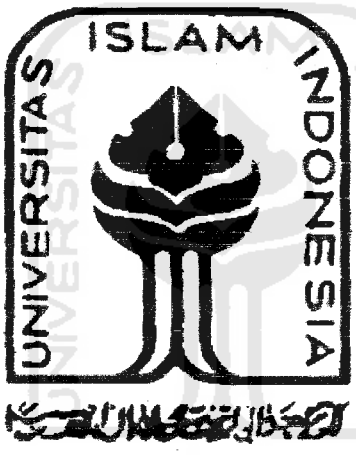
PENAMPIL INFORMASI DOT MATRIK DENGAN ANIMASI TERPROGRAM BERBASIS MIKROKONTROLER AT89C52

TUGAS AKHIR

Diajukan Sebagai Syarat Untuk Memperoleh Gelar Sarjana Pada
Jurusan Teknik Elektro, Fakultas Teknologi Industri
Universitas Islam Indonesia



disusun oleh :

Nama : Adisatya Pramardianto

No. Mahasiswa : 01 524 029

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

HALAMAN PENGESAHAN PEMBIMBING

TUGAS AKHIR

PENAMPIL INFORMASI DOT MATRIK DENGAN ANIMASI TERPROGRAM BERBASIS MIKROKONTROLER AT89C52

disusun oleh :

Nama : **Adisatya Pramardianto**

No. Mahasiswa : **01 524 029**

Yogyakarta, April 2007

Pembimbing I	Pembimbing II
(Wahyudi Budi Pramono, ST)	(Yusuf Aziz Amrulloh, ST)

HALAMAN PERSEMBAHAN

Syukur Alhamdulillah kepada Allah SWT, atas rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi ini.

Skripsi ini didedikasikan dan dipersembahkan sebagai ungkapan terima kasih dengan tulus kepada mereka yang sangat berarti dalam hidupku :

Ayahnda (Budi Kusdiyatno) dan ibunda tercinta (Ispartini)
Terimakasih atas do'a dan kasih sayang serta kepercayaan yang diberikan kepadaku.
Dan
Adikku tersayang (Bagas Raedianto) yang telah memberi semangat dan masukan yang sangat berarti

Semoga menjadi kenangan yang indah dan tak terlupakan
Amien Ya Rabbal 'Aalamien.....

MOTTO

- "Hai sekalian orang-orang yang beriman, mintalah pertolongan (kepada Allah) dengan sabar dan sholat, sesungguhnya Allah beserta orang-orang yang sabar."

(QS. Al Baqarah 153)

- "Katakanlah, "Setiap orang berbuat menurut keadaannya, maka Tuhan kamu lebih mengetahui siapa-siapa yang lebih benar jalannya."

(QS. Al Israa' 84)

- "Dan katakanlah, "Bekerjalah kamu. Allah dan Rasul-Nya serta orang-orang beriman akan melihat pekerjaanmu. Dan kamu akan dikembalikan kepada yang Maha Mengetahui yang gaib dan yang nyata; maka Dia akan memberitakan kepadamu tentang apa yang kamu kerjakan"."

(QS. At Taubah 105)

"Jangan Menyerah Sebelum Berusaha"

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Assalamu'alaikum Wr. Wb.

Alhamdulillah, puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan laporan Tugas Akhir ini sebagai salah satu persyaratan untuk memperoleh gelar Sarjana (S1) Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta. Tak lupa shalawat dan salam senantiasa tercurah pada junjungan kita Nabi Besar Muhammad SAW yang telah membawa kita keluar dari zaman kegelapan.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan Tugas Akhir ini dengan judul **"Penampil Informasi Dot Matrik Dengan Animasi Terprogram Berbasis Mikrokontroler AT89C52"** masih terdapat banyak kekurangan didalamnya.

Sedangkan keberhasilan penulis dalam menyelesaikan laporan ini tidak terlepas dari bantuan dan partisipasi berbagai pihak, baik berupa bimbingan, kritik, saran, maupun do'a. Untuk itu dalam kesempatan yang khusus ini, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Bapak Fathul Wahid, ST.,M.Sc. selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
2. Bapak Tito Tuwono, ST.,M.Sc. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia.

3. Bapak Yusuf Aziz Amrulloh, ST. selaku Sekretaris Jurusan Teknik Elektro dan selaku pembimbing II yang telah banyak membantu dalam penyusunan dan penulisan laporan Tugas Akhir ini.
4. Bapak Wahyudi Budi Pramono, ST. selaku Dosen Pembimbing I yang telah banyak membantu dalam penyusunan dan penulisan laporan Tugas Akhir ini.
5. Segenap Dosen di lingkungan Jurusan Teknik Elektro, Universitas Islam Indonesia, terima kasih atas ilmu yang diberikan sewaktu kuliah.
6. Ayah dan Ibundaku, yang telah memberikan segala-galanya, terima kasih untuk cinta dan kasih sayang, serta do'a selama ini.
7. Teman-teman Elektro angkatan 2001.
8. Semua pihak yang telah banyak membantu.

Kepada mereka penulis mengucapkan terima kasih yang sebanyak-banyaknya, semoga Allah SWT membalas segala kebaikan dan bantuannya.

Akhir kata, semoga laporan Tugas Akhir ini dapat bermanfaat bagi semua pihak, bagi perkembangan ilmu pengetahuan dan teknologi pada umumnya, dan Jurusan teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia pada khususnya.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, April 2007

Penulis

ABSTRAK

Perkembangan teknologi yang semakin maju membuat manusia berusaha mengembangkan dan memanfaatkan teknologi untuk berbagai macam kebutuhan. Dalam dunia bisnis, khususnya pemasaran, para pedagang membutuhkan suatu penampil informasi untuk menawarkan produk dagangannya kepada konsumen. Selain pada dunia bisnis, kadang orang membutuhkan suatu penampil informasi untuk kebutuhan layanan publik, terutama pada sarana-sarana umum, seperti bandara, terminal, stasiun, dan pusat kota. Penampil informasi tersebut sangat penting untuk menyebarkan informasi kepada masyarakat. Salah satu cara untuk mengatasi permasalahan diatas adalah dengan membuat alat penampil informasi. Alat ini dibuat dengan menggunakan beberapa komponen, yaitu : mikrokontroler AT89C52, SEEPROM AT24C64, demultiplexer 74LS138, resistor, kapasitor, transistor, dot matrik 10 buah, dan rangkaian catu daya. Mikrokontroler AT89C52 berfungsi untuk mengendalikan kerja alat. SEEPROM AT24C64 digunakan untuk menyimpan karakter. Transistor digunakan untuk scanning baris, sedangkan untuk scanning kolom digunakan demultiplexer 74LS138. Untuk menjalankan alat ini, pertama-tama karakter yang akan ditampilkan pada dot matrik, diketik terlebih dahulu melalui komputer, lalu dikirim ke mikrokontroler. Karakter-karakter tersebut akan disimpan di memori SEEPROM. Setelah seluruh karakter yang akan ditampilkan disimpan di SEEPROM, lalu karakter tersebut ditampilkan satu per satu pada dot matrik. Ada dua macam animasi yang dapat ditampilkan pada dot matrik, yaitu : karakter berjalan dari kanan ke kiri normal atau berkedip. Dari hasil penelitian yang telah dilakukan, alat ini telah dapat bekerja dengan baik, yaitu dengan menampilkan karakter berjalan maksimal sampai 1000 karakter.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Dunia teknologi saat ini berkembang sangat pesat. Terlebih lagi dalam bidang elektronika komunikasi dan elektronika kendali. Dalam kehidupan sehari-hari, hampir semua orang memanfaatkan piranti elektronika, baik untuk media komunikasi, media kendali, maupun kebutuhan lain. Dalam dunia bisnis, khususnya pemasaran, para pedagang menginginkan sesuatu yang istimewa pada produk dagangan ataupun tempat jualnya. Salah satunya adalah dengan memasang penampil informasi yang akan menawarkan produk dagangannya.

Selain pada dunia bisnis, kadang orang membutuhkan suatu penampil informasi untuk kebutuhan layanan publik, terutama pada sarana-sarana umum, seperti bandara, terminal, stasiun, dan pusat kota. Penampil informasi tersebut sangat penting untuk menyebarkan informasi kepada masyarakat. Kemudian timbul suatu pemikiran bagaimana menciptakan suatu alat yang mudah dioperasikan dan mampu menjawab pertanyaan, serta memenuhi kebutuhan diatas, sehingga didapatkan pemanfaatan teknologi tepat guna yang efektif dan efisien.

1.2. Rumusan Masalah

Agar arah dari tugas akhir ini menjadi jelas, maka perlu dibuat suatu rumusan masalah, yaitu : Bagaimana merancang sistem penampil informasi pada

dot matrik, baik perangkat keras maupun lunak dengan memberikan fasilitas input untuk user ?

1.3. Batasan Masalah

Dalam penelitian ini, masalah difokuskan pada cara penampilan informasi pada sebuah penampil dot matrik. Ada beberapa batasan masalah, antara lain :

1. Jumlah karakter yang mampu ditampilkan secara bersamaan berjumlah 10 karakter, sedangkan untuk karakter berjalan maksimal sampai 1000 karakter.
2. Memberikan fasilitas masukan kepada user untuk mengganti informasi yang akan ditampilkan.

1.4. Tujuan Penelitian

Tujuan utama dari penelitian ini adalah untuk membuat alat penampil informasi pada dot matrik menggunakan mikrokontroler AT89C52, sedangkan tujuan khususnya adalah :

1. Mengaplikasikan ilmu-ilmu yang telah diperoleh.
2. Mengembangkan dan mempelajari pemrograman mikrokontroler lebih lanjut.
3. Mengaplikasikan penggunaan mikrokontroler Atmel AT89C52 untuk kebutuhan masyarakat.

1.5. Sistematika Penulisan

Dalam penyusunan laporan tugas akhir, sistematika penulisannya terdiri dari lima bab, yaitu :

BAB I PENDAHULUAN

Dalam bab ini dijelaskan tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Dalam bab ini dijelaskan tentang teori-teori yang digunakan dalam parancangan dan pembuatan sistem.

BAB III PERANCANGAN SISTEM

Dalam bab ini dijelaskan tentang perancangan sistem, desain perangkat keras, perangkat lunak, komponen yang digunakan, serta penjelasannya.

BAB IV PENGUJIAN DAN ANALISA

Bagian ini berisi hasil pengujian alat yang telah dibuat dan analisanya.

BAB V PENUTUP

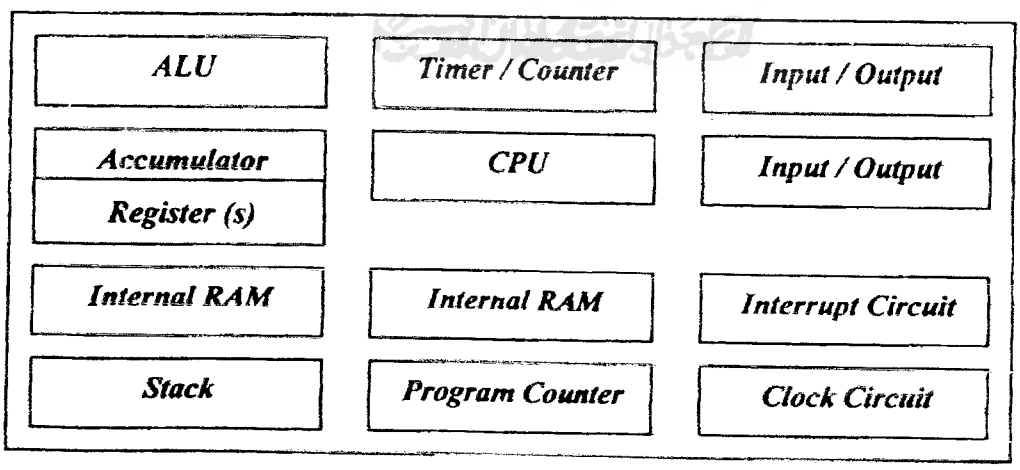
Bagian ini berisi kesimpulan dari alat yang dibuat dan saran-saran guna perbaikan dan pengembangan alat ini.

BAB II

LANDASAN TEORI

2.1. Mikrokontroler AT89C52

Mikrokontroler adalah suatu rangkaian terintegrasi yang tersusun atas beberapa komponen, antara lain : *Central Processing Unit (CPU), Read Only Memory (ROM), Random Access Memory (RAM), Timer,* dan *Input Output (I O)* yang dikemas dalam satu keping tunggal (*Chip*). Jadi, sebenarnya mikrokontroler merupakan sebuah piranti pengembangan mikroprosesor dengan teknik fabrikasi dan konsep pemrograman yang sama, sehingga memungkinkan pembuatan mikroprosesor multiguna. Pada gambar dibawah ini memperlihatkan bagian-bagian dalam mikrokontroler secara umum.



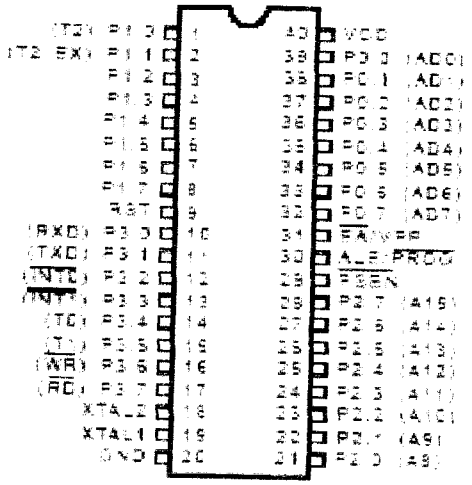
Gambar 2.1. Bagian-bagian dari suatu mikrokontroler

Mikrokontroler merupakan sistem mikroprosesor yang dirancang secara khusus untuk aplikasi dengan kendali sekuensial, yaitu digunakan untuk mengatur dan memonitor suatu sistem dengan urutan kerja tertentu. Gambar diatas memperlihatkan blok diagram mikrokontroler lengkap dengan komponen-komponennya dalam satu *chip*. Dari gambar tersebut terlihat bahwa mikrokontroler merupakan gabungan antara mikroprosesor dengan *Read Only Memory (ROM), Random Access Memory (RAM), Paralel I O, Serial I O, Counter, Timer,* dan pembangkit isyarat pulsa detak. Sedangkan suatu mikroprosesor sendiri atau CPU terdiri atas beberapa komponen, antara lain : *Aritmathic Logic Unit (ALU), Program Counter (PC), Stack Pointer (SP),* dan register-register.

Dalam sejarahnya, mikrokontroler MCS-51 merupakan jenis mikrokontroler yang termasuk tua, keluarga mikrokontroler MCS-51 adalah merupakan mikrokontroler yang paling populer saat ini. Keluarga ini diawali oleh Intel yang mengenalkan IC mikrokontroler type 8051 pada awal tahun 1980 an. Sampai saat ini sudah ada lebih 100 macam mikrokontroler turunan 8051, sehingga terbentuklah keluarga besar mikrokontroler dan biasa disebut MCS-51. Belakangan ini, pabrik IC Atmel ikut menambah anggota keluarga MCS-51. Produksi mikrokontroler MCS-51 Atmel dibagi dua macam, yang pertama yaitu mikrokontroler dengan jumlah pin 40 setara dengan jumlah pin 8051 yang asli, dan yang kedua adalah mikrokontroler dengan jumlah pin 20 yang merupakan mikrokontroler MCS-51 yang disederhanakan. Perbedaan keduanya adalah dalam hal kapasitas *Flash* PEROM (*Programmable and Erasable Read Only Memory*)

nya. AT89C51 mempunyai *Flash* PEROM dengan kapasitas 4 Kbytes, AT89C52 dengan kapasitas 8 Kbytes, AT89S53 dengan kapasitas 12 Kbytes, AT89C55 dengan kapasitas 20 Kbytes, dan AT 89C8252 berisikan 8 Kbytes *Flash* PEROM dan 2 Kbytes EEPROM (*Electrical Erasable and Programable ROM*). Penyederhanaan dilakukan pula pada mikrokontroler ukuran kecil ini, yaitu dengan cara mengurangi jalur I/O paralel, kemampuan yang lain sama sekali tidak mengalami pengurangan, penyederhanaan ini dimaksudkan untuk membentuk mikrokontroler yang ukuran fisiknya kecil, akan tetapi dengan kemampuan yang sama. Atmel memproduksi 3 buah mikrokontroler 'mini' ini masing-masing adalah AT89C1051 dengan kapasitas *Flash* PEROM 1 Kbyte, AT89C2051 dengan kapasitas 2 Kbytes, dan AT89C4051 dengan kapasitas 4 Kbytes.

AT89C52 adalah mikrokontroler 8 bit keluaran Atmel dengan 8 Kbytes *Flash* PEROM yang merupakan memori dengan teknologi *high density nonvolatile memory* dan kompatibel dengan mikrokontroler standar industri MCS-51, isi memori tersebut dapat diisi ulang atau dihapus berkali-kali sampai batas 1000 kali. Mikrokontroler ini merupakan *high performance* teknologi CMOS (*Complementary Metal Oxide Semiconductor*) dan dikemas dalam paket 40 pin dengan catu daya tunggal. Diagram susunan kaki dan simbol logika dari mikrokontroler AT89C52 dalam bentuk PDIP (*Plastic Dual In Line Package*) dapat dilihat pada gambar berikut ini.



Gambar 2.2. Konfigurasi pin dari chip mikrokontroler AT89C52.

Masing-masing pin pada gambar tersebut memiliki fungsi tersendiri. Satu kumpulan pin memiliki fungsi sama dan diwakili oleh sebuah register atau alamat tersendiri pada internal CPUnya, disebut juga port. Fungsi dari pin-pin tersebut adalah sebagai berikut :

1. Port 0

Port 0 dapat berfungsi sebagai I/O biasa, *low order multiplex address data* atau menerima kode byte pada saat *Flash Programing*. Pada fungsi sebagai I/O biasa, port ini dapat memberikan *output sink* ke delapan buah TTL input atau dapat diubah sebagai input dengan memberikan logika 1 pada port tersebut. Pada fungsi sebagai *low order multiplex address data*, port ini akan mempunyai *internal pull up*. Pada saat *Flash Programing* diperlukan *external pull up* terutama saat verifikasi program. Port 0 terdapat pada pin 32-39.

2. Port 1

Port 1 berfungsi sebagai I/O biasa atau menerima *low order address bytes* pada saat *Flash Programing*. Port ini mempunyai *internal pull up* dan berfungsi sebagai input dengan memberikan logika 1. Sebagai output, port ini dapat memberikan *output sink* ke empat buah input TTL. Port 1 terdapat pada pin 1-8.

3. Port 2

Port 2 berfungsi sebagai I/O biasa atau menerima *high order address bytes* pada saat mengakses memori secara 16 bit (Movx @DPTR). Pada saat mengakses memori secara bit, (Mov @Rn) port ini akan mengeluarkan isi dari P2 *Special Function Register (SFR)*. Port ini mempunyai *internal pull up* dan berfungsi sebagai input dengan memberikan logika 1. Sebagai output, port ini dapat memberikan *output sink* ke empat buah input TTL. Port 1 terdapat pada pin 21-28.

4. Port 3

Port 3 yang terdapat pada pin 10-17 berfungsi sebagai *input output* (I/O) yang mempunyai sifat sama dengan port 1 maupun port 2, sedangkan sebagai fungsi spesial, port-port ini mempunyai keterangan yang ditunjukkan pada tabel berikut ini :

Tabel 2.1. Fungsi khusus Port 3

PORT PIN	NO PIN	FUNGSI KHUSUS
P3.0	10	RXD (masukan data port serial)
P3.1	11	TXD (keluaran data port serial)
P3.2	12	INT0' (masukan interupsi 0 dari luar)
P3.3	13	INT1' (masukan interupsi 1 dari luar)
P3.4	14	T0 (masukan ke pencacah 0)
P3.5	15	T1 (masukan ke pencacah 1)
P3.6	16	WR' (sinyal baca untuk memori luar)
P3.7	17	RD' (sinyal tulis untuk memori luar)

5. PSEN (*Program Strobe Enable*)

PSEN adalah kontrol sinyal yang mengijinkan untuk mengakses program (code) memori *eksternal*. Pin ini dihubungkan ke pin OE (*output enable*) dari EPROM. Sinyal PSEN akan 0 pada tahap *fetch* (penjemputan) instruksi. PSEN akan selalu bernilai 0 pada pembacaan program memori *internal*. PSEN terdapat pada pin 29.

6. ALE (*Address Latch Enable*)

Pin ini dapat berfungsi sebagai *Address Latch Enable* (ALE) yang me–*latch low byte address* pada saat mengakses memori *eksternal*, sedangkan pada saat *Flash Programing* berfungsi sebagai *pulse input*. Pada operasi normal, ALE akan mengeluarkan sinyal clock sebesar 1/16 frekuensi oscilator, kecuali pada saat mengakses memori *eksternal*, sinyal clock pada pin ini dapat pula di *disable* dengan men-set bit 0 dari SFR dialamat 8Eh. ALE hanya akan aktif pada saat mengakses memory *eksternal* (Movx dan Movc). ALE terdapat pada pin 30.

7. EA (*External Access*)

Pada kondisi logika rendah, pin ini akan berfungsi sebagai EA, yaitu mikrokontroler akan menjalankan program yang ada pada memori *eksternal* setelah sistem direset. Jika berkondisi logika tinggi, pin ini akan berfungsi untuk menjalankan program yang ada pada memori *internal*. Pada saat *Flash Programing*, pin ini akan mendapat tegangan 12 Volt (VP). EA terdapat pada pin 31.

8. *On-Chip Oscillator*

AT89C52 telah memiliki *on-chip oscillator* yang dapat bekerja dengan menggunakan kristal *eksternal* yang dihubungkan ke pin XTAL1 dan XTAL2. Tambahan kapasitor untuk menstabilkan oscilator tersebut. Nilai kristal yang biasa dipakai oleh keluarga MCS-51 adalah 12 MHz. *On-chip oscillator* juga dapat menggunakan isyarat pulsa detak dari luar, misalnya AFG (*eksternal oscilator*) yang cukup dihubungkan pada pin XTAL1.

9. RST (*Reset*)

RST pada pin 9 merupakan reset dari AT89C52. Jika pada pin ini diberi masukan logika tinggi selama 2 *machine cycle*, maka register-register *internal* pada AT9C52 akan berisi nilai *default* setelah sistem di reset seperti ditunjukkan pada tabel berikut ini :

Tabel 2.2. Nilai register setelah direset

REGISTER	ISI
Program counter	0000H
Accumulator	00H
B register	00H
PSW	00H
SP	07H
DPTR	0000H
Port 0-3	FFH
IP (8031/8051)	XXX00000B
IP (8032/8052)	XX000000B
IE (8031/8051)	0XX00000B
IE (8032/8052)	0X000000B
Timer Register	00H
SCON	00H
SBUF	00H
PCON (HMOS)	0XXXXXXXB
PCON (CMOS)	0XXX0000B

10. Koneksi Catu Daya

Beroperasi pada tegangan 5 Volt. Pin Vcc terdapat pada pin 40, sedangkan Vss (*ground*) terdapat pada pin 20.

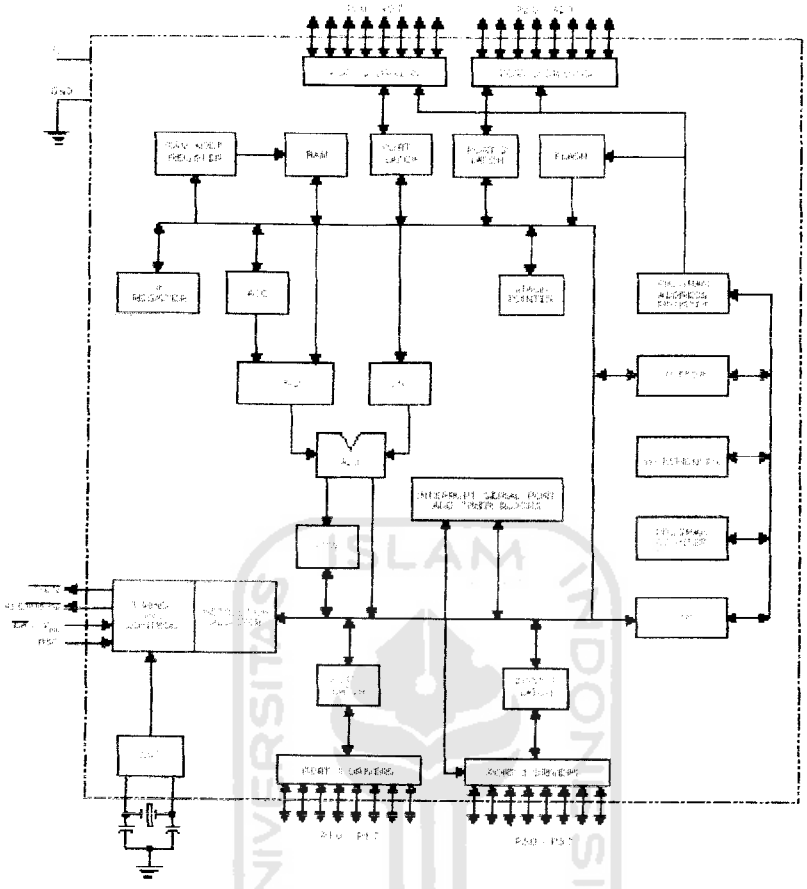
Untuk merancang suatu sistem yang menggunakan mikrokontroler AT89C52 sebagai basis utamanya, diperlukan pemahaman terhadap konstruksi, instruksi, dan pendukung operasi yang dimiliki oleh mikrokontroler tersebut. Berikut ini adalah sekilas gambaran dari mikrokontroler AT89C52 :

a. Konstruksi dasar mikrokontroler AT89C52.

Mikrokontroler ini dilengkapi dengan berbagai fasilitas, diantaranya adalah :

1. *Central Processing Unit* (CPU).
2. Memori data (RAM) didalam chip 256 bytes, *Flash* PEROM didalam chip sebesar 8 Kbytes yang dapat diisi ulang sebanyak 1000 kali.

3. Pengendali interupsi (*Interrupt Control*).
4. Bus Kendali (*Bus Control*).
5. UART (*Universal Asynchronous Receiver Transmitter*) yang digunakan untuk komunikasi data secara serial (Jalur seri berada pada pin RXD dan TXD).
6. Rangkaian oscilator didalam chip dengan frekuensi maksimal 24 MHz.
7. Empat buah port yang masing-masing berisi 8 bit, sifatnya 2 arah (I/O), dan setiap bitnya dapat dialamati. Salah satu portnya, yaitu port 3 (P3) juga dapat berfungsi untuk komunikasi data secara serial, interupsi, masukan untuk pencacah, dan masukan isyarat perintah baca dan tulis (R/W).
8. Pewaktu (*Timer*) / pencacah (*Counter*) 16 bit sebanyak 3 buah.



Gambar 2.3. Arsitektur perangkat keras AT89C52

Dari uraian tersebut, letak dan konstruksi dari komponen-komponen yang telah disebutkan dapat dilihat secara jelas seperti pada Gambar 2.3.

b. RAM, ROM, dan register dalam AT89C52.

Dalam pengertian MCS-51, *Flash* PEROM merupakan memori penampung program pengendali AT89C52, dikenal sebagai memori program (nomor $0000 sampai $07FF). Sedangkan RAM dalam chip AT89C52 adalah memori data (dialamati dari alamat $00 sampai $FF), yaitu memori yang dipakai untuk menyimpan data.

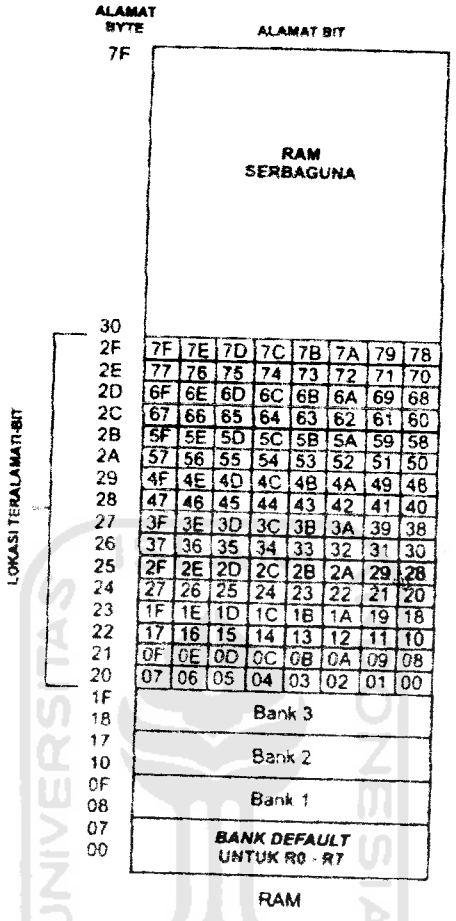
Memori data pada AT89C52 dibagi menjadi 2 bagian, yaitu :

1). Memori alamat $00 sampai $7F merupakan memori seperti layaknya RAM yang dipakai sebagai memori penyimpan data biasa, terdiri atas :

a). Register serbaguna (*General Purpose Register*) terdapat pada memori alamat $00 sampai $18, memori sebanyak 32 bytes ini dikelompokkan menjadi 4 kelompok register (*Register Bank*), 8 bytes memori dari masing-masing kelompok itu dikenal sebagai R0, R1, R2, R3, R4, R5, R6, dan R7.

b). Memori level bit (memori alamat $20 sampai $2F). Setiap byte memori didaerah ini bisa dipakai untuk menampung bit informasi yang masing-masing dialamati sendiri, dengan demikian dari 16 bytes memori yang ada dapat dipakai untuk menyimpan 128 bit (16 X 8 bit) yang dialamati dengan bit alamat $00 sampai $7F.

c). Memori alamat $30 sampai $7F (sebanyak 80 bytes) merupakan memori data biasa, dapat dipakai untuk menyimpan data maupun dipakai sebagai stack.



Gambar 2.4. Peta memori data

2. Memori alamat $80 sampai $FF dipakai sangat khusus yang dinamakan *Special Function Register* (SFR).

Pada AT89C52 terdapat register-register baku seperti yang dapat kita jumpai pada semua jenis mikrokontroler ataupun mikroprosesor, seperti *Program Counter* (PC), *Accumulator*, *Stack Pointer* (SP), *Register*, dan Program Status Register. Juga terdapat register-register khas yang hanya terdapat pada keluarga MCS-51, adapun register-register dan fungsi dari register dalam AT89C52 tersebut adalah sebagai berikut :

1) *Program Counter* (PC).

Register yang ditempatkan di tempat tersendiri didalam inti prosesor. PC mempunyai kapasitas 16 bit, didalam PC dicatat nomor memori program yang menyimpan instruksi berikutnya yang akan diambil (*fetch*) sebagai instruksi untuk dikerjakan.

2) Akumulator.

Sesuai dengan namanya, akumulator adalah sebuah register yang berfungsi untuk menampung (*accumulate*) hasil-hasil pengolahan data dari banyak instruksi MCS-51, register ini berada pada alamat E0h, akumulator dapat menampung data 8 bit (1 byte) dan merupakan register yang paling banyak kegunaannya, lebih dari setengah instruksi-instruksi MCS-51 melibatkan akumulator.

3) *Stack Pointer Register (SP Register).*

Merupakan salah satu bagian dari memori data yang dipakai sebagai stack (tumpukan), yaitu tempat yang dipakai untuk menyimpan sementara nilai PC sebelum prosesor menjalankan subrutin, nilai tersebut akan diambil kembali dari stack dan dikembalikan ke PC saat prosesor selesai menjalankan subrutin. Stack Pointer adalah register yang berada pada alamat 81h dan berfungsi untuk mengatur kerja stack, dalam SP disimpan alamat memori data yang dipakai untuk operasi stack berikutnya.

4) *Program Status Word* (PSW).

Program Status Word terletak pada alamat D0h dan berfungsi untuk mencatat kondisi prosesor setelah melaksanakan instruksi.

5) Register B.

Merupakan register dengan kapasitas 8 bit, sebagai register pembantu akumulator saat menjalankan instruksi perkalian dan pembagian, register ini dapat dialamati secara bit.

6) DPH dan DPL.

Data Pointer High Byte (DPH) dan *Data Pointer Low Byte* (DPL) masing-masing merupakan register dengan kapasitas 8 bit, tetapi dalam pemakaiannya kedua register ini digabungkan menjadi satu register 16 bit yang dinamakan sebagai *Data Pointer Register* (DPTR). Sesuai dengan namanya, Register ini dipakai untuk mengalamati data dalam jangkauan yang luas.

7) P0, Pl, P2, dan P3.

Seluruh port dapat diakses dengan pengalamatan secara bit, merupakan sarana I/0 dua arah. Selain dipakai sebagai port I/0, PO dan P2 dapat pula dipakai untuk saluran data (*Data Bus*) dan saluran alamat (*Address Bus*) yang diperlukan AT89C52 untuk dapat menambah memori diluar chip.

8) SBUF (*Serial Buffer*).

Register Serial Buffer dipergunakan untuk mengirim data dan menerima data dengan UART yang terdapat dalam IC AT89C52. Data yang disimpan di dalam SBUF akan dikirim keluar secara serial lewat kaki TXD, sebaliknya data seri yang diterima di kaki RXD dapat diambil di register SBUF, jadi SBUF akan berfungsi sebagai port *output* pada saat register ini diisi data dan SBUF akan menjadi port *input* apabila isinya diambil.

9) SCON (*Serial Control*).

Register SCON dipergunakan untuk mengatur mode operasi UART didalam IC AT89C52, hal-hal yang diatur meliputi penentuan kecepatan pengiriman data serial (*baud rate*), mengaktifkan fasilitas penerimaan data serial (fasilitas ini tidak perlu diatur). Register SCON digunakan pula untuk memantau proses pengiriman dan penerimaan data serial. Register ini dapat dialamati secara bit.

10) TL0 / TH0 (Timer 0 Low / High).

Kedua register ini bersama membentuk timer 0, yang merupakan pencacah naik (*up counter*), mode operasi kedua register ini diatur oleh register TMOD dan register TCON. Hal-hal yang dapat diatur antara lain adalah sumber detak untuk pencacah, nilai awal pencacah bilamana proses pencacahan mulai atau berhenti, dan lain sebagainya.

11) TL 1 / TH 1 (*Timer 1 Low / High*).

Kedua register ini bersama membentuk *timer* 1, yang merupakan pencacah naik (*up counter*), mode operasi kedua register ini diatur oleh register TMOD dan register TCON. Hal-hal yang dapat diatur antara lain adalah sumber detak untuk pencacah, nilai awal pencacah bilamana proses pericacahan mulai atau berhenti, dan lain sebagainya.

12) TMOD (*Timer Mode*)

Register TMOD dipergunakan untuk mengatur mode operasi *timer* 0 dan *timer* 1, lewat register ini masing-masing timer dapat diatur menjadi *timer* 16 bit, *timer* 13 bit, *timer* 8 bit yang dapat diisi ulang secara otomatis, atau 2 buah *timer* 8 bit yang terpisah. Selain itu, dapat diatur pula agar proses-proses

pencacahan *timer* dapat dikendalikan lewat sinyal dari luar IC AT89C52 atau *timer* dipakai untuk mencacah isyarat-isyarat dari luar IC.

13) TCON (*Timer Control*).

Register TCON dipergunakan untuk memulai atau menghentikan proses pencacahan *timer* dan dipakai untuk memantau apakah terjadi limpahan dalam proses pencacahan. Selain itu, masih tersisa 4 bit yang lain dalam register TCON yang tidak digunakan untuk mengatur *timer*, melainkan digunakan untuk mengatur isyarat interupsi yang diterima di pin INT0 atau INT1 dan digunakan untuk memantau apakah ada permintaan interupsi pada kedua pin tersebut. Register ini dapat dialamati secara bit.

14) IE (*Interrupt Enable*).

Register ini dipergunakan untuk mengaktifkan atau me-non aktifkan sarana interupsi, bit 0 sampai bit 6 dari register IE (IE.0, IE.1, ..., IE.6) digunakan untuk mengatur masing-masing sumber interupsi (sesungguhnya IE.6 tidak dipergunakan), sedangkan IE.7 digunakan untuk mengatur sistem interupsi secara keseluruhan, jika IE.7 = 0 mengakibatkan sistem interupsi non aktif tidak mempedulikan IE.0, IE.1, ..., IE6. Register ini dapat dialamati dengan cara pengalamatan bit.

15) IP (*Interrupt Priority*).

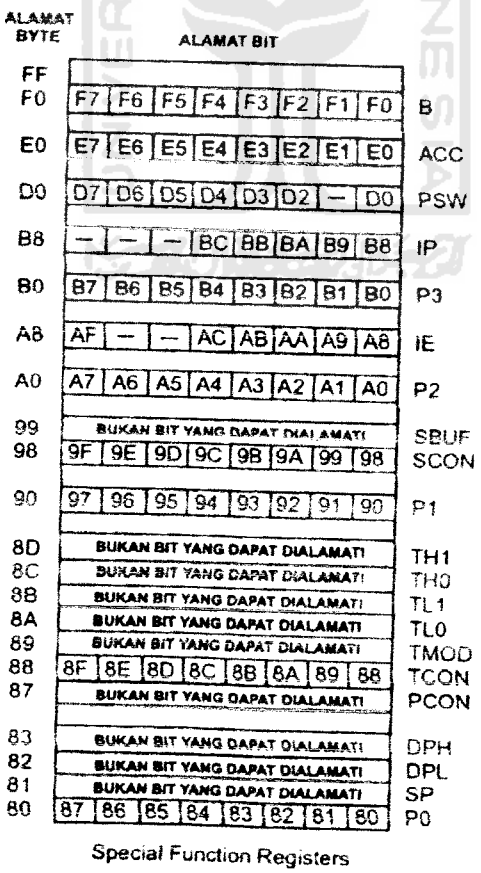
Register ini dipergunakan untuk mengatur prioritas dari masing-masing sumber interupsi, masing-masing sumber interupsi dapat diberi prioritas tinggi dengan memberikan logika '1' pada bit bersangkutan dalam register ini. Sumber interupsi yang prioritasnya tinggi dapat menginterupsi proses dari

sumber interupsi yang prioritasnya lebih rendah. Register ini dapat dialamati dengan cara pengalamatan bit.

16) PCON (*Power Control*).

Register PCON dipergunakan untuk mengatur pemakaian daya IC AT89C52, dengan cara 'menidurkan' IC tersebut sehingga hanya memerlukan arus kerja yang sangat kecil. Salah satu bit dalam register ini digunakan untuk menggandakan kecepatan pengiriman data seri (*baud rate*) dari UART didalam IC AT89C52.

Adapun letak register-register tersebut pada *Special Function Register* (SFR) seperti pada Gambar 2.5. berikut ini :



Gambar 2.5. Peta *Special Function Register* (SFR)

c. *Timer* Pada AT89C52.

Mikrokontroler AT89C52 dibuat dengan dibekali 3 buah *timer*, ketiganya dapat di kendalikan, diset, dibaca, dan dikonfigurasi sendiri-sendiri. *Timer* AT89C52 memiliki 3 fungsi umum, yaitu :

1) Menghitung waktu antara 2 kejadian (*event*).

2) Menghitung jumlah kejadian itu sendiri.

3) Membangkitkan *baud rate* untuk port serial.

Sebuah *timer* bekerja dengan mencacah, tidak tergantung pada fungsi sebagai register, *counter*, ataupun *generator baud rate*. Sebuah *timer* akan selalu membentuk tundaan waktu dengan melakukan operasi penambahan satu (*Increment*) secara terus menerus.

d. Mode Pengalamatan.

Program pengendali mikrokontroler disusun dari kumpulan instruksi. Instruksi-instruksi tersebut setara dengan kalimat perintah bahasa manusia yang hanya terdiri atas predikat dan obyek. Dengan demikian, tahap pertama pembuatan program pengendali mikrokontroler dimulai dengan pengenalan dan pemahaman predikat (kata kerja) dan obyek apa saja yang dimiliki oleh mikrokontroler.

Obyek dalam pemrograman mikrokontroler adalah data yang tersimpan didalam memori, register, atau alat *Input Output*. Sedangkan 'kata kerja' yang dikenal, secara umum dikelompokkan menjadi perintah untuk perpindahan data, operasi aritmatika, operasi logika, pengaturan alur program, dan beberapa hal

khusus. Kombinasi 'kata kerja' dan 'obyek' inilah yang membentuk perintah pengatur kerja mikrokontroler.

Instruksi Mov A, 7Fh merupakan contoh sebuah instruksi dasar yang sangat spesifik. Mov merupakan 'kata kerja' yang memerintahkan untuk memindahkan data, merupakan predikat dalam kalimat perintah ini. Sedangkan obyeknya adalah data yang dipindahkan, dalam hal ini adalah data yang berada didalam memori nomor 7Fh dipindahkan ke akumulator A.

Data dapat berada di berbagai tempat yang berlainan, dengan demikian dikenal beberapa cara untuk mengalamati atau menyebut alamat suatu data (dalam bahasa Inggris disebut sebagai *Addressing Mode*) yang setiap mode pengalamatan memberikan fleksibilitas khusus yang sangat penting. Mode pengalamatan ini antara lain adalah :

1) Pengalamatan Data Konstan (*Immediate Addressing Mode*).

Contoh instruksi : Mov A, #20h.

Data konstan merupakan data yang berada didalam instruksi. Contoh instruksi tersebut mempunyai makna data konstan #20h (sebagai data konstan ditandai dengan '#') dipindahkan ke akumulator A. Nilai dari suatu konstanta tersebut dapat segera menyatu dengan *op code* dalam memori program.

2) Pengalamatan Langsung (*Direct Addressing Mode*).

Contoh instruksi : Mov A, 30h.

Cara tersebut digunakan untuk menunjuk data yang berada di dalam memori dengan cara menyebut alamat memori tempat data tersebut berada. Contoh instruksi diatas mempunyai makna data yang berada di dalam alamat memori

30h dipindahkan ke akumulator A. Sekilas instruksi tersebut sama dengan instruksi data konstan diatas. Perbedaannya, instruksi diatas menggunakan tanda # yang menandai 20h adalah data konstan, sedangkan pada instruksi ini, karena tidak ada tanda #, maka 30h adalah alamat dari memori. Dalam pengalamatan ini, *operand-operand* ditentukan berdasar alamat 8 bit (1 *byte*) dalam suatu instruksi. Hanya RAM data internal dan SFR saja yang dapat diakses secara langsung.

3) Pengalamatan Tak Langsung (*Indirect Addressing Mode*).

Contoh instruksi : Mov A, @R0.

Cara tersebut digunakan untuk menunjuk data yang berada di dalam alamat memori, apabila memori penyimpan data ini letaknya berubah-ubah sehingga alamat memori tidak disebut secara langsung, tetapi di'titip'-kan ke register lain. Dalam instruksi ini, register serbaguna R0 digunakan untuk mencatat alamat memori, sehingga instruksi ini mempunyai makna memori yang alamatnya tercatat dalam R0 isinya dipindahkan ke akumulator A. Tanda @ dipakai untuk menandai alamat memori disimpan di dalam R0. Bandingkan dengan pengalamatan secara langsung diatas, dalam instruksi ini alamat memori terlebih dahulu disimpan di R0, dan R0 berperan menunjuk memori mana yang digunakan, sehingga apabila R0 berubah, memori yang ditunjuk juga akan berubah pula. Dalam hal ini R0 berfungsi sebagai register penampung alamat, selain R0, register serbaguna R1 juga dapat digunakan sebagai register penampung alamat. Dalam pengalamatan ini, baik RAM *internal* maupun *eksternal* dapat diakses secara tak langsung.

4) Pengalamatan Data dalam Register (*Register Addressing Mode*).

Contoh instruksi : Mov A, R5.

Instruksi ini mempunyai makna data dalam register serbaguna R5 dipindahkan ke akumulator A. Instruksi ini membuat register serbaguna R0 sampai R7 sebagai tempat penyimpan data yang praktis dan kerjanya sangat cepat.

5) Pengalamatan Kode (*Code Indirect Addressing Mode*).

Contoh instruksi : Movc A, @A+DPTR.

Perhatikan dalam instruksi ini, Mov digantikan dengan Movc, tambahan huruf c tersebut dimaksudkan untuk membedakan bahwa instruksi ini digunakan didalam memori program (mov tanpa huruf c artinya instruksi dipakai di memori data). Tanda '@' digunakan untuk memberi tanda bahwa A+DPTR digunakan untuk menyatakan alamat memori yang isinya di pindahkan ke akumulator A, dalam hal ini nilai yang tersimpan di dalam DPTR (Data Pointer Register 2 byte) ditambahkan dengan nilai yang tersimpan di akumulator A (1 byte) digunakan untuk menunjuk alamat memori program.

6) Pengalamatan Bit (*Bit Addressable*).

Proses pengalamatan ketika operand menunjuk ke alamat pada RAM internal ataupun register fungsi khusus yang mempunyai kemampuan dialamati secara bit.

Berdasarkan penulisannya, pengalamatan ini terdiri atas beberapa macam :

a) Langsung menunjuk ke alamat bit, contoh : Setb 0B0h.

b) Menggunakan operator titik, contoh : Setb P3.0.

b) Menggunakan lambang standar *Assembler*, contoh : Clr RXD.

c) Menggunakan lambang *Assembler* bebas, contoh : Baca bit P2. 5

C1r Baca

e. Dasar Kerja Program.

Program untuk mengendalikan kerja dari mikrokontroler disimpan dalam memori program. Program pengendali tersebut merupakan kumpulan dari instruksi kerja mikrokontroler. Suatu instruksi MCS-51 merupakan kode yang panjangnya antara 1 sampai 4 Bytes.

Selama mikrokontroler bekerja, instruksi kerja tersebut byte demi byte diambil ke CPU dan selanjutnya digunakan untuk mengatur kerja mikrokontroler. Proses pengambilan instruksi dari memori program disebut '*fetch cycles*' dan saat-saat CPU melaksanakan instruksi disebut dengan '*execute cycles*'.

Semua mikrokontroler maupun mikroprosesor dilengkapi dengan sebuah register yang berfungsi khusus untuk mengatur *fetch cycles*, register tersebut dinamakan sebagai *Program Counter* (PC). Nilai PC secara otomatis bertambah satu setiap kali selesai mengambil 1 byte isi memori program, dengan demikian isi memori program dapat berurutan diumpankan ke CPU.

Saat MCS-51 di reset, isi *program counter* di-reset menjadi 0000, artinya sesaat setelah reset, isi dari memori program alamat 0 dan seterusnya akan diambil ke CPU dan diperlakukan sebagai instruksi yang akan mengatur kerja mikrokontroler. Dengan demikian, awal dari program pengendali MCS-51 harus ditempatkan di memori alamat 0, setelah reset MCS-51 menjalankan program

mulai dari memori program alamat 0000, dengan melakukan proses *fetch cycles* dan *execute cycles* terus menerus tanpa henti.

Jika sarana interupsi diaktifkan dan tegangan di pin INT0 (pin nomor 6) berubah dari '1' menjadi '0', maka proses menjalankan program diatas akan dihentikan sebentar, mikrokontroler melayani dulu permintaan interupsi, CPU akan melanjutkan mengerjakan program utama lagi.

Untuk melaksanakan hal diatas, pertama-tama CPU menyimpan nilai *Program Counter* ke *Stack* (*Stack* merupakan satu bagian kecil dari data memori RAM), kemudian mengganti isi PC dengan 0003, artinya MCS-51 akan melaksanakan program yang ditempatkan di memori program mulai byte ke-3 untuk melayani interupsi yang diterima dari pin INT0. Adalah tugas *programmer* untuk mengatur dimana program yang dipakai untuk melayani interupsi lewat INT0 harus ditempatkan.

Selesai melayani interupsi, nilai PC yang semula disimpan di dalam *stack* akan dikembalikan ke PC, dengan demikian CPU bisa melanjutkan '*execute cycles*'-nya di program utama.

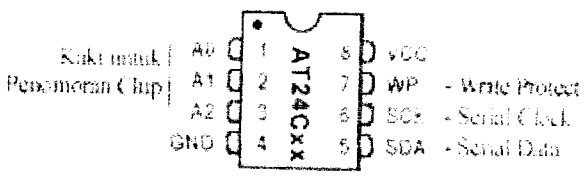
Selain INT0, AT89C52 dapat menerima interupsi dari INT1, UART, dan *Timer*. Agar permintaan interupsi itu dapat dilayani dengan program yang berlainan, maka masing-masing sumber interupsi itu mempunyai nomor awal program untuk layanan interupsi yang berlainan.

2.2. *Serial* EEPROM (*Electrically Erasable Programmable ROM*) AT24C64

Mengatasi terbatasnya jumlah kaki IC mikrokontroler, beberapa perusahaan IC mengembangkan teknik transfer data secara seri untuk menghubungkan IC mikrokontroler ke IC pendukungnya. Transfer data secara seri antar IC ini tidak ada hubungannya dengan transfer data seri yang biasa dipakai untuk modem.

Teknik transfer data secara seri antar IC dikembangkan oleh 3 perusahaan IC, yang pertama adalah teknik I^2C (*Inter Integrated Circuit*) yang dikenalkan oleh Philips, teknik SPI (*Serial Peripheral Interface*) dari Motorola, dan teknik *MicroWire* ciptaan National Semiconductor. Teknik I^2C memakai 2 jalur untuk keperluan transfer data secara seri, sedangkan SPI dan *MicroWire* memakai 3 jalur. Semua teknik mempunyai 1 jalur untuk *Clock*, I^2C hanya punya satu jalur data 2 arah, sedangkan SPI dan *MicroWire* mempunyai 2 jalur data satu arah, masing-masing untuk jalur data masuk dan jalur data keluar.

Salah satu jenis IC SEEPROM adalah IC dengan kode AT24C64 buatan Atmel. Kapasitas memori dari IC ini adalah 8192 bytes, dengan 1 byte berisi 8 bit, maka jumlah bitnya adalah 64 Kbit.



Gambar 2.6. Susunan kaki IC AT24C64

Kaki SDA (kaki nomor 5) dan kaki SCK (kaki nomor 6) merupakan kaki baku IC jenis I^2C, kedua kaki inilah yang mebentuk I^2C Bus.

Kaki nomor 7 (WP – *Write Protect*) merupakan kaki yang dipakai untuk melindungi isi yang disimpan di dalam IC SEEPROM, jika kaki ini diberi tegangan '1' maka IC dalam keadaan ter-proteksi, isinya tidak dapat diganti. Agar bisa menuliskan informasi ke dalam IC ini, kaki ini harus diberi tegangan '0'.

Kaki nomor 1 sampai dengan nomor 3 (A0, A1, dan A2) merupakan fasilitas untuk penomoran chip, hal ini diperlukan kalau dalam satu rangkaian dipakai lebih dari satu IC SEEPROM sejenis.

2.3. Sinyal Dasar I^2C

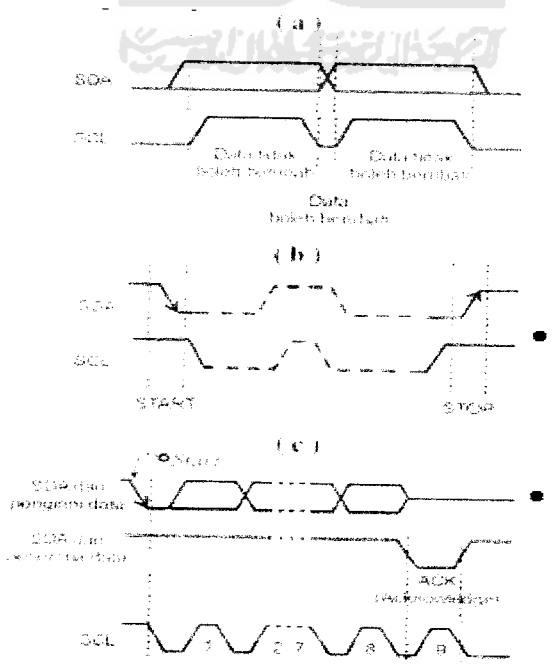
Mengingat hanya 2 saluran saja yang dipakai I^2C Bus, padahal I^2C Bus diharapkan bisa dipakai membentuk jaringan kecil dengan banyak peralatan I^2C, maka dalam konsep I^2C ditentukan sinyal dan tatacara dasar untuk memperlancar komunikasi antar peralatan I^2C tersebut. Sinyal dasar I^2C meliputi sinyal START, STOP, dan ACK sebagai berikut :

1. SCK merupakan sinyal clock untuk 'mendorong' data di SDA. Dalam keadaan tidak ada transfer data, SDA dan SCK harus dalam keadaan '1'. Data di SDA boleh berubah hanya pada saat SCK ='0', seperti digambarkan dalam diagram waktu Gambar 2.7.(a), isi SDA diambil peralatan I^2C pada saat SCL berubah dari '1' menjadi '0'. Jika terjadi perubahan SDA pada saat SCL = '1', perubahan itu diartikan sebagai sinyal *START* atau *STOP*.

2. Sinyal *START* menandakan master akan mulai mengirim data, sinyal ini terlihat di bagian kiri Gambar 2.7.(b) berupa perubahan tegangan SDA dari '1' menjadi '0' pada saat SCK='1'.
3. Sinyal *STOP* menandakan master akan mengakhiri komunikasi data, sinyal ini terlihat di bagian kanan Gambar 2.7.(b) berupa perubahan tegangan SDA dari '0' menjadi '1' pada saat SCK='1'.

Catatan : Sinyal *START* dan *STOP* muncul saat awal dan akhir pengiriman 1 blok data, bukan sinyal yang muncul pada awal dan akhir pengiriman 1 byte data.

Gambar 2.7.(c) memperlihatkan interaksi antara dua peralatan I^2C, setelah penerima data menerima data 8 bit, pada clock yang ke-9 penerima data membalas dengan '0' sebagai tanda data 8 bit tadi sudah diterima dengan baik, sinyal '0' ini dinamakan sebagai sinyal ACK.

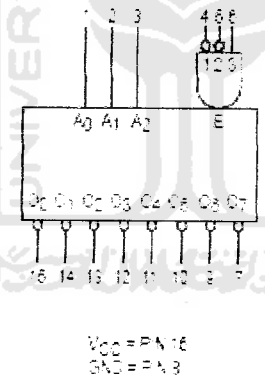


Gambar 2.7. Sinyal dasar I^2C

2.4. Demultiplexer 74LS138

IC LS138 adalah sebuah demultiplexer 3 ke 8 dengan kecepatan tinggi. Demultiplexer ini mempunyai 3 bit alamat (A0, A1,A2) dan 3 input (dua aktif rendah (E1,E2) dan satu aktif tinggi (E3)). Keluaran dari IC ini adalah aktif rendah. Bila E1 dan E2 *low* dan E3 *high*, maka salah satu keluaran yang bersesuaian dengan alamat yang diberikan pada alamatnya akan menjadi rendah, sementara yang lainnya berlogika tinggi.

Bila salah satu atau semua masukan, yaitu E1, E2, dan E3 tidak sesuai dengan ketentuan diatas, maka semua keluaran akan berkondisi *high*. Dibawah ini digambarkan pin-out dari IC 74LS138 tersebut.



Gambar 2.8. Pin-out dari IC 74LS138

Tabel kebenaran dari masukan dan keluaran IC tersebut tampak seperti pada tabel dibawah ini.

Tabel 2.3. Tabel kebenaran dari IC 74LS138

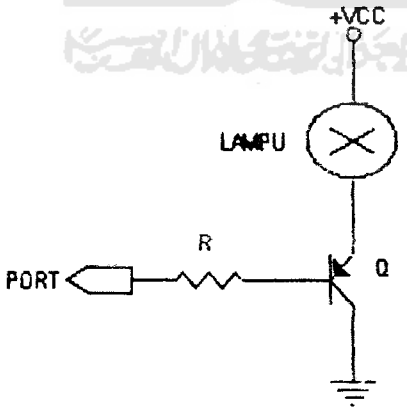
TRUTH TABLE

INPUTS						OUTPUTS							
E_1	E_2	E_3	A_0	A_1	A_2	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	L	L	L	L	L	L	H	H	H	H	H	H	H
L	L	L	H	L	L	H	L	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H
L	L	L	H	H	L	H	H	H	L	H	H	H	H
L	L	L	L	L	H	H	H	H	H	L	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H
L	L	L	L	H	H	H	H	H	H	H	H	L	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

2.5. Transistor Sebagai Saklar

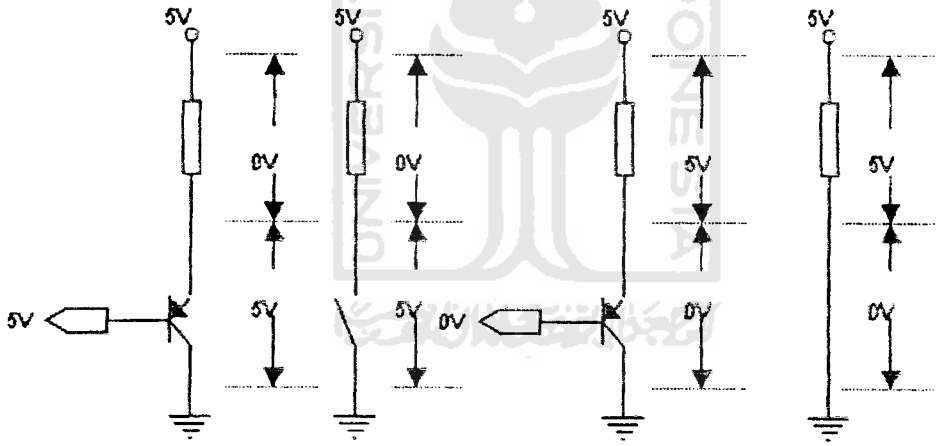
Salah satu penggunaan umum dari transistor dalam suatu rangkaian digital adalah sebagai saklar (*Switching*). Gambar 2.9 berikut merupakan rangkaian transistor sebagai saklar.



Gambar 2.9. Transistor sebagai saklar untuk menyalakan lampu

Untuk dapat menyalakan lampu, antara kaki emitor dengan kolektor harus dihubungsingkatkan. Untuk itu diperlukan arus yang cukup pada basis sebagai

pembias (menempatkan titik operasi transistor pada daerah jenuh), yaitu dengan membuat tegangan pada emitor lebih besar dari tegangan pada basis dengan selisih lebih besar dari tegangan *cut off* (umumnya diatas 0.6V, tergantung spesifikasi dari pabrik dan bahan yang digunakan). Arus pada basis yang diperlukan relatif sangat kecil (dibatasi oleh tahanan RB). Apabila dalam rangkaian pada Gambar 2.9, transistor yang digunakan memiliki tingkat penguatan sebesar 50, berarti arus sebesar 1 mA pada basis dapat digunakan untuk mengendalikan arus pada emitor ke kolektor sebesar 50 mA. Untuk lebih jelasnya diperlihatkan pada Gambar 2.10 berikut ini.



Gambar 2.10. Transistor sebagai saklar

Pada saat tegangan V_{CC} denganV_{BB} bernilai sama, tidak ada arus mengalir melewati kaki basis, sehingga transistor tidak terbias dan tidak ada arus melewati kaki emitor ke kolektor atau identik dengan kondisi saklar terbuka (OFF), sehingga beban tidak aktif. Pada saat tegangan V_{CC} jauh lebih besar dari V_{BB} atau pada kondisi kaki basis di ground-kan, arus akan mengalir melalui kaki basis

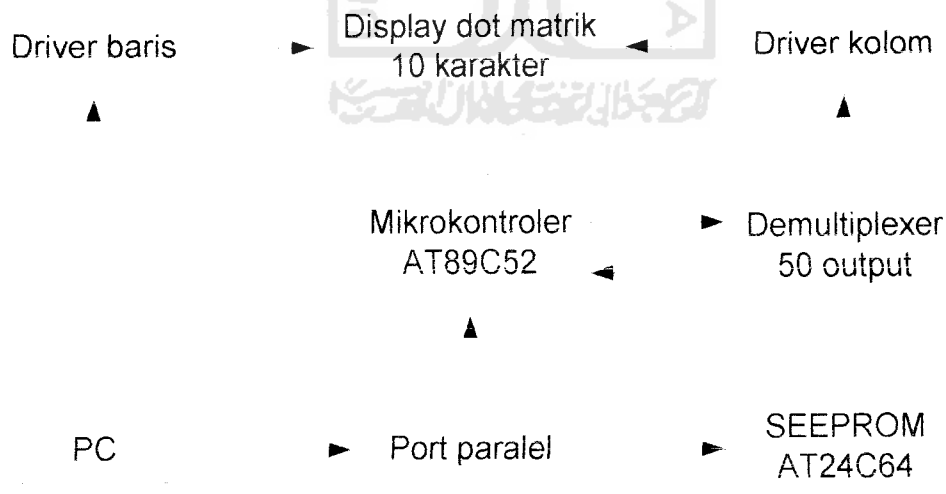
sebagai bias bagi transistor, akibatnya mengalir pula arus yang lebih besar melalui kaki emitor ke kaki kolektor sehingga aktiflah beban yang terpasang pada kaki emitor, keadaan ini identik dengan menutupnya saklar mekanis (ON).

BAB III

PERANCANGAN SISTEM

Dalam bab ini akan dibahas mengenai perancangan alat yang dibuat. Alat penampil informasi dot matrik ini mempunyai dua bagian dalam perancangannya, yaitu perancangan perangkat keras dan perangkat lunak. Untuk perancangan perangkat lunak juga ada dua bahasan, yaitu perangkat lunak yang akan diisikan kedalam mikrokontroler dan perangkat lunak untuk mengisikan karakter pada alat yang akan dijalankan pada komputer.

Sebelum masuk ke penjelasan mengenai perancangan alat secara lebih jelas, dibawah ini digambarkan diagram blok dari alat yang dibuat.



Gambar 3.1. Diagram blok dari alat yang dibuat

Jantung utama dari rangkaian ini adalah mikrokontroler keluaran Atmel dengan seri AT89C52 yang memiliki kapasitas *flash* memori (memori untuk tempat menyimpan program yang akan di isikan ke dalam mikrokontroler) sebesar 8 Kbytes.

Tugas-tugas mikrokontroler dari alat ini ialah menerima kiriman karakter dari komputer, lalu mengkodekan karakter yang diterimanya dalam bentuk kode tampilan pada dot matrik dan menyimpannya dalam IC memori AT24C64. Karena dot matrik yang digunakan adalah dot matrik 5 X 8, dimana 5 jalur untuk kolom dan 8 jalur untuk baris, maka untuk satu karakter membutuhkan alamat memori sebesar 5 byte (1 byte 8 bit, = 5 X 8) untuk menyimpan kode penampil karakternya.

Alat yang dibuat ini mempunyai kemampuan untuk menyimpan memori karakter untuk ditampilkan sebanyak 1000 karakter, dengan demikian maksimal memori dari SEEPROM yang digunakan adalah sebesar 1000 dikalikan dengan 5 byte, yaitu 5 kilobyte. SEEPROM yang digunakan mempunyai kapasitas memori sebesar 8192 bytes, berarti masih tersisa memorinya sebesar 3192 bytes.

Setelah semua karakter yang dikirim oleh komputer dikodekan dan disimpan dalam memori SEEPROM, tugas selanjutnya dari mikrokontroler adalah menampilkan kode-kode huruf yang telah disimpan dalam IC SEEPROM tadi ke penampil dot matrik dengan sarana bantuan dari *driver* baris sejumlah 7 jalur, IC demultiplexer, dan *driver* kolom sejumlah 50 jalur. Kemudian tampilan pada dot matrik ini digeser penyalaan hurufnya oleh mikrokontroler.

Berikut ini adalah uraian dari perancangan alat baik perancangan perangkat kerasnya maupunperangkat lunak.

3.1. Perancangan Perangkat Keras

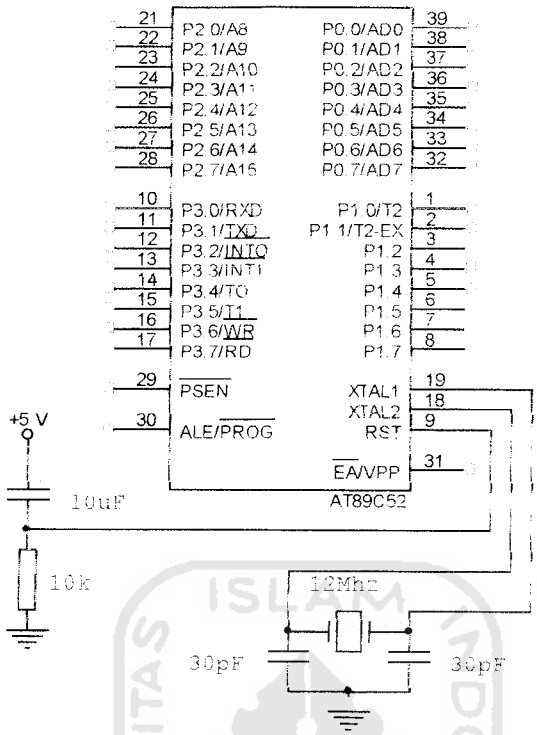
Perangkat keras yang dibuat memiliki beberapa bagian yang antara lain adalah sebagai berikut:

- Rangkaian sistem minimum mikrokontroler AT89C52
- Rangkaian *driver* baris
- Rangkaian demultiplexer dengan keluaran sebanyak 50 jalur
- Rangkaian *driver* kolom
- Rangkaian memori SEEPROM AT24C64
- Rangkaian catu daya

Untuk penjelasan dari masing-masing bagian akan diuraikan dibawah ini.

3.1.1. Rangkaian Sistem Minimum Mikrokontroler AT89C52

Rangkaian sistem minimum AT89C52 dibangun dengan menggunakan beberapa komponen, yaitu IC mikrokontroler itu sendiri, kristal 12 Mhz, sebuah resistor, dan beberapa buah kondensator. Mikrokontroler pada saat pertama kali diberi sumber arus dari catu daya akan memberikan nilai logika tinggi keseluruh kaki-kaki portnya, dengan demikian dapat dikatakan bahwa mikrokontroler ini kondisi aktifnya adalah rendah. Gambar 3.2 menunjukkan rangkaian dari sistem minimum mikrokontroler AT89C52.



Gambar 3.2. Rangkaian sistem minimum mikrokontroler AT89C52

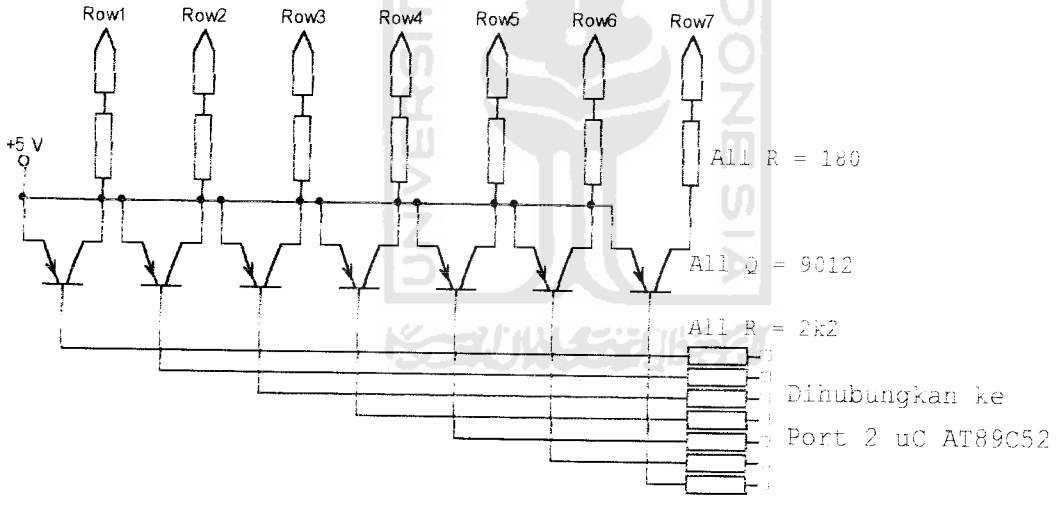
Dipilihnya mikrokontroler jenis ini sebagai komponen kontrol utama rangkaian, karena mudah memperolehnya dan mempunyai kapasitas *flash memory* yang cukup untuk program yang akan dimasukkan kedalamnya.

3.1.2. Rangkaian *Driver* Baris

Dot matrik yang digunakan pada alat penampil tulisan ini adalah dot matrik dengan jumlah dot sebanyak 5 X 8, dimana 5 adalah jumlah kolom dan 8 jumlah baris. Akan tetapi, pada alat ini untuk jalur barisnya hanya digunakan sebanyak 7 buah agar bentuk huruf yang ditampilkan kesimetrisannya antara bagian bawah dan atas menjadi baik.

Kutub dari catu yang dibutuhkan untuk jalur masukan baris ini adalah kutub positif. Karena mikrokontroler berada pada daerah aktif rendah dan aktif rendah pada mikrokontroler ini memiliki arus yang lebih kuat daripada logika *high*-nya, maka untuk dapat mengontrol keluaran positif pada baris adalah dengan aktif rendah. Keluaran dari mikrokontroler ini dihubungkan dengan rangkaian transistor jenis PNP dan transistor ini dapat juga dikatakan sebagai *driver* baris dari dot matrik atau rangkaian pensaklaran elektronis catuan positif dari catu daya untuk dot matrik.

Rangkaian dari *driver* baris tersebut nampak pada Gambar 3.3. berikut ini.



Gambar 3.3. Rangkaian *driver* baris

Transistor PNP yang digunakan pada rangkaian *driver* baris ini adalah transistor FCS 9012. Sifat dari transistor PNP adalah bila pada basisnya diberikan catu negatif atau logika rendah, maka antara emitor dan kolektornya akan terhubung, sehingga mengalirlah arus dari emitor ke kolektor, dan sebaliknya bila diberi logika tinggi, maka antara emitor dan kolektornya akan terputus.

3.1.3. Rangkaian Demultiplexer Dengan Keluaran Sebanyak 50 Jalur.

Komponen utama yang membentuk rangkaian demultiplexer dengan keluaran sebanyak 50 jalur adalah IC demultiplexer dengan keluaran sebanyak 8 jalur, yaitu IC 74LS138. Dengan demikian untuk mendapatkan keluaran sebanyak 50 jalur dibutuhkan IC jenis ini sebanyak 7 buah. 7 buah IC ini akan mendapatkan keluaran sebanyak 7 X 8 = 56, jadi sisanya yang 6 jalur tidak dipergunakan. Seperti yang telah dijelaskan pada bab dua bahwa IC ini memiliki 3 bit pengalamatan, 3 bit jalur masukan, dan 8 bit keluaran. Gambar 3.4 menunjukkan rangkaian demultiplexer tersebut.

Untuk membentuk rangkaian demultiplexer 50 keluaran dengan IC ini, dibutuhkan pula satu buah IC tambahan dari jenis yang sama untuk mengatur pengalamatan IC mana dari ke tujuh buah IC tadi yang akan diaktifkan. Satu buah IC ini akan mengatur masukan dari ketujuh buah IC tadi. Dari yang sudah dijelaskan bahwa keluaran dari IC ini adalah berada pada aktif rendah, dengan demikian untuk mengaktifkan salah satu dari ketujuh buah IC tadi, diberikan logika rendah pada masukannya. Sementara dua buah masukan yang lain langsung dihubungkan dengan kebutuhan masukan. Dari konfigurasi rangkaian IC demultiplexer ini untuk mengatur keluarannya mulai dari jalur pertama hingga jalur ke lima puluh dibutuhkan pengalamatan sebanyak 6 bit, dimana 3 bit bagian bawah untuk pengaturan pengalamatan dari masing-masing IC dan 3 bit bagian atas untuk mengatur IC mana dari ke tujuh IC tadi yang akan diaktifkan.

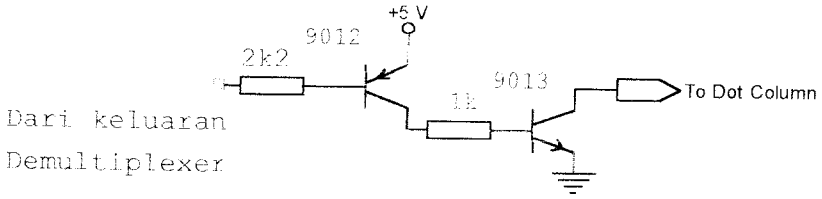
keluarannya yang pertama, dimana aktifnya akan memberikan logika rendah, sementara keluaran yang lain akan berlogika tinggi. Bila nilai pengalamatan dinaikkan menjadi 000 001, maka keluaran yang akan aktif adalah pada posisi berikutnya. Untuk mengganti IC lain yang akan diaktifkan keluarannya, maka pengaturannya ada pada 3 bit bagian atas, misalnya hendak diaktifkan IC yang ketiga dan pada keluarannya yang pertama, maka alamat yang harus diberikan adalah 010 000.

3.1.4. Rangkaian *Driver* Kolom

Sumber catu yang dibutuhkan untuk mensuplai jalur kolom adalah catu negatif. Keluaran dari demultiplexer memang telah berada pada posisi negatif pada saat aktifnya, akan tetapi keluarannya hanya dapat mensuplai arus yang sangat kecil yang mungkin hanya dapat menyalakan satu buah LED, sedangkan masing-masing kolom pada dot matrik terdapat tujuh buah LED yang ada kemungkinan untuk dinyalakan serentak. Oleh karena itu, keluaran dari IC demultiplexer diperlukan penguatan agar dapat mengontrol LED pada jalur kolom.

Komponen yang digunakan sama seperti pada rangkaian *driver* baris, yaitu transistor. Hanya saja pada rangkaian *driver* kolom ini diperlukan dua transistor untuk masing-masing kolom, dengan jenis PNP dan NPN. Transistor yang pertama dari jenis PNP adalah untuk menanggapi keluaran dari demultiplexer yang berada pada aktif rendah, dan transistor berikutnya yang berjenis NPN untuk menanggapi keluaran dari transistor PNP yang berada menjadi aktif tinggi untuk

dapat menyalurkan catu negatif ke jalur kolom pada dot matrik. Gambar 3.5. menunjukkan rangkaian dari *driver* kolom yang digunakan.



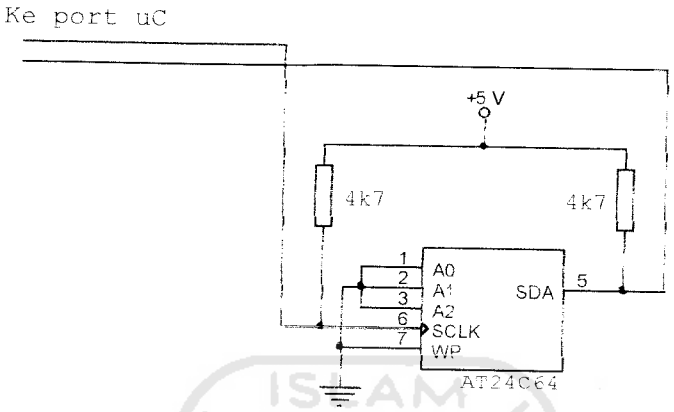
Gambar 3.5. Rangkaian *driver* kolom

Rangkaian *driver* kolom yang digambarkan diatas adalah *driver* kolom untuk satu jalur kolom, dengan demikian bila ada 50 buah jalur kolom, dibutuhkan 50 buah rangkaian yang sama. Seperti yang juga telah dijelaskan pada rangkaian *driver* baris, transistor rangkaian *driver* kolom ini juga berfungsi sebagai saklar.

3.1.5. Rangkaian Memori SEEPROM AT24C64

Untuk menghemat pemakaian kaki dari port mikrokontroler, IC memori yang digunakan untuk menyimpan data pada alat penampil informasi ini adalah dari jenis SEEPROM atau *Serial Erasable Electrically Programmable Read Only Memory*, dimana untuk menyimpan atau mengambil data dari IC memori ini hanya membutuhkan jalur dua bit dari mikrokontroler, karena prosesnya menggunakan sistem serial. Dua bit ini dipergunakan untuk jalur sinyal data dan sinyal *clock*. Untuk menstabilkan perubahan kondisi logika dari sinyal data dan

sinyal *clock* dari IC SEEPROM ini dibutuhkan resistor *eksternal pull up*. Gambar 3.6. menunjukkan rangkaian memori tersebut.



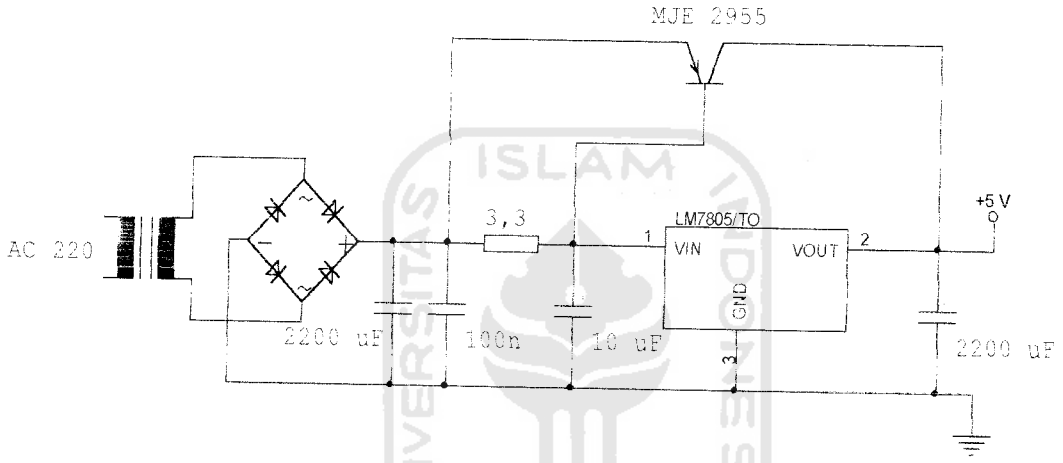
Gambar 3.6. Rangkaian memori SEEPROM AT24C64

Masukan A0, A1, dan A2 adalah untuk mengatur pengalamatan IC, masukan ini digunakan adalah apabila jumlah IC yang dipakai lebih dari satu, sementara untuk jalur data dan jalur *clock*-nya tetap menggunakan jalur yang sama. Karena pada alat ini hanya digunakan 1 buah IC, maka semua kaki pengalamatannya langsung dihubungkan ke titik 0, sehingga untuk mengidentifikasikan IC ini, mikrokontroler harus mengirimkan biner 000 pada saat mikrokontroler hendak mengirim atau membaca data dari SEEPROM.

3.1.6. Rangkaian Catu Daya

Catu daya yang dibutuhkan untuk mensuplai seluruh rangkaian adalah catu daya tunggal dengan keluaran 5 Volt. Untuk menstabilkan tegangan pada keluaran 5 Volt, digunakan IC regulator 7805. Karena arus yang dibutuhkan untuk

mensuplai rangkaian lebih dari 1 Ampere, maka penggunaan IC 7805 (arus maksimal yang dapat diberikan IC 7805 = 1 Ampere) ini harus dibantu dengan menggunakan transistor MJE 2955, agar catu daya dapat mengeluarkan arus lebih dari 1 Ampere. Gambar 3.7 menunjukkan skema dari rangkaian catu daya yang digunakan.



Gambar 3.7. Rangkaian catu daya

Fungsi IC 7805 yang terpasang pada rangkaian catu daya diatas memiliki fungsi yang hampir sama seperti sebuah zener dioda, yaitu membatasi tegangan keluaran pada 5 Volt, akan tetapi tentu saja dengan menggunakan IC ini disipasi yang dapat ditahan oleh IC ini melebihi kemampuan sebuah zener berkali-kali lipat.

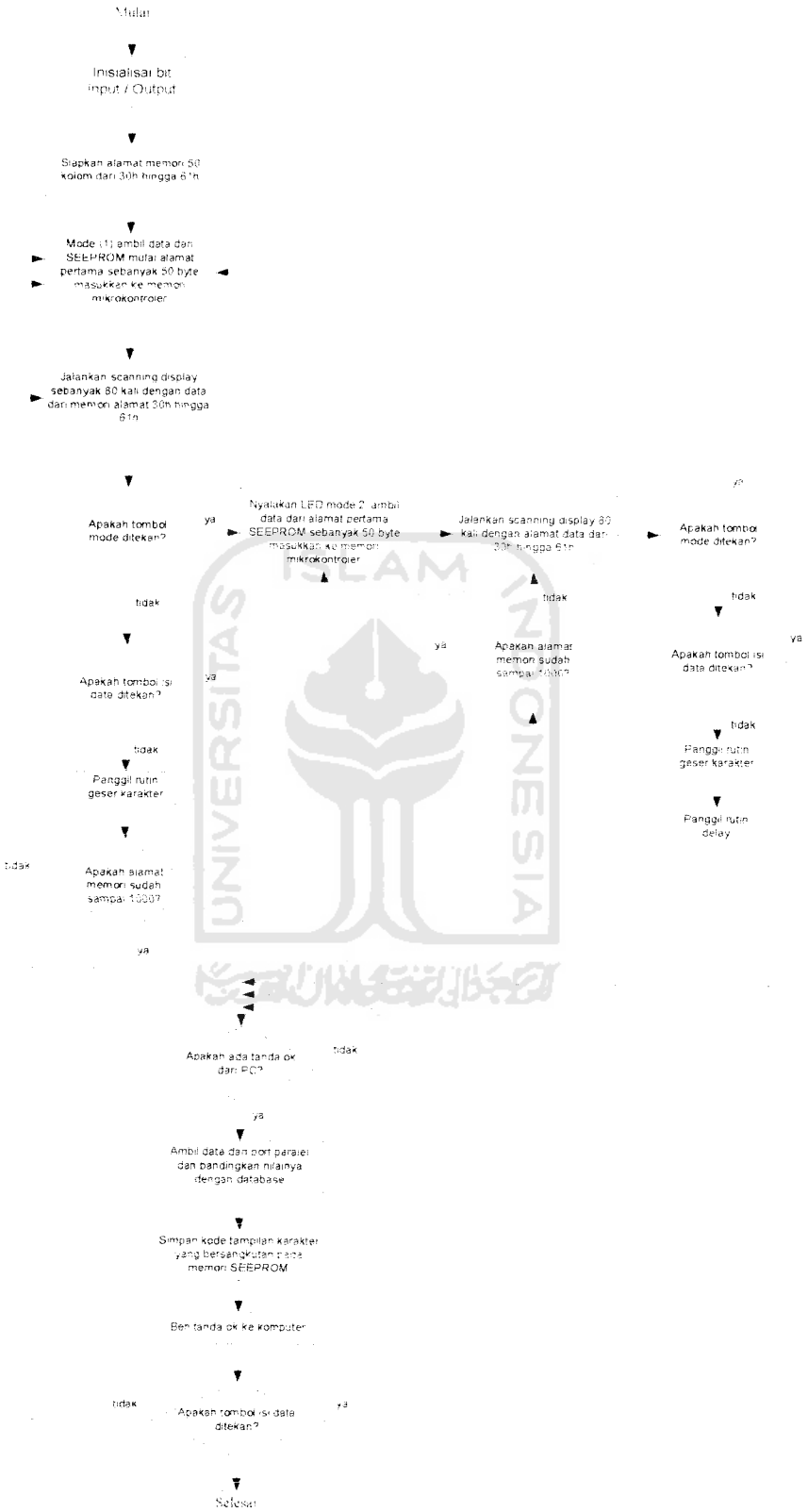
3.2. Perancangan Perangkat Lunak

Seperti yang telah disebutkan pada awal bab ini bahwa perangkat lunak yang dibuat ada dua buah, yaitu perangkat lunak untuk mikrokontroler dan komputer. Perangkat lunak yang dibutuhkan untuk mikrokontroler dibuat dengan menggunakan bahasa pemrograman asembler yang dalam hal ini adalah asembler 51. Sedangkan perangkat lunak yang digunakan untuk transfer karakter dari komputer dibuat dengan menggunakan bahasa pemrograman Visual Basic 6.0.

Untuk memudahkan penjelasan mengenai program yang dibuat, pada masing-masing program akan digambarkan diagram alirnya.

3.2.1. Program Untuk Mikrokontroler

Diagram alir dari program untuk mikrokontroler yang dibuat nampak pada gambar berikut ini.



Gambar 3.8. Diagram alir program pada mikrokontroler

Program dimulai dengan proses inisialisasi bit input dan output dari kaki-kaki port mikrokontroler, juga inisialisasi dari alamat memori yang digunakan. Default awal tampilan mode pada tulisan berjalan adalah mode satu, yaitu tulisan berjalan biasa, tanpa berkedip. Kemudian program dilanjutkan dengan proses pengambilan data kode karakter huruf dari SEEPROM ke memori internal dari mikrokontroler. Kode karakter yang diambil dimulai dari alamat pertama dari SEEPROM. Lokasi memori internal dari mikrokontroler yang digunakan sebagai penampung sementara kode karakter huruf ini pada alamat 30h hingga 61h. Kemudian setelah proses pengambilan selesai, program akan menjalankan proses *scanning* pada *display* dot matrik sebanyak 80 kali. Setelah itu program akan membaca apakah tombol mode dan tombol isi data ditekan, kalau tidak, maka program akan menjalankan rutin penggeseran huruf. Proses penggeseran huruf dilakukan dengan memindahkan isi memori internal dari mikrokontroler pertama ke kedua, kedua ke ketiga, dan seterusnya hingga alamat terakhir. Dan kemudian alamat yang pertama tadi diisikan satu byte data baru dari SEEPROM. Setiap kali penggeseran, mikrokontroler akan memeriksa apakah alamat pengambilan data telah sampai seribu atau belum. Kalau belum, maka mikrokontroler akan langsung melakukan *scanning* kembali dan kalau sudah, maka mikrokontroler akan mengulangi pengambilan data kode huruf dari SEEPROM.

Bila pada suatu saat tombol mode ditekan, maka program akan menjalankan mode dua, dimana proses penampilan tulisan berjalan akan berubah menjadi tulisan berjalan berkedip. Prosesnya hampir sama seperti pada mode satu, hanya saja pada mode dua ini setiap kali selesai satu siklus *scanning* dan pada saat

selesai memanggil rutin penggeseran huruf, diberi jeda waktu sesaat untuk mematikan display dan setelah itu baru dinyalakan kembali, pemberian waktu sesaat ini untuk menampilkan efek berkedip dari tulisan (*blink*). Untuk membedakan kondisi tampilan mode satu dan dua, pada mode satu, LED indikasi mode mati, sedangkan pada mode dua, LED indikasi menyala.

Juga bila pada suatu saat tombol isi data ditekan, maka program akan dialihkan ke proses pengkopian data baru dari komputer ke alat. Setiap kali komputer mengirimkan satu karakter, komputer akan selalu memberitahukan ke mikrokontroler dengan sinyal ok. Sinyal ok dari komputer merupakan logika satu yang diberikan pada pin 16 dari DB25. Dan mikrokontroler setiap kali mendapatkan sinyal ok ini, maka program akan memerintahkan untuk mengkopi data yang ada pada port data dari DB25, yaitu kaki 2-8, kemudian membandingkan dengan database yang ada. Dari hasil perbandingan, kemudian mikrokontroler akan memindahkan kode karakter yang bersangkutan pada databasenya ke memori SEEPROM. Setelah proses tersebut selesai dilakukan, maka mikrokontroler memberitahukan kepada komputer bahwa proses pengkopian selesai dan mikrokontroler siap menerima data berikutnya. Sinyal ok dari mikrokontroler ini diberikan pada pin 10 dari DB25. Proses ini terus berlangsung hingga komputer selesai mengirimkan semua karakter.

Untuk melihat penampilan karakter yang baru saja dikirim dari komputer, dapat dilakukan dengan kembali menekan tombol isi data. Dengan ditekannya tombol isi data ini, maka program akan kembali ke proses *scanning* pada mode

satu dan karena isi memori dari SEEPROM sudah berubah, maka tulisan yang tampilpun sudah berubah.

3.2.2. Program Untuk Komputer

Program untuk mentransfer karakter dari komputer ke alat dibuat dengan menggunakan bahasa pemrograman Visual Basic 6.0. Visual Basic tidak memiliki menu untuk dapat mengakses *hardware* pada komputer secara langsung. Dalam hal ini adalah penanganan port paralel LPT1 yang merupakan port printer. Oleh karena itu, sebagai jembatan penanganan port LPT1 oleh program Visual Basic ini, dibutuhkan modul program *port io.dll* yang dibuat dengan menggunakan bahasa pemrograman Delphi.

Diagram alir dari program Visual Basic untuk mengirim karakter dari komputer ke alat adalah sebagai berikut.

Gambar 3.9. Diagram alir program Visual Basic untuk transfer karakter

Pada saat pertama kali program dijalankan, program akan memanggil modul penanganan port, yaitu *port io.dll*, dimana modul ini digunakan oleh program untuk menjembatani penangananan port LPT1 yang akan digunakan. Kemudian program akan memberikan logika 0 pada pin 16 dari DB25. Logika 0 berarti karakter belum siap dikirim.

Proses selanjutnya adalah program akan memberikan kesempatan pengguna untuk mengetikkan karakter yang akan dikirim ke alat. Sementara pengguna mengetik, program akan menghitung karakter yang dimasukkan, apakah sudah sampai 1000 atau belum, kalau belum, program akan terus menunggu. Dan bila telah sampai 1000, maka program akan menahan karakter yang masuk dan menunggu hingga tombol *send* diklik.

Bila tombol *send* telah diklik, maka program akan mengirimkan tanda ok ke alat dibarengi dengan karakter dari huruf pertama. Program pada komputer tidak akan mengirimkan karakter berikutnya sebelum ada tanda ok dari alat yang menandakan bahwa karakter yang dikirim telah selesai dikopi oleh alat. Jika tanda ok dari alat telah masuk, program akan memeriksa apakah karakter yang dikirim telah sampai 1000 karakter belum. Kalau belum, maka program akan mengirimkan karakter berikutnya juga diikuti dengan tanda ok dan seterusnya. Jika 1000 karakter telah selesai dikirim, maka program akan memberikan informasi bahwa transfer karakter telah selesai.

BAB IV

PENGUJIAN DAN ANALISA

Dalam bab ini akan diuraikan mengenai pengujian dari alat yang dibuat disertai dengan pembahasannya. Pengujian mencakup pengujian perangkat keras dan perangkat lunak. Untuk pengujian perangkat keras, seperti juga tentang penjelasan alat pada bab tiga, juga dilakukan bagian perbagian. Sementara untuk pengujian perangkat lunak, untuk program mikrokontroler dapat diuji bila program telah dimasukkan kedalam mikrokontroler, yang berarti juga pengujian alat secara keseluruhan, sementara untuk pengujian perangkat lunak untuk komputer, diuji dengan menggunakan modul uji yang dibuat dengan menggunakan LED dan saklar.

4.1. Pengujian Perangkat Keras

4.1.1. Pengujian Rangkaian *Driver* Baris

Pengujian *driver* baris dilakukan dengan memberikan logika 0 atau 1 pada masing-masing basis transistor. Transistor yang digunakan adalah transistor jenis PNP, yaitu transistor FCS 9012, karena transistor ini apabila diberi catu negatif atau logika rendah pada basisnya, maka antara emitor dan kolektor akan terhubung, sehingga mengalirlah arus dari emitor ke kolektor. Sebaliknya, apabila diberi logika tinggi, maka antara emitor dan kolektornya akan terputus.

Jika, arus yang mengalir sebesar 10,6 mA, maka nilai R dapat dicari dengan persamaan :

$V_{CC} - V_{CE} - V_R - V_{LED} = 0$

$5V - 0 - I.R - 1,5V = 0$

$3,5V = 10,6 \text{ ma} \times R$

$$R = \frac{3,5}{10,6mA}$$

$= 330$ Ohm

Karena, I_c mendekati I_c, maka arus basis (I_b) pada transistor FCS 9012 dapat diketahui. Nilai $hf_e = 40$ (berdasarkan *datasheet* transistor FCS 9012 dan 9013).

$$hf_e = \frac{Ic}{Ib}$$

$$I_b = \frac{Ic}{hfe}$$

$$= \frac{10,6mA}{40}$$

$= 0,265$ mA x 8 (faktor penguatan)

$= 2,12$ mA

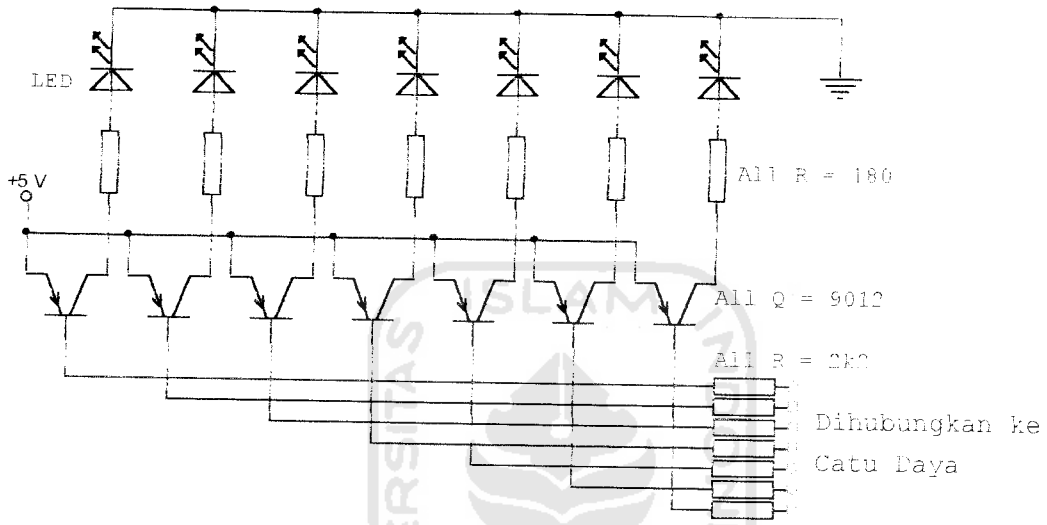
Schingga, $R_b = \frac{V}{Ib}$

$$= \frac{5}{2,12mA}$$

$= 2.358$ Ohm

Untuk membuktikan apakah rangkaian berada dalam kondisi baik atau tidak, maka keluaran dari transistor yang dalam hal ini adalah kaki kolektornya, dipasang LED. Saat basisnya diberi logika 0, LED akan menyala, sedangkan bila diberi logika 1, LED akan mati. Untuk memberikan logika 0, basis pada masing-

masing transistor dihubungkan dengan titik 0 atau negatif dari rangkaian power suplai, sementara untuk memberikan logika 1, basis diberikan sumber tegangan positifnya, yaitu sebesar 5 Volt. Berikut ini adalah skema pengujian dari rangkaian *driver* baris.



Gambar 4.1. Skema pengujian rangkaian *driver* baris

Dari pengujian yang telah dilakukan, diperoleh data pengamatan seperti pada tabel berikut.

Tabel 4.1. Data pengamatan rangkaian *driver* baris

	Logika Masukan	Kondisi LED
Transistor 1	0	Hidup
	1	Mati
Transistor 2	0	Hidup
	1	Mati
Transistor 3	0	Hidup
	1	Mati
Transistor 4	0	Hidup
	1	Mati
Transistor 5	0	Hidup
	1	Mati
Transistor 6	0	Hidup
	1	Mati
Transistor 7	0	Hidup
	1	Mati

Dari tabel diatas, diperoleh data bahwa rangkaian *driver* baris telah bekerja dengan baik seperti yang direncanakan.

4.1.2. Pengujian Rangkaian Demultiplexer Dengan 50 Keluaran

Pengujian rangkaian demultiplexer dilakukan dengan menghubungkan dip-*switch* dan resistor *pull-up* pada masukannya. Resistor *pull-up* sebesar 10k digunakan untuk menstabilkan perubahan kondisi tegangan dan sebagai pembatas arus. Arus keluaran maksimal dari port 0 mikrokontroler sebesar 1,6 mA. Dari hasil pengukuran arus, diperoleh I = 0,5 mA. Jika, tegangannya sebesar 5 Volt, maka nilai R dapat ditentukan dengan rumus : $R = \frac{V}{I}$

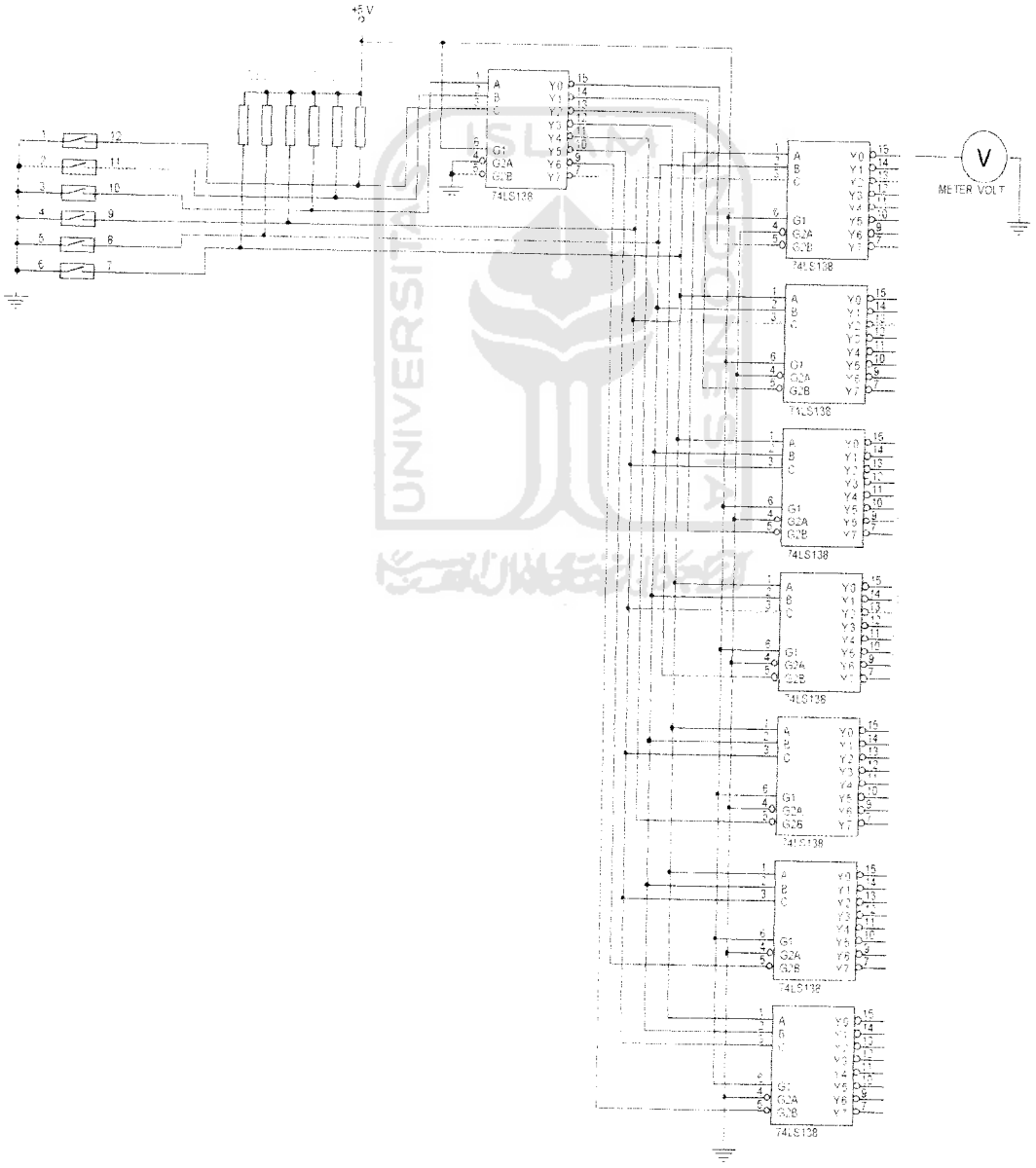
$$= \frac{5}{0{,}5mA}$$

$$= 10.000 \text{ Ohm}$$

Untuk mengatur pemberian nilai biner 6 bit pada masukannya, digunakan dip-*switch*, dimana dip-*switch* ini pada saat ON akan menghubungkan ke titik 0, dan pada saat OFF, resistor *pull-up* akan menghubungkan masukan ke titik 1. Nlai masukan biner 6 bit yang akan diberikan mewakili nilai desimal dari 0 hingga 49, sehingga dengan demikian jika masukan diberikan biner 000 000, maka seharusnya keluaran ke 0 akan berlogika rendah dan yang lainnya akan berlogika tinggi. Dengan dinaikkannya nilai hitungan biner 6 bit tadi setingkat, misalnya dari 000 000 menjadi 000 001, maka akan menggeser logika rendah tadi ke keluaran berikutnya dan keluaran sebelumnya akan kembali menjadi berlogika tinggi.

Untuk memeriksa keadaan logika keluaran, pada pengujian ini dilakukan dengan mengukur tegangan keluarannya menggunakan Volt meter dari multitester. Untuk keadaan tegangan 0 hingga 0,4 Volt mewakili nilai biner 0, sedangkan untuk tegangan 4,5 hingga 5,5 Volt mewakili nilai biner 1.

Gambar skema pengujian dari rangkaian tersebut tampak seperti pada Gambar 4.2. berikut ini.



Gambar 4.2. Skema pengujian rangkaian demultiplexer dengan 50 keluaran

Dari pengujian yang telah dilakukan terhadap rangkaian demultiplexer ini, diperoleh data pengamatan seperti pada Tabel 4.2.

Tabel 4.2. Data pengamatan rangkaian demultiplexer 50 keluaran

Masukan		Kondisi yang berlogika 0	
000	000	Q0	Yang lain 1 (high)
000	001	Q1	Yang lain 1 (high)
000	010	Q2	Yang lain 1 (high)
000	011	Q3	Yang lain 1 (high)
000	100	Q4	Yang lain 1 (high)
000	101	Q5	Yang lain 1 (high)
000	110	Q6	Yang lain 1 (high)
000	111	Q7	Yang lain 1 (high)
001	000	Q8	Yang lain 1 (high)
001	001	Q9	Yang lain 1 (high)
001	010	Q10	Yang lain 1 (high)
001	011	Q11	Yang lain 1 (high)
001	100	Q12	Yang lain 1 (high)
001	101	Q13	Yang lain 1 (high)
001	110	Q14	Yang lain 1 (high)
001	111	Q15	Yang lain 1 (high)
010	000	Q16	Yang lain 1 (high)
010	001	Q17	Yang lain 1 (high)
010	010	Q18	Yang lain 1 (high)
010	011	Q19	Yang lain 1 (high)
010	100	Q20	Yang lain 1 (high)
010	101	Q21	Yang lain 1 (high)
010	110	Q22	Yang lain 1 (high)
010	111	Q23	Yang lain 1 (high)
011	000	Q24	Yang lain 1 (high)
011	001	Q25	Yang lain 1 (high)
011	010	Q26	Yang lain 1 (high)
011	011	Q27	Yang lain 1 (high)
011	100	Q28	Yang lain 1 (high)
011	101	Q29	Yang lain 1 (high)
011	110	Q30	Yang lain 1 (high)
011	111	Q31	Yang lain 1 (high)
100	000	Q32	Yang lain 1 (high)
100	001	Q33	Yang lain 1 (high)
100	010	Q34	Yang lain 1 (high)
100	011	Q35	Yang lain 1 (high)
100	100	Q36	Yang lain 1 (high)
100	101	Q37	Yang lain 1 (high)
100	110	Q38	Yang lain 1 (high)
100	111	Q39	Yang lain 1 (high)

Tabel 4.2. Lanjutan

Masukan		Kondisi yang berlogika 0	
101	000	Q40	Yang lain 1 (high)
101	001	Q41	Yang lain 1 (high)
101	010	Q42	Yang lain 1 (high)
101	011	Q43	Yang lain 1 (high)
101	100	Q44	Yang lain 1 (high)
101	101	Q45	Yang lain 1 (high)
101	110	Q46	Yang lain 1 (high)
101	111	Q47	Yang lain 1 (high)
110	000	Q48	Yang lain 1 (high)
110	001	Q49	Yang lain 1 (high)

Dari tabel pegamatan diatas, dapat disimpulkan bahwa rangkaian demultiplexer dengan 50 keluaran telah bekerja dengan baik dan sesuai dengan yang direncanakan.

4.1.3. Rangkaian *Driver* Kolom

Pengujian rangkaian *driver* kolom hampir sama seperti pengujian rangkaian *driver* baris, hanya saja karena keluarannya banyak, maka pengujiannya tidak dilakukan dengan memasang keluarannya pada LED, akan tetapi diukur keluarannya dengan menggunakan Voltmeter pada multitester. Setelah dilakukan pengukuran arus pada LED, diperoleh I_c = 10,6 mA. Jika, diketahui hf_e = 40 (berdasarkan *datasheet* transistor FCS 9012 dan 9013), maka R dapat dicari dengan rumus :

$V_{CC} - 0,7 - I.R - 0,7 = 0$

$5V - 0,7 - 10,6\ mA.R - 0,7 = 0$

$$R = \frac{3,6}{10,6mA} = 340\ \text{Ohm}$$

Karena, I_c mendekati I_c, maka arus basis (I_b) dapat diketahui.

$$I_b = \frac{Ic}{hfc}$$

$$= \frac{10{,}6mA}{40}$$

= 0,265 mA x 8 (faktor penguatan)

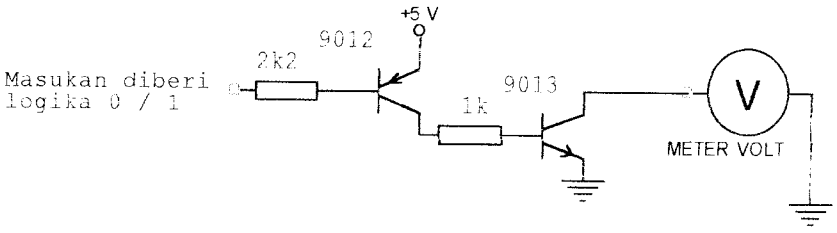
= 2,12 mA

Sehingga, $R_b = \frac{V}{Ib}$

$$= \frac{5}{2{,}12mA}$$

= 2.358 Ohm

Untuk menyalakan LED pada dot matrik, keluaran dari *driver* baris harus berlogika *high*, sedangkan keluaran *driver* kolom berlogika *low*. Karena keluaran dari demultiplexer 74LS138 adalah *low*, maka *output*-nya dihubungkan ke basis transistor PNP untuk mensuplai arus positif yang akan mengaktifkan transistor berikutnya, yaitu transistor NPN. Transistor NPN jika diberi logika tinggi pada basisnya, maka antara kolektor dan emitornya akan terhubung, sehingga keluaran dari *driver* kolom akan berlogika *low*. Rangkaian dapat dikatakan bekerja dengan baik, bila pada saat masukannya diberi logika 0, keluarannya juga akan berlogika 0. Gambar 4.4. menunjukkan skema pengujian rangkaian *driver* kolom.



Gambar 4.3. Skema pengujian rangkaian *driver* kolom

Dari pengujian yang telah dilakukan terhadap rangkaian *driver* kolom ini, didapat data pengamatan sebagai berikut.

Tabel 4.3. Data pengamatan pada rangkaian *driver* kolom

Masukan	Keluaran
0	0
1	-

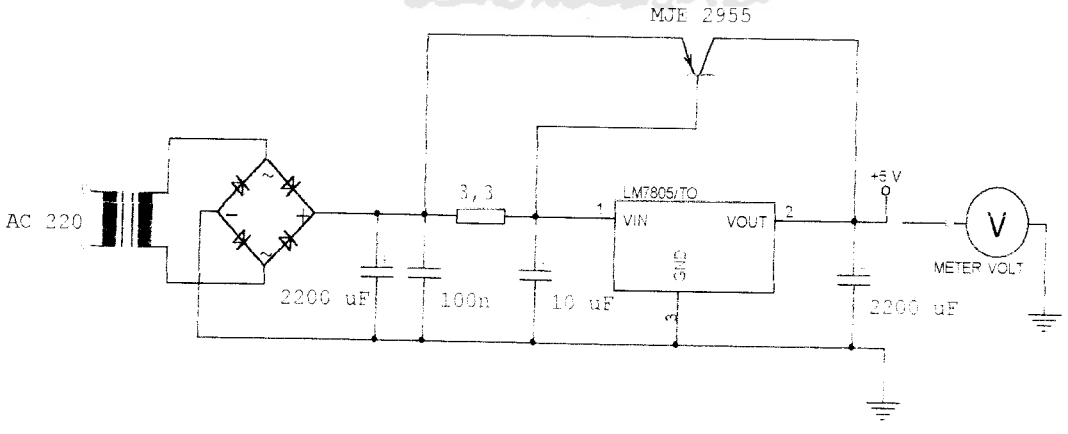
Driver kolom yang digunakan pada alat ini semuanya berjumlah 50 buah, sesuai dengan banyaknya kolom yang ada. Tabel pengamatan diatas merupakan perwakilan dari pengujian seluruh rangkaian *driver* kolom, karena hasil pengujian untuk masing-masing *driver* menunjukkan hasil yang relatif sama antara yang satu dengan yang lainnya.

Dari pengujian yang telah dilakukan terhadap seluruh rangkaian *driver* kolom yang ada, didapat kesimpulan bahwa rangkaian *driver* kolom yang dibuat telah bekerja dengan baik.

4.1.4. Pengujian Rangkaian Catu Daya

Pengujian rangkaian catu daya dilakukan dengan mengukur tegangan keluaran dari rangkaian catu daya tersebut. Rangkaian catu daya dapat dikatakan

bekerja dengan baik apabila tegangan yang dikeluarkan sesuai dengan kebutuhan rangkaian alat, yaitu sebesar 5 Volt. Tegangan dari PLN sebesar 220 Volt dihubungkan ke transformator stepdown 3 Ampere untuk menurunkan tegangan menjadi 9 Volt. Keluaran dari transformator dihubungkan ke dioda bridge. Fungsi dioda bridge untuk menyearahkan arus dan tegangan. Keluaran dari dioda bridge dihubungkan ke IC regulator 7805. IC ini berfungsi untuk menurunkan tegangan 9 Volt menjadi 5 Volt dan juga untuk menstabilkan tegangan. Karena arus yang dibutuhkan untuk mensuplai rangkaian lebih dari 1 Ampere, sedangkan IC 7805 hanya mampu mensuplai arus maksimal sampai 1 Ampere, maka dibutuhkan transistor MJE 2955 agar dapat mensuplai arus lebih dari 1 Ampere. Resistor 3,3 Ohm digunakan untuk membatasi arus yang masuk ke IC regulator 7805 agar kenaikan arus pada beban tidak merusak IC. Kapasitor digunakan untuk menstabilkan tegangan dan mengurangi riak. Gambar 4.5. menunjukan titik pengukuran dari rangkaian catu daya.



Gambar 4.4. Titik pengukuran rangkaian catu daya

Dari hasil pengukuran yang dilakukan terhadap rangkaian catu daya ini, diperoleh hasil pengukuran sebesar 5,1 Volt. Dengan demikian dapat dapat disimpulkan bahwa rangkaian catu daya yang dibuat telah bekerja dengan baik.

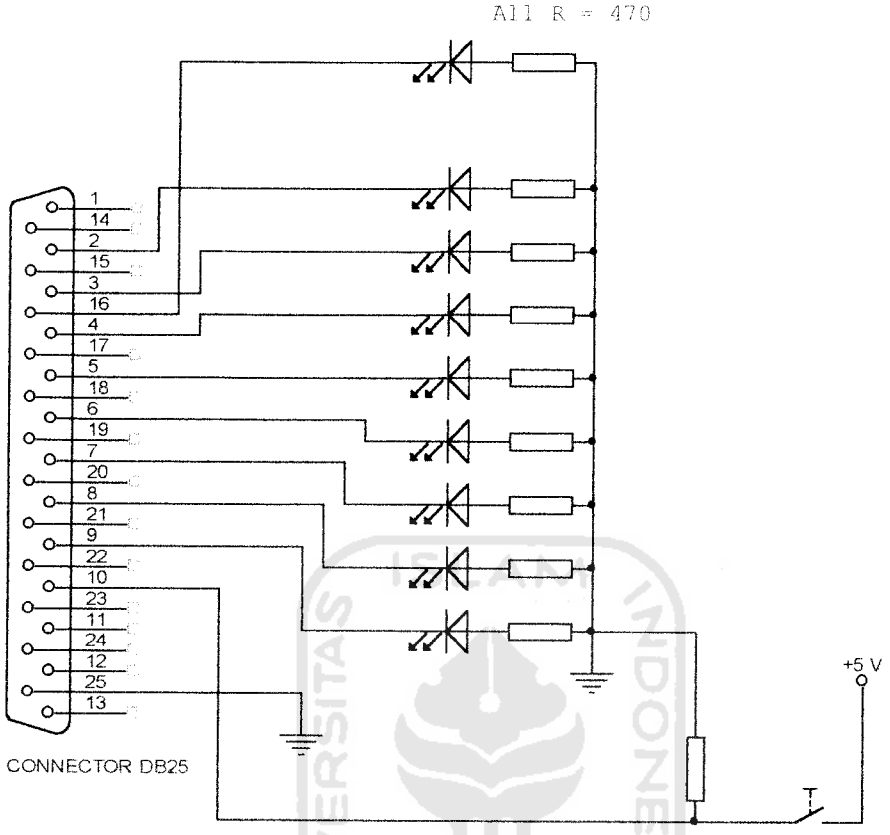
4.2. Pengujian Perangkat Lunak

Pengujian perangkat lunak yang digunakan untuk mengirim karakter dari komputer ke alat dilakukan dengan menggunakan alat bantu *hardware* yang berupa lampu-lampu LED untuk indikasi data dan sinyal ok yang diberikan oleh komputer, sementara untuk sinyal ok yang diberikan oleh alat, dibuat simulasinya dengan menggunakan saklar yang akan menghubungkan jalur ke titik + 5 Volt, yang akan mengindikasikan pemberian logika 1.

Sementara pengujian perangkat lunak yang digunakan untuk mikrokontroler dapat dilakukan setelah program tersebut dimasukkan kedalam mikrokontroler, yang dalam hal ini dapat juga dikatakan sebagai pengujian alat secara keseluruhan.

4.2.1. Pengujian Perangkat Lunak Untuk Pengiriman Karakter

Rangkaian *hardware* yang digunakan untuk menguji perangkat lunak pengiriman karakter terlihat pada gambar dibawah ini.



Gambar 4.5. Rangkaian *hardware* yang digunakan untuk menguji perangkat lunak pengiriman karakter dari komputer

LED yang terpasang dari pin 2 - 9 pada konektor DB25 merupakan LED yang mengindikasikan nilai biner karakter kode ASCII yang dikirim oleh komputer, sementara LED yang terpasang pada pin 16 merupakan indikasi sinyal ok yang dikirim oleh komputer bersamaan dengan dikirimnya kode karakter 8 bit tadi. Pada saat tombol ok belum diklik (pengiriman karakter belum dimulai), LED yang menandakan sinyal ok dari komputer berada pada kondisi mati.

Sinyal ok sebagai indikasi ke komputer bahwa alat telah melakukan pengkopian data diberikan pada pin 10. Tanda ok dari alat juga diberikan dalam logika biner 1.

Arus yang mengalir sebesar 10,6 mA, maka nilai R dapat dicari dengan persamaan :

$V_{CC} - V_R - V_{LED} = 0$

$5V - I.R - 1,5V = 0$

$3,5V = 10,6 \text{ ma x } R$

$$R = \frac{3,5}{10,6mA}$$

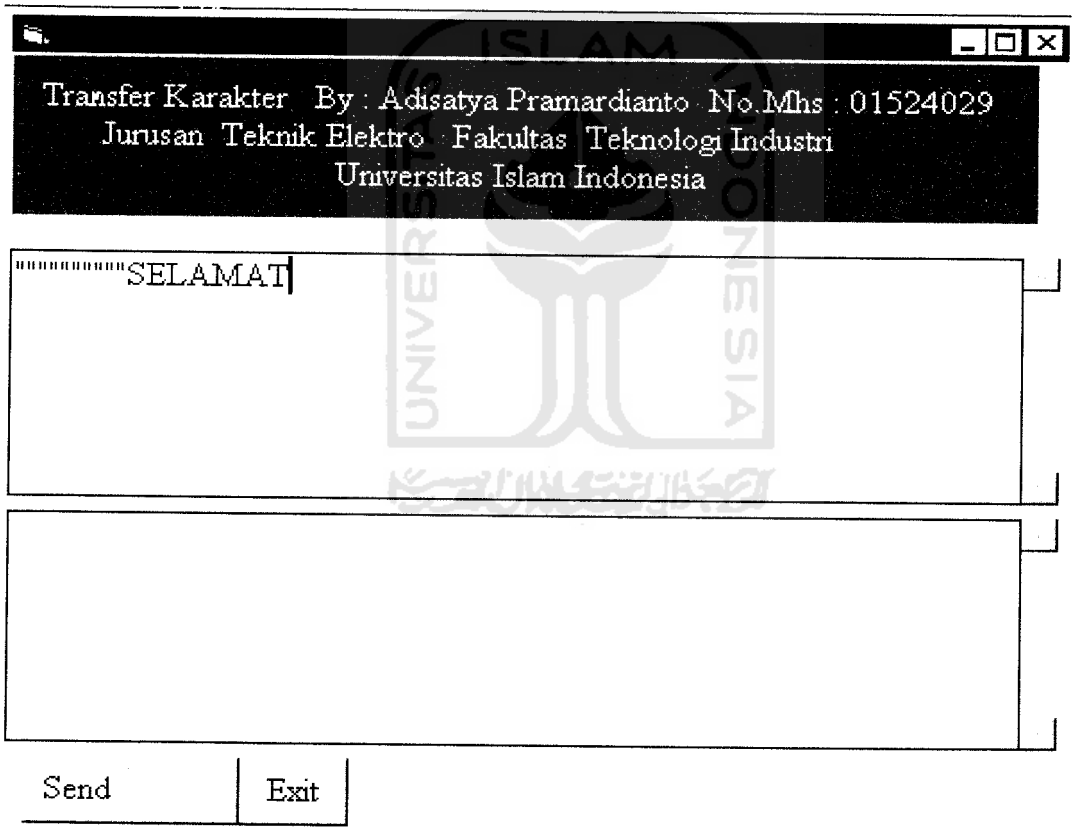
$= 330$ Ohm

4.3. Pengujian Alat Keseluruhan

Seperti yang telah dituliskan sebelumnya bahwa pengujian perangkat lunak untuk mikrokontroler dapat juga dikatakan menguji alat secara keseluruhan, karena perangkat lunak yang dibuat akan terlihat kinerjanya setelah alat dijalankan.

Langkah-langkah pengujian alat secara keseluruhan adalah, pertama semua bagian *hardware* dipasang dan diperiksa konektifitasnya apakah telah terhubung dengan benar atau belum. Bila telah terhubung dengan benar, maka alat diberikan catu daya, dan koneksi DB25 dari alat dihubungkan ke komputer.

Pada awal pertama kali alat dihidupkan, pada dot matrik tidak terlihat ada yang menyala, karena memang IC SEEPROM saat pertama tidak berisi data. Kemudian karakter diketik pada komputer, diawali dengan tanda petik 10x (“”””””””””””””””””””). Setelah selesai, kemudian tombol isi data yang tersedia pada alat ditekan. Dengan ditekannya tombol isi data ini, LED indikasi untuk proses pengisian karakter akan menyala dan ini menandakan alat siap menerima karakter. Lalu tombol *send* pada layar komputer di klik.



Gambar 4.6. Setelah karakter ditulis, maka tombol *Send* harus diklik

Dengan di kliknya tombol *send* tadi, pada layar komputer akan tampak pengkopian karakter demi karakter dari layar bagian atas ke bagian bawah. Dan

BAB V

PENUTUP

5.1 Kesimpulan

Dari keseluruhan kerja yang telah dilakukan, penulis mengambil beberapa kesimpulan sebagai berikut :

1. Alat yang dibuat telah dapat bekerja dengan baik, yaitu menampilkan tulisan bergeser maksimal sampai 1000 karakter.
2. Memori SEEPROM sangat baik digunakan sebagai media penyimpanan, karena hanya membutuhkan koneksi jalur yang sangat sedikit dengan mikrokontroler.
3. IC *demultiplexer* memiliki kelebihan dibandingkan *shift register*, karena kolom yang akan dinyalakan dapat dialamati.

5.2 Saran

Dari hasil perancangan sistem penampil informasi dot matrik ini, para peneliti berikutnya dapat melakukan pengembangan agar diperoleh alat yang lebih baik, terutama dalam hal :

1. Mengganti koneksi port paralel ke komputer dengan port USB.
2. Menambah animasi terprogram pada mikrokontroler.
3. Memberikan akses penerimaan karakter yang luas, yaitu mencakup seluruh karakter ASCII.
4. Menambah jumlah dot matrik yang dipasang.

DAFTAR PUSTAKA

Ibrahim, K F. 1998. *Teknik Digital*. Alih Bahasa : Ir. P. Insap Santoso, MSc. Yogyakarta : Andi.

Irawan Hsr, dkk, Agus. 1991. *Pintar Elektronika Jilid 1*. Pekalongan : C.V. Bahagia.

MADCOMS. 2005. *Mahir Dalam 7 Hari Pemrograman Visual Basic 6.0*. Yogyakarta : Andi.

Malvino, Albert Paul. 2003. *Prinsip-prinsip Elektronika Jilid 1*. Alih Bahasa : Prof. M. Barmawi, Ph.D. Jakarta : Salemba Teknik.

Millman, Halkias. 1972. *Integrated Electronics*. International Student Edition. McGraw-Hill & Kasugawa LTD.

Millman, Jacob. 1984. *Mikroelektronika Sistem Digital dan Rangkaian Analog*. Alih bahasa : Sutanto. Jakarta : Universitas Indonesia.

Nalwan, Paulus Andi. 2003. *Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51*. Jakarta : Elek Media Komputindo.

Prasetya, Retna. 2004. *Interfacing Port Paralel dan Port Serial Komputer Dengan Visual Basic 6.0*. Yogyakarta : Andi.

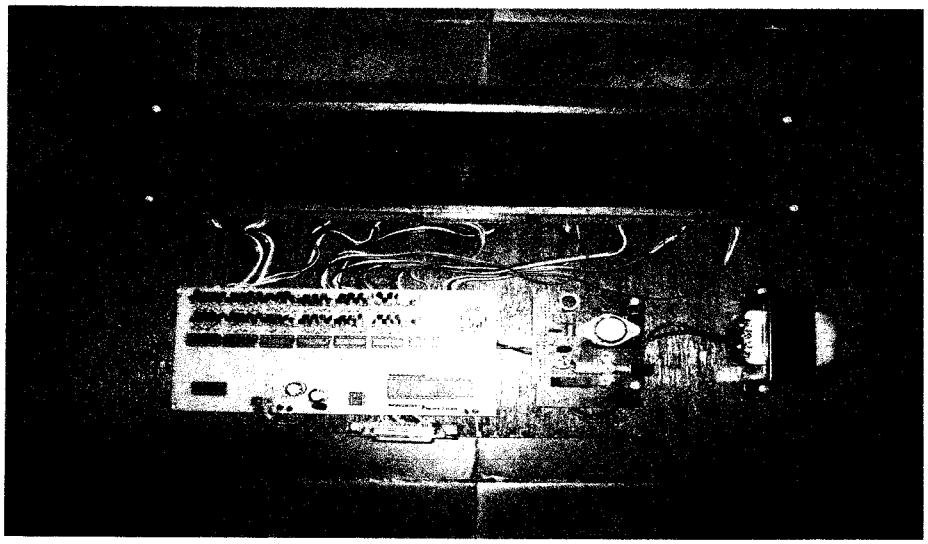
Putra, Agfianto Eko. 2002. *Belajar Mikrokontroler AT89C51 52 53*. Yogyakarta : C.V. Gava Media.

Woollard, Barry G. 2003. *Elektronika Praktis*. Alih Bahasa : H. Kristono. Jakarta : Pradnya Paramita.

hal ini mengindikasikan bahwa proses pengiriman data per karakter sedang berlangsung. Setelah selesainya pengiriman data karakter, program pada komputer akan menginformasikan bahwa proses pengiriman data telah selesai. Dengan informasi ini, maka pengguna diperkenankan untuk menekan tombol isi data pada alat untuk mengembalikan posisi alat ke penampilan tulisan bergeser.

Dengan ditekannya tombol isi data, LED indikasi proses pengisian data kembali mati. Dan kini pada tampilan dot matrik akan tampak tulisan yang tadi ditulis pada layar komputer. Tulisan tersebut akan bergeser dari kanan ke kiri. Langkah terakhir proses pengujian adalah menekan tombol mode. Dengan ditekannya tombol mode, LED indikasi mode kedua menyala, tampilan tulisan pada dot matrik kini bergeser dan berkedip.

Dari uraian mengenai pengujian alat yang telah dilakukan, disimpulkan bahwa alat yang dibuat telah dapat bekerja sebagaimana yang telah direncanakan pada awal penelitian.



Gambar 4.7. Foto alat penampil informasi dot matrik

BAB V

PENUTUP

5.1 Kesimpulan

Dari keseluruhan kerja yang telah dilakukan, penulis mengambil beberapa kesimpulan sebagai berikut :

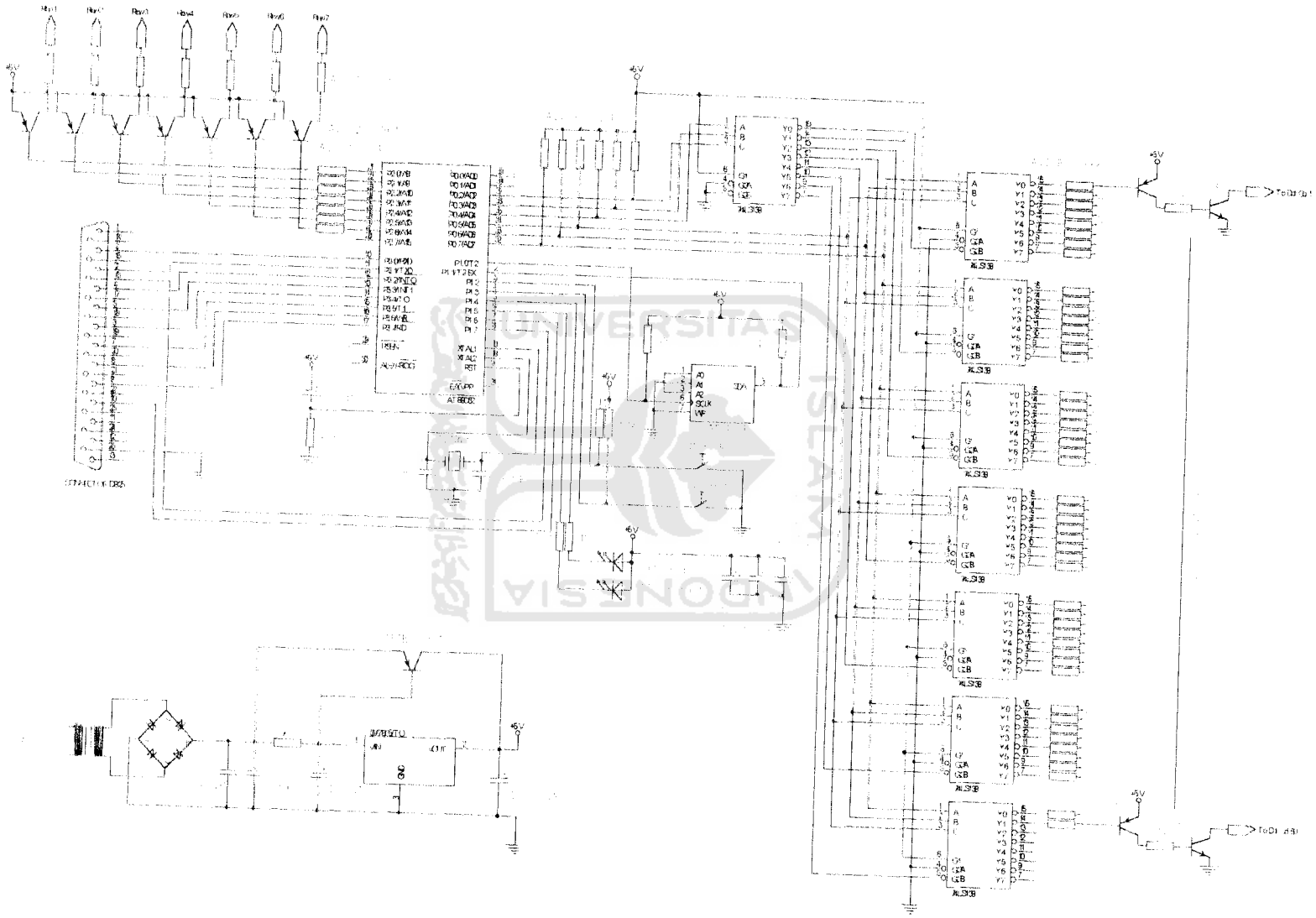
1. Alat yang dibuat telah dapat bekerja dengan baik, yaitu menampilkan tulisan bergeser maksimal sampai 1000 karakter.
2. Memori SEEPROM sangat baik digunakan sebagai media penyimpanan, karena hanya membutuhkan koneksi jalur yang sangat sedikit dengan mikrokontroler.
3. IC *demultiplexer* memiliki kelebihan dibandingkan *shift register*, karena kolom yang akan dinyalakan dapat dialamati.

5.2 Saran

Dari hasil perancangan sistem penampil informasi dot matrik ini, para peneliti berikutnya dapat melakukan pengembangan agar diperoleh alat yang lebih baik, terutama dalam hal :

1. Mengganti koneksi port paralel ke komputer dengan port USB.
2. Menambah animasi terprogram pada mikrokontroler.
3. Memberikan akses penerimaan karakter yang luas, yaitu mencakup seluruh karakter ASCII.
4. Menambah jumlah dot matrik yang dipasang.

LAMPIRAN



```
;*****************************************************************
;Program tulisan berjalan dengan kapasitas memori 1000 karakter
;oleh Adisatya Pramardianto, No.Mahasiswa:01524029
;Jurusan Teknik Elektro, Fakultas Teknologi Industri
;Universitas Islam Indonesia, 2007
;*****************************************************************
tampung1                Equ   70h
Dummyaddr_down          Equ   R3
Dummyaddr_up            Equ   R2
Repeat                  Equ   23h
FixW                    Equ   0A0h
FixR                    Equ   0A1h
TimeDELAY               Equ   -1000
SCL                     bit   P1.0
SDA                     bit   P1.1
tombol_mode             bit   P1.2
tombol_isi_data         bit   P1.3
led_mode                bit   P1.4
led_isi_data            bit   P1.5
bit_ok_fhw              bit   P1.6
bit_ok_fpc              bit   P1.7
Kode_mode               Equ   75h
;-------------------------------------
;data dari pc dimasukkan ke port 3
;data byte scanning pada port 2
;data byte kolom pada port 0
;-------------------------------------
            org   0h
            clr   bit_ok_fhw          ;buat low bit ack ke pc
            mov   R1, #00
;-------------------------------------
;Mode 1 Scanning
;-------------------------------------
mode1:      setb  led_mode
            mov   kode_mode, #50
            call  delay
modesatu:   call  ambildata
scan:
            mov   R4, #35             ;Diulang sebanyak 35 kali
scan1:
            djnz  R4, scan2
            mov   P2, #0FFh
            sjmp  geserhuruf
scan2:
            jnb   tombol_mode, mode2
            jnb   tombol_isi_data, isidata
            mov   R0, #30h
            mov   P0, #0ffh
scan3:
            mov   P2, @R0
            inc   P0
            call  tunggu
            inc   R0
            cjne  R0, #60h, scan3
            sjmp  scan1
```

```
;--------------------------------
;penggeseran huruf ke kiri model
;--------------------------------

geserhuruf:
            inc   dptr
            mov   dummyaddr_up, dph
            mov   dummyaddr_down, dpl
            cjne  dummyaddr_up, #13h, lagi6
            cjne  dummyaddr_down, #0BDh, lagi6
            jmp   modesatu

lagi6:
            call  rdeeprom
            jmp   scan
;-------------------
;rutin-rutin delay
;-------------------
Tunggu:
            MOV   R6, #100
            DJNZ  R6, $                ;Menunggu disini selama 100 uS
            RET

delay:      mov   R7, #4
delay1:     mov   R6, #0FFh
delay2:     mov   R5, #0
            djnz  R5, $
            djnz  R6, delay2
            djnz  R7, delay1
            ret

;-----------------------------------------
;menjalankan program pengisian data dari pc
;-----------------------------------------
isidata:    mov   P2, #0ffh
            clr   led_isi_data
            call  delay
            ljmp  isidatasee
;---------------------------------------
;Mode 2 scanning
;---------------------------------------
mode2:      clr   led_mode
            call  delay
            mov   kode_mode, #100
modedua:    call  ambildata
scanb:
            mov   R4, #80              ;Diulang sebanyak 80 kali
scan1b:
            djnz  R4, scan2b
            mov   P2, #0FFh
            call  delay
            sjmp  geserhuruf_b
scan2b:
            jnb   tombol_mode, model
            jnb   tombol_isi_data, isidata
            mov   R0, #30h
```

```
                mov     P0, #0ffh
scan3b:
                mov     P2, @R0
                inc     P0
                call    tunggu
                inc     R0
                cjne    R0, #62h, scan3b
                sjmp    scan1b


;---------------------------------
;penggeseran huruf ke kiri mode2
;---------------------------------

geserhuruf_b:
                inc     dptr
                mov     dummyaddr_up, dph
                mov     dummyaddr_down, dpl
                cjne    dummyaddr_up, #13h, lagi6b
                cjne    dummyaddr_down, #0BDh, lagi6b
                jmp     modedua
lagi6b:
                call    rdeeprom
                jmp     scanb

;-------------------------------------
;rutin pengambilan data dari seeprom
;-------------------------------------

ambildata:
                clr     A
                mov     dpl, #01h
                mov     dph, A

rdeeprom:
                mov     R0, #30h
                Mov     Repeat,#10
RdLagi:         Call    Start
                Jnc     Dummy
                Call    DLY1ms
                Ajmp    RdLagi
Dummy:          Mov     A,#FixW
                Call    Out8Bit
                Jc      RERR
                Mov     A,dph                   ;Alamat awal pembacaan memory
                Call    Out8Bit
                Jc      RERR
                mov     A, dpl
                call    Out8bit
                jc      rerr
Real:           Call    Start
                Jnc     Real1
                Call    DLY1ms
                Sjmp    Real
Real1:          Mov     A,#FixR
                Call    Out8Bit
```

```
                JC      RERR
RdSUKSES:       Call    In8Bit
                cjne    A, #0FFh, sip
                inc     R1
                cjne    R1, #65, sip2
                mov     R1, kode_mode
                cjne    R1, #50, kemode
                mov     R1, #00h
                ljmp    modesatu
kemode:         mov     R1, #00h
                ljmp    modedua
sip:            mov     R1, #00h
sip2:           Mov     @R0,A           ; Data yang sudah dapat disalin ke
                inc     R0              ; Ram Mikrokontroler
                cjne    R0, #62h, lagi5
                call    NAK
                jmp     Sukses

lagi5:          call    ACK
                sjmp    Rdsukses

sukses:         call    stop
                clr     C
                sjmp    exitRW
xrwerr:         setb    C
ExitRW:         RET

RERR:           CLR     C
                Call    Stop
                Call    DLY1ms
                DJNZ    Repeat,RdLAGI
                sjmp    XRWERR

;************************************
; Rutin Penulisan data ke SEEPROM
;************************************
isidatasee:
                clr     A
                mov     dummyaddr_down, A
                mov     dummyaddr_up, A
lagi10:         JB      bit_ok_fpc, copydata
                jnb     tombol_isi_data, lagi20
                sjmp    lagi10

lagi20:         setb    led_isi_data
                call    delay
                ljmp    model

copydata:
                mov     A, P3
                call    database
                call    dly1ms
                setb    bit_ok_fhw
ulangi:         jnb     bit_ok_fpc, lanjut
                sjmp    ulangi
lanjut:         call    dly1ms
```

```
                clr   bit_ok_fhw
                call  dly1ms
                sjmp  lagi10

;-----------------------------
;rutin pengisian seeprom
;-----------------------------
isisee:
;               mov   repeat, #10
                mov   tampung1, A
                inc   dummyaddr_down
                cjne  dummyaddr_down, #00h, wrlagi
                inc   dummyaddr_up
wrlagi:
                call  start
                jnc   wrlagi1
                call  dly1ms
                sjmp  wrlagi
wrlagi1:        mov   A, #FixW
                call  out8bit
                jc    Werr
                mov   A, dummyaddr_up
                call  out8bit
                jc    Werr
                mov   A, dummyaddr_down
                call  out8bit
                jc    Werr
                mov   A, tampung1
                call  dly1ms
                call  out8bit
                jc    Werr
                call  stop
                call  stop
                clr   C
                ret
Werr:           clr   C
                call  stop
                call  stop
                call  Dly1ms
                sjmp  wrlagi

;*****************
;Delay Timer 1ms
;*****************
DLY1ms:
                Mov   TH1, #High TimeDelay
                Mov   TL1, #Low TimeDelay
                SETB  TR1
                JNB   TF1,$
                CLR   TR1
                CLR   TF1
                RET
;**************
;Sinyal START
;**************
START:
```

```
        SETB   SDA
        SETB   SCL
;
        JNB    SDA,BusBusy ; kalau SDA=0 I2C Bus tidak siap pakai
        JNB    SCL,BusBusy ; kalau SCL=0 I2C Bus tidak siap pakai
;
        NOP                ; tunggu sebentar
        CLR    SDA
        NOP                ; tunggu agar data benar-benar stabil
        NOP
        NOP
        NOP
        CLR    SCL
        CLR    C           ; C=0 berarti berhasil membuat START
        RET
;
BusBusy:
        SETB   C           ; C=1 berarti gagal membuat START
        RET

;***************
; Sinyal STOP
;***************
STOP:
        CLR    SDA         ; SDA=0 low beberapa saat
        NOP
        NOP
        SETB   SCL         ; SCL=0
        NOP                ; tunggu sebetar
        NOP
        NOP
        NOP
        NOP
        SETB   SDA         ; SCL=1
        nop
        nop
        clr    scl
        RET

;**********************
; Sinyal ACK dan NAK
;**********************
ACK:    CLR    SDA         ; ACK adalah SDA=0
        SJMP   MakeACK
NAK:
        SETB   SDA         ; NAK adalah SDA=1
MakeACK:
        NOP                ; tunggu sebentar
        NOP
        SETB   SCL         ; SCL menjadi 1
        NOP                ; tunggu biar stabil
        NOP
        NOP
        NOP
        CLR    SCL         ; SCL kembali =0
```

```
      RET
;************************
;Kirim Data Ke Seeprom
;************************
Out8bit:
      PUSH  B
      Mov   B,#8          ; akan digeser 8 kali (bit)
OutLoop:
      RLC   A             ; bit A.7 digeser ke C di PSW
      Mov   SDA,C         ; nilai C di SDA
      NOP                 ; tunggu sebentar sebelum.. ..
      SETB  SCL           ; SCL dibuat = '1'
      NOP                 ; tunggu lagi
      NOP
      NOP
      NOP
      CLR   SCL           ; SCL dibuat ='0', data diambil 'slave'
      DJNZ  B,OutLoop     ; ulangi terus sampai 8 kali
;
      SETB  SDA           ; SDA dibuat '1'
      NOP                 ; agar 'slave' bisa mengirim ACK
      NOP
      SETB  SCL           ; clock ke 9 untuk menerima ACK
      NOP                 ; tunggu dulu
      NOP
      NOP
      NOP
      Mov   C,SDA         ; ambil ACK yang dikirim 'slave'
      CLR   SCL           ; SCK kembali ke '0'
      POP   B
      RET
;***************************
;Ambil Data Dari Seeprom
;***************************
In8bit:
      PUSH  B
      SETB  SDA           ; SDA='1' agar 'slave' bisa kirim data
      Mov   B,#8          ; akan digeser 8 kali (bit)
InLoop:
      NOP                 ;  'slave'  boleh  merubah  data  selama
SCK='0'
      NOP
      NOP
      SETB  SCL           ; SCK='1'
      NOP                 ; tunggu sebentar
      NOP
      Mov   C,SDA         ; ambil kiriman bit dari 'slave'
      RLC   A             ; ditampung di A
      CLR   SCL           ;  'slave'  boleh  merubah  data  selama
SCK='0'
      DJNZ  B,InLoop
      POP   B
      RET

;=================================
```

```
;pengolahan database huruf
;=========================================
database:
      mov   R5, #6
;-------------------------------------
;hurufA
      cjne  A, #01000001b, go1
      mov   dptr, #charA
lag1:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem1
      ljmp  kembali
isimem1:
      call  isisee
      jc    isi1
i1:   inc   dptr
      sjmp  lag1
isi1: call  WRlagi
      jc    isi1
      sjmp  i1
;-------------------------------------
;hurufB
go1:
      cjne  A, #01000010b, go2
      mov   dptr, #charB
lag2:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem2
      ljmp  kembali
isimem2:
      call  isisee
      jc    isi2
i2:   inc   dptr
      sjmp  lag2
isi2: call  WRlagi
      jc    isi2
      sjmp  i2
;-------------------------------------
;hurufC
go2:
      cjne  A, #01000011b, go3
      mov   dptr, #charC
lag3:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem3
      ljmp  kembali
isimem3:
      call  isisee
      jc    isi3
i3:   inc   dptr
      sjmp  lag3
isi3: call  WRlagi
      jc    isi3
```

```
        sjmp  i3
;-------------------------------------
;hurufD
go3:
        cjne  A, #01000100b, go4
        mov   dptr, #charD
lag4:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem4
        ljmp  kembali
isimem4:
        call  isisee
        jc    isi4
i4:     inc   dptr
        sjmp  lag4
isi4:   call  WRlagi
        jc    isi4
        sjmp  i4
;-------------------------------------
;hurufE
go4:
        cjne  A, #01000101b, go5
        mov   dptr, #charE
lag5:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem5
        ljmp  kembali
isimem5:
        call  isisee
        jc    isi5
i5:     inc   dptr
        sjmp  lag5
isi5:   call  WRlagi
        jc    isi5
        sjmp  i5
;-------------------------------------
;hurufF
go5:
        cjne  A, #01000110b, go6
        mov   dptr, #charF
lag6:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem6
        ljmp  kembali
isimem6:
        call  isisee
        jc    isi6
i6:     inc   dptr
        sjmp  lag6
isi6:   call  WRlagi
        jc    isi6
        sjmp  i6
;-------------------------------------
```

```
;hurufG
go6:
      cjne  A, #01000111b, go7
      mov   dptr, #charG
lag7:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem7
      ljmp  kembali
isimem7:
      call  isisee
      jc    isi7
i7:   inc   dptr
      sjmp  lag7
isi7: call  WRlagi
      jc    isi7
      sjmp  i7
;-------------------------------------
;hurufH
go7:
      cjne  A, #01001000b, go8
      mov   dptr, #charH
lag8:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem8
      ljmp  kembali
isimem8:
      call  isisee
      jc    isi8
i8:   inc   dptr
      sjmp  lag8
isi8: call  WRlagi
      jc    isi8
      sjmp  i8
;-------------------------------------
;hurufI
go8:
      cjne  A, #01001001b, go9
      mov   dptr, #charI
lag9:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem9
      ljmp  kembali
isimem9:
      call  isisee
      jc    isi9
i9:   inc   dptr
      sjmp  lag9
isi9: call  WRlagi
      jc    isi9
      sjmp  i9
;-------------------------------------
;hurufJ
go9:
```

```
        cjne  A, #01001010b, go10
        mov   dptr, #charJ
lag10:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem10
        ljmp  kembali
isimem10:
        call  isisee
        jc    isi10
i10:    inc   dptr
        sjmp  lag10
isi10:  call  WRlagi
        jc    isi10
        sjmp  i10
;----------------------------------------
;hurufK
go10:
        cjne  A, #01001011b, go11
        mov   dptr, #charK
lag11:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem11
        ljmp  kembali
isimem11:
        call  isisee
        jc    isi11
i11:    inc   dptr
        sjmp  lag11
isi11:  call  WRlagi
        jc    isi11
        sjmp  i11
;----------------------------------------
;hurufL
go11:
        cjne  A, #01001100b, go12
        mov   dptr, #charL
lag12:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem12
        ljmp  kembali
isimem12:
        call  isisee
        jc    isi12
i12:    inc   dptr
        sjmp  lag12
isi12:  call  WRlagi
        jc    isi12
        sjmp  i12
;----------------------------------------
;hurufM
go12:
        cjne  A, #01001101b, go13
        mov   dptr, #charM
```

UNIVERSITAS ISLAM INDONESIA

```
lag13:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem13
      ljmp  kembali
isimem13:
      call  isisee
      jc    isi13
i13:  inc   dptr
      sjmp  lag13
isi13:      call  WRlagi
      jc    isi13
      sjmp  i13
;---------------------------------------
;hurufN
go13:
      cjne  A, #01001110b, go14
lag14:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem14
      ljmp  kembali
isimem14:
      call  isisee
      jc    isi14
i14:  inc   dptr
      sjmp  lag14
isi14:      call  WRlagi
      jc    isi14
      sjmp  i14
;---------------------------------------
;hurufO
go14:
      cjne  A, #01001111b, go15
      mov   dptr, #charO
lag15:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem15
      ljmp  kembali
isimem15:
      call  isisee
      jc    isi15
i15:  inc   dptr
      sjmp  lag15
isi15:      call  WRlagi
      jc    isi15
      sjmp  i15
;---------------------------------------
;hurufP
go15:
      cjne  A, #01010000b, go16
      mov   dptr, #charP
lag16:
      clr   A
      movc  A, @A+dptr
```

```
        djnz   R5, isimem16
        ljmp   kembali
isimem16:
        call   isisee
        jc     isi16
i16:    inc    dptr
        sjmp   lag16
isi16:          call   WRlagi
        jc     isi16
        sjmp   i16
;---------------------------------------
;hurufQ
go16:
        cjne   A, #01010001b, go17
        mov    dptr, #charQ
lag17:
        clr    A
        movc   A, @A+dptr
        djnz   R5, isimem17
        ljmp   kembali
isimem17:
        call   isisee
        jc     isi17
i17:    inc    dptr
        sjmp   lag17
isi17:          call   WRlagi
        jc     isi17
        sjmp   i17
;---------------------------------------
;hurufR
go17:
        cjne   A, #01010010b, go18
        mov    dptr, #charR
lag18:
        clr    A
        movc   A, @A+dptr
        djnz   R5, isimem18
        ljmp   kembali
isimem18:
        call   isisee
        jc     isi18
i18:    inc    dptr
        sjmp   lag18
isi18:          call   WRlagi
        jc     isi18
        sjmp   i18
;---------------------------------------
;hurufS
go18:
        cjne   A, #01010011b, go19
        mov    dptr, #charS
lag19:
        clr    A
        movc   A, @A+dptr
        djnz   R5, isimem19
        ljmp   kembali
```

```
isimem19:
      call  isisee
      jc    isi19
i19:  inc   dptr
      sjmp  lag19
isi19:      call  WRlagi
      jc    isi19
      sjmp  i19
;--------------------------------------
;hurufT
go19:
      cjne  A, #01010100b, go20
      mov   dptr, #charT
lag20:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem20
      ljmp  kembali
isimem20:
      call  isisee
      jc    isi20
i20:  inc   dptr
      sjmp  lag20
isi20:      call  WRlagi
      jc    isi20
      sjmp  i20
;--------------------------------------
;hurufU
go20:
      cjne  A, #01010101b, go21
      mov   dptr, #charU
lag21:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem21
      ljmp  kembali
isimem21:
      call  isisee
      jc    isi21
i21:  inc   dptr
      sjmp  lag21
isi21:      call  WRlagi
      jc    isi21
      sjmp  i21
;--------------------------------------
;hurufV
go21:
      cjne  A, #01010110b, go22
      mov   dptr, #charV
lag22:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem22
      ljmp  kembali
isimem22:
      call  isisee
```

```
        jc      isi22
i22:    inc     dptr
        sjmp    lag22
isi22:          call    WRlagi
        jc      isi22
        sjmp    i22
;---------------------------------------
;hurufW
go22:
        cjne    A, #01010111b, go23
        mov     dptr, #charW
lag23:
        clr     A
        movc    A, @A+dptr
        djnz    R5, isimem23
        ljmp    kembali
isimem23:
        call    isisee
        jc      isi23
i23:    inc     dptr
        sjmp    lag23
isi23:          call    WRlagi
        jc      isi23
        sjmp    i23
;---------------------------------------
;hurufX
go23:
        cjne    A, #01011000b, go24
        mov     dptr, #charX
lag24:
        clr     A
        movc    A, @A+dptr
        djnz    R5, isimem24
        ljmp    kembali
isimem24:
        call    isisee
        jc      isi24
i24:    inc     dptr
        sjmp    lag24
isi24:          call    WRlagi
        jc      isi24
        sjmp    i24
;---------------------------------------
;hurufY
go24:
        cjne    A, #01011001b, go25
        mov     dptr, #charY
lag25:
        clr     A
        movc    A, @A+dptr
        djnz    R5, isimem25
        ljmp    kembali
isimem25:
        call    isisee
        jc      isi25
i25:    inc     dptr
```

```
        sjmp  lag25
isi25:  call  WRlagi
        jc    isi25
        sjmp  i25
;-------------------------------------
;HurufZ
go25:
        cjne  A, #01011010b, go26
        mov   dptr, #charZ
lag26:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem26
        ljmp  kembali
isimem26:
        call  isisee
        jc    isi26
i26:    inc   dptr
        sjmp  lag26
isi26:  call  WRlagi
        jc    isi26
        sjmp  i26
;-------------------------------------
;Angka0
go26:
        cjne  A, #00110000b, go27
        mov   dptr, #char0
lag27:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem27
        ljmp  kembali
isimem27:
        call  isisee
        jc    isi27
i27:    inc   dptr
        sjmp  lag27
isi27:  call  WRlagi
        jc    isi27
        sjmp  i27
;-------------------------------------
;Angka1
go27:
        cjne  A, #00110001b, go28
        mov   dptr, #char1
lag28:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem28
        ljmp  kembali
isimem28:
        call  isisee
        jc    isi28
i28:    inc   dptr
        sjmp  lag28
isi28:  call  WRlagi
```

```
;---------------------------------------
;Angka5
go31:
      cjne  A, #00110101b, go32
      mov   dptr, #char5
lag32:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem32
      ljmp  kembali
isimem32:
      call  isisee
      jc    isi32
i32:  inc   dptr
      sjmp  lag32
isi32:      call  WRlagi
      jc    isi32
      sjmp  i32
;---------------------------------------
;Angka6
go32:
      cjne  A, #00110110b, go33
      mov   dptr, #char6
lag33:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem33
      ljmp  kembali
isimem33:
      call  isisee
      jc    isi33
i33:  inc   dptr
      sjmp  lag33
isi33:      call  WRlagi
      jc    isi33
      sjmp  i33
;---------------------------------------
;Angka7
go33:
      cjne  A, #00110111b, go34
      mov   dptr, #char7
lag34:
      clr   A
      movc  A, @A+dptr
      djnz  R5, isimem34
      ljmp  kembali
isimem34:
      call  isisee
      jc    isi34
i34:  inc   dptr
      sjmp  lag34
isi34:      call  WRlagi
      jc    isi34
      sjmp  i34
;---------------------------------------
;Angka8
```

```
go34:
       cjne  A, #00111000b, go35
       mov   dptr, #char8
lag35:
       clr   A
       movc  A, @A+dptr
       djnz  R5, isimem35
       ljmp  kembali
isimem35:
       call  isisee
       jc    isi35
i35:   inc   dptr
       sjmp  lag35
isi35:       call  WRlagi
       jc    isi35
       sjmp  i35
;-------------------------------------
;Angka9
go35:
       cjne  A, #00111001b, go36
       mov   dptr, #char9
lag36:
       clr   A
       movc  A, @A+dptr
       djnz  R5, isimem36
       ljmp  kembali
isimem36:
       call  isisee
       jc    isi36
i36:   inc   dptr
       sjmp  lag36
isi36:       call  WRlagi
       jc    isi36
       sjmp  i36
;-------------------------------------
;charMIN
go36:
       cjne  A, #00101101b, go37
       mov   dptr, #charMIN
lag37:
       clr   A
       movc  A, @A+dptr
       djnz  R5, isimem37
       ljmp  kembali
isimem37:
       call  isisee
       jc    isi37
i37:   inc   dptr
       sjmp  lag37
isi37:       call  WRlagi
       jc    isi37
       sjmp  i37
;-------------------------------------
;charPLUS
go37:
      cjne  A, #00101011b, go38
```

```
        mov   dptr, #charPLUS
lag38:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem38
        ljmp  kembali
isimem38:
        call  isisee
        jc    isi38
i38:    inc   dptr
        sjmp  lag38
isi38:  call  WRlagi
        jc    isi38
        sjmp  i38
;---------------------------------------
;charGARING
go38:
        cjne  A, #00101111b, go39
        mov   dptr, #charGARING
lag39:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem39
        ljmp  kembali
isimem39:
        call  isisee
        jc    isi39
i39:    inc   dptr
        sjmp  lag39
isi39:  call  WRlagi
        jc    isi39
        sjmp  i39
;---------------------------------------
;charSERU
go39:
        cjne  A, #00100001b, go40
        mov   dptr, #charSERU
lag40:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem40
        ljmp  kembali
isimem40:
        call  isisee
        jc    isi40
i40:    inc   dptr
        sjmp  lag40
isi40:  call  WRlagi
        jc    isi40
        sjmp  i40
;---------------------------------------
;charTANYA
go40:
        cjne  A, #00111111b, go41
        mov   dptr, #charTANYA
lag41:
```

```
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem41
        ljmp  kembali
isimem41:
        call  isisee
        jc    isi41
i41:    inc   dptr
        sjmp  lag41
isi41:        call  WRlagi
        jc    isi41
        sjmp  i41
;---------------------------------------
;charKURBUK
go41:
        cjne  A, #00101000b, go42
        mov   dptr, #charKURBUK
lag42:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem42
        ljmp  kembali
isimem42:
        call  isisee
        jc    isi42
i42:    inc   dptr
        sjmp  lag42
isi42:        call  WRlagi
        jc    isi42
        sjmp  i42
;---------------------------------------
;charKURTUP
go42:
        cjne  A, #00101001b, go43
        mov   dptr, #charKURTUP
lag43:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem43
        ljmp  kembali
isimem43:
        call  isisee
        jc    isi43
i43:    inc   dptr
        sjmp  lag43
isi43:        call  WRlagi
        jc    isi43
        sjmp  i43
;---------------------------------------
;charBAGI
go43:
        cjne  A, #00111010b, go44
        mov   dptr, #charBAGI
lag44:
        clr   A
        movc  A, @A+dptr
```

```
        djnz  R5, isimem44
        ljmp  kembali
isimem44:
        call  isisee
        jc    isi44
i44:    inc   dptr
        sjmp  lag44
isi44:      call  WRlagi
        jc    isi44
        sjmp  i44
;-------------------------------------------
;charPETIK
go44:
        cjne  A, #00100111b, go45
        mov   dptr, #charPETIK
lag45:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem45
        ljmp  kembali
isimem45:
        call  isisee
        jc    isi45
i45:    inc   dptr
        sjmp  lag45
isi45:      call  WRlagi
        jc    isi45
        sjmp  i45
;-------------------------------------------
;charTITIK
go45:
        cjne  A, #00101110b, go46
        mov   dptr, #charTITIK
lag46:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem46
        ljmp  kembali
isimem46:
        call  isisee
        jc    isi46
i46:    inc   dptr
        sjmp  lag46
isi46:      call  WRlagi
        jc    isi46
        sjmp  i46
;-------------------------------------------
;charKOMA
go46:
        cjne  A, #00101100b, go46a
        mov   dptr, #charKOMA
lag47:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem47
        ljmp  kembali
```

```
isimem47:
        call  isisec
        jc    isi47
i47:    inc   dptr
        sjmp  lag47
isi47:  call  WRlagi
        jc    isi47
        sjmp  i47
;-------------------------------------
;charpetikdua
go46a:
        cjne  A, #'"', lag48
        mov   dptr, #charTIKOM
lag47a:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem47a
        ljmp  kembali
isimem47a:
        call  isisee
        jc    isi47a
i47a:   inc   dptr
        sjmp  lag47a
isi47a: call  WRlagi
        jc    isi47a
        sjmp  i47a

;-------------------------------------
;char space
lag48:
        mov   dptr, #charspace
lag48a:
        clr   A
        movc  A, @A+dptr
        djnz  R5, isimem48
        ljmp  kembali
isimem48:
        call  isisee
        jc    isi48
i48:    inc   dptr
        sjmp  lag48a
isi48:  call  WRlagi
        jc    isi48
        sjmp  i48
;-------------------------------------
kembali:    ret

;-------------------------
;database
charA:          DB    81H,0F6H,0F6H,0F6H,81H,00H
charB:          DB    80H,0B6H,0B6H,0B6H,0C9H,00H
charC:          DB    0C1H,0BEH,0BEH,0BEH,0DDH,00H
charD:          DB    80H,0BEH,0BEH,0BEH,0C1H,00H
charE:          DB    80H,0B6H,0B6H,0B6H,0BEH,00H
charF:          DB    80H,0F6H,0F6H,0F6H,0F6H,00H
charG:          DB    0C1H,0B6H,0B6H,0B6H,0C5H,00H
```

```
charH:          DB      80H,0F7H,0F7H,0F7H,80H,00H
charI:          DB      0FFH,0FFH,80H,0FFH,0FFH,00H
charJ:          DB      0CFH,0BFH,0BEH,0BEH,0C0H,00H
charK:          DB      80H,0F7H,0EBH,0DDH,0BEH,00H
charL:          DB      80H,0BFH,0BFH,0BFH,0BFH,00H
charM:          DB      80H,0FDH,0E3H,0FDH,80H,00H
charN:          DB      80H,0FBH,0F7H,0EFH,80H,00H
charO:          DB      0C1H,0BEH,0BFH,0BEH,0C1H,00H
charP:          DB      80H,0F6H,0F6H,0F6H,0F9H,00H
charQ:          DB      0C1H,0BEH,0AEH,9EH,0C1H,00H
charR:          DB      80H,0F6H,0E6H,0D6H,0B9H,00H
charS:          DB      0D9H,0B6H,0B6H,0B6H,0CDH,00H
charT:          DB      0FEH,0FEH,80H,0FEH,0FEH,00H
charU:          DB      0C0H,0BFH,0BFH,0BFH,0C0H,00H
charV:          DB      0E0H,0DFH,0BFH,0DFH,0E0H,00H
charW:          DB      0C0H,0BFH,8FH,0BFH,0C0H,00H
charX:          DB      9CH,0EBH,0F7H,0EBH,9CH,00H
charY:          DB      0FCH,0FBH,87H,0FBH,0FCH,00H
charZ:          DB      9EH,0AEH,0B6H,0BAH,0BCH,00H
char1:          DB      0FFH,0BDH,80H,0BFH,0FFH,00H
char2:          DB      0BDH,9EH,0AEH,0B6H,0B9H,00H
char3:          DB      0B6H,0B6H,0B6H,0B6H,0C9H,00H
char4:          DB      0F0H,0F7H,0F7H,0F7H,80H,00H
char5:          DB      0D8H,0BAH,0BAH,0BAH,0C6H,00H
char6:          DB      0C1H,0B6H,0B6H,0B6H,0CDH,00H
char7:          DB      0BEH,0DEH,0EEH,0F6H,0F8H,00H
char8:          DB      0C9H,0B6H,0B6H,0B6H,0C9H,00H
char9:          DB      0D9H,0B6H,0B6H,0B6H,0C1H,00H
char0:          DB      0C1H,0AEH,0B6H,0BAH,0C1H,00H
charTITIK:      DB      0FFH,0FFH,0BFH,0FFH,0FFH,00H
charKOMA:       DB      0FFH,0BFH,0CFH,0FFH,0FFH,00H
charGARING:     DB      0DFH,0EFH,0F7H,0FBH,0FDH,00H
charTANYA:      DB      0FDH,0FEH,0A6H,0F6H,0F9H,00H
charSERU:       DB      0FFH,0FFH,0A0H,0FFH,0FFH,00H
charKURTUP:     DB      0FFH,0FFH,0BEH,0C1H,0FFH,00H
charKURBUK:     DB      0FFH,0C1H,0BEH,0FFH,0FFH,00H
charBAGI:       DB      0FFH,0FFH,0DBH,0FFH,0FFH,00H
;charTIKOM:     DB      0FFH,0BFH,0CBH,0FFH,0FFH,00H
charTIKOM:      DB      07FH,07FH,07FH,07FH,07FH,00H
charPLUS:       DB      0F7H,0F7H,0C1H,0F7H,0F7H,00H
charMIN:        DB      0F7H,0F7H,0F7H,0F7H,0F7H,00H
charPETIK:      DB      0FFH,0FBH,0F8H,0FCH,0FFH,00H
charspace:      DB      0FFH,0FFH,0FFH,0FFH,0FFH,00H
end
```

```
;*************************************************************
;Program Visual Basic untuk transfer karakter dari PC
;oleh Adisatya Pramardianto,  No.Mahasiswa:01524029
;Jurusan Teknik Elektro,     Fakultas Teknologi Industri
;Universitas Islam Indonesia,   2007
;*************************************************************

Dim J As Integer

Private Sub Command1_Click()
J = 0
    Text1.Text = ""
    Text2.Text = ""
    Text1.Enabled = True
    TblStop.Enabled = False
    Command1.Enabled = False
    Text1.SetFocus
End Sub

Private Sub Form_Load()
    Text1.Text = ""
    Text2.Text = ""
    J = 0
    Text2.Enabled = False
    TblSend.Enabled = False
    Command1.Enabled = False
    Port_Out 890, 8
    Port_Out 888, 255
End Sub



Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)
    If MsgBox("Anda yakin akan keluar", vbYesNo + vbQuestion, "")
= vbYes Then
        Cancel = 0
    Else
        Cancel = 1
    End If
End Sub

Private Sub Label1_Click()

End Sub

Private Sub TblSend_Click()
    TblSend.Enabled = False
    TblStop.Enabled = False
    Text1.Enabled = False
    Text2.Text = Mid(Text1.Text, 1, 1)
    Tunda (100)
    Port_Out 888, Asc(Mid(Text1.Text, 1, 1))
    Port_Out 890, 12
    For J = 0 To 1000
        Do While Port_In(889) = 56
```

```
        Loop
        If J <= Len(Text1.Text) Then
            Port_Out 888, Asc(Mid(Text1.Text, J, 1))
        Else
            Port_Out 888, Asc(" ")
        End If
        Text2.Text = Text2.Text +
        Mid(Text1.Text, J, 1)
        Tunda (100)
        Do While Port_In(889) = 100
            Port_Out 890, 8
        Loop
        Port_Out 890, 12
    Next J
        Port_Out 890, 8
        Command1.Enabled = True
        TblStop.Enabled = True
        J = J + 1
      MsgBox    "Transfer    karakter    selesai",    vbOKOnly    +
vbInformation, ""
End Sub

Private Sub TblStop_Click()
    Unload Me
End Sub


Private Sub Text1_Change()
    If Len(Text1.Text) > 0 Then
        TblSend.Enabled = True
    Else
        TblSend.Enabled = False
    End If
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
Dim X As String
    If ((KeyAscii >= 65) And (KeyAscii <= 90)) Or
        ((KeyAscii >= 97) And (KeyAscii <= 122)) Or
        (KeyAscii = 33) Or (KeyAscii = 34) Or
        ((KeyAscii >= 39) And (KeyAscii <= 41)) Or
        ((KeyAscii >= 43) And (KeyAscii < 47)) Or
        (KeyAscii = 58) Or (KeyAscii = 61) Or
        (KeyAscii = 63) Or (KeyAscii = 8) Or (KeyAscii = 32) Or
        ((KeyAscii >= 48) And (KeyAscii <= 57)) Or (KeyAscii = 47)
Then
            X = Chr(KeyAscii)
            X = UCase(X)
            KeyAscii = Asc(X)
    Else
        KeyAscii = 0
    End If
    If Len(Text1.Text) > 999 Then
    If KeyAscii <> 8 Then
    MsgBox   "Maksimal   huruf   sudah   tercapai   ",   vbOKOnly   +
vbInformation, ""
```

```
        KeyAscii = 0
        End If
        End If

End Sub


Private Sub Timer1_Timer()
  If (J > 1000) And _
  (Port_In(889) = 120) Then
    MsgBox "Transfer karakter selesai", vbOKOnly + vbInformation,
""
    End If
End Sub
```