunia industri teknologi digital membawa perubahan yang sangat besar dalam hal pengontrolan dan otomatisasi yang bertujuan untuk meningkatkan kualitas dan efisiensi dalam proses produksi. Perkembangan yang sangat pesat tersebut disebabkan oleh keunggulan-keunggulan yang ditawarkan pada sistem digital dan fleksibilitas yang tinggi dengan adanya software pendukungnya.

Salah satu aplikasi teknologi digital di bidang industri adalah pengaturan kecepatan motor induksi 3 fasa secara PWM menggunakan adaptif kontrol. Dengan metoda ini, motor dikontrol secara digital oleh μP 8088 minimum sistem yang dikomunikasikan dengan komputer. Pengontrolan tersebut mencakup frekuensi yang harus diberikan ke motor, besarnya tegangan, pengaturan kecepatan dan respon terhadap terjadinya perubahan parameter. Frekuensi yang diberikan ke motor diatur dengan jalan memberikan data faktor pengali pada rangkaian pengali frekuensi. Pengaturan tegangan dilakukan dengan jalan memilih data yang tersimpan di Eprom, sedangkan pengaturan respon motor dilakukan dengan mengirim data hasil identifikasi ke PC untuk diolah dan hasilnya dikirim kembali ke 8088 minimum sistem untuk menentukan tegangan dan frekuesi yang harus diberikan ke motor.

Proses adaptif yang dilakukan oleh komputer dirancang untuk putaran motor antara 550 - 2500 Rpm, dengan pengaturan kecepatan dan respon motor secara PWM menggunakan adaptif kontrol tersebut diharapkan diperoleh kecepatan dan respon motor induksi 3 fasa seperti yang dikehendaki sesuai dengan penerapannya.

Al Qur'an ini adalah penjelasan (yang lengkap) bagi manusia, supaya mereka mendapat peringatan dan memahami bahwasanya Dia-lah Tuhan Yang Maha Tunggal dan supaya orang-orang yang berakal mengerti. (Q.S. Ibrahim:52)

KATA PENGANTAR

KATA PENGANTAR

Atas berkat rahmat dan hidayah Allah SWT, maka penulis berhasil menyelesaikan perencanaan dan pembuatan alat serta penyusunan naskah Tugas Akhir yang berjudul :

SISTEM PENGATURAN ADAPTIF PADA MOTOR ASINKRON 3 PHASA MENGGUNAKAN KOMPUTER SEBAGAI UNIT PENGONTROL

Tugas akhir ini adalah merupakan salah satu syarat yang harus ditempuh oleh setiap mahasiswa untuk meraih gelar kesarjanaan di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam melaksanakan Tugas Akhir ini penulis banyak mendapatkan bantuan, bimbingan dan dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis menyampaikan rasa terima kasih sebesar - besarnya kepada :

- Ir. Harmani S, selaku Dosen Pembimbing I yang telah memberikan bimbingan dan pengarahan kepada penulis dalam perencanaan dan

pembuatan alat serta penyusunan naskah Tugas Akhir ini

- Ir. Hendra K, selaku Dosen Pembimbing II yang telah banyak memberikan bimbingan dan penarahan kepada penulis dalam perencanaan dan pembuatan alat serta penyusunan naskah Tugas akhir ini.
- Ayahanda dan Ibunda, yang telah memberikan segenap pengorbanan, motivasi dan do'a hingga terselesaikannya Tugas akhir ini.
- Kedua saudara saya yang telah begitu sabar memberikan dorongan serta do'a selama pengerjaan tugas akhir ini.
- Rekan - rekan di Laboratorium Medika, Laboratorium Elektronika, Laboratorium RL, dan Laboratorium Mikro Teknik Elektro ITS.

Akhir kata, penulis berharap semoga segala sesuatu yang telah dihasilkan dalam pelaksanaan Tugas Akhir ini dapat bermanfaat dan memberikan sumbangan ilmu pengetahuan serta memberikan kesejahteraan bagi umat manusia.

Surabaya, Februari 1995

Penulis

Sesungguhnya dalam penciptaan langit dan bumi, dan silih bergantinya malam dan siang terdapat tanda-tanda bagi orang-orang yang berakal. (Q.S. Al Imran:190)

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Katakan : "Sesungguhnya shalatku, ibadahku, hidupku dan matiku, hanyalah untuk Allah, Tuhan semesta alam." (Q.S. Al An'aam:162)

BAB I
PENDAHULUAN

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kehadiran komputer dalam kehidupan manusia telah membawa perubahan yang sangat besar dalam berbagai segi kehidupan terutama dalam bidang industri. Otomatisasi dan komputerisasi berkembang sangat pesat, hal ini bertujuan untuk meningkatkan kualitas hasil produksi dan efisiensi dalam proses produksi. Kondisi tersebut sangat ditunjang oleh berkembangnya ilmu pengetahuan yang langsung diaplikasikan dengan komputer sehingga tercipta peralatan-peralatan yang dikontrol oleh komputer.

Salah satu bagian peralatan yang dapat dikontrol oleh komputer adalah motor listrik yang dipakai sebagai tenaga penggerak dalam dunia industri. Pemakaian motor listrik tersebut sangat luas sekali, baik pada alat-alat ringan maupun alat berat yang perlu ketelitian tinggi ataupun yang tidak. Untuk peralatan yang memerlukan ketelitian tinggi diperlukan motor listrik dengan karakteristik dan respon khusus, karena itu diperlukan pengaturan yang khusus pula pada motor listrik penggeraknya. Dengan pengaturan tersebut maka dapat diperoleh karakteristik yang tepat untuk sistem secara keseluruhan. Untuk mendapatkan karakteristik dan respon tersebut salah satu hal yang dapat dilakukan adalah dengan mengaplikasikan sistem kontrol adaptif.

Sistem kontrol adaptif adalah sistem kontrol yang secara kontinyu dan otomatis mengukur karakteristik dinamik plant, kemudian membadingkan karakteristik yang terukur tersebut dengan karakteristik dinamik yang diinginkan dan menggunakan selisih yang terjadi untuk mengubah parameter-parameter sistem yang dikontrol. Dengan kemampuannya tersebut, maka sistem kontrol adaptif mampu mempertahankan performansi optimal dari plant yang dikontrol tanpa terpengaruh oleh perubahan- perubahan yang terjadi pada lingkungannya.

1.2. Permasalahan

Dalam dunia industri, peralatan mekanik selalu menggunakan motor listrik sebagai komponen penggeraknya. Untuk peralatan tertentu dibutuhkan karakteristik dan respon yang tertentu pula, dimana hal tersebut berhubungan langsung dengan proses produksi dan hasil produksi. Jika karakteristik dan respon peralatan yang digunakan tidak sesuai dengan yang diperlukan, maka akan menghambat proses produksi dan berakibat fatal pada hasil produksi. Sejalan dengan berkembangnya teknologi, maka dunia industripun berkembang dengan pesat, hal ini mengakibatkan peralatan yang digunakan semakin banyak dan komplek sehingga koordinasi dan pengendalian peralatan juga menjadi semakin rumit.

Berkaiatan dengan hal tersebut, maka pada tugas akhir ini akan dibuat alat yang mampu mengatur kecepatan putar, karakteristik serta respon dari motor induksi 3 fasa. Dengan perkembangan teknologi yang semakin canggih terutama dalam sistem digital maka pengendalian motor

induksi bisa lebih modern yaitu menggunakan mikroprosesor 8088 yang dikomunikasikan dengan komputer PC sebagai unit pengontrol, dengan demikian memungkinkan pengendalian sistem yang terpusat.

1.3. Pembatasan Masalah

Pada tugas akhir ini direncanakan dan dibuat alat untuk mengatur kecepatan putar motor induksi 3 fasa dengan menggunakan sistem kontrol Adaptif. Dengan sistem ini maka karakteristik dan respon motor terhadap perubahan beban bisa ditentukan sesuai dengan kriteria yang diperlukan. Untuk keperluan pembebanan tersebut, maka sistem akan disimulasikan pada motor dc sebagai bebannya. Perencanaan alat dibatasi pada pembangkit PWM, pengali frekuensi, rangkaian masukan / keluaran (I/O port) untuk papan kunci, tampilan dan data pengali frekuensi, pewaktu terprogram dan driver motor induksi 3 fasa menggunakan transistor sebagai switching. Rangkaian- rangkaian tersebut kemudian digabungkan dengan sistem minimum 8088 yang dikomunikasikan dengan komputer PC secara serial sehingga membentuk satu sistem pengatur kecepatan putar motor induksi 3 fasa secara adaptif.

Perancangan sistem kontrol adaptif pada dasarnya adalah suatu proses pengamatan yang dilakukan secara terus menerus (kontinyu) terhadap masukan dan keluaran dari plant yang akan diatur. Dari proses pengamatan tersebut akan didapatkan karakterstik dinamik dari plant. Dengan membandingkan antara karakteristik dinamik plant yang terukur dan karakteristik yang diinginkan, maka akan diperoleh selisih (error) yang

selanjutnya digunakan untuk mengubah parameter-parameter sistem yang dikontrol.

Berdasarkan cara melakukan proses tersebut di atas, sistem kontrol adaptif dapat dibagi menjadi tiga macam yaitu :

- Gain Scheduling.
- Self Tuning Control.
- Model Reference Adaptif Control (MRAC).

Dari ketiga macam metoda tersebut yang akan dibahas dalam Tugas Akhir ini adalah Sistem kontrol Adaptif dengan Self Tuning Control.

1.4. Metodologi

Permasalahan yang dikemukakan pada sub bab 1.2 perlu dipecahkan dengan metoda yang tepat, adapun tahab-tahab yang ditempuh adalah sebagai berikut:

a. Studi litelatur dan lapangan tentang metoda pengaturan kecepatan motor induksi dari buku referensi dan sumber data yang lain.
b. Menyusun berbagai alternatif untuk mendapatkan pemecahan yang diperkirakan paling sesuai.
c. Menganalisa data dan memadukannya dengan alternatif yang ada sehingga dapat ditentukan cara yang dipakai sebagai pemecahan. Pemecahan ini berupa hardware dan software yang dipadukan dengan sistem minimum 8088 dan dikomunikasikan secara serial dengan komputer PC.
d. Merancang jenis dan kelakuan regulasi yang diinginkan.

e. Menentukan parameter dinamis dari plant, yang dilakukan dengan identifikasi.

f. Merancang Kontrol adaptif self tuning control.

1.5. Sistematika Pembahasan

Dalam pembahasan Tugas Akhir ini dilakukan melalui langkah-langkah yang tertuang dalam beberapa bab dengan perincian sebagai berikut :

Bab I : Pendahuluan, membahas latar belakang, perumusan masalah, tujuan serta batasan-batasan yang diperlukan dalam Tugas Akhir ini.

Bab II : Teori Penunjang , membahas tentang motor induksi tiga fasa, cara-cara pengaturan kecepatan motor induksi, mikroprosesor 8088 dengan peripheralnya, serta sistem kontrol adaptif.

Bab III : Perencanaan dan Pembuatan Alat, membahas perencanaan dan pembuatan keseluruhan sistem meliputi perangkat keras yang digunakan untuk mengatur kecepatan motor induksi 3 fasa secara adaptif dan perangkat lunak pengendalinya.

Ban IV : Pengujian dan analisa hasil-hasil perencanaan.

Bab V : Kesimpulan dan Saran.

Maha Suci Alloh yang di tangan-Nya lah segala kerajaan dan Dia Maha Kuasa atas segala sesuatu. (Q.S. Al Mulk : 1)

BAB II
TEORI PENUNJANG

BAB II

TEORI PENUNJANG

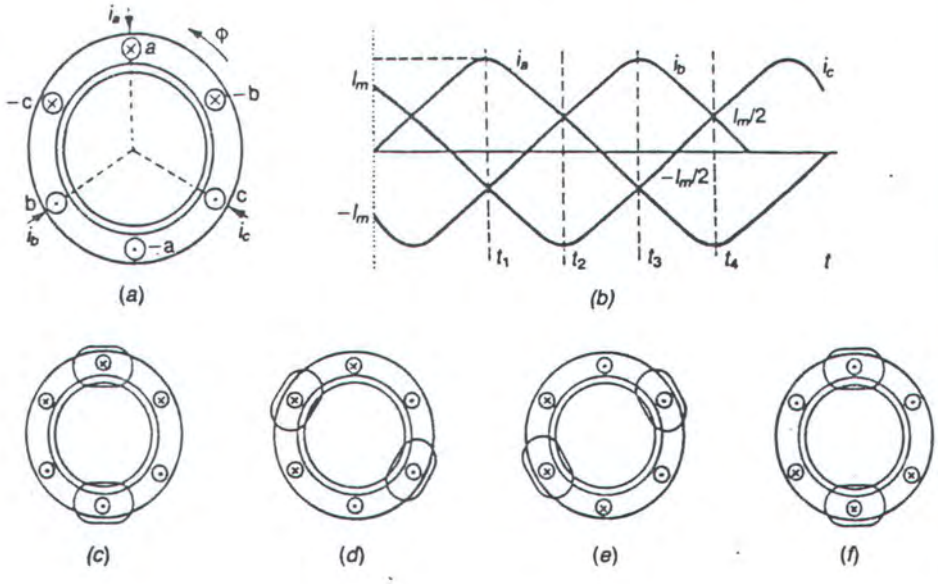
2.1 Motor Induksi 3 Fasa

2.1.1 Prinsip Kerja Motor Induksi

Motor induksi merupakan jenis motor AC yang paling banyak digunakan. Jenis motor ini disebut motor induksi, sebab arus rotor motor ini bukan diperoleh dari sumber tegangan tertentu, tetapi merupakan arus yang terinduksi sebagai akibat adanya perbedaan relatif antara putaran rotor dengan medan putar yang dihasilkan oleh arus stator.

Jika kumparan stator diberi tegangan bolak-balik 3 fasa maka akan terjadi medan magnet yang berputar dengan kecepatan sinkron. Medan putar pada stator tersebut akan memotong kumparan pada rotor, sehingga terinduksi arus yang menyebabkan rotor ikut berputar mengikuti medan putar stator. Perbedaan putaran relatif antara stator dan rotor disebut slip. Apabila kumparan pada stator adalah *a-a, b-b* dan *c-c* seperti yang diperlihatkan pada gambar 2.1a dihubungkan ke tegangan 3 fasa dan dialiri arus sinusoida dengan beda fasa masing-masing 120 derajat, maka distribusi arus sebagai fungsi waktu dapat diperlihatkan pada gambar 2.1b. Pada keadaan *t1, t2, t3,* dan *t4* fluks yang ditimbulkan masing-masing diperlihatkan pada gambar 2.1c, d, e dan f.

Kecepatan putar motor induksi dengan kutub berjumlah *P* dan pada statornya



Gambar 2.1 [1]
Pembentukan Medan Putar Stator.

diberi tegangan 3 fasa dengan frekuensi f dapat dinyatakan dengan persamaan berikut :

$$n_s = \frac{120\,f}{P}$$

Torsi yang ditimbulkan pada motor induksi adalah hasil interaksi dari fluks dan arus rotor. Jika rotor berputar pada kecepatan sinkron, maka tidak terdapat fluks yang memotong kumparan rotor sehingga tidak ada tegangan yang diinduksikan, karena itu rotor selalu berputar searah dengan putaran fluks tetapi lebih rendah kecepatan putarnya. Perbedaan kecepatan rotor

[1] Zuhal, Dasar Teknik Tenaga Listrik dan Elektronika Daya, hlm. 102.

dan kecepatan sinkron disebut dengan slip, yang dapat dinyatakan dengan persamaan :

$$S = \frac{(n_s - n_r)}{n_s}$$

Dimana :

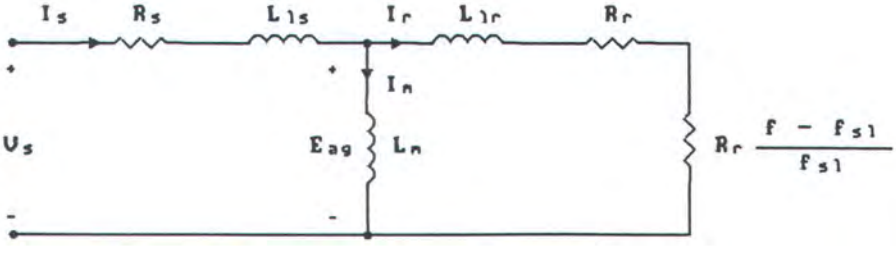
S = Slip yang terjadi.

n_s = Kecepatan medan putar stator.

n_r = Kecepatan putar rotor.

Karena kecepatan motor induksi (n_r) berubah-ubah, maka slip yang terjadi juga berubah dari 100% pada saat start dan menjadi kecil pada saat motor berputar pada putaran nominal.

Rangkaian ekuivalen motor induksi untuk setiap fasanya diperlihatkan pada gambar 2.2.



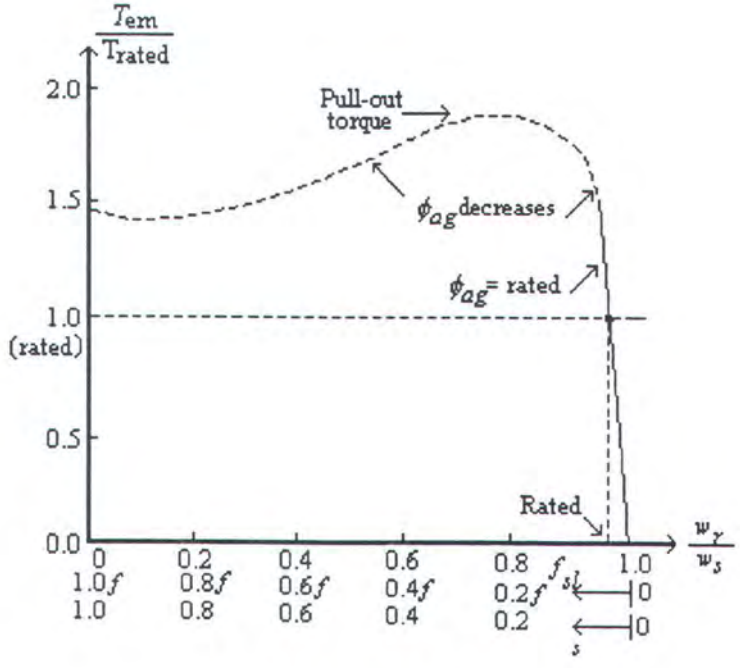
Gambar 2.2[2]
Rangkaian Ekuivalen Motor Induksi.

[2] Ned Mohan, Tore M. Undeland, W.P. Robbins, Power Elektronic: Converter, Applications, and Design, hlm. 311.

Tegangan yang diberikan pada kumparan stator dapat dinyatakan dengan :

$$V_s = E_{ag} + (R_s + j2\pi fL_{ls})\, I_s$$

Gambar 2.3 adalah karakteristik torsi dari motor induksi. Pada gambar tersebut terlihat grafik T_{em} sebagai fungsi dari kecepatan rotor dan f_{sl}. Pada nilai f_{sl} yang rendah, T_{em} bervariasi secara linier terhadap f_{sl}, tetapi pada slip yang besar perbedaan ini tidak linier sebab nilai induktansi kumparan rotor tidak dapat diabaikan lagi seperti pada slip yang kecil. Pada umumnya, motor induksi dijalankan pada slip kecil, yang pada gambar 2.3 ditunjukkan grafik dengan garis penuh.



Gambar 2.3[3]
Karakteristik Torsi Motor Induksi .

[3] Ned Mohan, Tore M. Undeland, William P. Robbins, Power Elektronics: Converters, Applications, and Design, hlm. 316.

Jika sebuah motor induksi dihubungkan dengan jala-jala listrik tanpa menggunakan sebuah kontroler elektronik, arus start yang dibutuhkan sangat besar, dan dapat mencapai 6 - 8 kali arus nominal motor.

2.1.2 Pengaturan Kecepatan Motor Induksi

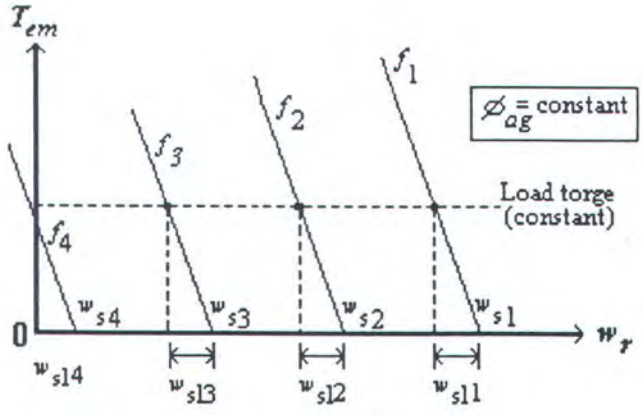
Terdapat beberapa metoda untuk mengatur kecepatan putar motor induksi yaitu :

- Mengubah jumlah kutub motor.
- Mengubah frekuensi yang diberikan ke motor.
- Mengubah tegangan yang diberikan ke motor.
- Pengaturan tahanan luar.

Diantara ke empat metoda tersebut pengaturan kecepatan dengan mengubah frekuensi adalah yang paling baik karena bisa menghasilkan kecepatan yang bervariasi dalam range yang lebar, dengan alasan tersebut pada tugas akhir ini akan dibahas pengaturan kecepatan motor induksi dengan mengatur frekuensi secara PWM yang dikontrol secara adaptif untuk motor induksi 3 fasa jenis rotor sangkar.

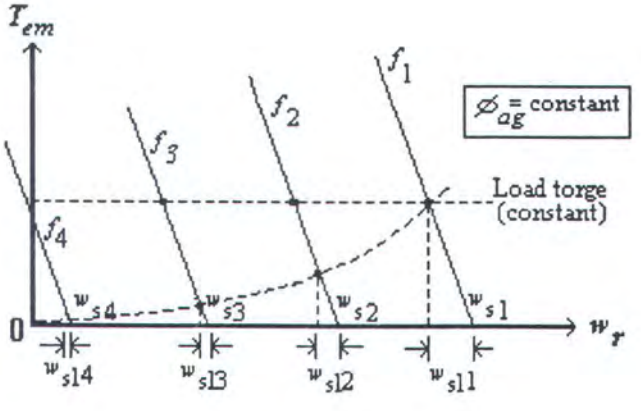
Kecepatan motor induksi dapat diatur dengan jalan mengubah frekuensi tegangan yang diberikan pada stator. Frekuensi ini akan menentukan frekuensi sinkron. Untuk mendapatkan fluks yang konstan maka perubahan frekuensi ini memerlukan perubahan tegangan pula.

Gambar 2.4 & 2.5 memperlihatkan karakteristik motor induksi yang dikontrol dengan merubah frekuensinya untuk 2 jenis beban. Gambar



Gambar 2.4 [4]
Karakteristik Torsi-Frekwensi Motor Induksi untuk Frekwensi Bervariasi, Beban Konstan.

2.4 memperlihatkan karakteristik torsi untuk frekuensi yang berbeda dengan torsi beban yang besarnya konstan. Sedangkan gambar 2.5 memperlihatkan karakteristik motor induksi untuk frekuensi yang berbeda dengan torsi beban berubah.



Gambar 2.5 [5]
Karakteristik Torsi-Frekwensi Motor Induksi, untuk Torsi Beban yang Bervariasi.

[4,5] Ibid, hlm. 318.

2.2 Inverter PWM

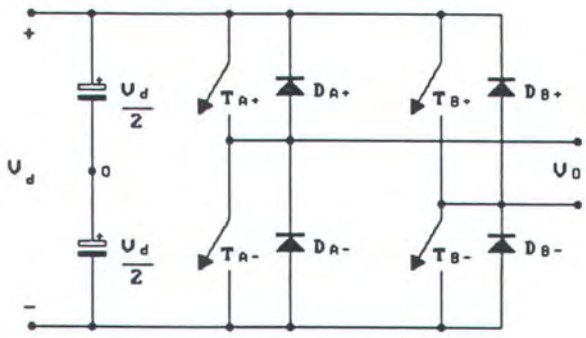
Untuk mengatur kecepatan motor induksi dengan jalan merubah frekuensi secara PWM, hal pertama yang dilakukan adalah menyearahkan tegangan AC jala-jala, kemudian dengan menggunakan teknik *switching*, tegangan DC yang dihasilkan diubah kembali menjadi tegangan AC yang terkontrol frekuensinya. Karena itu diperlukan rangkaian penyearah (*rectifier*), kontrol frekuensi dan *inverter*.

2.2.1 Inverter Satu Fasa

Bentuk gelombang PWM sinusoida, dapat diperoleh dengan jalan membandingkan sinyal sinus dan sinyal segitiga pada sebuah komparator. Frekuensi switching dari sinyal PWM yang dihasilkan ditentukan oleh frekuensi dari sinyal segitiga sebagai sinyal pemodulasi bagi sinyal sinus (sinyal termodulasi).

Modulasi duty ratio ditentukan oleh tegangan kontrol (*Vkontrol*), dari sinyal sinus yang memiliki frekuensi dasar *f1*, yaitu frekuensi dasar yang diinginkan bagi sinyal PWM sinusoida.

Untuk mendapatkan gelombang PWM sinusoida dari tegangan DC dilakukan menggunakan rangkaian *inverter* yang pada inputnya diberikan sinyal PWM sinusoida. Gambar 2.6 memperlihatkan rangkaian *inverter* satu fasa yang dapat menghasilkan gelombang PWM sinusoida.



Gambar 2.6 [6]
Rangkaian inverter satu fasa.

Perbandingan amplitudo *Vkontrol* dengan *Vtri* disebut dengan *amplitudo modulation ratio* dan dinyatakan oleh persamaan :

$$m_a = \frac{V_{kontrol}}{V_{tri}}$$

Sedangkan *frekuensi modulasi ratio* dinyatakan dengan persamaan :

$$m_f = \frac{f_s}{f_1}$$

PWM yang dihasilkan oleh inverter satu fasa ditunjukkan pada gambar 2.7. Switch T_{A+} dan T_{A-} dikontrol berdasarkan perbandingan *Vkontrol* dengan *Vtri*. Output yang dihasilkan adalah sebagai berikut :

---

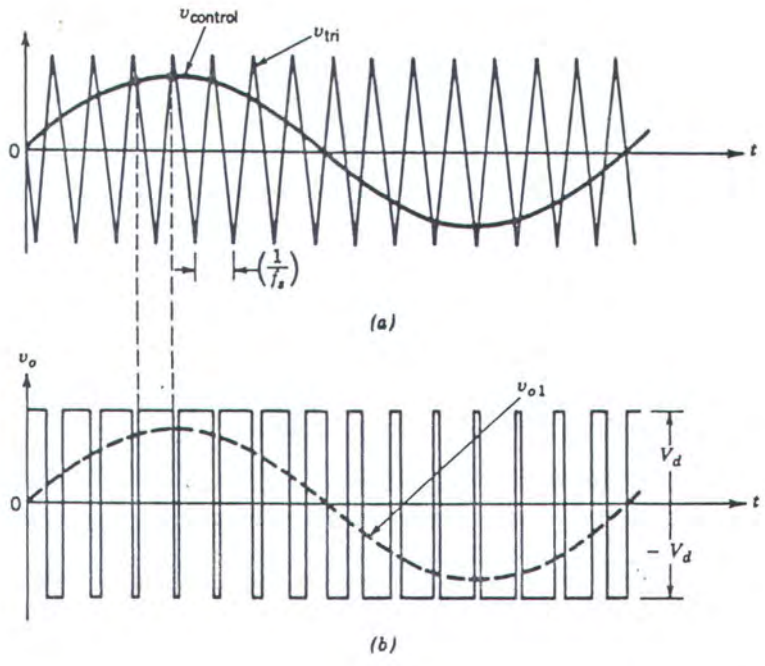
[6] Ibid, hlm. 115.

$$V_{kontrol} \langle V_{tri}, \quad T_{A-}\ on, \quad V_{AO} = -\frac{V_d}{2}$$

$$V_{kontrol} \rangle V_{tri}, \quad T_{A+}\ on, \quad V_{AO} = \frac{V_d}{2}$$

Besarnya tegangan ouput gelombang PWM Sinusoida yang dihasilkan oleh rangkaian *inverter* tergantung pada *amplitudo modulation ratio* dan dapat dinyatakan sebagai berikut :

$$(V_{AO})_1 = m_a \cdot \frac{V_d}{2}$$



Gambar 2.7[7]
Modulasi Lebar-Pulsa.

[7] Ned Mohan, Torem M. Undeland, William P. Robbins, op cit, hlm. 106.

2.2.2 Inverter Tiga Fasa

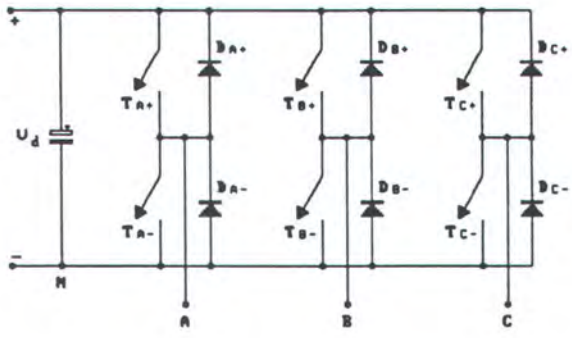
Inverter tiga fasa akan memiliki 3 output dengan beda fasa 120 derajat antara satu output dengan output yang lainnya. Rangkaian *inverter* 3 fasa diperlihatkan pada gambar 2.8.

Output dari *inverter* 3 fasa diperoleh dengan cara yang sama dengan *inverter* satu fasa yaitu dilakukan proses pembandingan antara *Vkontrol* dengan *Vtri*. Perbedaannya dengan *inverter* satu fasa adalah diperlukannya 3 buah *Vkontrol* pada *inverter* 3 fasa. Jika masing-masing *Vkontrol* merupakan gelombang sinus yang memiliki beda fasa 120 derajat satu dengan yang lainnya, maka output dari inverter akan berupa 3 buah gelombang PWM sinusoida yang masing-masing berbeda 120 derajat.

Tegangan *line to line* dari frekuensi dasar pada *inverter* 3 fasa dengan *amplitudo modulation ratio* yang lebih kecil atau sama dengan satu dapat dinyatakan pada persamaan sebagai berikut :

$$V_{LL} = \frac{\sqrt{3}}{\sqrt{2}} \cdot V_{AN} = 0{,}612\ m_a \cdot V_d$$

Amplitudo modulation ratio yang lebih besar dari satu akan menghasilkan *over modulation*. Pada daerah ini, tegangan output dari *inverter* tidak lagi linier terhadap *amplitudo modulation ratio*. *Over modulation* terjadi jika puncak dari *Vkontrol* yang digunakan lebih besar dari puncak *Vtri*.



Gambar 2.8[8]
Rangkaian inverter 3 fasa.

2.3 Sistem Minimum μP 8088

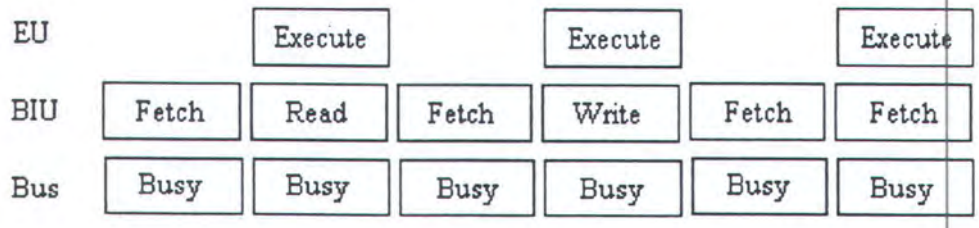
Mikroprocesor 8088 adalah mikroprosesor 8 bit yang dapat dioperasikan dalam dua mode, yaitu mode maximum dan mode minimum. Mode minimum digunakan untuk proses-proses yang sederhana yang tidak melibatkan proses perhitungan *floating point,* sedangkan mode maximum digunakan jika proses yang ditangani oleh mikroprocesor sangat komplek dan memerlukan proses perhitungan bilangan-bilangan *floating point.*

2.3.1 Arsitektur Dasar μP 8088

Pemahaman tentang arsitektur mikroprosesor 8088 mutlak diperlukan sehingga dapat memanfaatkan mikroprocesor secara optimal, baik pada saat melakukan *interfacing* maupun penyusunan program pengendali bagi perangkat keras yang di*interface*kan.

---

[8] Ibid, hlm. 130.



Gambar 2.9[9]

BIU, EU dan Bus Timing μP 8088.

Berikut ini dijelaskan hal-hal yang berhubungan dengan arsitektur dasar mikroprocesor 8088, yang meliputi :

1. Arsitektur internal dasar.
2. Sisten arsitektur.

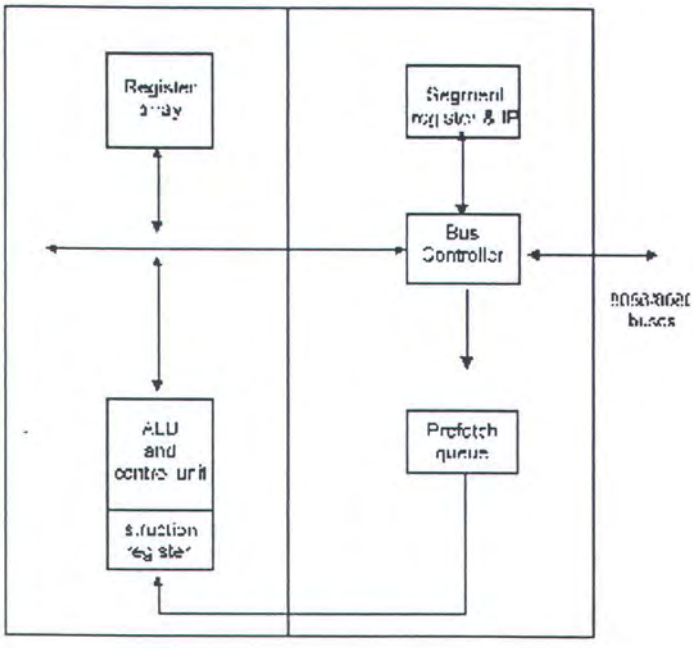
2.3.1.1 Arsitektur Internal Dasar

Mikroprosesor 8088 berbeda dengan mikroprosesor generasi sebelumnya dalam hal pengambilan instruksi dari memori, hal ini karena adanya perbedaan dalam struktur internalnya. Gambar 2.9 memperlihatkan operasi normal pada mikroprocesor 8088. Terdapat dua unit dalam μP 8088 yaitu *Internal Excution Unit (EU)* dan *Bus Interface Unit (BIU)*. BIU

---

[9] Barry b. Brey, The Intel Microprosesor, International Publishin Group, hlm. 4.

Bertugas menangani proses pengambilan instruksi, operand dari instruksi tersebut dan data dari memori. Sedangkan EU digunakan untuk menangani

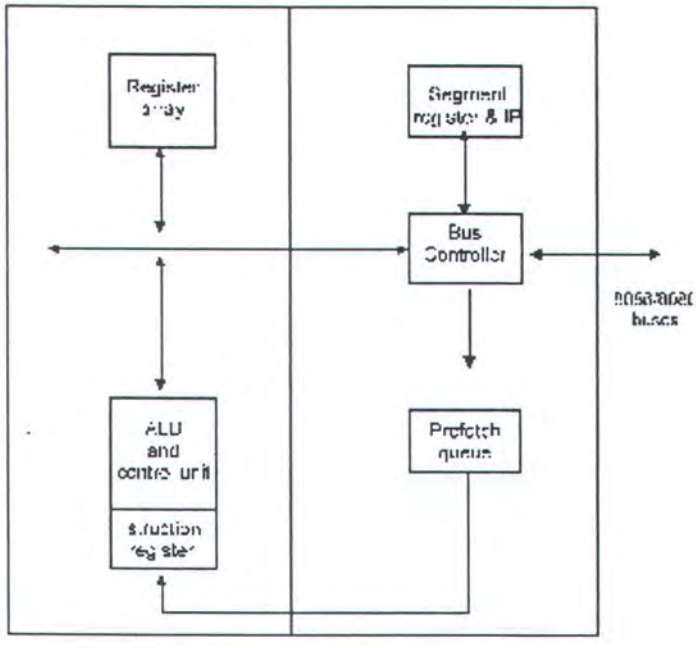


Gambar 2.10[10]
Struktur internal μP8088 meliputi BIU dan EU.

eksekusi intruksi-intruksi tersebut. Mikroprosesor 8088 dapat memanfaatkan bus secara efisien hal ini karena dimilikinya memori internal yang mempunyai bentuk *queue* atau *FIFO (First In First Out)*. Gambar 2.10 memperlihahatkan letak dari BIU dan EU. Dengan memori internal sistem FIFO ini membuat μP 8088 dapat memakai sistem memori secara lebih efektif. *Bus Interface Unit* (BIU), mengandung *prefetch* untuk *Queue*, *Bus Controller*, *Segment Register* dan *Instruction Pointer* (IP). Tujuan utama dari

[10] **Ibid, hlm.5.**

Bertugas menangani proses pengambilan instruksi, operand dari instruksi tersebut dan data dari memori. Sedangkan EU digunakan untuk menangani



Gambar 2.10[10]
Struktur internal μP8088 meliputi BIU dan EU.

eksekusi intruksi-intruksi tersebut. Mikroprosesor 8088 dapat memanfaatkan bus secara efisien hal ini karena dimilikinya memori internal yang mempunyai bentuk *queue* atau *FIFO (First In First Out)*. Gambar 2.10 memperlihahatkan letak dari BIU dan EU. Dengan memori internal sistem FIFO ini membuat μP 8088 dapat memakai sistem memori secara lebih efektif. *Bus Interface Unit* (BIU), mengandung *prefetch* untuk *Queue*, *Bus Controller*, *Segment Register* dan *Instruction Pointer* (IP). Tujuan utama dari

[10] **Ibid, hlm.5.**

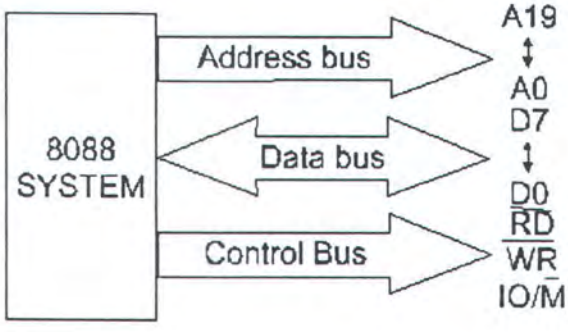
BIU adalah untuk menjaga prefetch terisi dengan instruksi, untuk membangkitkan dan menerima sinyal kontrol, menyediakan addres untuk memori dan port input/ output, serta berlaku sebagai window antara memori dan EU untuk data. BIU menjamin bahwa queue diisi dengan instruksi-instruksi dengan mengambil byte tunggal berikutnya dari instruksi jika queue μP 8088 mempunyai ruang kosong. Prefecth memungkinkan EU untuk mendapatkan instruksi selanjutnya secara langsung dari BIU sebagai pengganti memori seperti yang diperlukan oleh mikroprosesor generasi yang terdahulu. Karena instruksi berikutnya sudah berada dalam μP 8088 ketika intruksi akan dilaksanakan, maka mikroprosesor bisa mengeksekusi program lebih cepat dibanding dengan jika setiap intruksi diambil langsung dari memori.

Tujuan dari EU adalah untuk mengeluarkan instruksi yang diambil dari prefetch queue. EU mengandung Aritmatika Logika Unit, Register Intruksi, dan Register Array. ALU menyelenggarakan operasi logika dan aritmatika pada memori atau register data. Regester instruksi menerima instruksi dari prefetch instruksi pada saat mereka didekodekan untuk mengatur operasi dari EU. Register array, dan Scratchpad memori, memegang informasi sementara waktu.

EU juga mengandung Register Index dan Pointer yang dipakai untuk mengalamati operand data yang terletak pada memori. Operand data juga dialamati dan ditransfer melalui BIU.

2.3.1.2 Arsitektur Sistem

Gambar 2.11 memperlihatkan arsitektur sistem dari μP 8088. Pada keadaan ini komunikasi ke sistem terjadi melalui tiga bus : Address bus, Data bus, dan Kontrol bus. Address bus menyediakan address memori ke memori sistem dan address I/O ke perangkat keras I/O. Data bus



Gambar 2.11[11]
Bus Sistem μP 8088 meliputi Addtess, Data dan Kontrol Bus.

mentransfer data antara mikroprosesor ke/dari memori dan I/O yang diinterfacekan pada sistem. Kontrol bus menyediakan sinyal kontrol untuk proses Read dan Write pada memori atau I/O port.

2.3.2 Fungsi Pin-Pin μP 8088 pada Mode Minimum

Mode minimum dari μP 8088 didapat dengan menghubungkan pin MN/MX ke + 5 Volt. Pin ini tidak boleh dihubungkan dengan pull-up resistor karena akan membuat μP 8088 tidak akan bekerja, adapun pin yang aktif pada mode minimum adalah :

1. **IO/M**, Memori atau Input output yaitu pin yang menunjukkan kapan

---

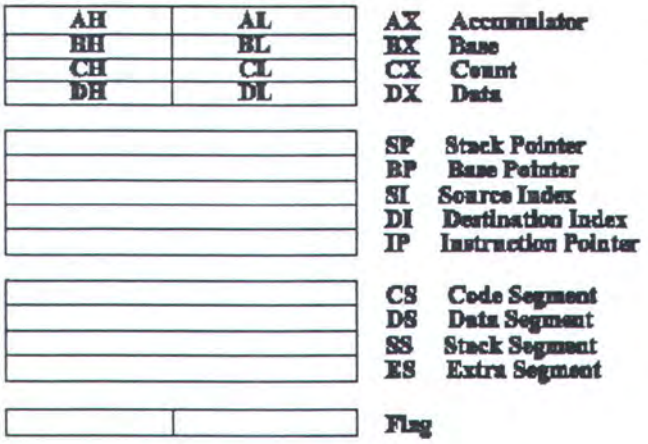
[11] Ibid, hlm. 6.

bus addres mengandung informasi memori address atau informasi address I/O. Output ini mengambang untuk state high-impendasi selama Hold Acknowledge.

2. **WR**, Write, strobe yang dipakai bahwa bus data mengandung data valid untuk ditulis ke memori atau I/O. Output ini mengambang untuk state high-impendasi selama Hold Acknowledge.
3. **INTA**, Interupt Acknowlege, adalah sebuah respon terhadap INTR. Selama interupt request pin ini berada pada logika 0, menunjukkan bahwa bus μP 8088 menunggu vektor number yang diberikan ke data bus yang terhubung.
4. **ALE**, Address Latch Enabel adalah pin yang dipakai untuk menunjukkan address data bus yang terhubung berisi address memori valid atau address port I/O valid. ALE tidak pernah mengambang ke state high-impendance.
5. **DT/R**, Data transmit/receive yaitu pin yang dipakai untuk mengontrol arah aliran data melalui hubungan buffer data bus eskternal. Output ini mengambang untuk state high-impendance selama Hold Acknowledge.
6. **DEN**, Data Bus Enabel yaitu sinyal yang menunjukkan bahwa bus data/address mengandung data valid. Ouput ini mengambang untuk state high-impendance selama Hold Acknowledge.
7. **HOLD**, yaitu pin input yang dipakai untuk meminta direct memory access (DMA). Ketika HOLD aktif μP 8088 akan mengambang bus address, data, dan kontrol sehigga kontroler DMA ekstrenal bisa

mendapatkan access ke memori atau I/O.

8. **HLDA**, Hold acknowledge yang menunjukkan pin HOLD sudah menjadi tinggi dan bahwa bus-bus diambangkan ke state high impendance.



Gambar 2.12 [12]
Register-register pada μP 8088.

9. **SSO**, Status Line yaitu pin ini bisa dikombinasikan dengan I/O dan DT/R untuk mendekodekan siklus bus yang sedangkan berlangsung.

2.3.3 Register-Register pada μP 8088

Gambar 2.12 memperlihatkan register internal pada μP 8088. Register-register tersebut dapat dikelompokkan dalam empat grup register yaitu general purpose register, pointer dan index register, segment register dan Register pembantu yang berkaitan dengan operasi khusus dari μP 8088 dan operasi aritmatik dan logik unit.

---

[12] Ibid, hlm. 9.

2.3.3.1 General Purpose register

General purpose register dipakai sebagai register pada pemrosesan data. Dalam pemakaiannya dapat dialamati sebagai register 16 bit maupun sebagai register 8 bit. Terdapat 4 register yang termasuk dalam kelompok general purpose register. Keempat register dan fungsi dari masing-masing register tersebut adalah sebagai berikut :

a. Accumulator atau Register AX.

Register ini dipakai untuk melakukan operasi aritmatika, logik, shift dan lintas data dari/ke port.

b. Base Register atau Register BX.

Register ini dipakai sebagai pencatat address memori yang isinya akan diakses.

c. Counter Register atau Register CX.

Register ini memiliki sifat khusus sebagai counter register terhadap perintah Loop dan perintah blok transfer pada operasi string.

d. Data Register atau Register DX.

Register ini dipakai untuk mencatat address port, dengan demikian operasi lintas data port harus diarahkan ke register DX.

2.3.3.2 Pointer dan Index Register

Register-register yang masuk dalam kelompok ini dipakai sebagai index atau petunjuk lokasi memori yang menangani data operand untuk bermacam instruksi. Register-register yang termasuk dalam kelompok ini berikutnya adalah:

a. *Stack Pointer atau Register SP.*

Register SP digunakan untuk mencatat address stack. Stack merupakan memori yang dikhususkan untuk menyimpan/membaca isi register pada saat perintah Push dan Pop.

b. *Base Pointer atau Register BP.*

Register BP digunakan untuk mencatat offset address data yang disimpan pada stack, pada saat data tersebut akan diakses.

c. *Source Index atau Register SI.*

Register ini digunakan untuk mencatat address memori yang isinya akan diakses. Address yang dicatat oleh register SI merupakan offset address bagi operand sumber.

d. *Destination Index atau Register DI.*

Register DI identik dengan register SI, address yang dicatat oleh register DI merupakan offset address yang menentukan lokasi dari operand tujuan.

e. *Instruction Pointer atau Register IP.*

Register IP digunakan untuk mencatat offset address dari instruksi berikutnya yang akan dieksekusi oleh mikroprosesor.

2.3.3.3 Segment Register

Mikroprosesor 8088 mencatat address untuk menunjukkan lokasi data di memori menggunakan segment register ditambah dengan ofsset atau selisih address. Selisih address tersebut ditempatkan pada salah satu termasuk segment register.

Setiap segment register adalah register 16 bit dan digunakan khusus untuk mencatat address awal segment memori. Satu segment memori mencakup lokasi maksimum 64 Kb. Register-register yang termasuk dalam kelompok ini adalah :

a. *Code Segment atau Register CS.*

   Register ini digunakan untuk mencatat address segment memori tempat kode operasi (kode bahasa mesin) suatu program. Kode itu secara urut dibaca dan dilaksanakan oleh CPU.

b. *Data Segment atau Register DS.*

   Register ini digunakan untuk mecatat address segment memori tempat data-data yang digunakan oleh suatu program disimpan di memori.

c. *Stack Segment atau Register SS.*

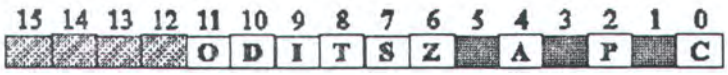
   Register ini dipakai untuk mencatat address segment memori yang dipergunakan sebagai stack yaitu berupa memori yang digunakan untuk tempat simpan-baca pada operasi Push dan Pop.

d. *Extra Segment atau Register ES.*

   Register ini digunakan untuk mencatat address segment memori bagi data yang tidak terletak di Code Segment, Data Segment ataupun Stack Segment.

2.3.3.4 Status Register

Register ini merupakan register 16 bit dan digunakan untuk menyimpan tanda sebagai hasil suatu operasi. Dari 16 bit tersebut yang dipergunakan hanya 9 bit yaitu :



Gambar 2.13[13]
Bit-bit status pada Register Status.

- Tanda yang berkaitan dengan operasi mikroprosesor, terdiri dari bit-bit :

  Overflow Flag (OF), Direction Flag (DF), Interrupt Flag (IF) dan bit Trap Flag (TF).

- Tanda sebagai akibat operasi arimatik dan logik, terdiri dari bit-bit :

  Sign Flag (SF), Zero Flag (ZF), Auxiliary Carry Flag (AF), Parity Flag (PF), dan Carry Flag (CF).

Posisi dari masing-masing bit pada register F ini diperlihatkam pada gambar 2.13.

2.3.4 Unit Input-Output

Untuk melakukan proses transfer data, mikroprosesor memerlukan unit input-output. Terdapat dua cara dalam transfer data yaitu secara paralel dan secara serial. Transfer data secara paralel jauh lebih cepat dibanding secara serial, tetapi transfer data secara serial mempunyai keunggulan dalam hal jarak yang lebih jauh dibanding secara paralel.

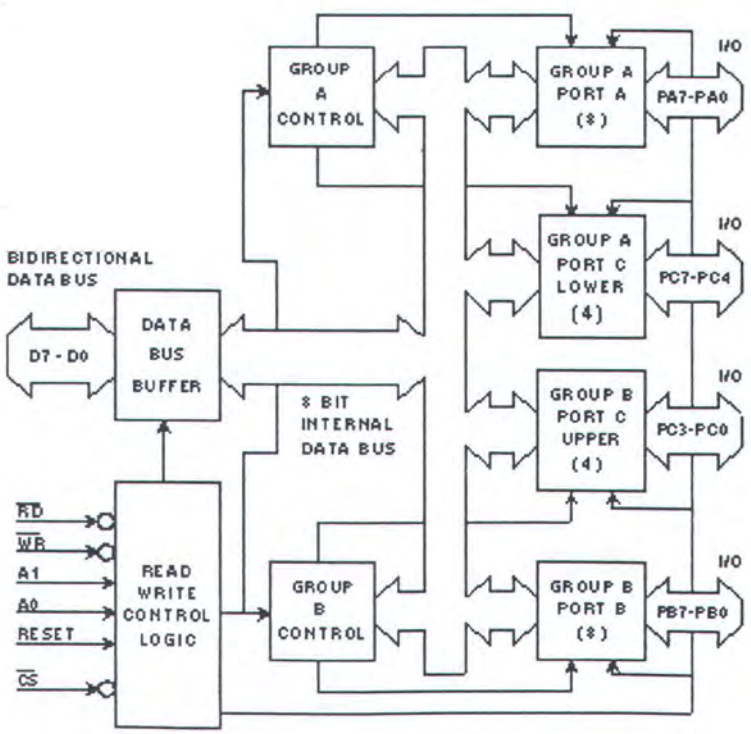
---

[13] Ibid, hlm. 14.

Mikroprosesor memerlukan sarana pembantu untuk dapat melakukan proses transfer data, yaitu berupa IC yang dikenal dengan sebutan *Programable Peripheral Interface (PPI 8255)* untuk tranfer data paralel dan *Universal Asynchronous Receiver Transmiter (UART 8250)* atau *Universal Synchronous Asynchronous Receiver Transmiter (USART 8251)* untuk komunikasi serial.

2.3.4.1 Programmable Peripheral Interface (PPI 8255)

PPI 8255 menyediakan 4 metoda transfer data paralel, yaitu Simple input-output, Simple strobe input-output, Single handshake dan Double handshake. Gambar 2.14 memperlihatkan blok diagram PPI 8255.



Gambar 2.14[14]
Blok diagram PPI 8255.

[14] J.P.M. Steeman, Data Sheet Book 2, hlm. 240.

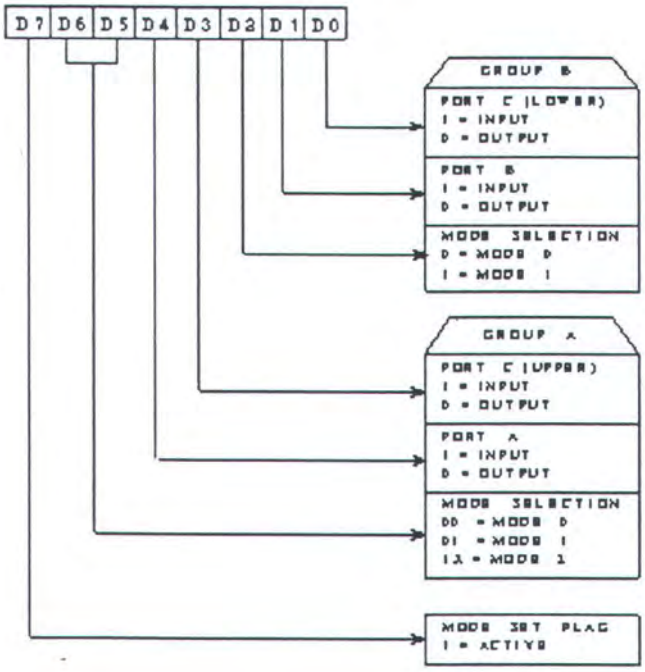
Pada sebelah kiri gambar 2.14 terdapat 8 bit data bus bidirectional (D_0 sampai D_7), sinyal-sinyal A_0, A_1, RD, WR, RESET, dan CS. Sisi kanan dari blok diagram menunjukkan 3 buah I/O port, masing-masing berukuran 1 byte. Port-port ini adalah Port A, Port B, dan Port C.

Melalui data bus dapat ditransfer data, intruksi atau informasi status. Data ini ditransfer bila µP 8088 melakukan siklus baca atau tulis, proses ini dilakukan dengan sinyal RD dan R.

Register port A, port B, port C dan control word dipilih dengan menggunakan dua bit A_0 dan A_1. Sinyal input CS dipakai untuk mengaktifkan PPI ini. Input RESET dipakai untuk menginisalisasi IC PPI, bila berlogika satu, maka semua register internal pada IC ini akan diclear. Sisi kanan dari blok diagram menunjukkan 3 buah I/O port, masing-masing berukuran 1 byte. Port-port ini dapat dikonfigurasikan untuk operasi input dan output. Ini memberikan total 24 jalur I/O.

2.3.4.1.1 Initialisasi PPI 8255

Sebelum PPI 8255 dapat digunakan, terlebih dahulu harus diinitialisasi. Initialisasi dilakukan dengan jalan menuliskan pada control word pada Control Word Register. Penulisan bertujuan untuk menentukan mode operasi bagi masing-masing port PPI 8255. Gambar 2.15 memperlihatkan Contol word untuk PPI 8255.



Gambar 2.15[15]
Control Word. PPI 8255.

2.3.4.1.2 Mode Kerja PPI 8255

Mode kerja dari PPI 8255 ditentukan dengan menuliskan pada Register Control word PPI 8255. Register ini merupakan write only register, dan isinya dapat diubah dengan melakukan siklus tulis ke PPI 8255 pada saat A_0 A_1 = 11.

PPI 8255 dapat dioperasikan dalam 3 mode, pada tugas akhir ini dipakai PPI 8255 pada mode 0, karena itu disini hanya dibahas mode 0, sedangkan mode yang lain dapat dilihat pada lembar lampiran.

Mode 0 juga disebut dengan simple I/O operation. Pada mode ini port A dan Port B berfungsi sebagai masukan atau keluaran tanpa strobe. Bila

---

[15] **Ibid, hlm. 242..**

port A dan port B bekerja pada mode 0, maka port C dapat dipakai sebagai port 8 bit ataupun port 4 bit yang terpisah. Dengan demikian pada mode 0 ini, PPI 8255 menyediakan 3 port input-output 8 bit atau 2 port input-output 8 bit dan 2 port input-output 4 bit.

2.3.4.2 Universal Asynchronous Receiver-Transmitter 8250

UART 8250 merupakan peripheral yang diperlukan mikroprosesor untuk melakukan transfer data serial. Dengan menggunakan peripheral ini data dikirim dari/ke mikroprosesor secara asinkron, yaitu data dikirim tiap karakter dengan dengan start dan stop bit. Jadi pada mode pengiriman ini tiap-tiap karakter diidentifikasi secara individu.

Pada tugas akhir ini digunakan Uart 8250 berupa Asynchronous Communication Adapter yaitu Serial interface card RS 232C yang dipakai pada IBM PC, dengan demikian level tegangan output dari card ini sudah sesuai untuk melakukan transfer data serial dengan IBM PC yang menggunakan standard RS 232 C dari EIA, atau V.24/V.28 dari CCITT.

2.3.4.2.1 Register pada UART 8250

Sebelum menggunakan Uart 8250 untuk melakukan transfer data, terlebih dahulu harus dilakukan pemrograman dengan jalan menuliskan pada register-register Uart 8250 spesifikasi pengiriman data yang diinginkan.

Tabel 2.1 memperlihatkan address register-regiaster yang terdapat pada Uart 8250 yang digunakan pada Asynchronous

Communiction Adaptor untuk IMB PC. Fungsi dari masing-masing registe tersebut adalah sebagai berikut :

a. TX Buffer.

Menampung dan menyimpan data yang akan dikirim ke luar. Data ini dikirim oleh CPU ke TX Buffer setelah mengecek kepastian tentang diperbolehkannya melakukan pengiriman.

b. RX Buffer.

Menampung dan menyimpan data yang diterima dari luar. Data ini diambil oleh CPU dari RX Buffer setelah mengecek kepastian tentang masuknya data.

Tabel 2.1 [16]
Address Register-register Uart 8250.

Nama Register	COM 1	COM 2
TX Buffer (Transmite Buffer)	03F8H	02F8H
RX Buffer (Receive Buffer)	03F8H	02F8H
Baut rate Divisor Latch LSB	03F8H	02F8H
Baut rate Divisor Latch MSB	03F9H	02F9H
Interrupt Enable Register	03F9H	02F9H
Intterupt Identification Register	03FAH	02FAH
Line Control Register	03FBH	02FBH
Modem Control Register	03FCH	02FCH
Line Status Register	03FDH	02FDH
Modem Status Register	03FEH	

[16] **Hartono Partoharsodjo, Bahasa Assembly, hlm. 557.**

c. *Baut Rate Divisor LSB.*

Register ini digunakan untuk menampung bilangan LSB untuk pembagi clock agar diperoleh baut rate yang diperlukan.

d. *Baut Rate Divisor MSB.*

Register ini digunakan untuk menampung bilangan MSB untuk pembagi clock agar diperoleh baut rate yang diperlukan.

e. *Interrupt Enable Register.*

Register ini digunakan untuk menampung pemrograman agar dapat melangsungkan inturupsi ke CPU untuk operasi Tx, Rx.

f. *Interrupt Identification Register.*

Menampung pemrograman untuk memberikan urutan prioritas operasi yang melangsungkan interupsi ke CPU.

g. *Line Control Register.*

Menampung pemrograman format data yang meliputi jumlah bit setiap data, jumlah stop bit dan parity check.

h. *Modem Control Register.*

Register ini digunakan untuk mengatur pengoperasian modem.

i. *Line Status Register.*

Register ini menampung bit-bit yang menyatakan kehadiran data dan kesalahan operasi, dengan demikian register ini berlaku sebagai register status.

j. *Mode Status Register.*

Register ini menampung bit-bit yang menyatakan keadaan hubungan modem.

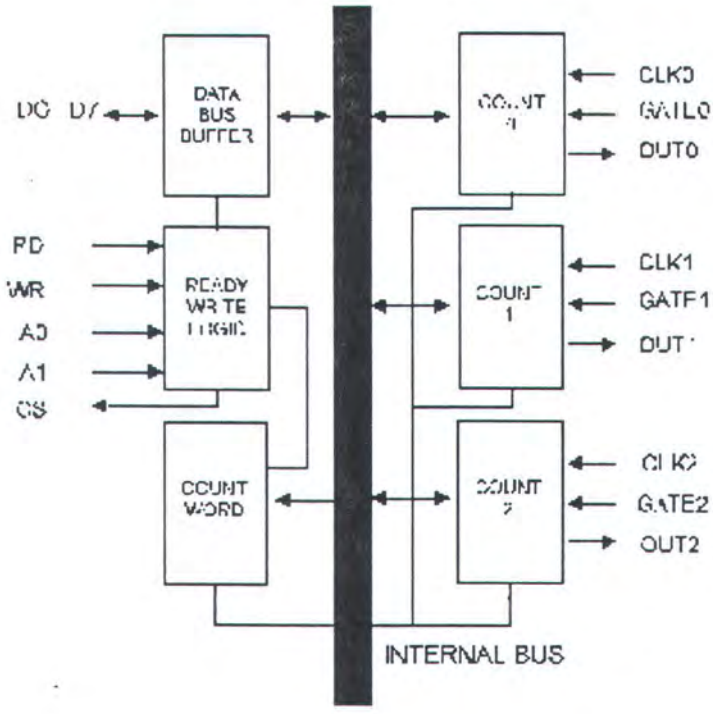
2.3.5 Programmable Interval Timer (PIT 8253)

PIT 8253 terdiri dari tiga counter yang saling independen dan dapat diprogram secara terpisah. Ketiga counter tersebut merupakan counter 16 bit, pre-settable, Down counter.

Pada suatu sistem mikroprosessor 8088, PIT 8253 dipakai sebagai perangkat peripheral, dimana CPU dapat menjalankan operasi read atau write pada register-register internal PIT 8253. Dalam hal ini, CPU dapat membentuk konfigurasi mode operasi dari suatu pewaktu, memuat harga initial ke dalam counter, atau membaca harga pada suatu saat dari counter tersebut.

Dalam PIT 8253 terdapat 8 bit *bidirectional data bus*, D_0 sampai D_7 dimana data ditransfer antara mikroprosesor dan PIT 8253. Masukan A_0 dan A_1 dihubungkan dengan address bus dan berfungsi untuk memilih Control word register dan counter yang akan diaktifkan. Sinyal kontrol *RD* dan *WR* merupakan sinyal aktif low. Pada saat sinyal *RD* low berarti mikroprosesor sedang membaca isi register counter PIT 8253. Sedangkan ketika sinyal *WR* low berarti mikroprosesor sedang mengadakan penulisan ke register PIT 8253. Penulisan tersebut bisa berupa penentuan mode kerja maupun setting nilai ke counter. Pin *CS* dibuat low untuk mengaktifkan PIT 8253.

Arsitekture internal PIT 8253 diperlihatkan pada gambar 2.16. Terlihat pada gambar tersebut bagian-bagian dari PIT 8253, yaitu Bus buffer, Read/Write logic, Control word register dan 3 buah counter.



Gambar 2.16[17]
Arsitektur Internal PIT 8253.

2.3.5.1 Initialisasi PIT 8253

Masing-masing counter dapat diprogram secara terpisah dengan cara menuliskan serangkaian control word dalam control word register. Format control word diperlihatkan pada gambar 2.17. Fungsi dari tiap-tiap bit pada kontrol word adalah sebagai berikut :

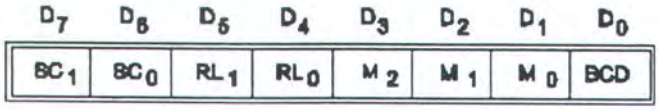
SC_0, SC_1 : Bit-bit ini menentukan counter yang diprogram.

RL_0, RL_1 : Bit-bit ini menentukan pembacaan/penulisan isi counter apakah LSB, MSB atau LSB dan MSB.

M_0, M_1, dan M_2 : Digunakan untuk menentukan mode operasi dari counter.

---

[17] J.P.M. Steeman, Data Sheet Book 2, hlm. 231.



Gambar 2.17[18]
Format Control Word PIT 8253.

BCD : Menentukan mode hitungan, bentuk biner atau BCD.

2.3.5.2 Mode Operasi PIT 8253

Frekuensi maksimum yang dapat diterima pada input *Clk* PIT 8253 adalah 2,6 MHz, sedangkan dalam pengoperasiannya PIT 8253 dapat diprogram dalam 6 mode. Dalam tugas akhir ini dipakai PIT 8253 dalam mode 0, 2 dan 3, mode yang digunakan tersebut dapat diuraikan sebagai berikut :

a. Mode 0 : Interrupt on Terminal Count.

Mode 0 ini biasanya digunakan untuk menghitung suatu kejadian. Setelah control word ditulis, Output mula-mula low dan akan tetap low sampai counter mencapai nol. Output kemudian menjadi high sampai hitungan yang baru atau control word mode 0 yang baru dituliskan pada counter.

Setelah control word dan initial count ditulis pada counter,

---

[18] Ibid, hlm. 234.

initial count akan diambil pada pulsa clock berikutnya. Pulsa clock ini tidak mengurangi hitungan, maka untuk initial count N, Out tidak akan menjadi high sampai N+ 1 pulsa clock setelah initial count ditulis.

Jika hitungan baru dituliskan pada counter, akan diambil pada pulsa clock berikutnya dan hitungan akan diteruskan mulai dari hitungan yang baru. Untuk dua byte hitungan yang diprogramkan, pada penulisan akan terjadi :

- Penulisan byte pertama akan mencegah hitungan, Out akan segera diset low (tidak diperlukan pulsa clock).
- Penulisan byte kedua menyebabkan hitungan yang baru diambil pada pulsa clock berikutnya.

Dengan begitu urutan perhitungan dapat disinkronkan dengan software. Jadi Out tidak akan menjadi high sampai N+ 1 pulsa clock setelah hitungan baru byte kedua dituliskan.

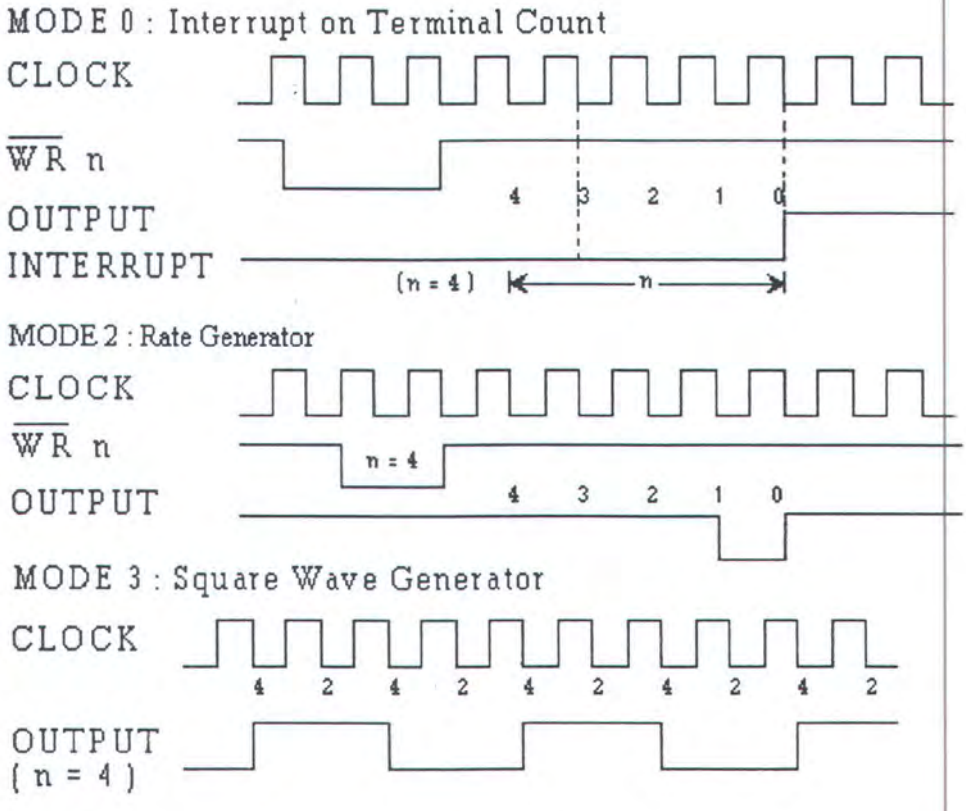
Jika initial count ditulis pada saat Gate = 0, akan diambil pada pulsa clock berikutnya. Jika Gate = 1, Out akan menjadi high pada N pulsa clock, jadi tidak diperlukan clock untuk mengambil initial count.

b. Mode 2 : Rate Generator.

Fungsi dari mode 2 ini adalah seperti counter devide by N. Pada mode ini Ouput mula-mula high dan akan menjadi low selama satu pulsa clock setelah mencapai N hitungan pulsa. Mode 2 ini berlangsung secara periodik dan terus menerus. Untuk initial count N, urutan seperti diatas terjadi setiap N pulsa clock.

Setelah control word dan initial count dituliskan, counter akan dimuati pada pulsa clock berikutnya. Out akan menjadi low setelah N pulsa clock sejak initial count dituliskan. Hal ini menyebabkan counter dapat disinkronkan dengan software.

Pada penulisan hitungan yang baru, initial count akan dimuat pada pulsa clock berikutnya jika pada Gate diberikan trigger. Jika tidak, initial count akan diambil setelah periode hitungan yang lama berakhir. Pada mode 2 ini tidak diperbolehkan memberikan nilai initial count = 1.



Gambar 18. [19]
Timing Diagram PIT mode 0, mode 2, dan mode 3.

[19] Ibid, hlm.236.

c. Mode 3 : Square Wave Generator.

Mode ini digunakan untuk menghasilkan gelombang persegi. Terdapat dua kondisi dalam mode ini, kondisi pertama adalah jika count yang diberikan genap, maka penghitungan berkurang setiap dua hitungan dan naik setiap dua hitungan. Kondisi kedua jika count yang diberikan ganjil, maka naik turunnya setiap dua hitungan, tetapi untuk low hitungan sampai nol, untuk high sampai 2.

Gambar 2.18 memperlihatkan Timing Diagram untuk mode 0, mode 2 dan mode 3 PIT 8253. Mode-mode operasi PIT 8253 yang lain dapat dilihat pada lampiran yang tersedia.

2.3.6 Programmable Interrupt Controller 8259

Programmable interrupt controller berfungsi mengatur seluruh sistem *interrupt driven*. PIC menerima permintaan dari peripheral dan menentukan prioritas mana yang tertinggi. PIC 8259 terdiri dari 8 pin input *interrupt request* dan dapat dikaskade sampai 64 permintaan interrupt.

Blok diagram internal 8259 diperlihatkan pada gambar 2.19, yang meliputi *ISR, IRR, IMR* dan *Priority resolver.* Masing-masing blok tersebut mempunyai fungsi sebagai berikut :

IRR : Digunakan untuk menampung kondisi-kondisi input.

IMR : Digunakan untuk meng-enable input regist.

ISR : Untuk menjaga input yang sedang dikerjakan.

Priority resolver : Untuk menentukan prioritas input.

2.3.6.1 Initialisasi PIC 8259

Initialisasi 8259 dilakukan dengan menuliskan pada Command word-command wordnya. Terdapat dua command word pada 8259, yaitu :

- Initialitation Command Word (ICW) yang berfungsi menentukan titik start.
- Operation Command Word (OCW) yang berfungsi menentukan mode operasi intterrupt. Mode-mode OCW tersebut adalah :
  a. Fully Nested Loop.
  b. Rotating Priority Mode.
  c. Special Mask Mode.
  d. Polled Mode.

Untuk meng-initialisasi 8259 diperlukan urutan khusus dalam penulisan Control wordnya. Urutan penulisan tersebut ditunjukkan dengan flow chart pada gambar 2.20. Dari flow chart tersebut, ICW1 dan ICW2 harus selalu dituliskan, sedangkan ICW4 diperlukan jika digunakan sistem 8086/8088. Jika menggunakan PIC yang dicascade, maka ICW3 juga harus dituliskan.

ICW2 digunakan untuk memberitahu PIC 8259 bilangan yang akan dipakai sebagai type interrupt terendah yang harus dikirim ke CPU. Pada mikroprosesor 8088, tipe interrupt 0 - 7 adalah merupakan dedicated interrupt, sehingga type interrupt 8 adalah yang paling rendah. Jika pada ICW2 dituliskan bilangan 8 decimal atau 00001000B, maka 8259 mengirimkan bilangan ini ke mikroprosesor 8088 sebagai jawaban dari pin interrupt IRQ.

OCW1 harus dituliskan untuk meng-enable berapa input IR yang ingin diaktifkan untuk beroperasi. OCW2 digunakan untuk mereset ISR. Jika direset, maka ia akan menjalankan interrupt yang lebih rendah. Jika diperlukan rotasi dari prioritas maka digunakan OCW3.

2.3.6.2 Mode Operasi PIC 8259

Didalam pemakaiannya PIC dapat dioperasikan dalam 6 mode. Keenam mode tersebut dapat dijelaskan sebagai berikut :

a. *Fully Nested Loop.*

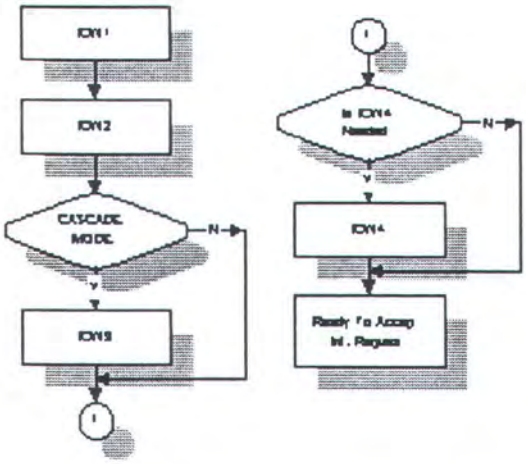
Pada mode ini, prioritas pelayanan tertinggi adalah IR0 dan terendahnya adalah IR7. Dalam semua mode, bit IS yang berhubungan harus direset agar prioritas yang lebih rendah dapat diketahui dengan cara menuliskan non specifik EOI.

b. *Special Fully Nested Loop.*

Mode ini seperti Fully nested loop, tetapi diperluas pemakaiannya untuk sistem kaskade.

c. *Non Specific Rotating Mode.*

Mode ini ditujukan untuk sistem yang mempunyai prioritas yang sama. Saat EOI ditulis, bit IS yang sesuai menjadi reset dan menuju ke prioritas yang lebih rendah.



Gambar 2.20[21]
Flow Chart Penulisan Control Word PIC 8253.

d. Specific Rotating.

Prioritas dalam mode ini dapat diputar dan EOI menunjukkan bit IS yang reset dan menjadi prioritas terendah.

e. Special Mask.

Pada mode ini interrupt-interrupt dapat diterima kecuali yang sedang dikerjakan.

f. Pool.

Pada mode ini, output INT dari PIC dihindari dan komponen digunakan sebagai prioritas poller.

---

[20] John Uffenbeck, The 8086/8088 Family Desing Programming And Interfacing, hlm. 464.

2.4 Sistem Kontrol Adaptif

Sistem kontrol yang ada banyak sekali jenisnya, pada tugas akhir ini akan dibahas salah satu jenis sistem kontrol yaitu adaptif. Penggunaan kontrol adaptif ini mempunyai tujuan :

- Memperkecil gangguan dari luar.
- Memperkecil gangguan dari dalam.
- Mengatasi keterbatasan perancangan sistem umpan balik biasa.

Adapun hal-hal yang diperlukan untuk merancang kontroler adalah :

- Kriteria performansi.
- Informasi proses (plant).

Kriteria performansi sistem yang diperlukan dalam mendisain kontroler meliputi beberapa hal yaitu :

- *Maximum overshoot.*
- *Rise time.*
- *Settling time.*
- *Peak time.*

Sedangkan informasi proses (*plant*) adalah mengenai model matematika proses (*plant*) didapatkan dengan cara melihat output dan input dari awal sampai akhir proses dengan metode identifikasi tertentu.

Pada kenyatannya sukar merancang kontroler yang tetap parameternya, hal ini disebabkan oleh :

- Parameter plant tidak diketahui.

- Parameter plant berubah terhadap waktu.
- Parameter plant tidak linier.

Dengan mengacu pada hal tersebut maka diperlukan pengaturan adaptif, dimana perubahan yang terjadi pada plant diimbangi dengan perubahan pada kontroler yang meliputi :

- Perubahan parameter kontroler.
- Perubahan struktur kontroler, meliputi perubahan derajat (orde) serta perubahan sistem/metode adaptasi.

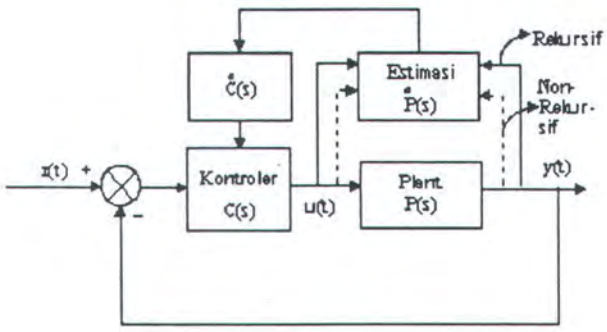
Untuk merancang suatu sistem kontrol adaptif, maka satu hal yang harus dilakukan adalah mengetahui model dari plant yang akan diatur. Untuk mengetahui model dan parameter-paremeter plant tersebut maka dilakukan proses identifikasi dan selanjutnya data hasil identifikasi dianalisa untuk membangun sistem kontrol adaptif sehingga bisa diperoleh kebijaksanaan adaptasi.

2.4.1 Sistem Kontrol Adaptif Self Tuning Regulator

Ide dari disain kontrol self tuning ini adalah membuat suatu strategi kontrol yang dihasilkan dari hasil pengolahan data parameter sistem yang didapatkan dari proses estimasi. Karenanya pada model kontrol ini terdapat dua proses terpisah yang saling terkait yaitu :

- Estimasi parameter plant.
- Disain kontroler.

Blok diagram kontrol adaptif *Self Tuning* ini ditunjukkan pada gambar 2.21. Estimasi parameter plant dapat dilakukan secara rekursif atau



Gambar 2.21
Blok Diagram Kontrol Adaptif Self Tuning Regulator.

non rekursif. Jadi, mula-mula dilakukan proses estimasi parameter, setelah parameter sistem didapatkan, dilakukan disain kontroler.

Metode *Self Tuning* Regulator dibedakan menjadi dua, yaitu :

a. Metoda tak langsung (Eksplisit).

Pada perancangan kontrol adaptif tak langsung selalu melalui dua tahap yaitu : tahap estimasi parameter dan tahap disain kontrol. Karenanya jika pada tahap disain kontrol terjadi perubahan parameter sistem, maka sinyal kontrol yang dihasilkan dari disain sudah tidak cocok lagi (kurang cocok) untuk menghasilkan output yang sesuai. Oleh karena itu pada model disain ini perlu diasumsikan bahwa : "parameter plant adalah konstan atau berubah perlahan" dan pada saat start up juga perlu diperhitungkan tentang bagaimana bentuk sinyal kontrol yang diperlukan karena parameter sistem belum diketahui. Maka muncul permasalahan baru yaitu : "bagaimana bentuk parameter kontroler pada saat start up".

b. Metoda langsung (Implisit).

Dalam metoda kontrol adaptif self tuning langsung ini, metode Self Tuning Regulator tak langsung dapat dimodifikasi menjadi langsung yaitu dengan memodifikasi estimasi parameter sistem menjadi estimasi parameter kontrol. Jadi pada metoda ini mempunyai dua tahapan, yaitu :

- Mengestimasi parameter kontroler secara langsung dari pengukuran input/output plant.
- Melakukan reparameterisasi dari sistem kontrol berdasarkan hasil estimasi.

Dalam tugas akhir ini strategi kontrol yang digunakan adalah strategi kontroler Adaptif Self Tuning langsung.

2.4.2 Identifikasi Sistem

Identifikasi sistem adalah suatu proses yang dilakukan untuk merekontruksi model dan mengestimasikan parameter-parameter model sehingga didapatkan model yang terbaik berdasarkan data-data eksperimen.

Secara garis besar ada dua macam metoda dalam melakukan identifikasi suatu sistem, yaitu :

- **Identifikasi Off Line**

Identifikasi sistem secara off line dilakukan dengan memberikan identifikasi input tertentu pada suatu proses, selanjutnya input dan output dibaca pada selang waktu tertentu. Dari data yang diperoleh dilakukan analisis secara terpisah untuk mendapatkan hasil yang relatif lebih teliti.

- **Identifikasi On Line**

Dalam identifikasi on line, pengambilan data dan analisis dilakukan secara serentak selama proses berlangsung. Estimator menerima data input dan output dari proses, selanjutnya dianalisa dan menghasilkan estimasi yang dihitung secara rekursif. Dengan cara demikian perubahan parameter yang terjadi dapat langsung dideteksi secara kontinyu, sehingga sistem kontrol dapat ditala kembali berdasarkan data yang dihasilkan.

Estimasi off line biasanya diperlukan untuk menentukan nilai awal parameter sistem yang akan diestimasikan secara on line. Jadi seting harga awal estimator on line didapat dari hasil estimasi off line, dengan demikian akan mempercepat keadaan konvergen.

2.4.2.1 Algoritma Identifikasi Rekursif

Data hasil pengukuran input dan output pada saat sampling ke t dituliskan dalam vektor data $Z(k)$ sebagai berikut :

$$Z(t) = \begin{bmatrix} y(t) \\ u(t) \end{bmatrix}$$

Sampai pada sampling ke t diperoleh hasil pengukauran $Z(1)$, $Z(2)$, ... $Z(t)$ yang secara keseluruhan dapat dituliskan sebagai berikut :

$$Z^t = \{ Z(t)\, Z(t-1)\, Z(t-2) \ldots Z(1) \}$$

Tujuan dari identifikasi adalah memperoleh model dari data-data Z, dengan jalan menentukan harga-harga parameter model yang

disusun dalam vektor parameter Q. Jadi identifikasi adalah untuk menentukan harga vektor Q.

Dengan demikian permasalahan identifikasi adalah menentukan pemetaan dari data Z^t ke parameter model Q :

$$Z^t \rightarrow \hat{Q}(t;Z^t)$$

Disini harga adalah diestimasi pada tiap t. Pada prinsipnya $Q(t)$ tetap merupakan fungsi dari data-data yang terdahulu. Tetapi karena ruang memori dan waktu komputasi terbatas, maka harus dibatasi sehingga $Q(t)$ dapat dibentuk. Oleh karena itu kita harus memperkecil banyaknya data hasil observasi ke dalam memory pembantu $S(t)$ dengan ukuran tertentu.

Matrik $S(t)$ diperbarui berdasarkan algoritma yang berstruktur sebagai berikut :

$$\hat{Q}(t) = F(\hat{Q}(t-1), S(t), Z(t))$$

$$S(t) = H(S(t-1), \hat{Q}(t-1), Z(t))$$

Dengan $F(.,.,.)$ dan $H(.,.,.)$ adalah fungsi yang telah diketahui, terlihat bahwa estimasi $Q(t)$ dibentuk dari data sekarang, hasil estimasi terdahulu dan variabel pembantu $S(t)$.

Informasi yang perlu disimpan pada saat t adalah $\{Q(t), S(t)\}$, harga dari informasi ini diperbarui dengan algoritma yang tetap dengan jumlah operasi tidak bergantung pada waktu t.

Sesungguhnya kami telah memberikan kepadamu nikmat yang banyak. Maka dirikanlah sholat karena Tuhanmu dan berkorbanlah. Sesungguhnya orang yang membencimulah yang putus sejarahnya. (Q.S. Al Kautsar:1-3)

BAB III
PERENCANAAN DAN PEMBUATAN ALAT

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

Pada bab ini dibahas perencanaan perangkat keras pengatur kecepatan motor induksi 3 fasa menggunakan sistem kontrol adaptif yang meliputi modul pengontrol, transfer data, identifikasi, sistem adaptif dan perangkat lunak pengendali sistem.

3.1 Uraian Umum

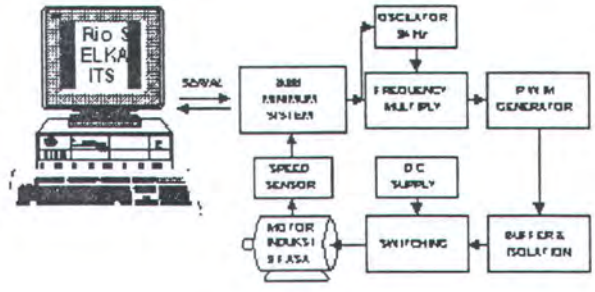
Sistem yang dirancang digunakan untuk mengontrol kecepatan dan respon terhadap perubahan beban pada motor induksi 3 fasa. Pengaturan kecepatan tersebut dilakukan dengan jalan mengendalikan frekuensi yang disalurkan ke motor secara PWM.

Pengendalian dilakukan dengan mengubah arus bolak-balik dengan frekuensi 60 Hz menjadi arus bolak-balik dengan frekuensi yang dapat dikontrol. Arus bolak-balik yang dihasilkan tidak berupa sinus murni, tetapi merupakan PWM. Arus bolak-balik dari jala-jala terlebih dahulu disearahkan oleh suatu rangkaian penyearah. Dari arus searah yang dihasilkan dibuat sinyal PWM dengan menggunakan rangkaian swiching yang dikontrol secara PWM.

Untuk tujuan pengontrolan, maka kecepatan putar motor dikendalikan oleh μP 8088 sistem minimum yang mendapat input dari sebuah komputer IBM PC AT sebagai unit pengontrol.

μP 8088 memantau secara on line putaran motor, kemudian mengirim hasilnya ke komputer untuk diproses dan hasil proses dikirim kembali ke μP 8088 dan digunakan untuk menentukan besarnya frekuensi serta tegangan yang harus diberikan ke motor.

Blok diagram dari sistem pengatur kecepatan motor induksi dengan metoda PWM dan memakai kontrol adaptif diperlihatkan pada gambar 3.1.



Gambar 3.1
Blok diagram pengatur kecepatan motor induksi secara PWM memakai kontrol adaptif.

3.2 Perangkat Keras

Perangkat keras yang digunakan untuk mengendalikan motor induksi 3 fasa tersusun dari beberapa bagian. Adapun keseluruhan dari masing-masing bagian tersebut dirancang sebagai berikut :

3.2.1 Motor Induksi 3 Fasa

Motor induksi yang dikontrol adalah motor induksi 3 phasa produksi Mitsubishi Electric Japan dengan spesifikasi :

Type	:	SB - ER	Ph	:	3
Out	:	25 Watt	Rpm	:	1400
Volt	:	220 / 380 V	Cycles	:	50 Hz
Amps	:	0,44 / 0.25 A			

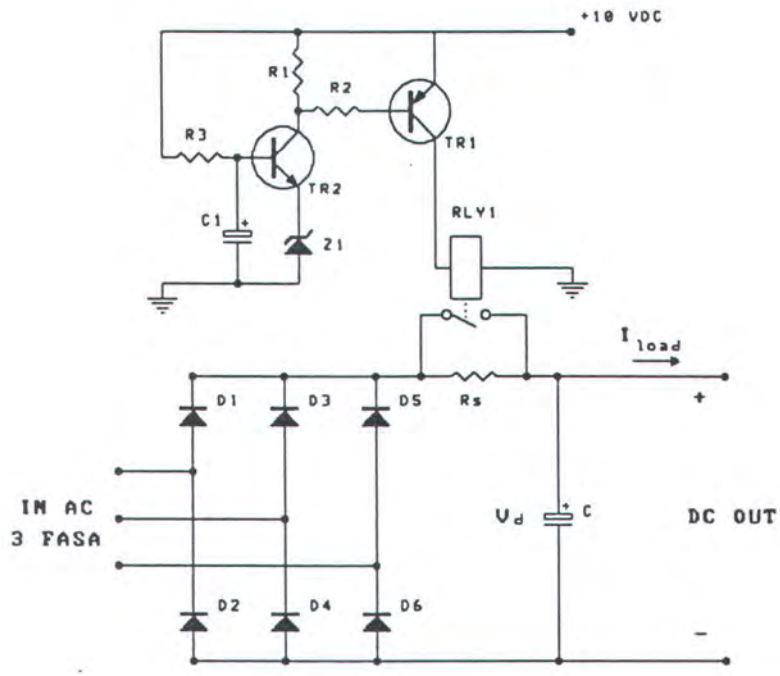
Motor induksi tersebut adalah motor induksi jenis rotor sangkar dengan jumlah kutup 4.

3.2.2 Penyearah Arus

Daya yang disalurkan ke motor diperoleh dari arus searah yang dibuat menjadi arus bolak-balik melalui proses switching. Dengan mengatur waktu switching secara PWM maka bisa dilakukan pengaturan kecepatan motor secara PWM.

Arus searah yang akan diswitching diperoleh dari tegangan jala-jala yang disearahkan. Karena motor induksi yang dikontrol adalah motor tiga fasa, maka untuk memperoleh torsi penuh pada motor, penyearahan juga dilakukan pada tegangan tiga fasa. Jika tidak diperlukan torsi penuh, maka penyearahan dapat dilakukan pada tegangan jala-jala satu fasa. Untuk menyearahkan tegangan tiga fasa diperlukan tiga pasang dioda, jadi diperlukan dua buah dioda untuk setiap fasa. Gambar 3.2 memperlihatkan rangkaian penyearah tiga fasa yang digunakan. Untuk rangkaian penyearah satu fasa hanya diperlukan dua pasang dioda. Jadi dengan melepaskan satu pasang dioda pada rangkaian gambar 3.2, diperoleh rangkaian penyearah satu fasa.

Pemilihan dioda didasarkan pada pertimbangan tegangan



Gambar 3.2
Rangkaian Penyearah 3 Fasa.

maksimum dari jala-jala harus lebih kecil dari pada tegangan dadal dioda yang digunakan, selain itu arus yang ditarik oleh motor harus lebih kecil dari arus maju dioda.

Dengan pertimbangan tersebut maka dipilih dioda dengan type 1N5407 yang memiliki rating tegangan 800 volt dan arus 3 amper. Arus searah yang terbentuk difilter menggunakan kapasitor sehingga diperoleh arus searah yang lebih rata.

Pada saat awal penyearah dihubungkan ke tegangan jala-jala kemungkinan kapasitor dalam kondisi tidak bermuatan / kosong. Karena itu jika diberi tegangan maka kapasitor akan berlaku sebagai hubung singkat untuk beberapa saat. Kondisi demikian bisa mengakibatkan

rusaknya dioda yang dipakai sebagai penyearah. Untuk mengatasi hal tersebut maka digunakan rangkaian tunda, sehingga pada saat penyearah dihubungkan ke tegangan jala-jala, resistor start Rs terhubung antara jembatan dioda dan kapasitor. Setelah waktu tunda terpenuhi maka relay menjadi aktif dan kontaknya digunakan untuk menghubung singkatkan resistor Rs. Dengan demikian antara jembatan dioda dan kapasitor menjadi terhubung langsung tanpa melalui Rs. Dengan adanya Rs ini maka arus start bisa dibatasi. Jika Rs yang terpasang bernilai 4 KΩ, maka arus awal yang melalui jembatan penyearah dapat dibatasi sebesar :

$$I_s = \frac{V_{Max}}{R_s} = \frac{\sqrt{2} \cdot 380}{4000} = 134\ mA$$

3.2.3 Inverter PWM Tiga Fasa

Inverter digunakan untuk mengkonversi tegangan DC menjadi Tegangan AC. Inverter yang dirancang adalah inverter jenis switching, dan sebagai komponen switchingnya digunakan transistor. Untuk mengisolasi bagian kontrol dengan inverter, maka digunakan optocoupler yang dipasang antara pembangkit PWM dengan inverter. Dengan demikian jika inverter rusak maka sistem kontroler terproteksi terhadap tegangan DC Bus yang digunakan pada inverter, sehingga kerusakan yang lebih besar pada sistem sebagai akibat rusaknya inverter bisa dicegah.

Pemilihan transistor sebagai komponen switching didasarkan pada alasan teknis yaitu kemampuan menyalurkan daya dan kecepatan switching dari transistor sudah memenuhi kriteria desain rangkaian yang

mecakup frekuensi kerja, arus maksimum , tegangan maksimum dan daya total. Selain itu komponen mudah didapat di pasaran dengan harga yang tidak terlalu tinggi sehingga bisa menekan biaya keseluruhan sistem.

Transistor-transistor Switching dikontrol sedemikian rupa sehingga berada pada keadaan terhubung dan terbuka pada waktu-waktu tertentu. Pengontrolan tersebut dilakukan melalui pembangkit sinyal PWM, dengan demikian pada output inverter tersedia daya PWM tiga fasa yang akan disalurkan ke motor. Gambar 3.3 memperlihatkan rangkaian inverter tiga fasa yang digunakan. Inverter tiga fasa sebenarnya adalah tiga buah inverter 1 fasa, karena itu pada gambar tersebut hanya diperlihatkan satu buah.

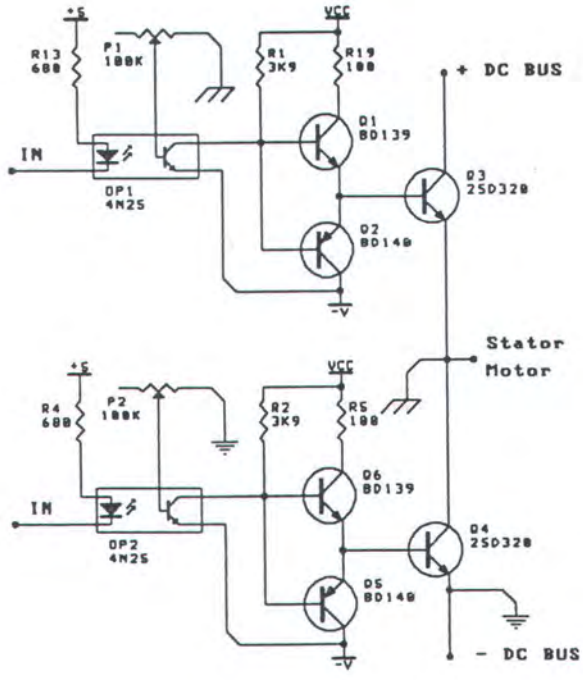
Komponen switching yang dipakai dipilih dengan memperhitungkan nilai tegangan dan arus yang terjadi pada pengontrolan kecepatan putar motor induksi. Tegangan maksimum terminal Kolektor dan Emitor dari transistor harus lebih besar dari tegangan DC output dari penyearah tiga fasa. Arus Kolektor maksimum dari transistor harus lebih besar dari arus maksimum yang diperlukan untuk menggerakkan motor. Dari spesifikasi motor induksi 3 fasa yang digunakan, diketahui arus nominal motor adalah sebesar 0,25 A pada tegangan efektif 380 volt. Arus start motor adalah 6 - 8 kali arus nominal yaitu sebesar 1,75 A.

Tegangan maksimum yang diberikan pada transistor switching adalah tegangan maksimum hasil penyearahan. Tegangan ini berasal dari tegangan AC tiga fasa yang disearahkan. Dengan demikian tegangan maksimum yang akan diterima oleh transistor adalah :

$$V_{DC} = \sqrt{2} \cdot 380 = 537{,}4\ Volt$$

Dengan memperhatikan batas tegangan dan arus yang digunakan, maka dipilih transistror type 2SD 320 sebagai transistor switchingnya. Adapun spesifikasi dari transistor tersebut adalah :

V_{CE}	Max	=	600 V
I_C	Max	=	5 A
H_{FE}	Min	=	10
P Tot		=	50 W



Gambar 3.3
Switching Transistor, Buffer dan Isolator.

Apabila transistor dipakai sebagai switching berarti dikerjakan pada dua daerah kerjanya yaitu daerah *saturasi* dan *cut off*. Kondisi terbuka atau tertutupnya switch transistor ditentukan oleh arus basis.

Dengan mengasumsikan arus maksimum yang melalui kolektror transistor (Ic Max) sebesar 0,5 Amper maka arus basis yang diperlukan untuk mencapai kondisi tersebut dihitung sebagai berikut :

$$I_B = \frac{I_C}{H_{FE}} = \frac{500}{10} = 50\ mA$$

Arus sebesar 50 mA terlalu besar jika harus disupply oleh opto-coupler, untuk itu output dari optocoupler dikuatkan terlebih dahulu dengan menggunakan transistor. Jika digunakan transistor BD 139 yang mempunyai $H_{FE\ Min}$ = 40 maka arus sebesar 50 mA yang diperlukan untuk membuat saturasi transistor switching adalah merupakan arus emiter dari transistor BD 139. Resistor yang terhubung pada kolektor dapat dihitung dengan pendekatan Ic= Ie= 50mA, sehingga diperoleh nilai 100Ω. Arus basis dari transistor BD 139 yang diperlukan adalah sebesar 1,22 mA. Dengan mengacu pada gambar 3.3, maka nilai resistor R yang terpasang pada terminal kolektor optocoupler dapat dihitung sebagai berikut :

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

$$= \frac{5\ Volt}{1{,}22\ mA} = 4098{,}4\ \Omega$$

Dengan memperhatikan komponen yang tersedia di pasaran maka diambil nilai terdekat yang tersedia yaitu resistor sebesar 3,9 KΩ. Optocoupler yang digunakan adalah 4N25 mempunyai kemampuan

mengisolasi tegangan sampai 2500 Volt dan mempunyai frekuensi pacung 300 KHz. Arus maju Led harus dibatasi maksimum sama dengan arus yang mampu diterima oleh output 7404 pada kondisi low yang digunakan pada pembangkit PWM. Jika arus tersebut diambil sebesar 5 mA dan tegangan maju Led adalah 1,7 Volt maka nilai resistor yang terhubung pada Led dihitung sebagai berikut :

$$R = \frac{5-1,7}{5.10^{-3}} = 660\ \Omega$$

3.2.4 Pembangkit Sinyal PWM

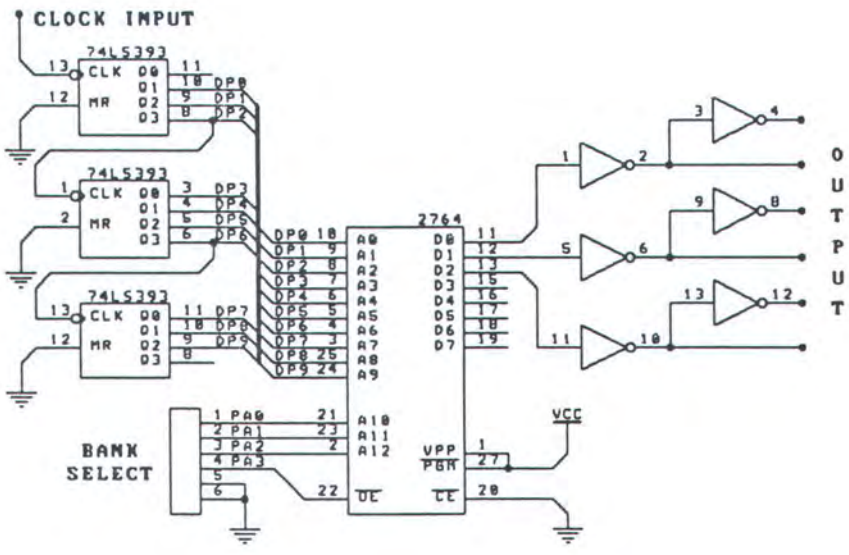
Sinyal PWM sinusoida bisa dihasilkan oleh komparator dengan input berupa sinyal segitiga dan sinusoida. Frekuensi dasar sinyal PWM yang dihasilkan ditentukan oleh frekuensi dari sinyal sinusoida. Pada tugas akhir ini perancangan rangkaian penghasil sinyal PWM dilakukan dengan terlebih dahulu melakukan simulasi komparator dengan input sinyal segitiga dan sinyal sinusoida secara grafik. Simulasi dilakukan pada sinyal sinusoida dalam satu perioda. Karena motor yang dipakai membutuhkan tegangan tiga fasa, maka dilakukan penggambaran 3 buah sinyal sinus yang masing-masing berbeda fasa 120 derajat pada kertas grafik sepanjang 1024 kolom. Ketiga buah sinyal tersebut kemudian dibandingkan dengan sinyal segitiga yang sama fasanya. Dari gambar pada grafik diperoleh data berupa PWM sinusoida dan selanjutnya data tersebut ditulis dalam Eprom. Data tiap kolom pada grafik dituliskan dalam Eprom 1 byte, dengan demikian sinyal

PWM sinusoida dalam satu periode memerlukan Eprom sebesar 1024 byte (1 Kbyte).

Untuk memperoleh data sinyal PWM sinusoida yang tersimpan dalam Eprom, maka address pin Eprom dihubungkan pada Pencacah yang selalu mencacah naik. Dengan demikian jika pencacah tersebut diberi input clock yang konstan maka data dari Eprom tersebut keluar dengan interval waktu yang konstan pula, hal ini menyebabkan diperolehnya sinyal PWM dengan frekuensi tertentu. Jika input clock dari pencacah diubah, maka sinyal PWM yang diperoleh juga akan berubah frekuensinya. Frekuensi clock dari pencacah tersebut akan dikendalikan oleh μP 8088 sehingga diperoleh sinyal PWM dengan frekuensi dan tegangan terprogram yang dipakai untuk mengatur kecepatan motor induksi 3 fasa.

Address A_0 - A_9 dari Eprom merupakan bit-bit yang harus selalu tercacah naik dengan frekuensi tertentu. Untuk kepentingan tersebut digunakan pencacah berupa IC TTL 74LS393 yaitu Dual Binary Counter. Gambar 3.4 memperlihatkan rangkaian pembangkit sinyal PWM yang dirancang.

Data pada Eprom hanya menempati 3 bit setiap bytenya, masing-masing 1 bit untuk satu fasa, sedangkan rangkaian switching yang dirancang menggunakan 2 transistor setiap fasanya dan waktu on dari transistor tersebut selalu bergantian, karena itu diperlukan dua buah logik yang saling berkomplemen setiap fasa. Karena itu output dari Eprom diberi inverter sehingga diperoleh dua logik yang berlawanan untuk setiap bitnya. Inverter yang dipakai adalah 74HC04, yaitu Hex Inverter.



Gambar 3.4
Pembangkit Sinyal PWM Sinusoida.

3.2.5 Pengatur Frekuensi PWM

Untuk mengeluarkan data sinyal PWM yang tersimpan pada Eprom, dilakukan dengan mencacah naik address pin A_0 - A_9 dari Eprom. Untuk tujuan tersebut maka pin A_0 - A_9 dihubungkan pada pencacah biner 10 bit yang diwujudkan dengan mengkaskade tiga pencacah biner 4 bit. Dengan mengatur frekuensi clock dari pencacah tersebut akan diperoleh sinyal PWM yang terkontrol frekuensinya.

3.2.5.1 Pembangkit Frekuensi Dasar PWM

Data sinyal PWM sinusoida dalam satu perioda menempati Eprom sebesar 1 Kbyte atau 1024 byte, karena itu untuk memperoleh sinyal PWM sinusoida dengan frekuensi 1 Hz diperlukan clock sebesar 1024 Hz.

Kecepatan putar sinkron motor induksi semakin tinggi jika

frekuensi tegangan naik. Hubungan tersebut ditunjukkan oleh persamaan dibawah ini :

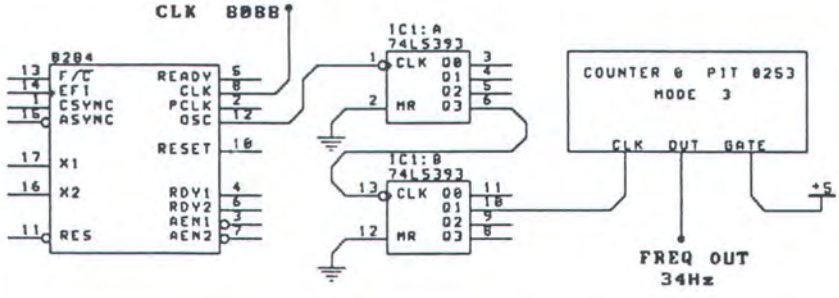
$$n_s = \frac{120f}{p}$$

Pada tugas akhir ini digunakan motor induksi dengan kutub (p) berjumlah empat, karena itu persamaan tersebut menjadi :

$$n_s = 30f$$

Ini berarti bahwa setiap kenaikan frekuensi sebesar 1 Hz menyebabkan putaran sinkron bertambah sebesar 30 Rpm.

Dengan mengacu pada persamaan tersebut, maka rangkaian pembangkit sinyal PWM harus dapat menghasilkan sinyal PWM dengan perubahan frekuensi sebesar 1/30 Hz tiap stepnya, dengan demikian tidak terjadi lonjakan kenaikan putaran motor yang tinggi yang disebabkan oleh perubahan frekuensi.



Gambar 3.5
Pembangkit Frekuensi 34 Hz.

Karena sinyal PWM sinusoida 1 Hz dihasilkan oleh clock sebesr 1024, maka untuk menghasilkan sinyal PWM sebesar 1/30 Hz diperlukan clock sebesar (1/30) x 1024 = 34,133 Hz. Frekuensi sebesar 34 Hz ini kemudian dipakai sebagai Base Clock dari rangkaian pengali frekuensi. Untuk menghasilkan frekuensi sebesar 34 Hz dilakukan dengan membagi frekuensi oscilator minimum sistem 8088 menggunakan pencacah biner 6 bit. Dengan demikian diperoleh faktor pembagian sebesar 64 dan diperoleh frekuensi sebesar 223,722 KHz. Frekuensi ini kemudian dipakai sebagai input Counter 0 dari PIT 8253 yang digunakan. Untuk mendapatkan frekuensi 34 Hz, maka Counter 0 dari PIT 8253 dikerjakan pada mode 3 yaitu *Square Wave* dengan *initial count* yang dituliskan adalah 6554, dengan demikian diperoleh frekuensi output dari Counter 0 tersebut sebesar 34 Hz yang akan dipakai sebagai input dari rangkaian pengali frekuensi. Gambar 3.5 memperlihatkan rangkaian penghasil frekuensi dasar untuk pembangkit sinyal PWM yang digunakan.

3.2.5.2 Pengali Frekuensi Terprogram

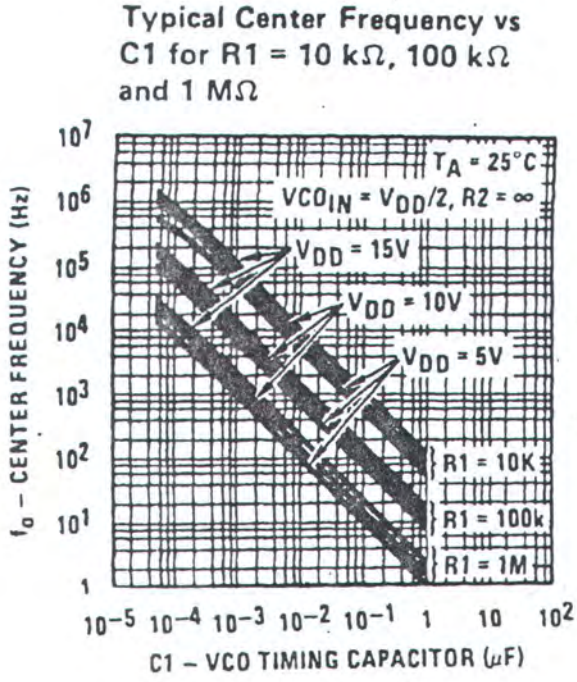
Dengan pengali frekuensi terprogram, maka frekuensi input dari rangkaian pembangkit sinyal PWM bisa dibuat variabel sehingga frekuensi output yang berupa sinyal PWM juga menjadi variabel.

Jika pada pengali frekuensi diberi sinyal input dengan frekuensi f maka akan diperoleh sinyal output dengan frekuensi nf dimana n adalah merupakan bilangan bulat tertentu. Dengan mengatur besar kecilnya faktor pengali n maka akan diperoleh frekuensi output sesuai dengan yang

diperlukan. Rangkaian pengali frekuensi yang yang dipakai dirancang menggunakan IC PLL 4046, dimana untuk mendapatkan faktor pengali diwujudkan dengan menggunakan pembagi *n* yang dihubungkan pada pin 3 dan pin 4 dari IC PLL 4046. Dengan demikian jika yang dihubungkan adalah pembagi / pencacah terprogram maka didapat rangkaian pengali frekuesi yang dapat diprogram.

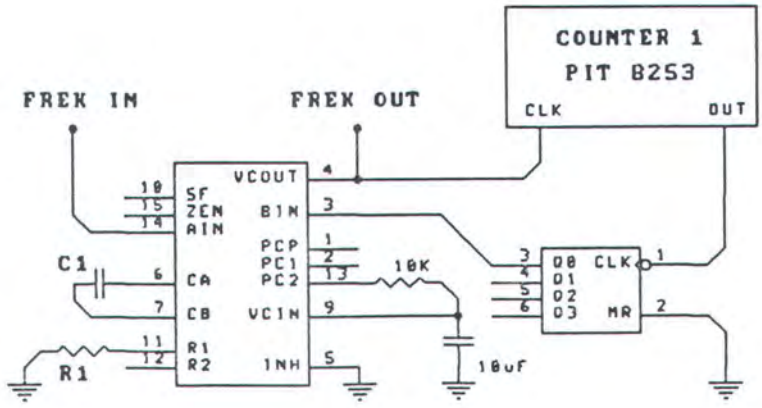
Pembagi terprogram yang dihubungkan pada pin 3 dan 4 IC 4046 adalah Counter 1 PIT 8353 yang dikerjakan pada mode 2 yaitu *Rate Generator*. Pada mode ini PIT berlaku sebagai pembagi *n*, dimana *n* adalah nilai initial counting yang dapat diubah dengan jalan menuliskan kembali setiap saat. Counter diinititalisasi untuk 2 Byte Read/Write, karena itu untuk memperoleh faktor pembagi *n*, diperlukan 2 kali siklus penulisan. Siklus pertama adalah LSB dan berikutnya adalah MSB.

Sebelum IC 4046 dapat digunakan, terlebih dahulu harus ditentukan frekuensi minimum dan frekuensi maksimum, dengan demikian dapat dihitung frekuensi tengahnya (*fo*). Dengan ditentukannya *fo* maka nilai komponen luar yang digunakan dapat ditentukan. Besarnya frekuensi tengah *fo* sama dengan frekuensi maksimum ditambah frekuensi minimum dibagi dua.Karena IC PLL 4046 dikerjakan tanpa memakai ofset ($R_2 = \sim$), maka mempunyai frekuensi kerja minimum 0 Hz. Jika frekuensi maksimum ditentukan sebesar 300 KHz, maka diperoleh nilai frekuensi tengah *fo* sama dengan 150 KHz. Nilai R_1 dan C_1 ditentukan dengan menggunakan grafik pada gambar 3.6.



Gambar 3.6
Grafik Penentuan Komponen Luar IC 4046.

Jika digunakan R1 sebesar 10 KΩ dan catu tegangan dari IC PLL 4046 dipakai sebesar 5 volt, maka diperoleh nilai C1 sekitar 300pF, sebagai harga pendekatan dipakai C1 sebesar 270 pF. Gambar 3.7 memperlihatkan rangkaian pengali frekuensi terprogram yang dirancang.



Gambar 3.7
Pengali Frekuensi Terprogram 16 Bit.

3.2.6 Pemantau Kecepatan Putar

Tinggi rendahnya frekuensi sumber menentukan kecepatan putar dari motor induksi. Semakin tinggi frekuensi sumber yang diberikan pada motor menyebabkan semakin cepat motor berputar. Pada motor induksi terdapat slip antara putaran rotor dengan medan putar stator, semakin besar beban yang terhubung pada rotor mengakibatkan slip yang terjadi semakin besar. Karena itu pemantauan kecepatan putar motor induksi tidak dapat dilakukan melalui frekuensi sumber, sehingga harus dilakukan secara langsung dengan mengukur putaran rotornya.

3.2.6.1 Metoda Pemantauan Kecepatan

Pemantauan kecepatan dilakukan dengan memakai optocoupler jenis pantul, karena itu poros motor diberi warna gelap kemudian pada bagian tertentu diberi warna terang yang digunakan sebagai pemantul dari optocoupler. Kecepatan putar motor dipantau selama poros berputar satu putaran. Untuk melakukan hal tersebut, maka pada saat optocoupler dilewati warna terang dibangkitkan sinyal interrupt yang mengaktifkan counter, sampai pada suatu saat dimana warna terang semula kembali melewati optocoupler dan sinyal interrupt kedua dibangkitkan guna menonaktifkan counter.

Kecepatan putar motor diperoleh dengan jalan membaca isi counter, kemudian isi tersebut dibandingkan dengan harga yang telah dikalibrasi sehingga diperoleh kecepatan putar motor. Hubungan yang digunakan untuk mendapatkan kecepatan putar motor adalah :

$$Rpm = \frac{Kal}{Count} Speed$$

Dimana :

Rpm : Kecepatan putar motor saat dipantau.

Kal : Isi counter pada saat kalibrasi.

Speed : Kecepatan yang digunakan pada saat dilakukan kalibrasi (*Base clock*).

Count : Isi counter pada saat dilakukan pemantauan kecepatan.

Counter yang digunakan pada pemantauan kecepatan adalah counter 2 dari PIT 8253 yang dikerjakan pada mode 0 yaitu *Interrupt On Terminal Count.* Input Clock dari counter 2 ini sama dengan input pada counter 0 yaitu sebesar 223,722 KHz yang diperoleh dari pin 12 IC penghasil clock 8284 yang dipakai pada minimum sistem 8088 setelah terlebih dulu dibagi dengan 64.

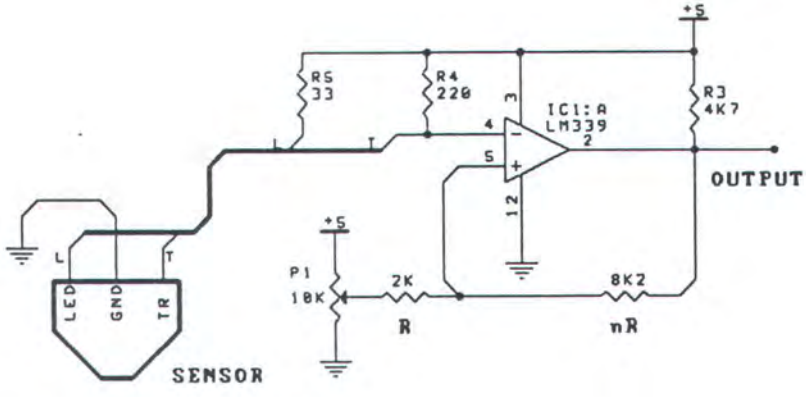
3.2.6.2 Detektor Level Tegangan

Output dari sensor kecepatan (optocoupler) tidak mampu membangkitkan sinyal interrupt , hal ini disebabkan level tegangan yang dihasilkan oleh sensor bukan level tegangan TTL, karena itu diperlukan suatu rangkaian yang dapat mengahasilkan output level tegangan TTL dengan input level tegangan dari sensor kecepatan. Rangkaian yang diperlukan tersebut diwujudkan dengan rangkaian Detektor level tegangan membalik. Untuk memperkecil pengaruh noise yang dihasilkan oleh motor

ataupun sumber-sumber noise yang lain, maka digunakan detektor yang mempunyai histerisis seperti yang ditunjukkan pada gambar 3.8.

Pada rangkaian yang digunakan, tegangan histerisis dibuat sebesar 1 Volt. Untuk catu tegangan tunggal, maka hubungan antara tegangan histerisis dan nilai resistor yang dipakai adalah sebagai berikut :

$$V_H = \frac{V_{Sat}}{n+1}$$



Gambar 3.8
Sensor Kecepatan Putar.

Jika V_{Sat} = 5 volt, maka diperoleh harga n = 4. Dengan mengambil harga R = 2 KΩ, maka sebagai pendekatan dipakai nR = 8,2 KΩ sehingga tegangan histerisis menjadi 0,98 volt. Tegangan V_{UT} dan V_{LT} besarnya tergantung pada V_{Ref}, karena itu V_{Ref} dibuat variabel dengan jalan memasang resistor variabel (Multitune). Output dari sensor saat aktif adalah sebesar 2,88 volt, karenanya penentuan V_{LT} harus lebih besar dari 2,88 volt. Jika ditentukan V_{LT} = 3,5 volt, maka V_{Ref} harus diset sebesar 4,46 volt. Penentuan harga V_{Ref} berdasarkan persamaan berikut :

$$V_{Ref} = \frac{n+1}{n} V_{LT}$$

$$= \frac{5,1}{4} 3,5 \quad = 4,46\ volt$$

Dengan ditentukannya V_{LT} = 3,5 volt, maka untuk V_H = 0,98 volt diperoleh V_{UT} = 4,48 volt. Data output optocoupler yang diperoleh dari percobaan adalah :

- Sensor aktif = 2.88 volt.
- Sensor non aktif = 4,97 volt.

Dari kedua data tersebut maka penentukan level tegangan telah sesuai, sehingga detektor level tegangan akan dapat bekerja dengan baik untuk kondisi output sensor optocoupler.

3.2.7 Pencacah dan Pewaktu

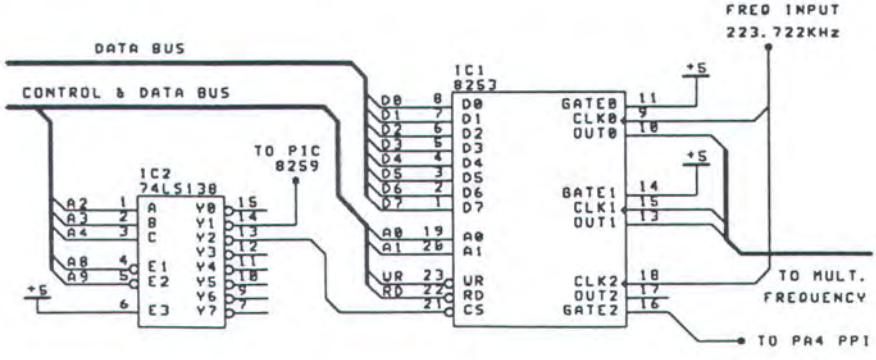
Dalam sistem alat yang dibuat pencacah yang digunakan diwujudkan dengan memakai *Programmable Interrupt Timer* 8253, yang mempunyai tiga buah pencacah, yaitu pencacah 0, pencacah 1, dan pencacah 2 yang bekerja bebas tidak tergantung satu dengan yang lainnya. Terdapat enam mode yang dapat dipakai sesuai dengan penggunaan pencacah, adapun mode dan aplikasi yang digunakan untuk ketiga pencacah didalam sistem adalah sebagai berikut :

- Pencacah 0 : Dikerjakan pada mode 3, digunakan untuk membangkitkan frekuensi dasar bagi rangkaian pengali frekuensi.

- Pencacah 1 : Dikerjakan pada mode 2, sebagai pembagai *n* untuk faktor pengali bagi rangkaian pengali frekuensi.
- Pencacah 2 : Dikerjakan pada mode 0 dan digunakan pada pemantauan kecepatan motor.

Agar modul PIT 8253 dapat beroperasi pada minimum sistem 8088 yang digunakan, maka *port address* harus ditentukan terlebih dahulu pada saat pembuatan modul. PIT 8253 memerlukan empat address dalam beroperasinya, untuk itu digunakan address 08h - 0Bh sebagai *port address* dari PIT tersebut. Untuk mendapatkan address tersebut diperlukan decoder yang mempunyai fungsi membedakan antara komponen atau modul yang satu dengan yang lain yang terhubung pada minimum sistem.

Decoder yang dipakai dirancang menggunakan IC 74LS138 yang mempunyai 6 saluran input dan 8 saluran output, dengan demikian decoder 74LS138 ini mampu membedakan address beberapa modul sekaligus.



Gambar 3.9
PIT 8253 dan Address Decoder.

Untuk mendapatkan address 08h - 0Bh maka input A, B, dan C 74LS138 dihubungkan dengan *address bus* A_2, A_3, dan A_4 minimum sistem yang digunakan. Dengan konfigurasi tersebut, maka address 08h - 0Bh yang diperlukan untuk PIT tersedia pada output 74LS138 pin 13 (Y_2). Gambar 3.9 memperlihatkan rangkaian PIT 8253 bersama addrress decoder yang digunakan.

3.2.8 Sistem Masukan dan Keluaran (PPI 8255)

Agar μP 8088 dapat berkomunikasi dengan rangkaian ataupun peralatan lain, diperlukan media yang menghubungkan antara keduanya, karena itu diperlukan sistem masukan dan keluaran yang memungkinkan bagi μP 8088 untuk melakukan komunikasi baik per bit ataupun per byte. Sistem masukan dan keluaran tersebut dapat diwujudkan dengan menggunakan PPI 8255, proses komunikasi yang diperlukan dapat terlaksana. Seperti halnya dengan modul-modul yang lain, maka untuk mengoperasikan PPI 8255 ini terlebih dahulu harus ditentukan *port address*nya. PPI 8255 mempunyai 3 buah *port* 8 bit yaitu *port A*, *port B*, dan *port C*. Pada alat yang dibuat seluruh proses masukan dan keluaran dapat dilakukan oleh satu buah PPI, karena itu digunakan PPI yang sudah disediakan pada minimum sistem yang mempunyai address port 00h - 03h.

Perincian penggunan port PPI untuk masing-masing bit adalah sebagai berikut :

- Port A sebagai output meliputi :
  - PA_0-PA_3 : Dipakai sebagai Bank select dan Output enable rangkaian

pembangkit PWM.

- PA4 : Dipakai sebagai Gate control pada counter 2 PIT 8253.
- PA5-PA7 : Dipakai sebagai control line pada display LCD.

- Port B Sebagai output dipakai sebagai data line pada display LCD.
- Port C Upper sebagai output dipakai pada Keypad.
- Port C Lower Sebagai input dipakai pada Keypad.

3.2.9 Sistem Perantara Keypad dan Display

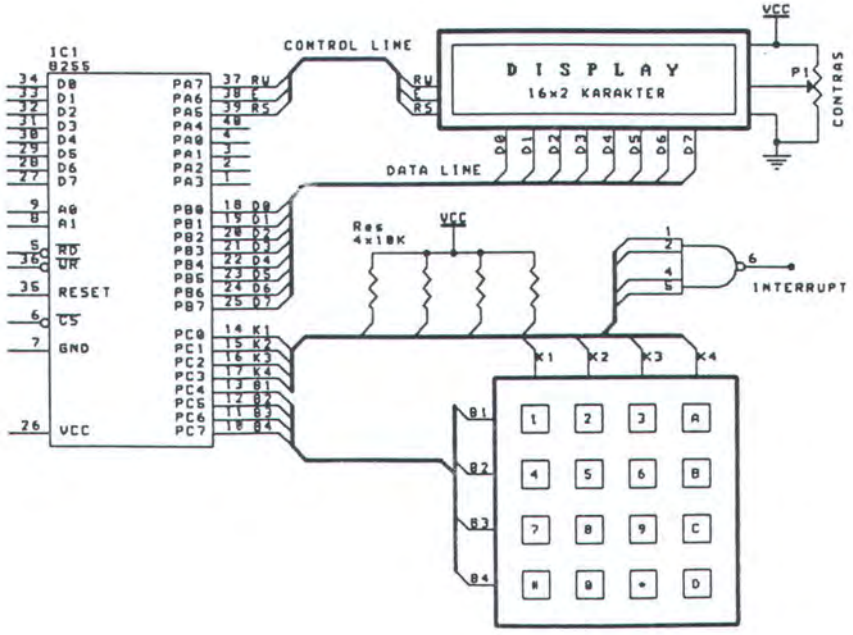
Keypad yang digunakan adalah berbentuk matrik 4x4, karena itu diperlukan 8 bit untuk mengoperasikannya. Kedelapan bit tersebut masing-masing adalah 4 bit untuk baris dan 4 bit untuk kolom. Pembacaan tombol yang ditekan dilakukan secara interrupt hal ini dapat diwujudkan karena keempat kolom keypad selain terhubung ke PPI juga dihubungkan pada input Nand gate yang outputnya dihubungkan pada Intr 2 PIC 8259. Dalam kondisi tidak ada tombol yang ditekan, maka semua kolom menjadi tinggi dan output Nand gare rendah. Jika ada tombol yang ditekan, maka pada salah satu kolom akan menjadi rendah, dan menyebabkan output Nand gate menjadi tinggi sehingga membangkitkan sinyal interupsi.

Proses pembacaan kode tombol yang ditekan dimulai saat sinyal interupsi yang dibangkitkan dikenali oleh µP 8088. Mula-mula pembacaan dilakukan untuk mengetahui posisi baris tombol yang ditekan, setelah ditemukan pembacaan dilanjutkan untuk mengetahui posisi kolom tombol yang ditekan. Dengan diketahuinya posisi baris dan kolom tombol yang

ditekan, maka diperoleh code tombol tersebut. Kemudian kode yang diperoleh digunakan sebagai address memory tempat karakter yang mewakili tombol yang ditekan tersimpan sehingga karakter dapat dibaca.

Untuk mengoperasikan alat yang dibuat, maka disusun menu operasi yang ditampilkan dalam display. Selain untuk menampilkan menu, display juga digunakan untuk menampilkan status alat dan putaran motor.

Sistem display yang dipakai adalah LCD yang mampu menampilkan 16x2 karakter. Selain itu sistem display yang digunakan juga dilengkapi dengan memory sebesar 80 byte yang digunakan untuk menyimpan karakter yang akan ditampilkan. Karakter yang telah tersimpan dalam memory display dapat ditampilkan dengan jalan memprogram sistem display. Pemrograman tersebut antara lain display geser kanan/kiri, set address awal yang akan ditampilkan, cursor home dll.



Gambar 3.10
Sistem Keypad dan Display.

Untuk keperluan pemrograman, maka pada sistem display yang digunakan selain terdapat jalur data juga terdapat jalur kontrol. Sebagai jalur kontrol disediakan 3 bit, masing-masing adalah *E*, *RS* dan *RW*. Jalur *E* digunakan untuk mengaktifkan modul, sedangkan *RS* digunakan untuk memilih register yaitu register instruksi jika *RS* low dan register data jika *RS* high. Jalur *RW* digunakan untuk menentukan operasi baca (*RW* high) dan operasi tulis (*RW* low). Sistem keypad dan display yang digunakan ditunjukkan pada gambar 3.10. Terlihat pada gambar tersebut untuk melayani keypad digunakan Port C PPI 8255 dan Intr 2 pada PIC 8259, sedangkan untuk sistem display digunakan Port B sebagai jalur data dan Port A (PA_5 - PA_7) sebagai jalur kontrol.

3.3 Perangkat Lunak

Perangkat keras yang telah dirancang tidak dapat beroperasi tanpa dilengkapi dengan perangkat lunak. Terdapat dua perangkat lunak yang diperlukan, pertama adalah perangkat lunak untuk mengendalikan 8088 minimum sistem dan peripheral yang terhubung, dalam hal ini digunakan bahasa Assembly. Kedua adalah perangkat lunak untuk mengendalikan PC guna pengolahan data yang dikirim oleh 8088 minimum sistem dan menampilkan hasil-hasil pengontrolan pada monitor. Perangkat lunak yang digunakan untuk mengolah data adalah bahasa C.

Perangkat lunak yang digunakan untuk mengendalikan sistem baik 8088 minimum sistem dan peripheral yang terhubung maupun perangkat lunak pengendali PC secara umum terdiri dari beberapa bagian

yaitu: Initialisasi, Input data, Pengolah data dan Pengirim/penerima data.

3.3.1 Perangkat Lunak 8088 Minimum Sistem

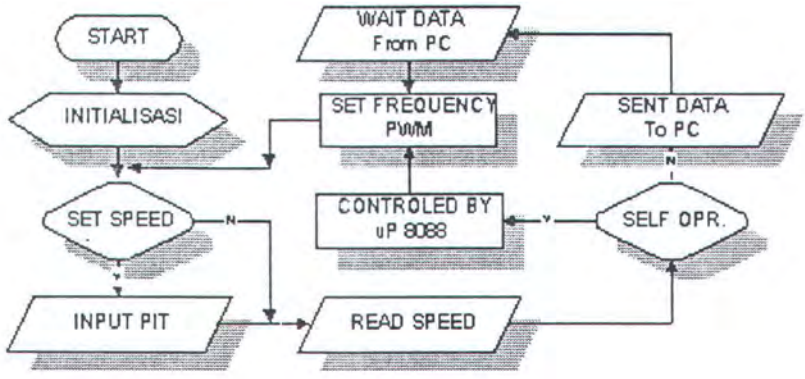
Initialisasi pada 8088 minimum sistem diperlukan agar peripheral-peripheral yang terhubung ke minimum sistem 8088 dapat beroperasi. Peripheral yang diinitialisasi meliputi UART 8250, PIT 8253, PPI 8255 dan PIT 8259. Untuk masing-masing peripheral tersebut initialisasi dilakukan sebagai berikut :

- Initialisasi UART 8250 meliputi :
  - Baut Rate : 9600 Bps.
  - Format data : 8 bit satu stop bit, dengan parity.
  - Interrupt Reg : Non interrupt.

- Initialisasi PIT 8253 meliputi :
  - Mode kerja tiap counter, Counter 0 mode 3, Counter 1 mode 2 dan Counter 3 mode 0.
  - Proses Read/Write data 2 byte (LSB dan MSB).
  - Menentukan initial load conting.

- Initialisasi PPI 8255 meliputi :
  - Mode kerja : menggunakan Mode 0.
  - Fungsi Port : PA output, PB output, PC lower input, PC upper output.

- Initialisasi PIT 8259 meliputi :
  - Mode kerja : Fully Nested.
  - Nomor interrupt untuk IRQ 0 : Int 08h.
  - Menentukan interrupt yang aktif.



Gambar 3.11
Proses Pengontrolan Pada 8088 Minimum Sistem.

Untuk keperluan input data dilakukan dengan memilih menu dan menekan tombol pada keypad sesuai dengan pilihan menu. Proses input data kecepatan dilakukan dengan terlebih dahulu memilih menu "SET SPEED". Setelah menu dipilih, maka akan tampil perintah untuk memasukkan harga kecepatan putar motor. Pada kondisi tersebut tombol yang valid adalah tombol 0 - 9, dan setiap penekanan tombol yang valid kodenya disimpan didalam memory sampai diakhirinya pemasukan data dengan menekan Enter berupa tombol $. Dengan ditekannya tombol $, maka kode tombol yang tersimpan di memory dibaca dan dikonversi ke

dalam satuan, puluhan, ratusan dan ribuan sesuai dengan urutan penekanan tombol, kemudian hasil konversi dijumlahkan sehingga diperoleh nilai kecepatan motor dalam Rpm. Untuk menggerakkan motor pada kecepatan yang telah dimasukkan, maka data dalam Rpm dimasukkan rumus sehingga diperoleh frekuensi yang harus dikeluarkan agar motor dapat berputar pada kecepatan yang dikehendaki. Frekuensi yang diperoleh tersebut merupakan harga yang ideal, karena itu frekuensi tersebut berubah selama proses sehingga tercapai kecepatan yang sesungguhnya.

Bagian pengolah data adalah bagian yang paling berperan dalam pengendalian sistem. Pada bagian ini dilakukan pembacaan data dari sensor kecepatan, kemudian diolah dan hasilnya dikirim ke PC. Selanjutnya PC akan mengolah data yang diterima dan hasilnya dikirimkan kembali ke 8088 minimum sistem dan digunakan untuk mengendalikan perubahan kecepatan yang terjadi. Jika sistem beroperasi secara mandiri, maka data hasil pembacaan sensor tersebut dikirim ke PC hanya untuk memonitor kecepatan motor. Jadi PC tidak mengolah dan mengirim balik data yang diterimanya. Pada 8088 minimum sistem data pembacaan kecepatan digunakan untuk mengetahui kondisi dari putaran motor dan diputuskan apakah putaran motor harus dinaikkan atau diturunkan, sehingga diperoleh kecepatan seperti yang disettingkan. Proses pengolahan data dan pengendalian sistem yang dilakukan oleh 8088 minimum sistem diperlihatkan pada gambar 3.11.

Bagian penerima/pengirim data digunakan pada saat μP 8088 melakukan komunikasi dengan PC. Komunikasi tersebut dilakukan jika

perbandingan hasil pembacaan kecepatan motor dan harga yang diset mempunyai error yang cukup tinggi sehingga diperlukan kebijaksanaan adaptasi untuk mengatasi perubahan kecepatan motor. Komunikasi yang dilakukan adalah secara serial dengan menggunakan standard RS-232 dan menggunakan Baut Rate 9600 bit per detik. Komunikasi juga dilakukan pada saat kecepatan motor diset baik melalui keyboard pada PC maupun melalui keypada pada alat yang dibuat. Pada kondisi ini komunikasi yang dilakukan bertujuan untuk menyamakan harga kecepatan yang disettingkan.

3.3.1.2 Perangkat Lunak PC-AT

Perangkat lunak yang digunakan pada PC-AT mempunyai struktur yang secara garis besar sama dengan pada 8088 minimum sistem, perbedaannya terletak pada proses yang dilakukan pada tiap bagian. Proses-proses yang dilakukan untuk tiap bagian adalah :

Pada bagian initialisasi, hal yang dilakukan adalah menginitialisasi UART 8250 dan initialisasi interrupt yang digunakan pada proses peneriamaan data. Initialisasi yang dilakukan pada UART 8250 secara umum sama dengan yang dilakukan pada 8088 minimum sistem, perbedaannya adalah diaktifkannya interrupt pada saat data masuk. Setelah UART 8250 diinitialisasi, hal selanjutnya yang harus dilakukan adalah membuka interrupt yang akan digunakan untuk komunikasi. Pada tugas akhir ini digunakan port Com1, dimana pada komputer PC port ini menggunakan Irq4. Instruksi yang digunakan untuk membukan Irq4 adalah sebagai berikut :

```
Mov  DX,21h
In   AL,DX
And  AL,11101111b
Out  DX,AL
```

Bagian input data digunakan untuk memasukkan harga kecepatan motor yang dikehendaki. Pemasukan tersebut dilakukan melalui pemilihan menu pada monitor menggunakan mouse. Data yang dimasukkan meliputi kecepatan motor, settling time, persen overshoot dan mode kerja dari sistem. Setelah proses pemasukan data berakhir maka PC akan mengirim data ke 8088 minimum sistem untuk memutar motor pada kecepatan yang dikehendaki.

Bagian pengolah data digunakan untuk memproses data kecepatan motor yang dikirimkan oleh μP 8088. Pengolah tersebut bertujuan untuk mendapatkan respon yang dikehendaki terhadap perubahan kecepatan yang terjadi pada motor dengan menggunakan *Gain Feedback* kontroler secara Adaptif. Output dari bagian ini adalah data kecepatan motor yang baru sebagai hasil dari kebijaksanaan adaptasi yang dilakukan.

Pada pengendalian motor AC 3 fasa ini harga variabel yang diinputkan adalah :

1. Kecepatan yang diinginkan (S_d).
2. Persen Overshoot yang diinginkan (Ovsht).
3. Settling time yang diinginkan (t_s).

Karena harga persen Overshoot dan Settling time diketahui, maka ξ dan ω_n dapat dihitung dengan persamaan berikut :

$$\xi = \sqrt{\frac{1}{1+(\frac{\pi}{Ln\,(Ovsht)})^2}}$$

$$\omega_n = \frac{4}{(\xi \,.\, t_s)}$$

Kemudian sistem pengaturan kecepatan motor induksi 3 fasa didekati dengan sistem orde 2, yang transfer functionnya adalah sebagai berikut :

$$G(s) = \frac{K}{S^2 + 2\xi\omega_n S + \omega_n^2}$$

Sistem yang dirancang adalah merupakan sistem digital, karenanya transfer fuction tersebut harus diubah ke dalam kawasan waktu diskrit dimana persamaannya adalah dalam bentuk fungsi z. Setelah ditransformasikan maka diperoleh :

$$G(z) = \frac{K}{2\omega_n} \cdot \frac{2ze^{-\xi\omega_n T}\, Sin(\omega_d T)}{z^2 - 2ze^{-\xi\omega_n T}\, Cos(\omega_d T) + e^{-2\xi\omega_n T}}$$

Dari identifikasi, diperoleh persamaan sistem secara umum :

$$\frac{Y(z)}{U(z)} = \frac{b_1 z}{z^2 + (a_1 - 1 - b_1 k_1)z + (b_1 k_1 - a_1)}$$

Dari kedua persamaan diatas harga ki dapat dihitung sebagai berikut:

$$b_1 k_1 - a_1 = e^{-2\xi\omega_n T} \quad \Rightarrow \quad k_1 = \frac{a_1 + e^{-2\xi\omega_n T}}{b_1}$$

Dengan demikian output kontroler / sinyal kontrolnya adalah :

$$U = (k_1 \times \textit{Kecepatan terukur}) + \textit{Kecepatan yang diinginkan.}$$

Apakah manusia itu mengira bahwa mereka dibiarkan (saja) mengatakan: "Kami telah beriman" sedang mereka tidak diuji lagi? (Q.S. Al Ankabut:2)

BAB IV
PENGUJIAN ALAT

BAB IV

PENGUJIAN ALAT

Agar sistem yang dirancang dapat bekerja dengan baik, maka perlu diadakan pengukuran, kalibrasi dan pengujian alat. Hal tersebut dilakukan pada bagian-bagian alat antara lain : sensor kecepatan putar, kalibrasi kecepatan putar pangukuran tegangan output 3 fasa dan bagian-bagian yang lain.

4.1 Pengukuran dan Seting Level Tegangan Sensor Kecepatan

Pengukuran dilakukan dengan tujuan mencari titik optimum dimana pada posisi tersebut sensor memberikan respon yang paling baik terhadap perubahan warna terang dan gelap dari poros motor. Respon terbaik sensor adalah tegangan paling rendah yang dapat dicapai saat sensor aktif. Tegangan tersebut diukur dan digunakan untuk menentukan seting level input dari rangkaian detektor level tegangan.

Dengan mengatur jarak antara sensor dan poros motor, maka diperoleh level tegangan output sensor terendah yaitu sebesar 2,88 volt. Sedangkan level tegangan saat sensor tidak aktif adalah sebesar 5 volt, dengan demikian rangkaian detektor tegangan harus mempunyai level input lebih besar dari 2,88 volt dan lebih kecil dari 5 volt agar rangkaian dapat mengetahui

apakah sensor dalam keadaan aktif atau tidak.

Detektor level tegangan yang dirancang mempunyai histerisis mendekati 1 volt, dimana V_{LT} dan V_{UT} dapat diset dengan mengatur tegangan referensi input melalui resistor variabel. Dengan mengacu pada level tegangan output dari sensor, maka V_{LT} diset sebesar 3,5 volt dengan demikian didapat V_{UT} sebesar 4,5 volt. Kedua nilai tersebut ternyata masih berada didalam jangkauan output sensor, karena itu detektor level tegangan akan dapat bekerja dengan baik dan pada outputnya akan tersedia tegangan pada level TTL sesuai dengan kondisi sensor kecepatan.

4.2 Pengujian Sistem Interrupt

Sistem interrupt digunakan pada proses pengukuran kecepatan, yaitu untuk mengaktifkan counter pada saat datangnya sinyal interrupt yang pertama dan menonaktifkan counter saat datangnya sinyal interrupt kedua.

Pengujian meliputi software yaitu routine pelayanan interrupt yang akan diakses saat datang sinyal interrupt, dan hard ware yaitu PIC 8253 yang mendapat input dari rangkaian detektor level tegangan.

Pengujian software dilakukan dengan memanggil routine pelayanan interrupt yang diuji melalui instruksi *Int n,* dimana n adalah nomor interrupt yang digunakan. Pada sistem yang dirancang, pengukuran kecepatan putar menggunakan *Irq1* dan *Irq2* dari PIC 8253. Karena *Irq0* dari PIC diinitialisasi pada interrupt no 8, maka *Irq1* dan *Irq2* masing-masing mempunyai nomor

interrupt 9 dan 10. Dengan demikian untuk menguji apakah routine interrupt yang digunakan untuk pegukuran kecepatan sudah berjalan dengan baik dilakukan dengan memakai instruksi INT 09h dan INT 0Ah. Jika routine sudah berjalan dengan baik maka saat diberikan instruksi INT 09h maupun INT 0Ah, program yang sedang dijalankan tidak akan terganggu ataupun membuat sistem macet.

Pengujian secara hard ware dilakukan setelah pengujian secara soft ware dapat berjalan dengan baik. Untuk keperluan pengujian secara hard ware, sebagai input untuk membangkitkan sinyal interrupt digunakan *Function Generator* dan frekuensinya diatur pada range putaran motor yang memungkinkan. Agar hasil pengujian secara hard ware dapat diketahui hasilnya, maka dibuat soft ware untuk membaca dan menampilkan ke monitor isi counter yang digunakan.

Dari hasil pengujian diporeh nilai yang ditampilkan di monitor semakin kecil jika frekuensi *Function Generator* semakin tinggi, ini berarti keseluruhan hardware dan routine interrupt yang digunakan sudah bekerja dengan baik dan siap untuk digunakan pada proses pengukuran kecepatan.

4.3 Kalibrasi Pengukur Kecepatan Putar

Kalibrasi ini dilakukan agar proses pengukuran kecepatan putar bisa memperoleh hasil pengukuran yang presisi. Metoda pengukuran kecepatan yang dirancang adalah dengan memantau poros motor dalam satu kali putaran,

dimana dalam selang waktu tersebut counter diaktifkan dan dibaca isinya setelah terjadi satu kali putaran. Kalibrasi bertujuan untuk mendapatkan harga referensi yang akan digunakan sebagai acuan dalam pengukuran.

Proses kalibrasi yang dilakukan menggunakan base time 60 Hz menggunakan IC MM 5369 dan kristal sebesar 3,579545 Mhz. Agar dapat mewakili kondisi sesungguhnya maka base time tersebut dibagi dua dengan menggunakan pecacah 1 bit, dengan demikian diperoleh clock sebesar 30 Hz atau sama dengan 1800 Rpm.

Dengan digunakan clock sebesar 30 Hz, maka diperoleh hasil pembacaan counter sebesar 7454. Nilai ini kemudian digunakan sebagai referensi, sehingga rumus yang digunakan pada pengukuran kecepatan seperti yang telah dibahas pada sub bab 3.2.6.1 menjadi :

$$Rpm = \frac{7454}{Count} . 1800$$

Rpm adalah kecepatan putar yang terukur, sedangkan *Count* adalah isi counter yang terbaca saat pengukuran.

4.4 Pengujian Sinyal PWM

Pengujian sinyal PWM dilakukan untuk mengetahui apakah fase ketiga sinyal PWM sudah benar. Hal ini bisa diketahui dengan mengamati pada osciloscope beda fase antara ketiga sinyal PWM. Sebelum pengujian beda fase dilakukan maka diadakan pengukuran rangkaian-rangkaian pembangkit sinyal

PWM, Buffer transistor daya dan Transintor daya (transistor swicthing).

Untuk mengetahui apakah pembangkit sinyal PWM sudah bekerja dengan baik dilakukan dengan melihat pada osciloscope output dari ke enam inverter seperti yang ditunjukkan pada gambar 3.4. Jika rangkaian bekerja dengan baik, maka pada osciloscope terlihat sinyal persegi (*Square Wave*) yang lebar pulsanya selalu berubah terus, yang berarti sinyal PWM sudah dapat dibangkitkan.

Untuk menguji apakah rangkaian buffer transistor daya sudah bekerja dengan baik maka output dari pembangkit sinyal PWM dihubungkan pada input buffer transistor daya. Jika output buffer transistor daya yang diamati pada osciloscope mempunyai bentuk seperti pada inputnya, maka dapat dipastikan rangkaian sudah bekerja dengan baik.

Langkah selanjutnya adalah menguji apakan fase dari sinyal PWM sinusoida yang dibangkitkan sudah benar, untuk itu rangkaian Buffer tansistor daya dihubungkan dengan Transistor daya. Sebagai sumber tegangan dari transistor daya digunakan power supply DC dari adaptor atau power supply DC yang lain. Tegangan yang digunakan cukup kecil saja, berkisar antar 5 - 10 volt. Dengan tegangan tersebut maka transistor daya cukup aman jika timing dari sinyal inputnya kurang tepat. Pada output transistor daya diberi beban hubung bintang yang dibuat dari 3 buah resistor. Pengamatan dengan osciloscope dilakukan dengan referensi titik tengah dari beban hubung bintang .

Dari pengukuran terlihat pada osciloscope bentuk gelombang

mendekati sinus dan mempunyai beda fase 120 dan 240 derajat, hal ini berarti seluruh rangkaian sudah bekerja dengan baik dan data PWM yang dibuat sudah benar.

4.5 Setting Timing Penyalaan Transistor daya dan Tegangan Antar Fasa

Timing penyalaan transistor daya merupakan hal yang sangat penting, sebab hal ini berhubungan dengan efisiensi daya yang disalurkan ke motor. Selain itu timing yang tidak tepat akan menyebabkan rusaknya transistor daya. Jadi harus diusahakan saat on dan off transistor switching setepat mungkin.

Pada rangkaian yang dibuat, penyalaan transistor daya dapat diatur dengan jalan mengatur multitune pada rangkaian buffer transistor daya. Terdapat 6 multitune yang harus di setting harganya, masing-masing dua buah untuk tiap fasa.

Pada pengukuran yang dilakukan, setting timing transistor dilakukan dengan memakai power supply DC Hawlett Pakard dengan spesifikasi :

Type : *6209B*
Volt : *0 - 320 V*
Amp. : *0 - 0.1 A*

Tegangan yang digunakan adalah sebesar 40 volt, dengan arus yang sekecil mungkin, hal ini bertujuan untuk mengetahui drop tegangan yang terjadi jika timing kurang tepat.

Pengukuran dilakukan per fasa dan tanpa beban. Sebagai inputnya digunakan Fuction Generator yang dihubungkan pada pin 11, 12 dan 13 socket ROM pada rangkaian pembangkit PWM setelah ROM yang dipakai dilepas. Bentuk output dari transistor daya diamati pada osciloscope, dan frekuensi input diatur dari rendah sampai beberapa KHz. Sepanjang jangkauan frekuensi tersebut drop tegangan yang terjadi harus relatif sama, dengan demikian diperoleh tegangan efektif yang disalurkan ke motor relatif sama untuk seluruh kecepatan.

Jika drop tegangan yang terjadi terlalu besar maka waktu on dari transistor daya terlalu lama, hal ini berbahaya jika transistor dihubungkan ke power supply yang diperoleh dari penyearahan tegangan jala-jala dan akan mengakibatkan rusaknya transistor. Jika waktu on terlalu kecil maka tegangan efektif yang disalurkan ke motor semakin kecil. karenanya multitune diset tepat pada saat tidak terjadi drop tegagan pada power supply.

Setelah waktu on dari transistor di set maka dilakukan pengukuran tegangan antar fasa. Untuk itu beban dan ROM kembali dipasang dan tegangan diukur terhadap titik tengah hubung bintang. Ketiga tegangan yang terukur harus relatif sama, hal ini diperoleh jika waktu on dan off dari transistor juga relatif sama. Untuk keperluan setting jika terjadi perbedaan tegangan digunakan multitune yang terpasang pada buffer transistor, diatur sedemikian rupa sehingga diperoleh hasil yang relatif sama baik pada saat pengukuran drop tegangan tiap fasa tanpa beban ataupun pada saat pengukuran tegangan berbeban.

4.6 Kalibrasi Putaran Motor

Kalibrasi putaran motor bertujuan men-set kecepatan putar motor ke harga yang disettingkan, hal tersebut dilakukan pada saat alat dioperasikan tanpa umpan balik. Dari hasil pengukuran diketahui terdapat perbedaan antara kecepatan putar motor dan kecepatan yang dimasukkan melalui keypad. Perbedaan yang terjadi tersebut disebabkan oleh slip dari motor induksi yang digunakan. Dengan melakukan koreksi terhadap frekuensi yang disalurkan ke motor, maka kecepatan putar motor dapat di set mendekati harga yang disettingkan.

Pengukuran dilakukan untuk setting kecepatan mulai dari rendah sampai tinggi dengan kenaikan sebesar 250 Rpm. Perbandingan antara kecepatan yang disettingkan dan kecepatan motor sesungguhnya setelah diadakan koreksi terhadap frekuensi dapat dilihat pada tabel C lampiran A.

Alloh tidak membebani seseorang melainkan sesuai dengan kesanggupannya ... (Q.S. Al Baqarah : 286)

BAB V
PENUTUP

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil uji coba mengatur kecepatan motor induksi secara PWM ternyata mempunyai unjuk kerja yang baik. Pengaturan kecepatan dengan metoda PWM ini mempunyai jangkauan yang lebar meliputi kecepatan dibawah maupun diatas kecepatan nominal motor. Dari alat yang telah dibuat diketahui bahwa torsi motor besar pada kecepatan di sekitar kecepatan nominal atau lebih rendah, tetapi torsi motor mengecil pada saat putaran motor tinggi.

Dari hal tersebut dapat ditarik kesimpulan bahwa :

1. Pengaturan kecepatan dengan PWM mengahasilkan jangkauan pengaturan yang lebar.
2. Torsi motor menjadi turun jika motor diputar diatas putaran nominal, turunnya torsi tersebut disebabkan oleh :
   - ❑ Spesifikasi motor yang memang didesain bekerja pada frekuensi 50 Hz.
   - ❑ Tidak liniernya komponen yang digunakan terutama optocoupler, sehingga semakin tinggi frekuensi semakin singkat waktu on di banding waktu offnya, hal ini mengabatkan turunnya tegangan efektif yang diterima motor.

3. Dengan memakai Adaptif kontrol pengaturan kecepatan motor secara PWM mempunyai performansi yang lebih baik yaitu respon motor untuk menjadi stabil terhadap pengaruh parameter yang ada.

5.2 Saran

Sistem yang telah dibuat ini masih bisa dikembangkan lebih lanjut untuk sistem yang lebih besar yaitu pengontrolan beberapa motor sekaligus oleh satu PC sebagai unit pemrosesnya, karena itu untuk pengembangan disarankan komunikasi data dipakai baut rate yang lebih tinggi. Pengontrolan beberapa motor sekaligus sangat memungkinkan karena komunikasi dilakukan secara interrupt, sehingga untuk sistem dengan beberapa motor sekaligus dapat diadakan pembagian prioritas dari motor yang dikontrol.

Demi Masa. Sesungguhnya manusia itu dalam kerugian. Kecuali orang-orang yang beriman dan melakukan amal saleh dan nasehat-menasehati supaya (menjalankan) kebenaran dan nasehat-menasehati supaya sabar (tabah menghadapi kesulitan). (Q.S. Al 'Ashr:1-3)

DAFTAR PUSTAKA

DAFTAR PUSTAKA

Bray, Barry B. THE INTEL MICROPROCESESOR, Devry Institute of Technologi, 1978.

Hall, Douglas V. MICROPROCESSOR AND DIGITAL SISTEM, Second Edition, McGraw-Hill Inc, Singapore 1983.

John Uffenbeck. THE 8086 / 8088 FAMILY DESING PROGRAMING AND INTERFACING, Prentice-Hall International, Inc, 1987.

Steean, J.P.M. DATA SHEET BOOK 2, PT. ELex Media komputindo, Jakarta 1988.

Partoharsono, Hartono. TUNTUNAN PRAKTIS PEMROGRAMAN BAHASA ASSEMBLY, PT. Elex Media Kompotindu, Jakarta, 1991.

Ned Mohan, Tore M. Undeland, Willian P. Robbins, POWER ELEKTRONICS: CONVERTER, APPLIKATIONS AND DESING, McGraw-Hill Inc, 1994.

...,DATA DAN PERSAMAAN TRANSISTOR, PT. Elex Media Kompotindo, Jakarta 1992.

...,DATA SHEET BOOK 1, PT. Elex Media Kompotindo, Jakarta 1994.

...,LINEAR DATA BOOK 2, National Semiconduktor Corporation, 1988.

Hai orang-orang yang beriman jadikanlah sabar dan sholat sebagai penolongmu, sesungguhnya Alloh beserta orang-orang yang sabar . (Q.S. Al Baqarah : 153)

LAMPIRAN

Lampiran A : Data Hasil Pengukuran.

Tabel A: Data Pengukuran Drop Tegangan Tanpa Beban.

(VDC = 40 Volt).

No	Setting Rpm	Tegangan Tiap Fasa (Volt)		
		R	S	T
1	500	43.5	43.5	43.5
2	750	43.1	43.1	43.1
3	1000	43.0	43.1	43.0
4	1250	42.8	42.9	42.9
5	1500	42.6	42.7	42.6
6	1750	42.4	42.4	42.4
7	2000	42.2	42.1	42.1
8	2250	41.9	41.8	41.8
9	2500	41.7	41.6	41.6
10	2750	41.5	41.5	41.6
11	3000	41.5	41.5	41.6

Lampiran A : Data Hasil Pengukuran.

Tabel B: Data Pengukuran Tegangan dengan Beban Motor.

(VDC = Jala-jala PLN 216 V).

No	Setting Rpm	Tegangan Tiap Fasa (Volt)		
		R	S	T
1	1000	111	111	112
2	1250	111	111	111
3	1750	110	110	112
4	2000	110	110	111
5	2250	111	110	111
6	2500	111	111	111
7	2750	112	111	111
8	3000	111	111	111
9	3250	111	110	111
10	2750	110	111	111
11	3500	110	111	111

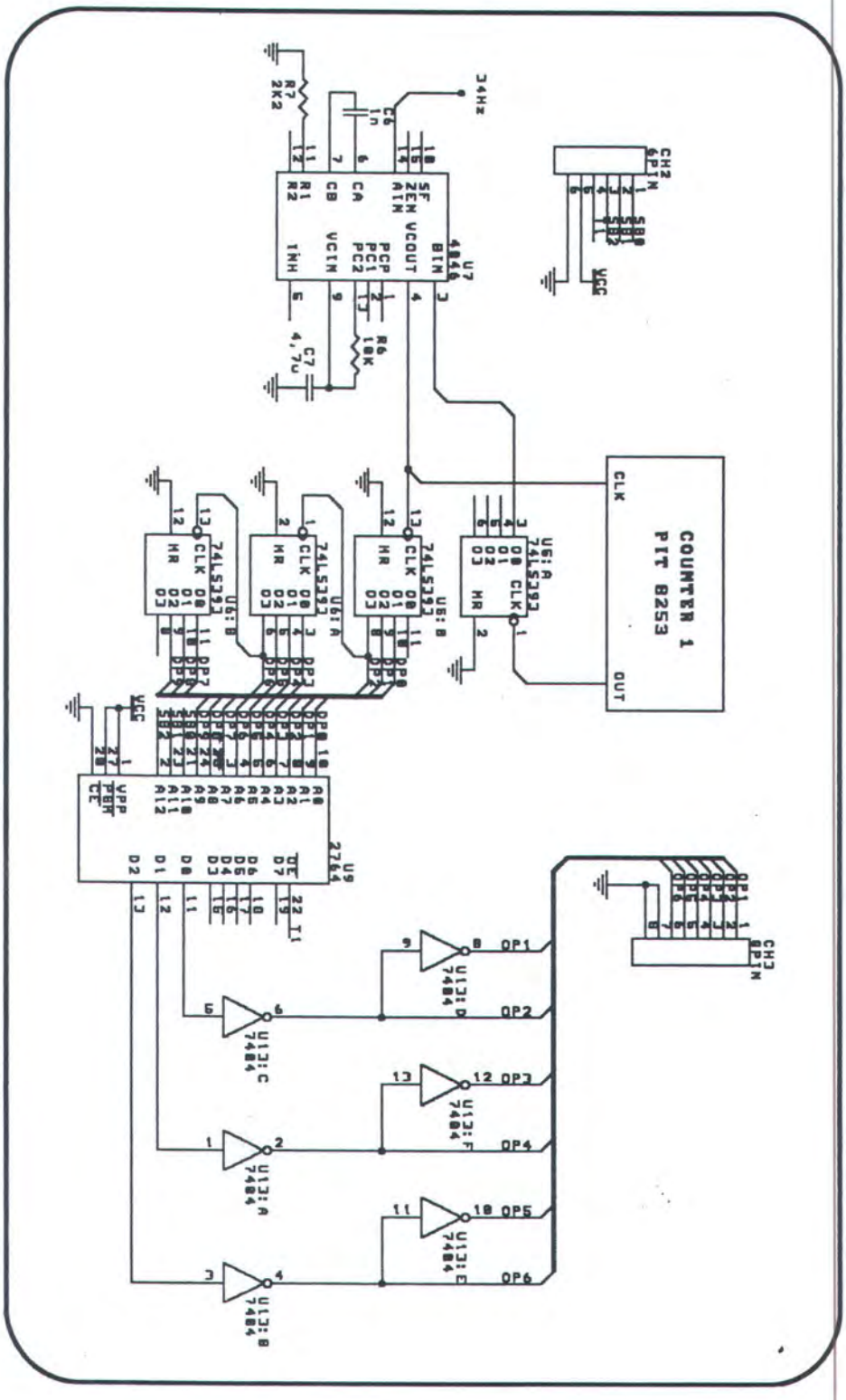
Lampiran A : Data Hasil Pengukuran.

Tabel C: Data Pengukuran Setting Kecepatan Motor dan Realitasnya .

No	Setting Rpm	Rpm Motor	Error
1	500	502	0,40%
2	750	752	0,27%
3	1000	1002	0,20%
4	1250	1252	0,16%
5	1500	1503	0,20%
6	1750	1753	0,17%
7	2000	2003	0,15%
8	2250	2253	0,13%
9	2500	2504	0,16%
10	2750	2754	0,15%
11	3000	3004	0,13%

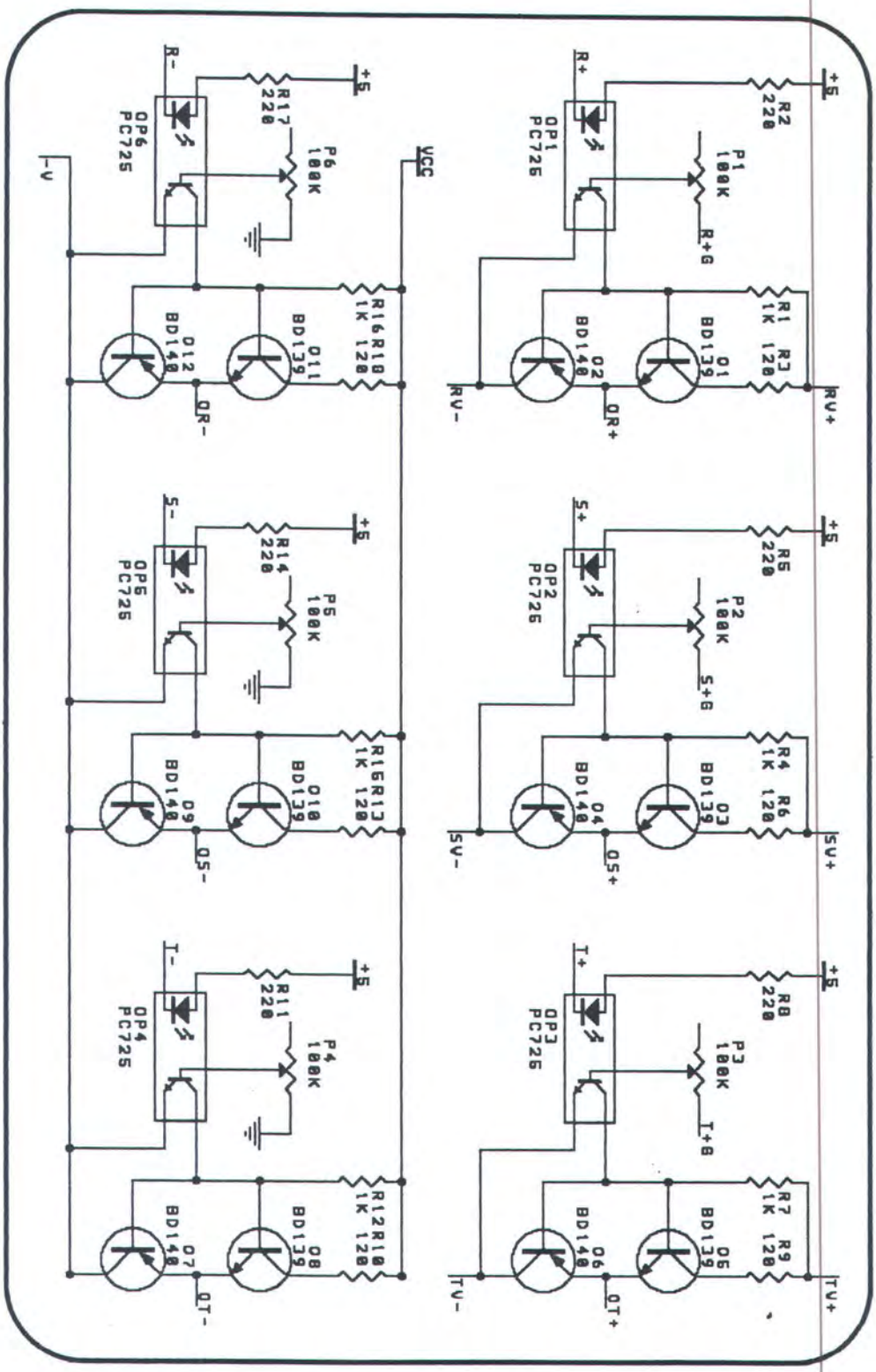
Lampiran B : Data Komponen dan Rangkaian.

Rangkaian Pengali Frekuensi dan Pembangkit Sinyal PWM Sinusoida.



Lampiran B : Data Komponen dan Rangkaian.

Rangkaian Driver Transistor Daya (Inverter 3 Phasa).



Lampiran C : Listing Program 8088 Minimum System

```
;********** Listing Program 8088 Minimum System **********

Mikro Segment
        Assume CS:Mikro, DS:Mikro
        Org 100h

Start:  CLI
        ;---------- Initial for 8088 Minimum System.
        Mov     AX,CS
        Mov     DS,AX
        Xor     AX,AX
        Mov     ES,AX
        Mov     SS,AX
        Mov     SP,800h

        ;---------- Initial for UART 8250.
        Mov     DX,Line_Cont_Reg
        Mov     AL,80h
        Out     DX,AL                   ;Bit_7 = 1, Setting Baut Rate.
        Mov     DX,Baut_Div_MSB         ;MSB Baut Rate Devider.
        Mov     AL,0
        Out     DX,AL

        Mov     DX,Baut_Div_LSB         ;LSB Baut Rate Devider.
        Mov     AL,2                    ;Baut Rate = 57600 Bps.
        Out     DX,AL

        Mov     DX,Line_Cont_Reg        ;Data Formating.
        Mov     AL,00000011b            ;8 bit / Character, 1 stop bit,
        Out     DX,AL

        Mov     DX,Int_Enbl_Reg         ;Interrupt Enable.
        Mov     AL,01h
        Out     DX,AL

        Mov     DX,Int_Id_Reg
        Mov     AL,00000001b
        Out     DX,AL

        Mov     DX,Modem_Cont_reg
        Mov     AL,00000000b
        Out     DX,AL

        Mov     DX,Line_Stat_reg
        Mov     AL,0
        Out     DX,AL

        ;---------- Initial for PIT 8253.
        Mov     AL,CW_0
        Out     Pit_Ctr,AL              ;Write Control Word Counter 0, used as
        Mov     AL,9Ah                  ;Base Clock for Multiply Frequency.
        Out     Pit_0,AL
        Mov     AL,19h
        Out     Pit_0,AL

        Mov     AL,CW_1                 ;Write Control Word Counter 1, used as
        Out     Pit_Ctr,AL              ;Multiply Frequency.
        Mov     AX,1500
        Out     Pit_1,AL                ;Default 1500 Rpm

        Mov     AL,CW_2                 ;Write Wontrol Word Counter 2, used as
        Out     Pit_Ctr,AL              ;Speed Counter.

        ;---------- Initial for PPI 8255.
        Mov     DX,Port_CTR
        Mov     AL,10001000b            ;Port A = Output, Port B = Output.
        Out     DX,AL                   ;Port C Low = Output, High = Input.

        Mov     DX,Port_CTR1            ;Second PPI the same as First PPI.
        Mov     AL,10000001b
        Out     DX,AL
                ;---------- Initial for PIC 8259.
```

```
            Mov     AL,ICW1
            Mov     DX,PIC_A
            Out     DX,AL
            Mov     AL,ICW2
            Mov     DX,PIC_B
            Out     DX,AL
            Mov     AL,ICW4
            Out     DX,AL
            Mov     AL,OCW1
            Out     DX,AL

            Lea     BX,IRQ0
            Mov     ES:[20h],BX
            Mov     ES:[22h],CS
            Lea     BX,Irq1
            Mov     ES:[24h],BX
            Mov     ES:[26h],CS
            Lea     BX,IRQ2
            Mov     ES:[28h],BX
            Mov     ES:[2Ah],CS
            Lea     BX,IRQ3
            Mov     ES:[2Ch],BX
            Mov     ES:[2Eh],CS

;=================== Main Program ===================

            Mov     Byte Ptr ES:[Menubuf],00h       ;Disp menu, Non Connected.
            Mov     Word Ptr ES:[RpmRead],00h
            Mov     Byte Ptr ES:[Speedbuf+8],00h
            Mov     Word Ptr ES:[IdentIrq0],00h
            Call    Identitas
            Call    WaitReady
            STI
            Call    Sinkron
            Mov     AL,11111000b
            Out     Pic_B,AL
            Mov     Word Ptr ES:[Over],50
            Mov     Word Ptr ES:[Seting],4

CheckSpeed: Call    Feedback
            Test    Byte Ptr ES:[Menubuf],01h
            Jnz     SpeedMenu
            Call    MenuDsp
            Jmp     CheckSpeed
SpeedMenu:  Call    SpeedDsp
            Jmp     CheckSpeed

;====================================================

RpmtoPit  Proc Near                                 ;AX = Data Rpm.
            Mov     BX,1024                         ;PIT = (1024*RPM)/1020.
            Mul     BX
            Mov     BX,1020
            Div     BX
            Out     Pit_1,AL
            Xchg    AL,AH
            Out     Pit_1,AL
            Ret
Endp RpmtoPit


DigitDsp Proc Near                                  ;AX Contain Digit for display RPM.
            Mov     Byte Ptr ES:[Data],04h          ;Set for Left Kursor Movement.
            Call    Instout
            Cmp     AX,00h
            Ja      RepDgDsp
            Mov     Byte Ptr ES:[Data],'0'
            Call    Dataout
            Jmp     QuitDgDsp
RepDgDsp:   Xor     DX,DX
            Mov     BX,10
```

```
                Div     BX
                Add     DL,30h
                Mov     Byte Ptr ES:[Data],DL
                Call    Datsout
                Cmp     AX,0
                Ja      RepDgDsp
QuitDgDsp:      Mov     Byte Ptr ES:[Data],06h          ;Set for Right Kursor Movement.
                Call    Instout
                Ret
DigitDsp        Endp


SpeedDsp Proc Near
                Push    CX
                Mov     CX,10
DispLoop:       Push    CX
                Mov     Byte Ptr ES:[Data],0Ch          ;Set for Display On, Cursor Off.
                Call    Instout
                Mov     Byte Ptr ES:[Data],67h
                Call    Setmem
                Lea     BX,Motorspd
                Call    Write
                Mov     Byte Ptr ES:[Data],06h
                Call    Setmem
                Mov     AL,Byte Ptr ES:[Menubuf]
                And     AL,02h
                Mov     AH,05h
                Mul     AH
                Lea     BX,Modemenu
                Add     BX,AX
                Call    Write
                Mov     Byte Ptr ES:[Data],4Fh          ;Space for speed motor display.
                Call    Setmem
                Mov     AX,Word Ptr ES:[RpmRead]
WriteRpm:       Call    DigitDsp
                Mov     Word Ptr ES:[Vardly],02FFFh
                Call    Delaymn
                Pop     CX
                Loop    Disploop
                Call    Feedback
                Pop     CX
                Ret
SpeedDsp Endp


MenuDsp Proc Near
                Push    CX
Repmn:          Call    Clear
                And     Byte Ptr ES:[Menubuf],0F3h      ;Default Menu Display.
                Mov     CX,04h
Repmn1:         Push    CX
                Mov     Byte Ptr ES:[Data],00h
                Call    Setmem
                Lea     BX,Menu_st
                Call    Write
                Mov     Byte Ptr ES:[Data],40h
                Call    Setmem
                Lea     BX,Menu_nd
                Call    Write
                Mov     Word Ptr ES:[Vardly],022FFh
                Call    DelayMn

                Test    Byte Ptr ES:[Menubuf],00001000b
                Jz      Unchanges
                Pop     CX                              ;Bit 3 indicate Changes
                Jmp     QuitMnDsp                       ;of Mode and Menu display.
Unchanges:      Call    Clear
                Mov     Word Ptr ES:[Vardly],014FFh
                Call    DelayMn
                Call    Feedback
                Pop     CX
                Loop    Repmn1
```

```
                Test    Byte Ptr ES:[Menubuf],00001000b
                Jz      ChoiseDsp
                Jmp     QuitMnDsp
ChoiseDsp:      Mov     Word Ptr ES:[Vardly],0AFFh
                Call    DelayMn
                Mov     Byte Ptr ES:[Data],67h
                Call    Setmem
                Lea     BX,Choice1
                Call    Write
                Mov     Byte Ptr ES:[Data],27h
                Call    Setmem
                Lea     BX,Choice2
                Call    Write
                And     Byte Ptr ES:[Menubuf],11111011b     ;Initially speed not set.
                Mov     Word Ptr ES:[Vardly],0EFFFh
                Call    DelayMn
                Test    Byte Ptr ES:[Menubuf],00001100b
                Jnz     QuitMnDsp
                Mov     CX,20                               ;Shift Character 20 times.
Rep02:          Mov     Byte Ptr ES:[Data],00011000b
                Call    Instout
                Loop    Rep02
                Mov     Word Ptr ES:[Vardly],0EFFFh
                Call    DelayMn
                Call    Clear
                Test    Byte Ptr ES:[Menubuf],00001000b
                Jnz     QuitMnDsp
Ch3:            Mov     Byte Ptr ES:[Data],67h
                Call    Setmem
                Lea     BX,Choice3
                Call    Write
                Mov     Word Ptr ES:[Vardly],0CFFFh
                Call    DelayMn
                Call    Feedback
QuitMnDsp:      Pop     CX
                Ret
MenuDsp Endp


TestKey Proc Near
                Push    AX
                Push    BX
                Push    DX
                Cmp     Byte Ptr ES:[Key],'A'               ;Stop Motor.
                Jne     StartMtr
Stop:           Or      Byte Ptr ES:[BufPC],80h
                Mov     AL,Byte Ptr ES:[BufPC]
                Mov     DX,Port_C1
                Out     DX,AL
                Mov     AL,00h
                Mov     DX,Port_B1
                Out     DX,AL
                Jmp     QuitTest
StartMtr:       Cmp     Byte Ptr ES:[Key],'B'               ;Start Motor.
                Jne     SetSpeed
                And     Byte Ptr ES:[BufPC],7Fh
                Mov     AL,Byte Ptr ES:[BufPC]
                Mov     DX,Port_C1
                Out     DX,AL
                Mov     Word Ptr ES:[Vardly],03FFFh
                Call    Delay
                Mov     AL,01h
                Mov     DX,Port_B1
                Out     DX,AL
                Jmp     QuitTest
SetSpeed:       Cmp     Byte Ptr ES:[Key],'C'
                Je      InDsp
                Jmp     ModeOprs
InDsp:          Or      Byte Ptr ES:[Menubuf],04h           ;Indicate set speed.
                Call    Clear
                Mov     Byte Ptr ES:[Data],02h
                Call    Setmem
```

```
            Lea     BX,NewSpd1
            Call    Write
            Mov     Byte Ptr ES:[Data],43h
            Call    Setmem
            Add     BX,13
            Call    Write
            Mov     Byte Ptr ES:[Data],44h
            Call    Setmem
            Mov     Byte Ptr ES:[Data],0Fh          ;Set for Cursor On and Blink.
            Call    Instout
            Xor     BX,BX
            Mov     Byte Ptr ES:[Key],00h
InputKey:   Cmp     Byte Ptr ES:[Key],'0'
            Jl      RepKeyIn
            Cmp     Byte Ptr ES:[Key],'9'
            Jle     SaveKey
RepKeyIn:   Cmp     Byte Ptr ES:[Key],'S'
            Je      Calculate
            Cmp     Byte Ptr ES:[Key],'@'
            Jne     InputKey
            Test    BX,0FFh
            Jnz     Indsp
            Jmp     QuitTest
SaveKey:    Cmp     BX,03h
            Jg      RepKeyIn
            Mov     AL,Byte Ptr ES:[Key]
            Mov     Byte Ptr ES:[Data],AL
            Call    Dataout
            Mov     Byte Ptr ES:[Speedbuf+BX],AL
            Mov     Byte Ptr ES:[Key],00h
            Inc     BX
            Jmp     InputKey
Calculate:  Cmp     BX,01h
            Jg      OkCAl
Outrange:   Call    Clear
            Lea     BX,Choice4
            Call    Write
            Mov     Word Ptr ES:[Vardly],0AFFFh
            Call    Delay
            Jmp     InDsp
OkCal:      Mov     CX,BX
            Dec     CX
            Mov     AX,1
Mult:       Mov     BL,10
            Mul     BL
            Loop    Mult
            Push    AX                              ;AX = Bil berpangkat.
            Mov     BL,Byte Ptr ES:[Speedbuf]
            Sub     BL,30h
            Xor     BH,BH
            Mul     BX
            Mov     Word Ptr ES:[Speedbuf+4],AX
            Pop     AX                              ;Min 3 digit.
            Mov     BL,10
            Div     BL
            Push    AX
            Mov     BL,Byte Ptr ES:[Speedbuf+1]
            Sub     BL,30h
            Mul     BL
            Add     Word Ptr ES:[Speedbuf+4],AX
            Pop     AX
            Cmp     AL,10
            Jg      Devide
            Mov     AL,Byte Ptr ES:[Speedbuf+2]
            Sub     AL,30h
            CBW
            Add     AX,Word Ptr ES:[Speedbuf+4]
            Jmp     QuitCal
Outvalue:   Jmp     Outrange
Devide:     Mov     BH,10
            Div     BH
            Mov     BL,Byte Ptr ES:[Speedbuf+2]
```

```
            Sub     BL,30h
            Mul     BL
            CBW
            Add     Word Ptr ES:[Speedbuf+4],AX
            Mov     AL,Byte Ptr ES:[Speedbuf+3]
            Sub     AL,30h
            CBW
            Add     AX,Word Ptr ES:[Speedbuf+4]
QuitCal:    Cmp     AX,550
            Jl      Outvalue
            Cmp     AX,4500
            Jg      Outvalue
            Mov     Word Ptr ES:[Rpmref],AX
            Mov     Word Ptr ES:[RpmPit],AX
            Push    AX
            Mov     AL,0ABh
            Call    Transmite
            Pop     AX
            Push    AX
            Xchg    AH,AL
            Ccall   Ttransmite
            Xchg    AH,AL
            Call    Transmite
            Pop     AX
            Mov     BX,1024                         ;PIT = (1024*RPM)/1020.
            Mul     BX
            Mov     BX,1020
            Div     BX
            Sub     AX,4
            Out     Pit_1,AL
            Xchg    AL,AH
            Out     Pit_1,AL
            Jmp     QuitTest
ModeOprs:   Cmp     Byte Ptr ES:[Key],'D'
            Jne     On_Offmn
            Call    Clear
            Or      Byte Ptr ES:[Menubuf],08h
            Mov     Byte Ptr ES:[Data],67h
            Call    Setmem
            Lea     BX,ModeOprMn
            Call    Write
RepMode:    Cmp     Byte Ptr ES:[Key],'0'
            Jne     Conect
            And     Byte Ptr ES:[Menubuf],11111101b ;Bit 1 = Conect/Nonconect.
            Mov     AL,0ACh
            Call    Transmite
            Jmp     QuitTest
Conect:     Cmp     Byte Ptr ES:[Key],'1'
            Jne     RepMode                         ;1 = Connecting.
            Or      Byte Ptr ES:[Menubuf],02h
            Mov     AL,0ADh
            Call    Transmite
            Jmp     QuitTest
On_Offmn:   Cmp     Byte Ptr ES:[Key],'S'
            Jne     QuitTest
            Or      Byte Ptr ES:[Menubuf],08h
            Mov     AL,Byte Ptr ES:[Menubuf]        ;Menubuf 0 = Display menu.
            And     AL,01h
            Cmp     AL,00h
            Jg      Decr
            Inc     Byte Ptr ES:[Menubuf]
            Jmp     QuitTest
Decr:       Dec     Byte Ptr ES:[Menubuf]
QuitTest:   Mov     Byte Ptr ES:[Key],00h
            Mov     Byte Ptr ES:[Data],0Ch          ;Set for Cursor Off.
            Call    Instout
            Pop     DX
            Pop     BX
            Pop     AX
            Ret
TestKey     Endp
```

```
Write       Proc    Near
            Push    DI
            Mov     Byte Ptr ES:[Data],00010100b        ;Cursor Movement.
            Call    Instout
            Xor     DI,DI
RepW:       Mov     AL,Byte Ptr CS:[BX+DI]              ;Pre Lea BX...
            Cmp     AL,'#'
            Jne     OkW
            Jmp     QuitW
Okw:        Mov     Byte Ptr ES:[Data],AL
            Call    Dataout
            Inc     DI
            Jmp     Repw
QuitW:      Pop     DI
            Ret
Write       Endp

Setmem      Proc    Near
            Or      Byte Ptr ES:[Data],80h              ;Prepare : Inst = Lokasi - 1.
            Call    Instout
            Ret
Setmem      Endp

Clear       Proc    Near
            Mov     Byte Ptr ES:[Data],01h              ;Clear Display.
            Call    Instout
            Mov     Byte Ptr ES:[Vardly],03Fh
            Call    Delay
            Ret
Clear       Endp

Instout     Proc    Near
            Push    AX                                  ;'DATA' diisi dulu.
            Push    And  Byte Ptr ES:[BufPA],00011111b
            Mov     AL,Byte Ptr ES:[BufPA]
            Mov     DX,Port_A
            Out     DX,AL
            Mov     DX,Port_B
            Mov     AL,Byte Ptr ES:[Data]
            Out     DX,AL
            Call    Strobe
            Pop     DX
            Pop     AX
            Ret
Instout     Endp

DataOut Proc Near
            Push    AX
            Push    DX
            And     Byte Ptr ES:[BufPA],00011111b
            Or      Byte Ptr ES:[BufPA],00100000b
            Mov     AL,Byte Ptr ES:[BufPA]
            Mov     DX,Port_A
            Mov     AL,Byte Ptr ES:[BufPA]
            Out     DX,AL
            Mov     DX,Port_B
            Mov     AL,Byte Ptr ES:[Data]
            Out     DX,AL
            Call    Strobe
            Pop     DX
            Pop     AX
            Ret
DataOut     Endp

DelayMn Proc Near
            Push    CX
            Mov     CX,Word Ptr ES:[Vardly]
```

```
RepdlyMn: Push    CX
          Mov     CX,08h
          Loop    $
          Cmp     Byte Ptr ES:[Key],00
          Je      PassDly
          Call    TestKey
          Pop     CX
          Mov     CX,5                    ;Time Delay Reduced.
          Push    CX
PassDly:  Pop     CX
          Loop    RepdlyMn
          Pop     CX
          Ret
DelayMn   Endp


Keypad Proc Near
          Push    AX
          Push    BX
          Push    CX
          Push    DS
startKey: Xor     BL,BL                   ;Four time shifted.
          Mov     Byte Ptr ES:[Key+1],0EEh ;First Row.
Next_Brs: Mov     AL,Byte Ptr ES:[Key+1]  ;Row Output.
          Mov     DX,Port_C
          Out     DX,AL
          In      AL,DX                   ;Input Baris.
          And     AL,0F0h
          Cmp     AL,0F0h                 ;Equal if not Buttom position.
          Jne     ScanKlm
          Mov     AL,Byte Ptr ES:[Key+1]
          Rol     AL,1                    ;Shift righ for next row.
          Mov     Byte Ptr ES:[Key+1],AL
          Inc     BL
          Cmp     BL,03
          Jle     Next_brs
          Jmp     StartKey
ScanKlm:  Mov     BH,00h
          Mov     Byte Ptr ES:[Key+2],11h
Test1:    Test    AL,Byte Ptr ES:[Key+2]
          Jz      Getkey
Shiftklm: Inc     BH
          Rol     Byte Ptr ES:[Key+2],1
          Jmp     Test1
Getkey:   Mov     AL,04h
          Mul     BH
          Add     AL,BL
          Lea     BX,Karakter
          Mov     CX,CS
          Mov     DS,CX
          Xlat
          Mov     Byte Ptr ES:[Key],AL
RepKey:   Mov     AL,00h
          Mov     DX,Port_C
          Out     DX,AL
          In      AL,DX                   ;Row Input.
          And     AL,0F0h
          Cmp     AL,0F0h                 ;Equal if not Buttom position.
          Jne     RepKey
          Pop  DS
          Pop  CX
          Pop  BX
          Pop  AX
          RET
Keypad    Endp


Identitas Proc Near
          Mov     Byte Ptr ES:[Key],00h
Repid:    Mov     Byte Ptr ES:[Data],0Ch  ;Kursor Off.
          Call    Instout
          Mov     Byte Ptr ES:[Data],00h
```

```
                Call    Setmem
                Lea     BX,Kal1
                Call    Write
                Mov     Byte Ptr ES:[Data],40h
                Call    Setmem
                Lea     BX,Kal2
                Call    Write
                Mov     CX,31
Rep00:          Push    CX
                Mov     Word Ptr ES:[Vardly],01FFFh
                Call    Delay
                Mov     Byte Ptr ES:[Data],00011000b
                Call    Instout
                Pop     CX
                Loop    Rep00
                Ret
Identitas       Endp


Receive Proc Near
Tunggu_Data:
                Mov     DX,Line_Stat_reg
                In      AL,DX
                Test    AL,00000001b
                Jnz     Baca_Data
                Loop    Tunggu_Data
                Mov     AL,00h
                Jmp     QuitRx
Baca_Data:      Mov     DX,Rx_Buffer
                In      AL,DX
QuitRx:         Ret
Receive         Endp


Transmite Proc  Near
                Push    DX
                Push    AX
RepTx:          Mov     DX,Line_Stat_reg
                In      AL,DX
                And     AL,01100000b
                Cmp     AL,01100000b
                Jne     RepTx
                Pop     AX
                Mov     DX,Tx_Buffer
                Out     DX,AL
                Pop     DX
                Ret
Transmite       Endp


Sinkron Proc Near
                Mov     Byte Ptr ES:[Key],00h
                Call    Clear
                Lea     BX,WaitCN
                Call    Write
                Mov     CX,10
RxClear:        Mov     DX,RX_Buffer
                In      AL,DX
                Loop    RxClear
                Mov     AL,0AAh
                Call    Transmite
                Mov     CX,0FFFFh
RepSin:         Mov     DX,Line_Stat_Reg
                In      AL,DX
                Test    AL,00000001b
                Jnz     ReadSin
                Push    CX
                Mov     CX,08h
                Loop    $
                Pop     CX
                Loop    RepSin
                Jmp     FailSin
```

```
ReadSin:    Mov     DX,RX_Buffer
            In      AL,DX
            Cmp     AL,'#'
            Jne     Failsin
            Mov     AL,'#'
            Call    Transmite
            Call    Clear
            Lea     BX,Fault
            Add     BX,59                       ;Sistem Connected.
            Call    Write
            Or      Byte Ptr ES:[Menubuf],12h
            Jmp     QuitSin
FailSin:    Mov     Byte Ptr ES:[Data],67h
            Call    Setmem
            Lea     BX,Fault
            Call    Write
            And     Byte Ptr ES:[Menubuf],11101101b
Choose:     Cmp     Byte ptr ES:[Key],'A'
            Jne     Choose1
            Jmp     Sinkron
Choose1:    Cmp     Byte Ptr ES:[Key],'C'
            Jne     Choose
QuitSin:    Mov     Byte Ptr ES:[Key],00h
            Ret
Sinkron     Endp


Irq0 Proc Near
            Pushf
            Push    AX
            Call    Keypad
            Mov     Byte Ptr ES:[IdentIrq0],1
            Mov     AL,EOI
            Out     Pic_A,AL
            Pop     AX
            Popf
            Iret
Irq0 Endp


Irq1 Proc  Near
            Push    AX
            Push    CX
            Push    DX
            Mov     DX,RX_Buffer                ;Protokol.
            In      AL,DX
            Cmp     AL,0DDh                     ;Request data for PC.
            Jne     SetSpeedPC
            Mov     AL,0AEh
            Call    Transmite
            Mov     AX,Word Ptr ES:[RpmRead]
            Xchg    AH,AL
            Call    Transmite
            Xchg    AH,AL
            Call    Transmite
            Jmp     QuitIrq1
SetSpeedPC: Cmp     AL,0ABh                     ;Set Speed from PC.
            Jne     Disc88
            Mov     CX,0FFFFh
            Call    Receive
            Mov     AH,AL
            Mov     CX,0FFFFh
            Call    Receive
            Mov     Word Ptr ES:[RpmRef],AX
            Mov     Word Ptr ES:[RpmPit],AX
            Push    AX
            Call    Receive
            Mov     AH,AL
            Mov     CX,0FFFFh
            Call    Receive
            Mov     Word Ptr ES:[Over],AX
            Pop     AX
```

```
                Or      Byte Ptr ES:[Menubuf],00100000b
                Mov     BX,1024                         ;PIT = (1024*RPM)/1020.
                Mul     BX
                Mov     BX,1020
                Div     BX
                Sub     AX,4
                Mov     Word Ptr ES:[Rpmref],AX
                Out     Pit_1,AL
                Xchg    AL,AH
                Out     Pit_1,AL
                Jmp     QuitIrq1
Disc88:         Cmp     AL,0ACh                         ;Disconnected mode.
                Jne     Connect88
                And     Byte Ptr ES:[Menubuf],11111101b
                Jmp     QuitIrq1
Connect88:      Cmp     AL,0ADh                         ;Connected mode.
                Jne     QuitIrq1
                Or      Byte Ptr ES:[Menubuf],00000010b
QuitIrq1:       Mov     AL,EOI
                Out     Pic_A,AL
                Pop     DX
                Pop     CX
                Pop     AX
                Iret
Irq1 Endp


Irq2 Proc Near                                          ;Int 1&2 initialized with MenuBuf+1 = 00h.
                Push    AX
                Push    BX
                Push    DX
                Cmp     Byte Ptr ES:[Speedbuf+8],04h
                Jl      Irq3Akt                         ;Error protecting.
CountStp:       And     Byte Ptr ES:[BufPA],0FEh        ;Disable PIT Counter 2,
                Mov     AL,Byte Ptr ES:[BufPA]          ;1 Cycle.
                Mov     DX,Port_A
                Out     DX,AL
                Mov     AL,0FFh                         ;Disable all Irq.
                Out     Pic_B,AL
                Mov     DX,Port_C1
                In      AL,DX
                Test    AL,01h                          ;Bit 0 : Counter 2, bit 1 : Ready motor on.
                Jz      ReadPit
                Mov     Word Ptr ES:[RpmRead],00h
                Jmp     Irq2Rst
ReadPit:        In      AL,Pit_2
                Mov     AH,AL
                In      AL,Pit_2
                Xchg    AL,AH
                Mov     BX,0FFFFh
                Sub     BX,AX
                Test    Byte Ptr ES:[IdentIrq0],1
                Jnz     Irq2Rst
                Mov     DX,0CCh
                Mov     AX,0BAF0h
                Div     BX
RpmRead:        Mov     Word Ptr ES:[RpmRead],AX
Irq2Rst:        Mov     Byte Ptr ES:[Speedbuf+8],00h
                Mov     AL,11111000b                    ;Enable Irq2, Disable Irq3.
                Out     Pic_B,AL
                Jmp     PassIrq2
Irq3Akt:        Or      Byte Ptr ES:[BufPA],01h         ;Enable Gate Pit 2
                Mov     AL,Byte Ptr ES:[BufPA]
                Mov     DX,Port_A
                Out     DX,AL
                Mov     AL,0FFh                         ;Counter Initialized.
                Out     Pit_2,AL
                Out     Pit_2,AL
                Mov     Byte Ptr ES:[Speedbuf+8],00h    ;1 Cycle Detect.
                Mov     AL,11110100b                    ;Disable Irq2, Enable Irq3.
                Out     Pic_B,AL
PassIrq2:       Mov     AL,EOI
```

```
                Out     Pic_A,AL
                Mov     Byte Ptr ES:[IdentIrq0],0
                Pop     DX
                Pop     BX
                Pop     AX
                Iret
Irq2 Endp


Irq3 Proc Near
                Push    AX
                Push    DX
                Inc     Byte Ptr ES:[Speedbuf+8]
                Cmp     Byte Ptr ES:[Speedbuf+8],04h
                Je      Irq2Akt
                Jl      PassIrq3
IrqInit:        Mov     Byte Ptr ES:[Speedbuf+8],00h
Irq2Akt:        Mov     AL,11111000b                    ;Enable Irq2, Disable Irq3.
                Out     Pic_B,AL
PassIrq3:       Mov     AL,EOI
                Out     Pic_A,AL
                Pop     DX
                Pop     AX
                Iret
Irq3 Endp


                Org  20F0h
Reset           Db 0EAh,00,01,0F0h,0FDh
                Db 11 Dup (0FFh)
Mikro           Ends
                End Start
```

Lampiran C : Listing Program PC - AT.

```
#include "kotak.h"
#include "regseria.h"
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#define x1 33
#define y1 180
#define x2 368
#define y2 420
#define xx 75
#define yy 75
extern int RpmRef;
extern int RpmDisp;
extern int overshoot;
extern int setingtime;

extern void putar();
int data[304];
void inisialisasi()
{int i;
for (i=0;i<304;i++);data[i]=0;
}

void isidata(int di)
{
int jarak=1,old=data[jarak],t=0;//StatusMouse(HILANG);
for (t=0;t<304;t=t+jarak)
            {
setcolor(0);    line (x1+t+28,y2-22-old,x1+t+28+jarak,y2-22-data[t]);
                old=data[t];
                data[t]=data[t+jarak];
setcolor(10);   if(t>0)    line (x1+t+28,y2-22-old,x1+t+28+jarak,y2-22-data[t]);
            }
data[t]=di; StatusMouse(MUNCUL);
}

char Mask_Org;
void interrupt ( *oldhandler)(__CPPARGS);
void interrupt handler(__CPPARGS);


main()
{
  oldhandler = getvect(INTRirq2);                    // save the old interrupt vector
  setvect(INTRirq2, handler);                        // install the new interrupt handler

asm{        Mov     DX,Line_Cont_Reg
            Mov     AL,80h
            Out     DX,AL                          //Bit_7 = 1, Setting Baut Rate.
            Mov     DX,Baut_Div_MSB                //MSB Baut Rate Devider.
            Mov     AL,0
            Out     DX,AL
            Mov     DX,Baut_Div_LSB                //LSB Baut Rate Devider.
            Mov     AL,2                           //Baut Rate = 57600 Bps.
            Out     DX,AL
            Mov     DX,Line_Cont_Reg               //Data Formating.
            Mov     AL,00000011b                   //8 bit / Character, 1 stop bit,
            Out     DX,AL                          //Disable Parity & Untransmited,
                                                   //No Acces Baut rate.
            Mov     DX,Int_Enbl_Reg                //Interrupt Enable.
            Mov     AL,01h
            Out     DX,AL
            Mov     DX,Int_Id_Reg
            Mov     AL,00000010b
            Out     DX,AL
            Mov     DX,Modem_Cont_Reg
            Mov     AL,00001000b
            Out     DX,AL
```

```
            Cli
            Mov     DX,21h
            In      AL,DX
            Mov     Mask_Org,AL
            And     AL,11110111b            //Membuka Interupsi Irq3 (Com2).
            Mov     DX,21h
            Out     DX,AL
            Mov     DX,Line_Stat_Reg
            Mov     AL,0
            Out     DX,AL
            Sti
}
extern int Connect;
Connect=0;
RpmRef=1500;
RpmDisp=0;
overshoot=50;
setingtime=300;
inisgraph();
inisialisasi(); ResetMouse(); StatusMouse(MUNCUL);
kotak layar(0,0,639,479,"",7,15,8,5);
kotak judul(x1-15,y1-30,x2+5,y1-6,"GRAFIK MOTOR",3,15,8,3);
kotak graf(x1-15,y1-3,x2+5,y2+10,"",0,15,8,4);
kotak Speed(525,165,602,192,"",0,15,8,2);
kotak Speedhapus(528,168,599,188,"",0,0,0,0);
kotak tbEXIT(390,400,620,430,"EXIT",7,15,8,3);
kotak tbSETTING(390,231,620,345,"SETTING",7,15,8,3);
kotak tbConnect(390,360,500,390,"CONNECT",7,15,8,3);
kotak tbdConnect(510,360,620,390,"DISCONNECT",7,15,8,3);

layar.GbKotak();
judul.GbKotak();
graf.GbKotak();
Speed.GbKotak();
tbSETTING.GbKotak();
tbEXIT.GbKotak();
tbConnect.GbKotak();
tbdConnect.GbKotak();

StatusMouse(HILANG);
settextstyle(0,0,2);
setcolor(0);
outtextxy(321,50,"PENGENDALIAN MOTOR INDUKSI 3 FASA");
outtextxy(321,80,"SECARA ADAPTIF");
setcolor(9);
outtextxy(318,48,"PENGENDALIAN MOTOR INDUKSI 3 FASA");
outtextxy(318,78,"SECARA ADAPTIF");
setcolor(0);
outtextxy(458,180,"SPEED :");
setcolor(13);
outtextxy(455,182,"SPEED :");
setcolor(6);
for (int a=400;a<=402;a++)                  //Sumbu X
line(58,a,366,a);
line(335,413,365,413);
line(366,413,359,417);
line(366,413,359,409);

for (int b=58;b<=60;b++)                    //Sumbu Y
line(b,185,b,400);
line(47,186,47,216);
line(47,185,43,192);
line(47,185,51,192);
setcolor(4);
settextstyle(0,1,0);
outtextxy(33,203,"Rpm");
settextstyle(0,0,0);
outtextxy(345,420,"t");
outtextxy(308,410,"75");
outtextxy(225,410,"50");
```

```
outtextxy(142,410,"25");
outtextxy(39,240,"3000");
outtextxy(39,293,"2000");
outtextxy(39,345,"1000");
int d,i=0,SPEED=1, Ovrsht=50;
char speed[10],metu;

while(metu!='\x1B')
{
        if(kbhit())metu=getch();
        if(i<200)d=i;
        isidata(SPEED);                         //proc menampilkan grafik
        if (i>200){d=(200+(200-i));}
        if (i==400) i=0;
        i=i+1;

        SPEED=RpmDisp/19;
        itoa(RpmDisp,speed,10);                 //Display angka speed.
        Speedhapus.GbKotak();
        outtextxy(547,180,speed);               //tampil angka speed [ ]
        outtextxy(583,180,"RPM");

        asm{
          Mov     AL,0DDh
          Mov     DX,TX_Buffer
          Out     DX,AL     }

        StatusMouse(MUNCUL);
        if(tbSETTING.pilih())
{       printf("\a");
        putar();
        asm{
          Mov     AL,0ABh
          Mov     DX,TX_Buffer
          Out     DX,AL       }

RepTx:  asm{
          Mov     DX,Line_Stat_Reg
          In      AL,DX
          And     AL,01100000b
          Cmp     AL,01100000b
          Jne     RepTx
          Mov     AX,RpmRef
          Xchg    AH,AL
          Mov     DX,TX_Buffer
          Out     DX,AL       }

RepTx2: asm{
          Mov     DX,Line_Stat_Reg
          In      AL,DX
          And     AL,01100000b
          Cmp     AL,01100000b
          Jne     RepTx2
          Mov     AX,RpmRef
          Mov     DX,TX_Buffer
          Out     DX,AL       }
        delay(10);
RepTx3: asm{
          Mov     DX,Line_Stat_Reg
          In      AL,DX
          And     AL,01100000b
          Cmp     AL,01100000b
          Jne     RepTx3
          Mov     AX,overshoot
          Xchg    AH,AL
          Mov     DX,TX_Buffer
          Out     DX,AL       }
RepTx4: asm{
          Mov     DX,Line_Stat_Reg
          In      AL,DX
```

```
            And   AL,01100000b
            Cmp   AL,01100000b
            Jne   RepTx4
            Mov   AX,overshoot
            Mov   DX,TX_Buffer
            Out   DX,AL          }

          tbSETTING.GbKotak();
                }
          if(tbEXIT.pilih())
{         printf("\a");
          metu='\x1B';
          tbEXIT.GbKotak();      }

          if(tbConnect.pilihp())
{         Connect=1;
          asm{                                    //Kirim data tombol connect
            Mov   AL,0ADh
            Mov   DX,TX_Buffer
            Out   DX,AL       } }

          if(tbdConnect.pilihp())
{         Connect=0;                              //Kirim data tombol disconnect
          asm{
            Mov   AL,0ACh
            Mov   DX,TX_Buffer
            Out   DX,AL       } }

  if(Connect==0){ if(tbdConnect.Gbinfo())tbdConnect.GbKotakp();
                  if(!tbConnect.Gbinfo())tbConnect.GbKotak();}

          if(Connect==1){ if(tbConnect.Gbinfo())tbConnect.GbKotakp();
                          if(!tbdConnect.Gbinfo()) tbdConnect.GbKotak();}  }
          asm{
            Mov DX,21h
            Mov AL,Mask_Org
            Out DX,AL          }

setvect(INTRirq2, oldhandler);
return 0;
}
```

```
/* PROGRAM PENGENDALIAN ADAPTIF SELF TUNING REGULATOR
   UNTUK MOTOR INDUKSI 3 PHASA.
*/


 # include <math.h>
 # include <stdio.h>
 # include <stdlib.h>
 # include <dos.h>
 # include <conio.h>
 # include <string.h>
 # include "regseria.h"

 const float pi=3.142857;
 extern int sinyal, RpmRef, RpmDisp, setingtime, overshoot;
 float a, b, c, d, zheta, omega_n, e, f, g, T, h, i, var;
 float kali_rekursif, hitung, a1, b1, f11, f12, f21, f22;
 float u_awal, y_awal, alpha0, alpha1, alpha2, E, peny;
 float pbg1, pbg2, pbg3, pbg4, k, u, v;

 void Identifikasi(void);
 void kondisi_awal(void);
 void aksi_kontrol(void);

 main()
 {
  a = log(overshoot);
  b = pow(a,2);
  c = pow(pi,2);
  d = b / (c+b);
  zheta = sqrt(d);
  omega_n = 4.0 / (zheta * setingtime);
  e = pow(zheta,2);
  f = 1.0 - e;
  g = sqrt(f);  /* g adalah akar dari 1 - kuadrat zheta */
  T = 0.1; /* Time Sampling 1 milidetik */
  h = zheta * omega_n * T;
  i = -2 * h;
  var = exp(i);

  kondisi_awal();
  kali_rekursif = ceil(setingtime / T);
  hitung = 0.0;
  do
   {
   hitung = hitung + 1.0;
   Identifikasi();
   } while(hitung < kali_rekursif);
   }


  void kondisi_awal()
   {
   a1 = 1.0;  /* Estimasi awal parameter sistem */
   b1 = 1.0;
   f11 = 50.0;  /* Gain Adaptasi disini diambil sama dengan 50 */
   f12 = 0.0;
   f21 = 0.0;
   f22 = 50.0;
   alpha0 = 0.9;
   alpha1 = 0.97;
   alpha2 = 0.9;
   y_awal = 0.0;  /* Nilai awal input u dan output y dari plant */
   u_awal = 0.0;
   }

  void Identifikasi( )
   {
    E = RpmDisp - (a1 * (-y_awal) + b1 * u_awal);
    a1 = a1 - (E * f11 * y_awal) + (E * f12 * u_awal);
```

```
 b1 = b1 - (E * f21 * y_awal) + (E * f22 * u_awal);
 peny = (alpha1/alpha2) + (f11*pow(y_awal,2)) - (f21*u_awal*y_awal)
          - (f12*u_awal*y_awal) + (f22*pow(u_awal,2));
 pbg1 = (pow(f11,2)*pow(y_awal,2)) - (f11*f12*u_awal*y_awal)
          - (f11*f21*y_awal*u_awal) + (f12 * f21 * pow(u_awal,2));
 pbg2 = (f11*f12*pow(y_awal,2)) - (pow(f12,2)*u_awal*y_awal)
          - (f11*f22*y_awal*u_awal) + (f12*f22*pow(u_awal,2));
 pbg3 = (f21*f11*pow(y_awal,2)) - (f22*f11*u_awal*y_awal)
          - (pow(f21,2)*y_awal*u_awal) + (f22*f21*pow(u_awal,2));
 pbg4 = (f21*f12*pow(y_awal,2)) - (f22*f12*u_awal*y_awal)
          - (f21*f22*y_awal*u_awal) + (pow(f22,2)*pow(u_awal,2));
 f11 = (f11 - (pbg1 / peny)) / alpha1;
 f12 = (f12 - (pbg2 / peny)) / alpha1;
 f21 = (f21 - (pbg3 / peny)) / alpha1;
 f22 = (f22 - (pbg4 / peny)) / alpha1;
 alpha1 = (alpha0 * alpha1) + 1 - alpha0;
 aksi_kontrol();
 y_awal = RpmDisp;
 u_awal = u;
}

void aksi_kontrol()
{
 k = (a1 + var) / b1;
 u = k*RpmDisp + RpmRef;
 v = ceil(u);
 sinyal = v;
 asm{ Mov AL,0DDh
      Mov DX,TX_Buffer
      Out DX,AL }

 RepTx: asm{
      Mov DX,Line_Stat_Reg
      In AL,DX
      And AL,01100000b
      Cmp AL,01100000b
      Jne RepTx

      Mov AX,sinyal
      Xchg AH,AL
      Mov DX,TX_Buffer
      Out DX,AL }

 RepTx2: asm{
      Mov DX,Line_Stat_Reg
      In AL,DX
      And AL,01100000b
      Cmp AL,01100000b
      Jne RepTx2

      Mov AX,sinyal
      Mov DX,TX_Buffer
      Out DX,AL }
}
```