INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

SKRIPSI

PERANCANGAN DAN PEMBUATAN ALAT RETRIBUSI ANGKUTAN KOTA PADA TERMINAL DENGAN SISTEM PRABAYAR MENGGUNAKAN RFID

Oleh :

KHOTIFUL IMAM
NIM : 02.17.102

SEPTEMBER 2007

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN ALAT RETRIBUSI ANGKUTAN KOTA PADA TERMINAL DENGAN SISTEM PRABAYAR MENGGUNAKAN RFID

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik
Pada Jurusan Teknik Elektro Strata Satu (S-1) Konsentrasi Elektronika

Disusun oleh :

KHOTIFUL IMAM

NIM : 02.17.102

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Joseph Dedy Irawan, ST, MT)	(Sotyohadi, ST, MSc)
NIP. 132315178	

Mengetahui

Ketua Jurusan Teknik Elektro S-1

(Ir. F. Yudi Limpraptono, MT)

NIP.Y. 1039500274

FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
INSTITUT TEKNOLOGI NASIONAL MALANG
2007

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Karanglo KM. 2 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	:	Khotiful Imam
NIM	:	02.17.102
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika
Masa Bimbingan	:	22 Mei 2007 s/d 22 Nopember 2007
Judul Skripsi	:	Perancangan Dan Pembuatan Alat Retribusi Angkutan Kota Pada Terminal Dengan Sistem Prabayar Menggunakan RFID

Dipertahankan di hadapan Tim Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari	:	Senin
Tanggal	:	3 September 2007
Nilai	:	77,05 B+

PANITIA UJIAN SKRIPSI

KETUA

(Ir. Mochtar Asroni, MSME)
NIP.Y. 1018100036

SEKRETARIS

(Ir. F. Yudi Limpraptono, MT)
NIP.Y. 1039500274

ANGGOTA PENGUJI

PENGUJI I

(Ir. Widodo Pudji M., MT.)
NIP.Y. 102870017

PENGUJI II

DR. Cahyo Crysdian, Msc)
NIP. 1030400412

ABSTRAK

PERANCANGAN DAN PEMBUATAN ALAT RETRIBUSI ANGKUTAN KOTA PADA TERMINAL DENGAN SISTEM PRABAYAR MENGGUNAKAN RFID

(Khotiful Imam, 02.17.102, Teknik Elektro S.1/Elektronika)
(Dosen Pembimbing : Joseph Dedy Irawan, ST,MT dan Sotyohadi, ST, MSc.)

Kata Kunci : RFID, Mikrokontroller *ATS52*, MAX 232, Motor DC, Printer.

Pada saat ini proses penarikan retribusi di terminal angkutan kota masih ditangani secara manual yang dilakukan oleh beberapa orang petugas DLLAJ secara bergantian, biasanya terdiri dari 2 orang atau lebih yang bertugas sebagai pencatat angkutan kota yang keluar dari terminal dan mencatat apakah angkutan kota tersebut sudah membayar retribusi atau belum dan yang lain bertugas memberikan karcis dan menerima uang retribusi dari sopir angkutan kota.

Dari hal tersebut, maka timbulah suatu pemikiran tentang bagaimana membuat suatu alat yang dapat membantu proses pembayaran retribuis dengan sistem prabayar. Dalam hal ini digunakan sebuah devais kecil yang disebut tag RFID sebagai alat retribuis dan sebagai kartu identitas suatu angkutan kota.

Diharapkan nantinya alat ini dapat membantu proses kerja dari petugas retribuis dan juga dapat digunakan untuk membantu tugas tugas dinas perhubungan dalam pembuatan data statistik angkutan kota yang beroperasi setiap harinya.

Abstract

At present, collecting process retribution at public transportation Terminal still do manual process by some DLLAJ guard to another. Usually it contains with 2 persons and more who assignment as a register public transportations that out from terminal and it was registered that the public transportations paid retributions or not yet and the other assignment to give tickets and take money retributions from the public transportations driver.

From this problem, there is grow a think about how to make an instrument which help payment retributions process with prepaid system. There is use a little device, named tag RFID as a retribution instrument and also a public transportations ID card.

Next, this instrument wish to help work process from the retributions guard and also it use to help assignment the Connections Services on making publics transportations data statistics which operate everyday.

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat-Mu Ya Allah yang telah memberikan rahmat dan hidayah-Nya, sehingga dapat menyelesaikan skripsi yang berjudul "Perencanaan Dan Pembuatan Alat Ukur Tinggi Badan Dan Berat Badan Berbasis Mikrokontroler ATmega8535" ini dengan lancar. Skripsi ini merupakan persyaratan kelulusan Studi di Jurusan Teknik Elektro S-1 Konsentrasi Teknik Elektronika ITN Malang dan untuk mencapai gelar Sarjana Teknik.

Keberhasilan peyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terima kasih kepada :

1. Bapak Prof. DR. Ir. Abraham Lomi, MSEE selaku Rektor ITN Malang.
2. Bapak Ir. F. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S-1.
3. Bapak Joseph Dedy Irawan, ST, MT selaku Dosen Pembimbing I.
4. Bapak Sotyohadi, ST, MSc selaku Dosen Pembimbing II.
5. Ayah dan Ibu serta keluarga besar yang telah memberikan do'a restu, dorongan, semangat, dan biaya.
6. Rekan-rekan mahasiswa/i Elektronika 2002 yang telah memberikan semangat dan motivasi dalam penyelesaian skripsi ini.
7. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penyusun telah berusaha semaksimal mungkin dan meyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

Harapan penyusun semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, September 2007

(***Penyusun***)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dengan perkembangan pola tata letak wilayah kota-kota besar terutama semakin banyaknya jalan raya yang dibangun guna mengatasi kemacetan lalu lintas maka kebutuhan sarana transportasi umum atau angkutan kota bagi masyarakat agar sampai ketujuan dengan cepat akan meningkat sehingga diperlukan pengadaan angkutan kota dengan jurusan yang baru yang menggunakan jalur tersebut. Dengan semakin banyaknya angkutan kota dengan jurusan yang berbeda dengan kata lain akan terjadi penambahan jurusan yang akan dilalui angkutan kota tersebut maka akan semakin banyak jurusan angkutan kota pada terminal.

Penarikan retribusi kendaraan sebagai upaya untuk pemeliharaan sarana dan prasarana terminal juga dilakukan oleh petugas. Pada saat ini proses penarikan retribusi di terminal angkutan kota masih ditangani secara manual yang dilakukan oleh beberapa orang secara bergantian oleh petugas DLLAJ, biasanya terdiri dari 2 orang atau lebih yang bertugas sebagai pencatat angkutan kota yang keluar terminal dan mencatat apakah angkutan kota tersebut sudah membayar retribusi atau belum dan yang lain bertugas memberikan karcis dan menerima uang retribusi dari sopir angkutan kota.

Sepertinya tugas ini sangat merepotkan karena petugas harus berkali-kali melihat data angkutan kota terutama pada data penarikan retribusi dalam hal ini

sudah membayar atau belum sehingga lama kelamaan akan membuat kesulitan para petugas apabila angkutan kota yang ada di terminal tersebut jumlahnya banyak dengan berbagai macam jurusan sehingga dapat menimbulkan antrian yang panjang untuk angkutan kota yang akan keluar dari terminal terutama pada siang hari.

Selain itu diharapkan agar alat ini dapat digunakan untuk mambantu membuat data statistik keluarnya suatu angkutan kota dengan jurusan tertentu sehingga dapat digunakan untuk membantu tugas Dinas Perhubungan untuk mendata angkutan kota yang beroperasi setiap harinya. Data statistik tersebut bisa digunakan untuk menentukan perlu tidaknya diadakan penambahan jumlah armada angkutan kota dari suatu jurusan.

1.2. Rumusan Masalah

Dalam perencanaan dan alat retribusi angkutan kota pada terminal dengan sistem prabayar menggunakan RFID ini dapat dirumuskan beberapa masalah yang akan dibahas yaitu :

Bagaimana merancang dan membuat suatu sistem retribusi angkutan kota pada terminal dengan sistem prabayar dengan menggunakan tag RFID sebagai identitas angkutan kota.

1.3. Tujuan

Penulisan skripsi ini bertujuan untuk merancang dan membuat suatu alat retribusi angkutan kota pada terminal dengan sistem prabayar menggunakan teknologi RFID.

1.4. Batasan masalah

Untuk mencapai tujuan yang dimaksud dalam skripsi ini diperlukan batasan masalah dengan harapan agar alat yang dibuat ini dapat diselesaikan semaksimal mungkin. Batasan masalah pada tugas akhir ini meliputi :

1. Alat ini ditujukan untuk angkutan kota.
2. Komunikasi mikrokontroller dengan komputer mengggunakan RS232.
3. Motor yang digunakan adalah motor DC.
4. Tidak membahas secara detail tentang printer, catu daya dan mekanik.
5. Alat yang dicoba atau yang dibuat hanya dalam bentuk miniatur saja (simulasi).
6. Program PC menggunakan Delphi dan database menggunakan Microsoft Access.

1.5. Metodologi Penulisan

Metodologi yang dipakai dalam pembuatan skripsi ini adalah:

1. Studi Literatur

   Dengan mencari referensi-referensi yang berhubungan dengan perencanaan dan pembuatan alat yang akan dibuat.

2. Studi Lapangan

   Dengan melakukan penelitian secara langsung mengenai objek-objek yang berhubungan langsung dengan perencanaan alat yang akan dibuat.

3. Pengujian Alat

   Dengan melakukan pengujian perblok rangkaian dan kerja seluruh sistem pada alat tersebut.

4. Penyusunan Laporan Skripsi

   Membuat laporan yang terdiri dari: Pendahuluan, Landasan Teori, Perencanaan dan Pembuatan Alat, Pengujian Alat dan Penutup.

1.6. Sistematika Penulisan

Penulisan skripsi ini terbagi menjadi lima bab dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Membahas tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi dan sistematika penulisan.

BAB II LANDASAN TEORI

Membahas teori dasar penunjang perancangan dan pembuatan alat.

BAB III PERENCANAAN DAN PEMBUATAN ALAT

Membahas tentang perancangan alat yang terdiri dari perancangan perangkat keras dan perangkat lunak.

BAB IV PENGUJIAN ALAT

Membahas tentang pengujian peralatan secara keseluruhan dan analisa hasil pengujian.

BAB V PENUTUP

Berisikan kesimpulan dan saran.

BAB II

TEORI DASAR

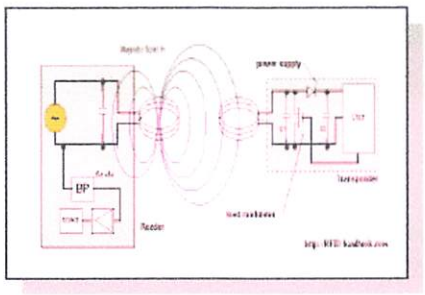
2.1. Pendahuluan

Seiring perkembangan, system identifikasi banyak dipergunakan dalam industri untuk menyediakan informasi mengenai manusia, hewan kendaraan, dan lainya. Salah satu teknik identifikasi adalah *RFID (Radio Frequency Identification*) dimana tidak diperlukan hubungan secara fisik antara obyek yang diidentifikasi dengan unit pembacanya karena menggunakan teknologi frekuensi radio.

Untuk dapat memahami alat yang akan dirancang, maka dalam bab ini akan dijelaskan mengenai teori dasar yang akan berkaitan dengan sistem yang digunakan pada perancangan dan pembuatan alat.

2.2. RFID (*Radio Frequency Identification*)

RFID adalah proses identifikasi seseorang atau objek dengan menggunakan frekuensi transmisi radio. RFID menggunakan frekuensi radio untuk membaca informasi dari sebuah devais kecil yang disebut tag atau *transponder* (*Transmitter* + *Responder*). Tag RFID akan mengenali diri sendiri ketika mendeteksi sinyal dari *devais* yang kompatibel, yaitu pembaca RFID (RFID *Reader*) dengan *range* kisaran pembacaan 12 cm serta bekerja pada frekuensi 125 KHz.



Gambar 2.1 Komunikasi Antara *Reader* dan *Transmiter* (Tag)
Sumber: RFID series datasheet, www.digiware.com,

RFID dapat disediakan dalam piranti (devais) yang hanya dapat dibaca saja (*Read Only*) atau dapat dibaca dan ditulis (*Read/Write*), tidak memerlukan kontak langsung maupun jalur cahaya untuk dapat beroperasi, dapat berfungsi pada berbagai variasi kondisi lingkungan, dan menyediakan tingkat integritas data yang tinggi. Sebagai tambahan, karena teknologi ini sulit untuk dipalsukan, maka RFID dapat menyediakan tingkat keamanan yang tinggi.

Pada sistem RFID umumnya, tag atau *transponder* ditempelkan pada suatu objek. Setiap tag dapat membawa informasi yang unik, di antaranya: serial number, model, warna, tempat perakitan, dan data lain dari objek tersebut. Ketika tag ini melalui medan yang dihasilkan oleh pembaca RFID yang kompatibel, tag akan mentransmisikan informasi yang ada pada tag kepada pembaca RFID, sehingga proses identifikasi objek dapat dilakukan.

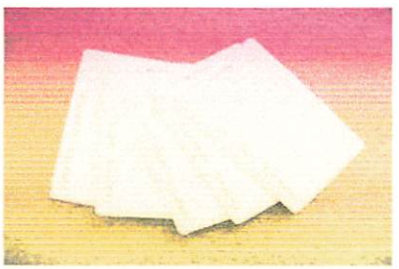
Sistem RFID terdiri dari empat komponen, di antaranya :

- Tag: Ini adalah devais yang menyimpan informasi untuk identifikasi objek. Tag RFID sering juga disebut sebagai *transponder*. Format dari tag pada perancangan ini adalah EM4001 atau tag kompatibel lainnya.

- Antena: untuk mentransmisikan sinyal frekuensi radio antara pembaca RFID dengan tag RFID.
- Pembaca RFID: adalah devais yang kompatibel dengan tag RFID yang akan berkomunikasi secara *wireless* dengan tag. Digunakan Tipe ID-12 sebagai RFID reader pada perancangan ini.
- Software Aplikasi: adalah aplikasi pada sebuah workstation atau PC yang dapat membaca data dari tag melalui pembaca RFID. Baik tag dan pembaca RFID diperlengkapi dengan antena sehingga dapat menerima dan memancarkan gelombang elektromagnetik.

2.2.1. RFID Card

RFID card chip yang di dalamnya juga terdapat nomor identitas kartu atau nomor seri kartu yang nantinya nomor tersebut akan diambil oleh reader kartu saat chip dari kartu tersebut dibaca oleh reader kartu, dimana keluaran nomor seri tersebut sudah berupa ASCII dan itu tergantung dari konfigurasi rangkaian reader kartunya.

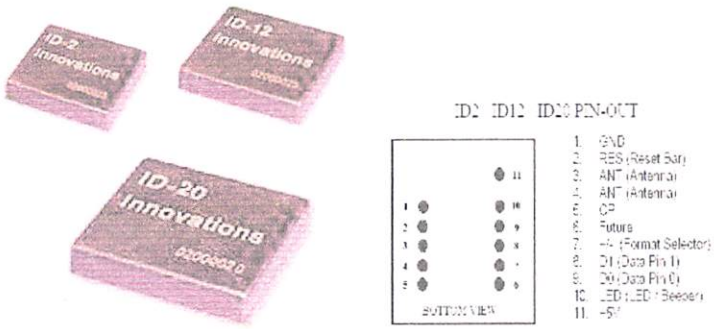


Gambar 2.2 RFID card
Sumber:ID series datasheet,

2.2.2. RFID Reader

RFID disini yang kita gunakan adalah jenis ID-12, ID-12 yang kita gunakan mempunyai jarak baca maksimal 12 cm, serta bekerja pada frekuensi 125

KHz. Sesuai dengan datasheet dari reader kartu ID-12 itu sendiri, untuk memperoleh keluaran yang berbentuk ASCII maka reader itu disusun seperti gambar di bawah ini :



Gambar 2.3 RFID Reader
Sumber: ID series datasheet, [illegible]

2.2.3. Pembacaan Format RFID

Saat ini model alat identifikasi sangatlah bermacam - macam ada yang berupa kartu dengan lubang, barcode, RFID, dll. RFID (RF Identification) merupakan suatu alat untuk identifikasi yang biasanya ditempelkan pada barang atau dibuat menjadi kartu. Disini akan dibahas mengenai cara membaca format data yang dikeluarkan oleh RFID reader dengan format output ASCII.

RFID reader mempunyai banyak sekali tipe, antara lain : ID-12, ID-20, EM-13, dll. biasanya RFID reader ini memiliki dua bentuk output serial yaitu : ASCII dan Wiegand 26-bit. Yang paling banyak digunakan adalah output dengan format ASCII, karena output ini sangat mudah untuk dihubungkan pada mikrokontroler atau PC menggunakan komunikasi serial UART.

2.2.4. Format Data ASCII

Output yang memiliki format ASCII memiliki struktur sebagai berikut :

02	10 data karakter ASCII	checksum	CR	LF	03

Gb1. Format data ASCII

Checksum merupakan hasil EXOR (Exclusive OR) dari 5 biner data byte. Untuk lebih jelasnya tentang cara pembacaan format ASCII, lihat contoh berikut. Misalnya data output serial (dalam hexadesimal) yang kita tangkap adalah sebagai berikut:

02	30	34	36	32	30	31	44	37	36	43	44	43	0D	0A	03

Langkah pertama adalah merubah semua nilai data diatas menjadi karakter ASCII. Misalnya 30H menjadi karakter "0", 34H menjadi karakter "4", dst. Langkah kedua adalah menyusun data - data tersebut ke dalam Format Data ASCII seperti gambar 1. Kemudian ambil 10 data karakter ASCII. Dalam contoh ini berarti data tersebut adalah :

[illegible]	[illegible]	36	32	30	31	44	37	36	43	Data Heksa
		6	2	0	1	D	7	6	C	Data ASCII

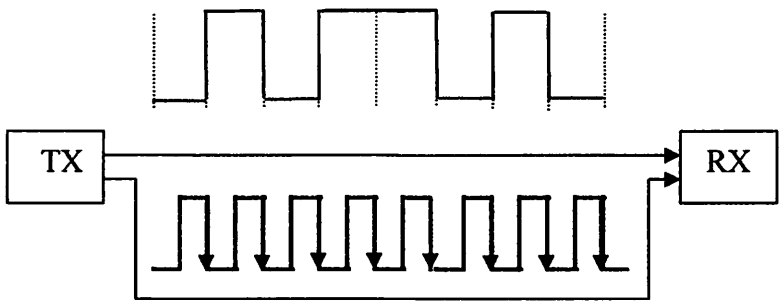
Untuk data dengan warna [illegible] merupakan data untuk jenis - jenis kartu dan tidak digunakan dalam proses konversi, yang akan dipakai disini adalah data yang ke 3 s/d 10. Hasil konversi dari data heksa ke dalam data ASCII adalah "6201D76C". Gabungkan karakter data ASCII menjadi bilangan Hexadesimal, kemudian konversikan bilangan hexadesimal tsb ke dalam desimal. Hasilnya sebagai berikut : 6201D76C H menjadi 1644287852 (ini merupakan nomor kartu sebenarnya yang tertera pada badan kartu tsb). Cara ini hanya berlaku pada kartu yang tidak dienkripsi.

2.3. Komunikasi Serial

Komunikasi serial ialah pengiriman data secara serial (data dikirim satu per satu secara berurutan) sehingga komunikasi serial jauh lebih lambat daripada komunikasi paralel. Serial *port* lebih sulit ditangani karena peralatan yang dihubungkan ke serial *port* harus berkomunikasi mengunakan transmisi serial sedangkan data di komputer diolah secara paralel. Oleh karena itu data dari / ke serial port harus dikonversikan ke / dari bentuk paralel untuk bisa digunakan. Kelebihan komunikasi serial ialah jangkauan panjang kabel yang lebih jauh dibandingkan paralel karena serial *port* mengirimkan logika 1 dengan kisaran tegangan -3 hingga -25 *volt* dan logika 0 sebagai +3 hingga +25 *volt* sehingga kehilangan daya karena panjangnya kabel bukan masalah utama.

Komunikasi serial ada dua macam, *asynchronous* serial dan *synchronous* serial. *Synchronous* serial adalah komunikasi dimana hanya ada satu pihak (pengirim atau penerima) yang menghasilkan *clock* dan mengirimkan *clock* tersebut bersama - sama dengan data. Contoh pengunaan *synchronous* serial terdapat pada transmisi data *keyboard. Asynchronous* serial adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan *clock* namun hanya data yang ditransmisikan, tanpa *clock.* Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi *clock* harus sama dan harus terdapat sinkronisasi. Setelah adanya sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi *clock* pengirim dan penerima akan membaca data sesuai dengan frekuensi *clock* penerima.

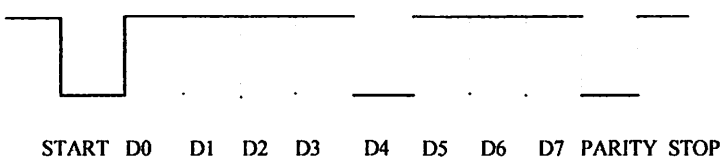
2.3.1 Komunikasi *Sinkron*



Gambar 2.4 Komunikasi Serial Dengan Sinyal Sinkronisasi
Sumber :RS-232 Standart Configuratioan

Setiap TX mengirimkan Dx atau bit ke x dari satu *byte* data akan diikuti dengan sinyal sinkronisasi yang berupa sinyal transisi dari rendah ke tinggi atau tinggi ke rendah. RX akan mengetahui bahwa dijalur data ada data milik Dx, sesuai dengan banyaknya sinyal sinkronisasi yang diterima, saat sinyal sinkronisasi pertama, berarti data milik D0, kedua milik D1dan seterusnya.

2.3.2 Komunikasi *Asinkron*



Gambar 2.5 Format Sinyal Serial Asinkron
Sumber :RS-232 Standart Configuratioan

Cara kedua dengan komunikasi *asynchron*, yaitu dengan menetapkan kecepatan bit dan menyisipkan beberapa bit protokol, yaitu bit *START*, *PARITY* bit dan *STOP* seperti diperlihatkan pada gambar 2.5. diatas.

- Kecepatan bit disebut sebagai *baud rate* atu pesat bit disingkat *bps (bit per second)*, pada standart komunikasi diantaranya adalah 1200, 4800, 9600 bps.

- Makin besar bit, makin cepat data ditransmisikan, tetapi memerlukan *bandwidth* jalur yang semakin lebar, penggunaan kabel biasa atau kabel telpon kecepatan transmisi data dibatasi oleh *bandwidth* kabel tersebut.
- Saluran tanpa data bertegangan '1' , start bit selebar 1 pulsa, selalu '0'.
- Setelah *bit start*, diikuti serial data, jumlah data dapat 7 atau 8.
- Setelah data - data bit bisa diiukuti (jika diperlukan) oleh *parity bit*, jika dipilih *parity even*, maka bit parity akan menggenapkan jumlah bit '1' nya, jika dipilih *parity odd*, maka bit parity akan mengganjilkan jumlah bit '1'.
- Akhir data adalah stop *bit*, yang selalu '1'.

2.3.3 Arah Pengiriman Data

Dikenal tiga macam arah pengiriman data, yaitu *Simplex, Half Duplex,* dan *Full Duplex.*

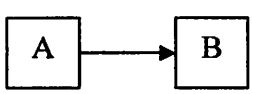
Simplex adalah sistem pemindahan data yg hanya satu arah saja, misalnya dari A ke B, dimana A sebagai pengirim dan B sebagai penerima, dan tidak dapat mengirimkan data dari B ke A.

Half Duplex adalah sistem pemindahan data dua arah, tetapi tidak dapat dilakukan secara bersamaan, harus bergantian.

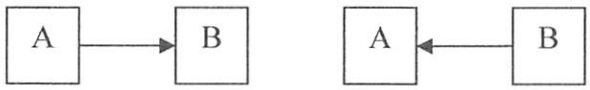
Full Duplex adalah sistem pemindahan data dua arah dan dapat berlangsung secara bersamaan dalam satu waktu.

Untuk dapat jelasnya, dapat dilihat pada gambar berikut :

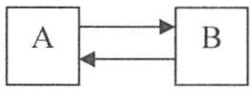
a. Komunikasi data *Simplex*



b. Komunkasi data *Half Duplex*

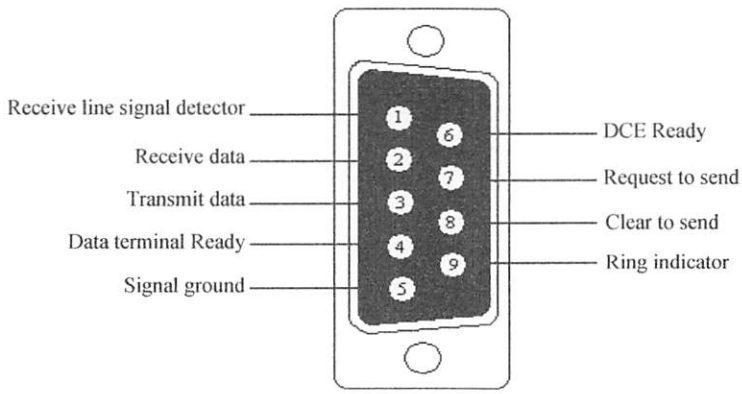


c. Komunikasi data *Full Duplex*



2.3.4 Interface

Istilah *interface* kalau diterjemahkan, mengandung arti sebagai penghubung. Namun bagi mereka yang awam dalam dunia komputer arti diatas mungkin sulit dicerna. Dalam suatu sistem komputer, kejadian dialat diluar komputer yang sedang dihubungkan dengan komputer mungkin dapat mengganggu komputer itu sendiri. Sinyal - sinyal yang tak dikenal seperti itu tidak diperkenankan untuk merusak rangkaian komputer yang ada. Untuk itu dibutuhkan alat perantara yang berfungsi sebagai penghubung dua lingkungan yang berbeda. Alat tersebut dinamakan *interface*. Karena interface merupakan semacam pintu gerbang maka interface sering disebut port I/O.



Gambar 2.6 Skematik DB 9.

Sumber :RS-232 Standart Configuratioan

2.3.5 Dasar-dasar *serial interface*

Proses transfer serial dengan menggunakan EIA RS 232 antara dua terminal biasanya memerlukan sebuah DTE (*Data Terminal Equipment*) untuk masing-masing terminal, kadang diperlukan seperangkat peralatan untuk komunikasi yang lebih kompleks misalnya dengan memanfaatkan modem. Perangkat tersebut sering disebut dengan DCE (*Data Communication Equipment*). Pada prinsipnya proses transfer data menggunakan sebuah serial *interface* ini sangat sederhana. Data yang ditransfer dari suatu terminal akan diterima oleh terminal lainnya.

Jenis data yang ditransfer adalah dalam bentuk biner (bit perbit transfer) dengan bantuan baud untuk kecepatan proses transfernya (bit per detik). Dalam proses transfer ini harus terdapat suatu peralatan yang berfungsi sebagai *hand-shake* (jabat tangan) yang sebagai pemantau status yang diterima untuk memberikan suatu respon yang sesuai. Dalam merancang *softwere* serial, *hand-shake* disempurnakan dengan menambahkan karakter pengendali dalam deretan atau jumlah bit data data yang ditransfer yang biasa disebut sebagai start bit dan stop bit. Konektor dari kabel - kabel ini sebenarnya bersifat pasif karena yang mengendalikan semua itu adalah sebuah alat yang disebut UART (*Universal Asynchronous Receiver/Transmitter*) yang dihubungkan dengan RS 232 untuk trasnsmisi data. Secara praktis untuk kebutuhan transfer data/komunikasi data terdapat dua macam konektor RS 232, jenis 25 pin dan 9 pin.

2.3.6 *Sinyal-sinyal RS-232*

Standar antarmuka RS-232 mendefinisikan sinyal - sinyal yang dipakai baik untuk mentransmisikan / menerima data maupun untuk proses jabat tangan (*handshaking*). Dalam aplikasi sinyal - sinyal RS-232 ini dihubungkan dengan konektor 25 pin (DB-25) atau 9 pin (DB-9). Pada tabel 2-1 berikut ini diberikan nama sinyal penting beserta hubungannya dengan konektor DB-9 maupun DB-25.

Tabel 2.1 Sinyal-Sinyal RS 232

DB-9	DB-25	NAMA SINYAL
1	8	DCD, DATA CARRIER DETECT
2	3	RD, RECEIVER DATA
3	2	TD, TRASMIT DATA
4	20	DTR, DATA TERMINAL READY
5	7	SG, SIGNAL GROUND
6	6	DSR, DATA SET READY
7	4	RTS, REQUEST TO SEND
8	5	CTS, CLEAR TO SEND
9	22	RI, RING INDICATOR

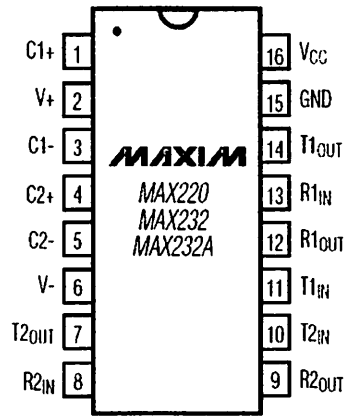
Sumber :RS-232 Standart Configuratioan

2.3.7 *IC MAX 232*

RS 232 merupakan salah satu jenis antarmuka (*interface*) dalam proses transfer data antar komputer dalam bentuk serial transfer. RS 232 merupakan kependekan dari (*Recommended Standart number 232*). Alat ini dibuat oleh Electronic Industry Assosiation, untuk interface antara peralatan terminal data dan peralatan komunikasi data, dengan menggunakan data biner serial sebagai data yang ditransmisikan. IC Max 232 merupakan konverter tegangan dari level - level TTL C-mos ke level RS 232. IC Max 232 ini mempunyai empat buah bagian konverter yaitu dua buah *driver receiver* dan dua buah *driver transmitter*.

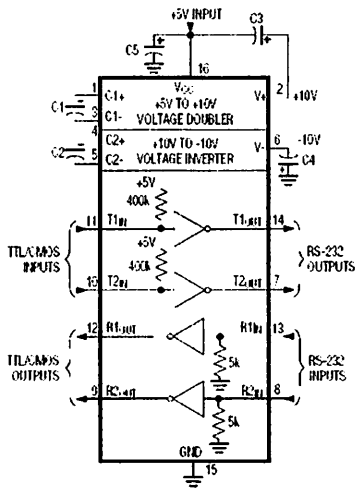
Saluran data pada *port* serial *PC* menggunakan standard RS232, dimana *logic* 0 (*low*) dinyatakan sebagai tegangan antara +3 *Volt* sampai +15 *Volt* dan *logic* 1 (*high*) dinyatakan sebagai tegangan antara -3 *Volt* sampai -15 *Volt*. Level tegangan ini tidak sesuai dengan level tegangan yang dipakai pada *port* seri AT89S52 yang menggunakan standard *TTL (Transistor Transistor Logic)*, yaitu level tegangan baku dalam rangkaian - rangkaian digital.

Dalam standar *TTL logic* 0 (*low*) dinyatakan sebagai tegangan antara 0 *Volt* sampai 0.8 *Volt*, dan *logic* 1 (*high*) dinyatakan sebagai tegangan antara 3.5 *Volt* sampai 5 *Volt*. Karena perbedaan tegangan tersebut, agar *port* seri PC tidak merusak *port* seri AT89S52 antara keduanya dipasangkan IC MAX232 sebagai penyesuai tegangan.



Gambar 2.7 Konfigurasi Pin IC MAX232
Sumber : Datasheet MAX 232

Rangkaian dasar dari MAX 232 dapat dilihat pada gambar berikut ini:



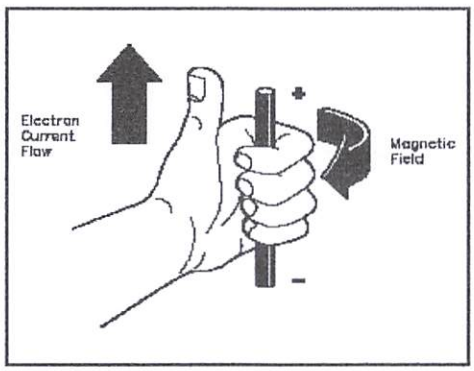
Gambar 2.8 Rangkaian Operasi MAX 232
Sumber : Datasheet MAX 232

2.4. Motor DC dan Driver

2.4.1. Teori Dasar Motor DC

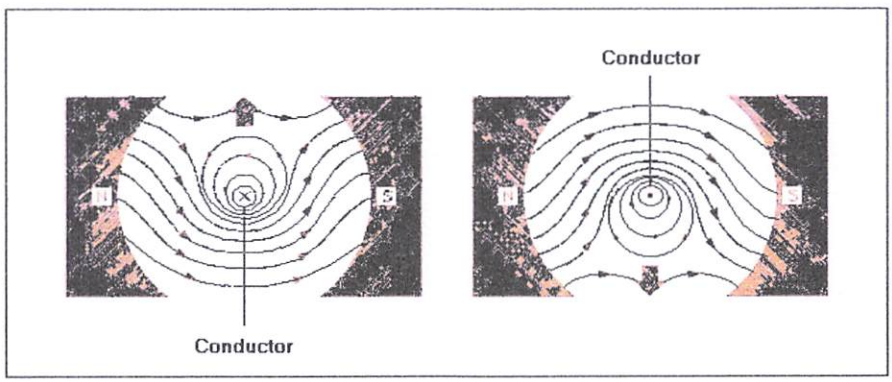
Setiap arus yang mengalir melalui sebuah konduktor akan menimbulkan medan magnet. Arah medan magnet dapat ditentukan dengan kaidah tangan kiri. Ibu jari tangan menunjukkan arah aliran arus listrik sedangkan jari - jari yang

lain menunjukkan arah medan magnet yang timbul, seperti yang ditunjukkan oleh gambar 2.9 berikut ini.



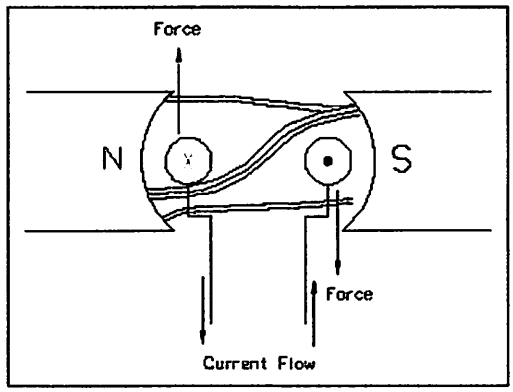
Gambar 2.9 Kaidah Tangan Kiri
Sumber : Fisika Dasar

Jika suatu konduktor yang dialiri arus listrik ditempatkan dalam sebuah medan magnet, kombinasi medan magnet akan ditunjukkan oleh gambar 2.10. Arah aliran arus listrik dalam konduktor ditunjukkan dengan tanda "x" atau ".". Tanda "x" menunjukkan arah arus listrik mengalir menjauhi pembaca gambar, tanda "." menunjukkan arah arus listrik mengalir mendekati pembaca gambar.



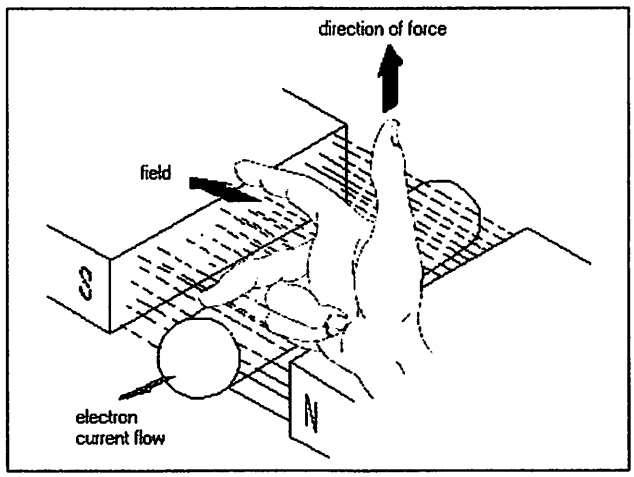
Gambar 2.10 Konduktor Berarus Listrik Dalam Medan Magnet
Sumber : Fisika Dasar

Pada gambar sebelah kiri, arah medan magnet pada sisi atas yang dihasilkan oleh konduktor berlawanan dengan arah medan magnet yang



Gambar 2.11 Bergeraknya Sebuah Motor
Sumber : Fisika Dasar

Sebuah cara lagi untuk menunjukkan hubungan antara arus listrik yang mengalir didalam sebuah konduktor, medan magnet dan arah gerak, adalah kaidah tangan kanan untuk motor seperti yang diperlihatkan pada gambar 2.12.



Gambar 2.12 Kaidah Tangan Kanan Untuk Motor
Sumber : Elektronika dalam Industri

Kaidah tangan kanan untuk motor menunjukkan arah arus yang mengalir didalam sebuah konduktor yang berada dalam medan magnet. Jari tengah menunjukkan arah arus yang mengalir pada konduktor, jari telunjuk menunjukkan arah medan magnet dan ibu jari menunjukkan arah gaya putar.

dihasilkan oleh magnet permanen. Sementara pada sisi sebelah bawah, arah medan magnet yang dihasilkan oleh konduktor searah dengan arah medan magnet yang dihasilkan oleh magnet permanen. Dengan kata lain, pada sisi sebelah atas kerapatan fluks magnet lebih sedikit dari pada sisi sebelah bawah. Sebuah gaya dorong akan menyebabkan konduktor bergerak ke sisi sebelah atas.

Pada gambar sebelah kanan, arah medan magnet pada sisi atas yang dihasilkan oleh konduktor searah dengan arah medan magnet yang dihasilkan oleh magnet permanen. Sementara pada sisi sebelah bawah, arah medan magnet yang dihasilkan oleh konduktor berlawanan dengan arah medan magnet yang dihasilkan oleh magnet permanen. Dengan kata lain, pada sisi sebelah bawah kerapatan fluks magnet lebih sedikit dari pada sisi sebelah atas. Sebuah gaya dorong akan menyebabkan konduktor bergerak ke sisi sebelah bawah.

Pada sebuah motor DC, konduktor dibentuk menjadi sebuah loop sehingga ada dua bagian konduktor yang berada didalam medan magnet pada saat yang sama, seperti diperlihatkan pada gambar 2.11.

Konfigurasi konduktor seperti ini akan menghasilkan distorsi pada medan magnet utama dan menghasilkan gaya dorong pada masing - masing konduktor. Pada saat konduktor di tempatkan pada rotor, gaya dorong yang timbul akan menyebabkan rotor berputar searah dengan jarum jam, seperti diperlihatkan pada gambar 2.11.

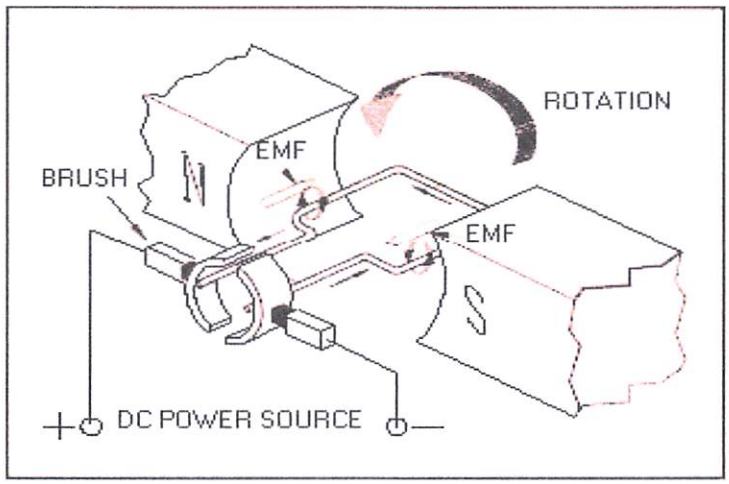
Adapun besarnya gaya yang bekerja pada konduktor tersebut dapat dirumuskan dengan :

F = B.L.I (Newton)

Dimana : B = kerapatan fluks magnet (weber)

L = panjang konduktor (meter)

I = arus listrik (ampere)



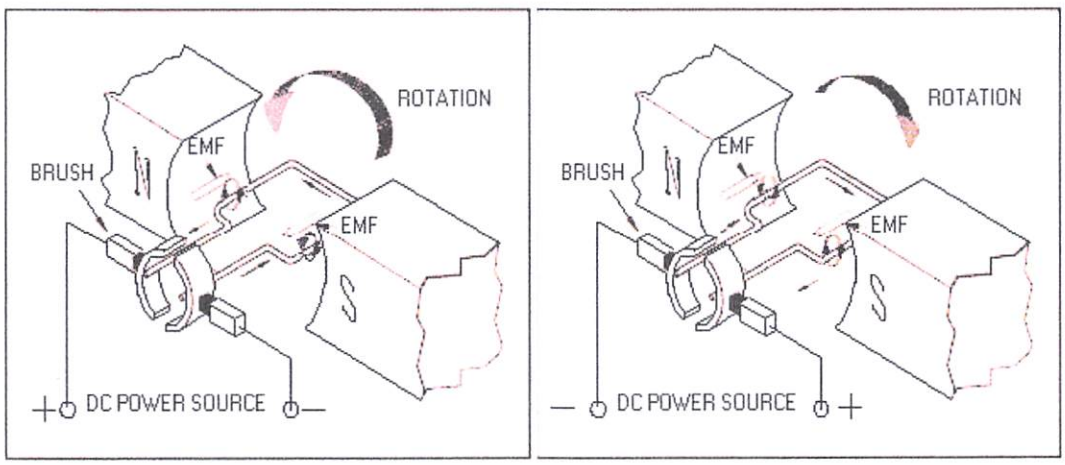
Gambar 2.13 Konstruksi Dasar Motor DC

Sumber : Elektronika dalam Industri

Pada gambar 2.13 diatas tampak sebuah konstruksi dasar motor dc, pada gambar diatas terlihat bahwa pada saat terminal motor diberi tegangan dc, maka arus elektron akan mengalir melalui konduktor dari terminal negatif menuju ke terminal positif. Karena konduktor berada diantara medan magnet, maka akan timbul medan magnet juga pada konduktor yang arahnya seperti terlihat pada gambar 2.13 diatas. Arah garis gaya medan magnet yang dihasilkan oleh magnet permanen adalah dari kutub utara menuju ke selatan. Sementara pada konduktor yang dekat dengan kutub selatan, arah garis gaya magnet disisi sebelah bawah

searah dengan garis gaya magnet permanen sedangkan di sisi sebelah atas arah garis gaya magnet berlawanan arah dengan garis gaya magnet permanen. Ini menyebabkan medan magnet disisi sebelah bawah lebih rapat daripada sisi sebelah atas. Dengan demikian konduktor akan terdorong ke arah atas. Sementara pada konduktor yang dekat dengan kutub utara, arah garis gaya magnet disisi sebelah atas searah dengan garis gaya magnet permanen sedangkan di sisi sebelah bawah arah garis gaya magnet berlawanan arah dengan garis gaya magnet permanen. Ini menyebabkan medan magnet disisi sebelah atas lebih rapat daripada sisi sebelah bawah. Dengan demikian konduktor akan terdorong ke arah bawah. Pada akhirnya konduktor akan membentuk gerakan berputar berlawanan dengan jarum jam seperti terlihat pada gambar 2.13 diatas.

2.4.2. Pengendalian Arah Putaran Motor DC



Gambar 2.14 Arah Putaran Motor DC
Sumber : Elektronika dalam Industri

Dari gambar 2.14 diatas, agar arah putaran motor DC berubah, maka polaritas tegangan pada terminal motor harus dibalik.

2.5. Transistor

2.5.1. Bipolar Transistor

Prinsip kerja transistor adalah arus bias base-emiter yang kecil mengatur besar arus kolektor-emiter. Bagian penting berikutnya adalah bagaimana caranya memberi arus bias yang tepat sehingga transistor dapat bekerja optimal.

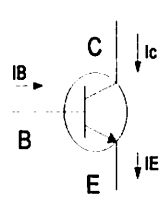
2.5.2. Arus bias

Ada tiga cara yang umum untuk memberi arus bias pada transistor, yaitu rangkaian *CE (Common Emitter), CC (Common Collector)* dan *CB (Common Base).* Namun dalam hal ini akan lebih detail dijelaskan bias transistor rangkaian CE. Dengan menganalisa rangkaian CE akan dapat diketahui beberapa parameter penting dan berguna terutama untuk memilih transistor yang tepat untuk berbagai aplikasi.

2.5.3. Arus *Emiter*

Dari hukum Kirchhoff diketahui bahwa jumlah arus yang masuk ke satu titik akan sama jumlahnya dengan arus yang keluar. Jika teorema tersebut diaplikasikan pada transistor, maka hukum itu menjelaskan hubungan :

$$I_E = I_C + I_B \quad \text{.....(2-1)}$$



Gambar 2.15 Arus Emitor
Sumber : Aswan Hamonangan , "rubric elka dasar", 2002

Persamanaan (2-2) tersebut mengatakan arus *emiter* I_E adalah jumlah dari arus kolektor I_C dengan arus base I_B. Karena arus I_B sangat kecil sekali atau disebutkan $I_B << I_C$, maka dapat dinyatakan :

$$I_E = I_C \quad \text{.....(2-2)}$$

2.5.4. *Alpha* (a)

Pada tabel data transistor (*databook*) sering dijumpai spesifikasi $_{dc}$ *(alpha dc*) yang tidak lain adalah :

$$\alpha_{dc} = I_C/I_E \quad \text{.....(2-3)}$$

Definisinya adalah perbandingan arus kolektor terhadap arus emitor. Karena besar arus kolektor umumnya hampir sama dengan besar arus emiter maka idealnya besar a_{dc} adalah = 1 (satu). Namun umumnya transistor yang ada memiliki a_{dc} kurang lebih antara 0.95 sampai 0.99.

2.5.5. *Beta* (ß)

Beta didefenisikan sebagai besar perbandingan antara arus kolektor dengan arus *base*.

$$\beta = I_C/I_B \quad \text{.....(2-4)}$$

Dengan kata lain, b adalah parameter yang menunjukkan kemampuan penguatan arus *(current gain)* dari suatu transistor. Parameter ini ada tertera di *data book* transistor dan sangat membantu para perancang rangkaian elektronika dalam merencanakan rangkaiannya.

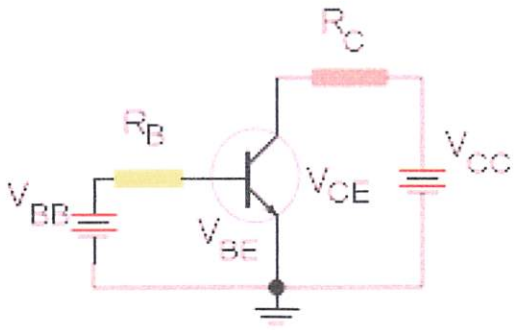
Misalnya jika suatu transistor diketahui besar β = 250 dan arus kolektor sebesar 10 mA, maka berapakah arus bias base yang diperlukan. Maka :

$$\mathbf{I_B = I_C/\beta = 10mA/250 = 40\ uA}$$

Dari rumusan ini lebih terlihat definisi penguatan arus transistor, yaitu sekali lagi, arus *base* yang kecil menjadi arus kolektor yang lebih besar.

2.5.6. *Common Emitter* (CE)

Rangkaian *Common Emitter* adalah rangkain yang paling sering digunakan untuk berbagai aplikasi yang mengunakan transistor. Dinamakan rangkaian *Common Emitter*, sebab titik *ground* atau titik tegangan 0 volt dihubungkan pada titik *emiter*.



Gambar 2.16 Rangkaian *Common Emitter*
Sumber : Aswan Hamonangan ,"rubric elka dasar", 2002

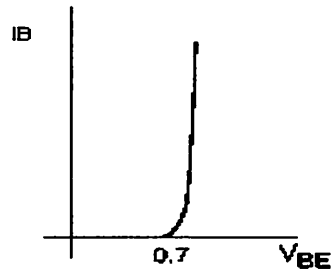
Sekilas tentang notasi, ada beberapa notasi yang sering digunakan untuk mununjukkan besar tegangan pada suatu titik maupun antar titik. Notasi dengan 1 *subscript* adalah untuk menunjukkan besar tegangan pada satu titik, misalnya V_C = tegangan kolektor, V_B = tegangan base dan V_E = tegangan emiter.

Ada juga notasi dengan 2 *subscript* yang dipakai untuk menunjukkan besar tegangan antar 2 titik, yang disebut juga dengan tegangan jepit. Diantaranya adalah :

- V_{CE} = tegangan jepit kolektor-emitor.
- V_{BE} = tegangan jepit base-emitor.
- V_{CB} = tegangan jepit kolektor-base.

Notasi seperti V_{BB}, V_{CC}, V_{EE} berturut-turut adalah besar sumber tegangan yang masuk ke titik base, kolektor dan emitor.

2.5.7. Kurva *Base*



Grafik 2.17 Kurva I_B –V_{BE}

sumber : Malvino, Prinsip-prinsip Elektronika,2002

Hubungan antara I_B dan V_{BE} tentu saja akan berupa kurva dioda. Karena memang telah diketahui bahwa junction *base-emitor* tidak lain adalah sebuah dioda. Jika hukum Ohm diterapkan pada loop base diketahui adalah :

$$I_B = (V_{BB} - V_{BE}) / R_B \quad \text{.....(2-5)}$$

V_{BE} adalah tegangan jepit dioda *junction base-emitor*. Arus hanya akan mengalir jika tegangan antara base-emitor lebih besar dari V_{BE}. Sehingga arus I_B mulai aktif

mengalir pada saat nilai V_{BE} tertentu. Besar V_{BE} umumnya tercantum di dalam *databook*. Tetapi untuk penyederhanaan umumnya diketahui V_{BE} = 0.7 volt untuk transistor silikon dan V_{BE} = 0.3 volt untuk transistor germanium.

Sampai disini akan sangat mudah mengetahui arus I_B dan arus I_C dari rangkaian berikut ini, jika diketahui besar b = 200. Katakanlah yang digunakan adalah transistor yang dibuat dari bahan silikon.

$I_B = (V_{BB} - V_{BE}) / R_B$

$= (2V - 0.7V) / 100\ K$

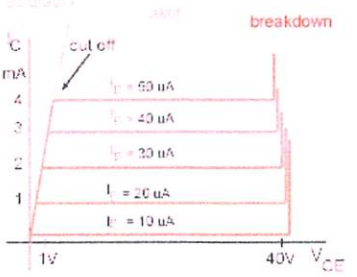
$= 13\ uA$

Dengan b = 200, maka arus kolektor adalah :

$I_C = b\ I_B = 200 \times 13uA = 2.6\ mA$

2.5.8. Kurva Kolektor

Sekarang sudah diketahui konsep arus base dan arus kolektor. Satu hal lain yang menarik adalah bagaimana hubungan antara arus base I_B, arus kolektor I_C dan tegangan kolektor-emiter V_{CE}. Pada grafik berikut telah diplot beberapa kurva kolektor arus I_C terhadap V_{CE} dimana arus I_B dibuat konstan.



Grafik 2.18 Kurva Kolektor
sumber : Malvino, Prinsip-prinsip Elektronika,2002

Dari kurva ini terlihat ada beberapa *region* yang menunjukkan daerah kerja transistor. Pertama adalah daerah *saturasi*, lalu daerah *cut-off*, kemudian daerah aktif dan seterusnya daerah *breakdown*.

2.5.9. Daerah Aktif

Daerah kerja transistor yang normal adalah pada daerah aktif, dimana arus I_C konstan terhadap berapapun nilai V_{CE}. Dari kurva ini diperlihatkan bahwa arus I_C hanya tergantung dari besar arus I_B. Daerah kerja ini biasa juga disebut daerah linier (*linear region*).

Jika hukum Kirchhoff mengenai tegangan dan arus diterapkan pada loop kolektor (rangkaian CE), maka dapat diperoleh hubungan :

$$V_{CE} = V_{CC} - I_C R_C \quad \text{.....(2-6)}$$

Dapat dihitung disipasi daya transistor adalah :

$$P_D = V_{CE}\, I_C \quad \text{.....(2-7)}$$

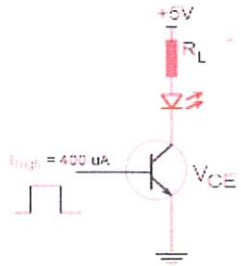
Rumus ini mengatakan jumlah disipasi daya transistor adalah tegangan kolektor-emitor dikali jumlah arus yang melewatinya. Disipasi daya ini berupa panas yang menyebabkan naiknya temperatur transistor. Umumnya untuk transistor power sangat perlu untuk mengetahui spesifikasi $P_D$max. Spesifikasi ini menunjukkan temperatur kerja maksimum yang diperbolehkan agar transistor masih bekerja normal. Sebab jika transistor bekerja melebihi kapasitas daya $P_D$max, maka transistor dapat rusak atau terbakar.

2.5.10. Daerah Saturasi

Daerah saturasi adalah mulai dari V_{CE} = 0 volt sampai kira-kira 0.7 volt (transistor silikon), yaitu akibat dari efek dioda kolektor-base yang mana tegangan V_{CE} belum mencukupi untuk dapat menyebabkan aliran elektron.

2.5.11. Daerah *Cut-Off*

Jika kemudian tegangan V_{CC} dinaikkan perlahan-lahan, sampai tegangan VCE tertentu tiba-tiba arus IC mulai konstan. Pada saat perubahan ini, daerah kerja transistor berada pada daerah *cut-off* yaitu dari keadaan saturasi (OFF) lalu menjadi aktif (ON). Perubahan ini dipakai pada system digital yang hanya mengenal angka biner 1 dan 0 yang tidak lain dapat direpresentasikan oleh status transistor OFF dan ON.



Gambar 2.19 Rangkaian Driver LED
sumber : Malvino, Prinsip-prinsip Elektronika,2002

Misalkan pada rangkaian driver LED di atas, transistor yang digunakan adalah transistor dengan Ic = I_{LED} = 20 mA. Penyalaan LED diatur oleh sebuah gerbang logika (*logic gate)* dengan arus *output high* = 400 uA dan diketahui tegangan forward LED, V_{LED} = 2.4 volt. Lalu pertanyaannya adalah, berapakah seharusnya resistansi R_L yang dipakai.

$ß = I_C/I_B = 20\ mA/\ 400uA = 50$

Arus sebesar ini cukup untuk menyalakan LED pada saat transistor *cut-off*. Tegangan VCE pada saat *cut-off* idealnya = 0, dan perkiraan ini sudah cukup untuk rangkaian ini.

$R_L = (V_{CC} - V_{LED} - V_{CE}) / I_C$

$= (5 - 2.4 - 0)V / 20\ mA$

$= 2.6V / 20\ mA$

$= 130\ Ohm$

2.5.12. Daerah *Breakdown*

Dari kurva kolektor, terlihat jika tegangan V_{CE} lebih dari 40V, arus I_C menanjak naik dengan cepat. Transistor pada daerah ini disebut berada pada daerah breakdown. Seharusnya transistor tidak boleh bekerja pada daerah ini, karena akan dapat merusak transistor tersebut. Untuk berbagai jenis transistor nilai tegangan V_{CE}max yang diperbolehkan sebelum breakdown bervariasi. V_{CE}max pada databook transistor selalu dicantumkan juga.

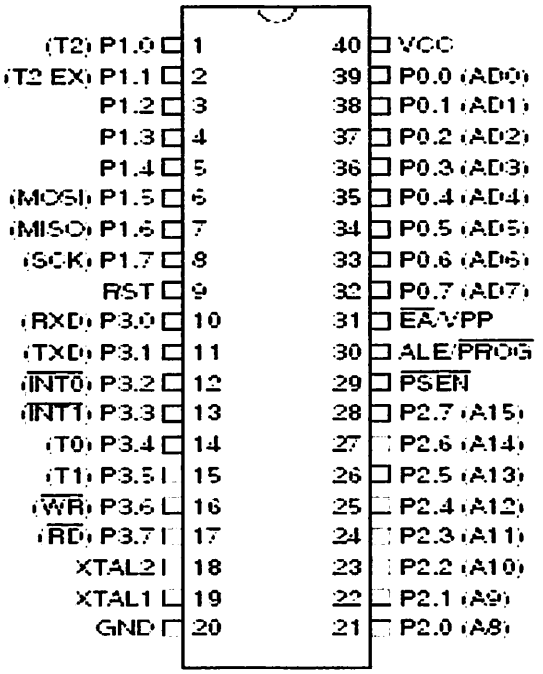
2.6. Mikrokontroller AT89S52

Mikrokontroler AT89S52 merupakan salah satu anggota keluarga dari MCS-51, yaitu suatu komponen produksi ATMEL yang berorientasi kontrol (*microcontroller*). Intel mengklarifikasikan dalam kelompok *embedded microcontroller*, yang artinya adalah mikrokontroler yang dapat diprogram ulang

(*reprogrammable*). Di dalam *chip* mikrokontroler AT89S52 ini sudah tersedia berbagai macam peralatan pendukung mikroprosesor seperti RAM, *serial port*, *bus-bus* data dan lainnya yang membuat pemakai *chip* ini dapat menekan penambahan komponen pendukung. Spesifikasi perangkat keras dari mikrokontroler AT89S52 adalah sebagai berikut :

- CPU (*Central Processing Unit*) dengan lebar data 8 *bit*.
- Prosesor *Boolean* untuk operasi logika 1 *bit*.
- Pembangkit *clock* internal.
- Tiga buah *timer/counter*
- Dua buah saluran interupsi eksternal.
- Jalur I/O dua arah (*bidirectional*) 32 buah.
- Memori program terpisah dari memori data.
- Memori data internal 256 *byte*.
- Alamat memori program eksternal 64 *Kilobyte*.
- Alamat memori data eksternal 64 *Kilobyte*.
- Memori program internal sebesar 8 *Kilobyte*.

Pada Gambar 2.20. memperlihatkan konfigurasi pin dan sambungan - sambungan keluar dari mikrokontroler AT 89S52.



Gambar 2.20 Konfigurasi pin AT 89S52

Data Sheet AT89S52, Atmel Corporation 2005, All Rights Reserved

Berikut ini adalah penjelasan dari masing-masing pin mikrokontroler AT 89S52:

- Pin 1 sampai 8

  Port 1 : merupakan 8-bit saluran masukan atau keluaran dua arah. Setiap saluran mampu melayani 4 masukan.

- Pin 9

  RST : Merupakan masukan *reset*. Logika *high* yang akan membuat mikrokontroler AT 89S52 menjalankan rutin *reset*.

- Pin 10 sampai 17

  Port 3 : *port* 3 terdiri dari 8 saluran masukan atau keluaran dua arah. Setiap salurannya mampu melayani 4 masukan. Selain sebagai *port* masukan atau keluaran, *port* 3 juga mempunyai fungsi-fungsi khusus yang dimiliki oleh keluarga MCS-51. Fungsi tersebut dapat dilihat dalam table 2.3.

- Pin 18 sampai 19

X1 (XTAL) dan X2 (XTAL2): Jika dikonfigurasikan bersama sebuah kristal akan membentuk rangkaian osilator on-chip pada mikrokontroler.

- Pin 20 sampai 27

Port 2 : *port* 2 terdiri dari 8 saluran masukan atau keluaran dua arah. Setiap salurannya mampu melayani 4 masukan. *Port* 2 mengeluarkan alamat bagian tinggi (A8-A15), selama pengambilan intruksi dari memori program eksternal dan pengambilan data dari memori data eksternal yang mengunakan mode pengalamatan 16-bit (dengan perintah MOVX @DPTR).

- Pin 29

$\overline{PSEN}$: *Program Store Enable* merupakan sinyal baca yang mengesekusi memori program eksternal.

- Pin 30

ALE/$\overline{PROG}$: *Addres Latch Enable* merupakan pulsa yang berfungsi menahan alamat rendah (A0-A7) pada port 0, selama dilakukan proses baca atau tulis memori eksternal. Pin ini juga berfungsi sebagai masukan pulsa program ($\overline{PROG}$), selama dilakukan pemprograman pada EEPROM eksternal.

- Pin 31

$\overline{EA}$/VP : *External Access.* $\overline{EA}$ dihubungkan dengan VSS untuk memungkinkan pengambilan intruksi pada memori program eksternal yang berlokasi 0000h sampai FFFFh. Jika diinginkan mengunakan memori porogram internal, maka $\overline{EA}$ dihubungkan VCC.

- Pin 32 sampai 39

Port 0 : *Port* 0 terdiri dari 8 saluran masukan atau keluaran dua arah. Setiap saluran mampu melayani 8 masukan. *Port* 0 merupakan saluran alamat bagian rendah (A0-A7), yang dimultipleks dengan saluran bus data (D0-D7), yang digunakan pada saat mengakses memori data eksternal dan memori program eksternal.

- Pin 40

VCC : Merupakan masukan catu daya 5 volt, dengan toleransi kurang lebih 10%.

Tabel 2.2 Keluarga MCS 51

Tipe	Tipe tanpa EPROM	Tipe ber-EPROM	Kapasitas ROM	Kapasitas RAM	*Port* I/O	Pewaktu
8051	8031	-	4K	128	4	2
8051AH	8031AH	8751H	4K	128	4	2
8052AH	8032AH	8752BH	8K	256	4	3
80C51BH	80C31BH	87C51	4K	128	4	2
80C52	80C32	-	8K	256	4	3
83C51FA	80C51BH	8751FA	8K	256	4	3
83C51FB	80C51FB	87C51FB	16K	256	4	3
83C152	80C152	-	8K	256	5	3
89S52	-	89S52	8K	256	4	3

Sumber : Atmel Data Book ,2006

Keluarga MCS®51 yang diproduksi Intel mempunyai konfigurasi yang berbeda-beda sesuai dengan jenisnya. Masing-masing jenis saling kompatibel serta mempunyai kelebihan tersendiri. Misalnya mikrokontroler AT89S52 merupakan padanan dari mikrokontroler 8051. Tabel 2.1. memperlihatkan sebagian dari keluarga MCS 51.

2.6.1. Organisasi Memori

Organisasi memori pada mikrokontroler AT89S52 dapat dibagi menjadi dua bagian besar yaitu memori program dan memori data. Pembagian tersebut didasarkan atas fungsi dari penyimpanan data maupun program. Memori program digunakan untuk menyimpan instruksi-instruksi yang akan dijalankan oleh mikrokontroler, sedangkan memori data digunakan sebagai tempat penyimpanan data yang sedang diolah mikrokontroler.

Program mikrokontroler disimpan dalam memori program berupa ROM. Mikrokontroler AT89S52 dilengkapi dengan ROM internal namun untuk program yang besar digunakan ROM eksternal yang terpisah dari mikrokontroler. Untuk dapat menggunakan memori program eksternal ini penyemat $/\overline{EA}$ dihubungkan dengan penyemat V_{SS} (logika 0).

Memori program mikrokontroler menggunakan alamat 16 *bit* mulai 0000_H-$FFFF_H$, sehingga kapasitas penyimpanan program maksimal adalah 2^{16} *byte* atau 64 Kb. Sinyal yang digunakan untuk membaca memori program eksternal adalah sinyal $/\overline{PSEN}$ ($\overline{\text{Pr}ogramStorageEnable}$).

Selain memori program mikrokontroler AT89S52 juga memiliki memori data internal berkapasitas 128 *byte* dan mampu mengakses memori data eksternal sebesar 64 Kb. Semua memori data internal dapat dialamati dengan pengalamatan langsung atau tidak langsung. Ciri dari pengalamatan langsung adalah *operand* berisi alamat data yang diolah. Sedangkan ciri dari pengalamatan tidak langsung adalah *operand* alamat *register* yang berisi alamat data yang akan diolah. Sebagian memori tersebut dapat dialamati dengan pengalamatan register, dan sebagian lagi dapat dialamati dengan memori satu *bit*. Untuk membaca data digunakan sinyal / RD, sedangkan untuk menulis data digunakan sinyal / WR.

2.6.2. Register Fungsi Khusus

Register fungsi khusus (*Special Function Register*) terletak pada 128 *byte* bagian atas memori data internal dan berisi *register-register* untuk pelayanan *latch port, timer, program status words, control peripheral* dan sebagainya. Alamat register fungsi khusus ditunjukkan pada Tabel 2.4.

Register-register ini hanya dapat diakses dengan pengalamatan langsung. Enam belas alamat pada register fungsi khusus dapat dialamati per*bit* maupun per-*byte* dan terletak pada alamat 80_H-FF_H. Secara perangkat keras, register fungsi khusus ini dibedakan dengan memori data internal.

Tabel 2.3 Nama dan Alamat *Register* pada *Register* Fungsi Khusus

Simbol	Nama Register	Nilai pada saat reset	Alamat
ACC	*Accumulator*	0000_H	$0E0_H$
B	*Register B*	00_H	$0F0_H$

PSW	*Program Status Word*	00_H	$0D0_H$
SP	*Stack Pointer*	07_H	81_H
DPTR	*Data Pointer 2 bytes*		
DPL	*Low bytes*	0000_H	82_H
DPH	*High bytes*	0000_H	83_H
P0	*Port 0*	FF_H	80_H
P1	*Port 1*	FF_H	90_H
P2	*Port 2*	FF_H	$0A0_H$
P3	*Port 3*	FF_H	$0B0_H$
IP	*Interrupt priority control*	$XXX00000_B$	$0B8_H$
IE	*Interupt enable control*	$0XX00000_B$	$0A8_H$
TMOD	*Timer/counter mode control*	00_H	89_H
TCON	*Timer/counter control*	00_H	88_H
TH0	*Timer/counter 0 high byte*	00_H	$8C_H$
TL0	*Timer/counter 0 low byte*	00_H	$8A_H$
TH1	*Timer/counter 1 high byte*	00_H	$8D_H$
TL1	*Timer/counter 1 low byte*	00_H	$8B_H$
SCON	*Serial control*	00_H	98_H
SBUF	*Serial data buffer*	Independen	99_H
PCON	*Power control*	HMOS $0XXXXXXX_B$ CHMOS $0XXX0000_B$	87_H

Sumber : Atmel Data Book ,2006

Beberapa macam register fungsi khusus yang sering digunakan, dijelaskan sebagai berikut :

- *Accumulator* (ACC) merupakan *register* untuk penambahan dan pengurangan. Perintah *Mnemonic* untuk mengakses akumulator disederhanakan sebagai A.
- *Register* B merupakan *register* khusus yang berfungsi melayani operasi perkalian dan pembagian.
- *Program Status Word* (PSW) terdiri dari beberapa *bit* status yang menggambarkan kejadian di akumulator sebelumnya. Yaitu *carry bit*, *auxiliary carry*, dua *bit* pemilih bank, bendera *overflow*, *parity bit*, dan dua bendera yang dapat didefinisikan sendiri oleh pemakai.
- *Stack pointer* (SP) merupakan *register* 8 *bit* yang dapat diletakkan di alamat manapun pada RAM internal. Isi *register* ini ditambah sebelum data disimpan, selama instruksi PUSH dan CALL. Pada saat *reset*, *register* SP diinisialisasi pada alamat 07_H, sehingga *stack* akan dimulai pada lokasi 08_H.
- *Data pointer* (DPTR) terdiri dari dua *register*, yaitu untuk *byte* tinggi (*Data pointer high*, DPH) dan *byte* rendah (*Data pointer Low*, DPL) yang berfungsi untuk mengunci alamat 16 *bit*.
- *Port* 0 sampai *port* 3 merupakan *register* yang berfungsi untuk membaca dan mengeluarkan data pada *port* 0, 1, 2, 3. Masing-masing *register* ini dapat dialamati per-*byte* maupun per-*bit*.

- *Serial data buffer* (SBUF) merupakan dua *register* yang terpisah, *register buffer* pengirim dan sebuah *register buffer* penerima. Meletakkan data pada SBUF berarti meletakkan pada *buffer* pengirim yang akan mengirimkan data melalui transmisi serial. Membaca data SBUF berarti menerima data dari *buffer* penerima.
- *Control register* terdiri dari *register* yang mempunyai fungsi kontrol. Untuk mengontrol sistem interupsi, terdapat dua register khusus, yaitu register IP (*interrupt priority*) dan *register* IE (*interrupt enable*). Untuk mengontrol pelayanan *timer/counter* terdapat *register* khusus, yaitu *register* TCON (*timer/counter control*) serta untuk pelayanan *port* serial menggunakan *register* SCON (*serial port control*)

2.6.3. Port Masukan dan Keluaran

Mikrokontroler AT89S52 mempunyai 4 *port* dan masing-masing *port* terdiri dari 8 saluran *bit*. Ke empat *port* ini bersifat *bidirectional* yaitu dapat digunakan sebagai masukan atau keluaran.

Port 0 digunakan sebagai saluran data yang dimultipleks dengan saluran alamat rendah untuk mengakses memori eksternal, baik memori program maupun memori data. *Port* 2 mengeluarkan bagian alamat tinggi untuk mode pengalamatan memori 16 *bit*. *Port* 1 dan 3 berfungsi sebagai saluran masukan dan keluaran multi fungsi. Jika dibutuhkan *port* 3 mempunyai fungsi khusus seperti ditunjukkan pada Tabel 2.5.

Tabel 2.4 Fungsi khusus *port* 3

Nama Penyemat	Fungsi Khusus
Port 3.0	RxD (*port* masukan serial)
Port 3.1	TxD (*port* keluaran serial)
Port 3.2	/INT0 (masukan interupsi eksternal 0)
Port 3.3	/INT1 (masukan interupsi)
Port 3.4	T0 (masukan pewaktu eksternal 0)
Port 3.5	T1 (masukan pewaktu eksternal 1)
Port 3.6	/WR (sinyal tulis memori data ekstenal)
Port 3.7	/RD (sinyal baca memori data eksternal)

Sumber : Atmel Data Book ,2006

2.6.4. Sistem Interupsi

Mikrokontroler AT89S52 mempunyai dua sumber interupsi eksternal dan sumber interupsi internal yang dapat diprogram agar sensitif terhadap perubahan level atau transisi. Interupsi *timer* aktif saat *register timer* yang bersangkutan mengalami *rollover*, interupsi serial akan aktif pada saat mikrokontroler mengirim/menerima data. Setiap sumber interupsi dapat diaktifkan/dimatikan melalui perangkat lunak.

Tabel 2.5 Tingkatan prioritas interupsi

Prioritas Interupsi	Sumber Interupsi	Alamat Vektor
1	IE0 (Interupsi eksternal 0)	0003_H
2	TF0 (*timer overflow flag* 0)	$000B_H$
3	IE1 (Interupsi eksternal 1)	0013_H
4	TF1 (*timer overflow flag* 1)	$001B_H$
5	R1 dan T1	0023_H
6	TF2 dan EXF2	$002B_H$

Sumber : Atmel Data Book ,2006

Hirarki tingkatan prioritas interupsi dapat dilihat dalam Tabel 2.6. Interupsi yang mempunyai tingkatan prioritas lebih tinggi tidak dapat diinterupsi oleh yang lebih rendah. Meskipun demikian melalui perangkat lunak hirarki tersebut dapat diubah, yaitu dalam *register interrupt priority* (IP).

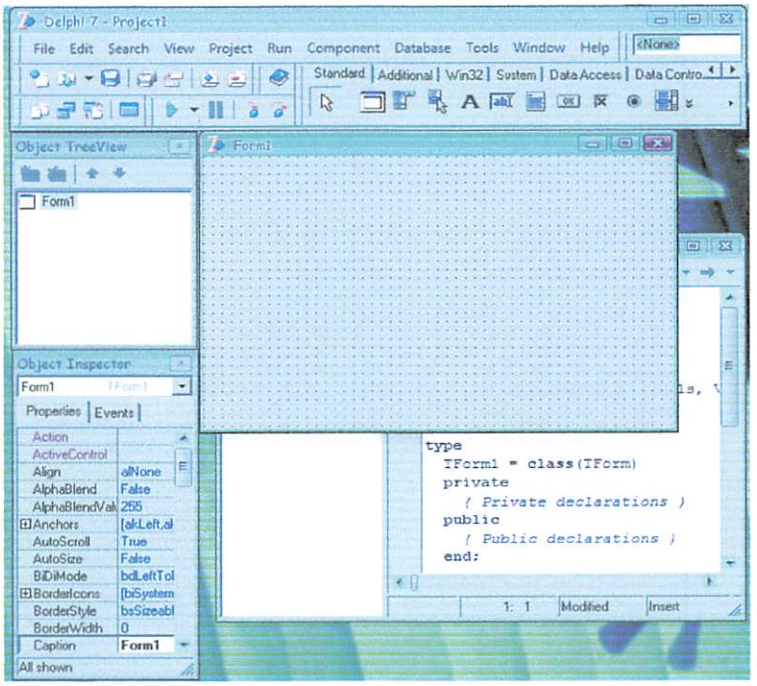
2.7. Pemrograman Delphi

Borland Delphi merupakan suatu bahasa pemograman yang memberikan berbagai fasilitas pembuatan aplikasi visual. Keunggulan bahasa pemograman ini terletak pada produktifitas, kualitas, pengembangan perangkat lunak, kecepatan kompilasi, pola desain yang menarik serta diperkuat dengan program yang terstruktur. Keunggulan lain dari *Delphi* ini adalah dapat digunakan untuk merancang program aplikasi yang memiliki tampilan seperti program aplikasi lain yang berbasis *Window*.Khusus untuk pemrograman *database, Borland Delphi*

menyediakan fasilitas objek yang kuat dan lengkap yang memudahkan *programmer* dalam membuat program. Lingkungan pengembangan terpadu atau *Integrated Development Environment* (IDE) dalam program *Delphi* terbagi menjadi beberapa bagian utama, yaitu :

- *Baris Menu* : Menyediakan menu-menu seperti : File, Edit, Search, View, Project, Run, Componen, Database, dll.
- *ToolBar* : Tombol-tombol pada Toolbar sebenarnya merupakan tombol speed dari menu-menu yang terdapat pada baris menu.
- *Component palette* : disini terdapat beberapa palette, dimana masing-masing palette didalamnya terdapat komponen-komponen yang akan digunakan dalam pembuatan program.
- *Form Designer* : Merupakan tampilan yang akan disajikan pada saat program kita jalankan *(running)*
- *Code Editor* : Pada Code Editor kita dapat menuliskan program yang akan dijalankan oleh komponen dalam form.
- *Object Inspector* : Disini kita dapat melakukan perubahan terhadap propertiesdan even dari komponen-komponen yang kita pergunakan.
- *Object TreeView* : Tree View merupakan daftar dari komponen-komponen apa saja yang telah kita pergunakan dan juga merupakan peta dari program yang kita buat.

Untuk lebih jelasnya perhatikan gambar 2.21.



Gambar 2.21 Lembar Kerja Delphi

Sumber : Abdul kadir, Pemrograman Delphi 7.0, 2006

IDE merupakan sebuah lingkungan dimana semua tombol perintah yang diperlukan untuk mendesain aplikasi, menjalankan dan menguji sebuah aplikasi disajikan dengan baik untuk memudahkan pegembangan program.

- **Struktur Menu Delphi**

  Struktur menu Delphi memberikan tools untuk mengakses lingkungan Delphi.

1. **File :** Menu file adalah menu paling penting dan akan dijabarkan pada bagian berikut:

   - ***New***. Digunakan untuk memulai obyek baru.
   - ***New* Application.** Dengan memilih menu ini, berarti akan membuat *project* baru. Jika belum membuka sebuah *project* atau *object* yang dibuka sudah disimpan ke disk. *Delphi* akan menutup *project* tersebut dan akan membuat *project* baru, termasuk membuat jendela *editor*

baru dengan nama file UNIT.PAS, *form baru* (form 1) dan menampilkan *object inspector*.

- ***New Form.*** Menu ini dipakai untuk membuat form baru.
- ***New frame.*** Untuk membuat frame kosong dan menambahkannya ke dalam *project.*
- ***Open.*** Menyatakan pada *Delphi* bahwa akan dibuka sebuah object dapat berupa sebuah program atau seluruh *project.*
- ***Open Project.*** Untuk membuka sebuah *project.*
- ***Reopen.*** Menu ini dipakai untuk membuka *object favorit* yang sudah pernah dibuka.
- ***Save.*** Menu ini dipakai untuk menyimpan *module* yang sedang aktif.
- ***Save as.*** Dipakai untuk menyimpan *module* dengan nama lain.
- ***Save project as.*** Menu ini dipakai untuk menyimpan project dengan nama baru.
- ***Save all.*** Menyimpan sebuah object yang dibuka.
- ***Close.*** Untuk menutup module program dengan formnya. Jika *module* tersebut belum disimpan, saat menutup maka Delphi akan menanyakan apakah modul tersebut akan disimpan.
- ***Close all.*** Menutup project.
- ***Use* Unit.** Delphi akan menambahkan *klauda uses* pada program yang dibuat. Artinya sebuah unit akan dipakai dalam project.
- ***Print.*** Mencetak item Delphi yang telah dipilih.
- ***Exit.*** Keluar dari aplikasi Delphi.

2. **Edit :** Dipakai untuk menyuting program.
3. **Search :** Dipakai untuk mencari dan mengganti kata-kata pada saat menyuting program.
4. **View :** Dipakai untuk menampilkan atau menyembunyikan jendela-jendela tertentu, misalnya *object inspector, code explorer, debug* dan lain-lain.
5. **Project :** Dipakai untuk mengelola *project. Form dapay* ditambah dan dibuang dari object, mengkompilasi project dan lain-lain.
6. ***Run*** **:** Menu ini dipakai untuk menjalankan program dan memantau jalannya program. Pada saat di run, apabila terjadi salah tulis akan dapat diketahui.
7. **Component :** Dengan menu ini komponen baru dapat ditambah atau diinstal.

2.7.1 Dasar *Pemrograman*

Dasar pemrograman Delphi diantaranya adalah : Variabel *(Intejer, Real, String)*. Percabangan *(If, Than, Else).* Case. Perulangan dengan *For.* Perulangan dengan *Repeat, Until, While, Do.*

- ***Variabel*** **:** Delphi menyediakan banyak sekali variabel, tetapi yang sering kita gunakan adalah :
  - Intejer : Kusus untuk bilangan bulat
  - Real : Bisa ditempati bilangan desimal
  - String : Untuk menyimpan data

- ***Percabangan (If, Than, Else)*** : Percabangan dilakukan dengan cara menguji suatu kondisi, jika kondisi tersbut bernilai benar *(True)* maka proses akan berlanjut ke program setelah *(Then)* tetapi jika kondisi bernilai salah *(false)* maka proses akan berlanjut ke program setelah *(Else)*.

- ***Case*** : Perintah bersarat Case umumnya digunakan untuk kondisi dengan bayak percabangan, syarat percabangan pada bentuk ini hanya boleh melibatkan satu buah parameter dengan tipe data bukan real, pemeriksaan kondisi disini lebih tepat disebutkan dalam hubungan relasi samadengan (=), dengan demikian bila parameter bernilai tertentu maka dilakukan suatu aksi terkait, bila bernilai lain maka dilakukan aksi yang lain juga.

- ***Perulangan dengan For*** : Pada perulangan dengan For, inisialisasi awal dan kondisi akhir ditentukan dengan menggunakan suatu variabel kendaliyang nilainya dibatasi dalam suatu range tertentu.

- **Perulangan dengan *While, Do*** : Pada metode pengulangan ini aksi hanya akan diproses bila kondosi pengulangan dipunuhi, selama kondisi pengulangan bernilai *True* maka aksi akan dilakukan, dan baru akan berhenti setelah kondisi pengulangan bernilai *False,* karena kondisi pengulangan diperiksa pada bagian awal, maka ada kemungkinan aksi tidak pernah dilakukan, yaitu bila kondisi pengulangan tidak pernah bernilai *True*.

- **Perulangan dengan *Repeat, Until* :** Metode pengulangan ini juga melakukan pengulangan berdasarkan pemeriksaan kondisi pengulangan hanya saja natur dari pengulangan ini adalah sistem seakan-akan memaksa untuk melakukan pengulangan, sampai diketahui adanya kondisi berhenti, berlawanan dengan *While*, yang akan memproses aksi hanya bila kondisi pengulangan berkondisi *True*, pada pengulangan *Repeat*, sistem akan memproses aksi selama kondisi berhenti bernilai *False*, dengan dimikian aksi akan selalu diproses (minimal satu kali), pada tipe ini pengulangan dapat terjadi terus-menerus (tidak pernah berhenti) yaitu bila kondisi berhenti tidak pernah bernilai *True*.

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

3.1 Pendahuluan

Dalam bab ini akan dibahas mengenai perencanaan dan pembuatan alat retribusi angkutan kota pada terminal dengan sistem prabayar menggunakan RFID. Pembahasan akan dilakukan pada setiap blok rangkaian, terdiri atas pemilihan komponen, cara kerja masing-masing blok rangkaian, perhitungan dan fungsi masing-masing blok rangkaian tersebut. Secara garis besar terdapat dua bagian perangkat yang ada, yaitu :

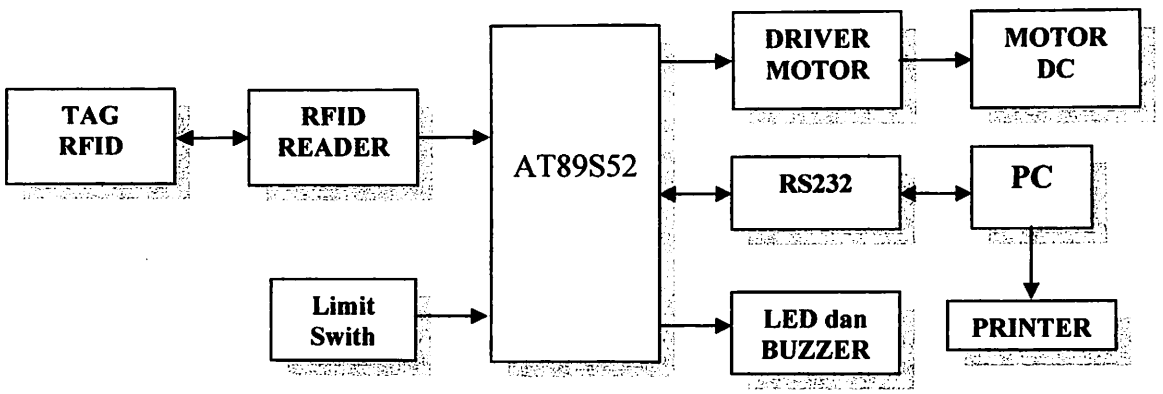
- Perencanaan perangkat keras (*Hardware*)
- Perencanaan perangkat lunak (*Software*)

Pada perencanaan perangkat keras meliputi seluruh peripheral yang digunakan pada sistem ini. Sedangkan pada perencanaan perangkat lunak meliputi *flowchart* dan *software* secara umum. Akan tetapi kedua perangkat tersebut dalam kerjanya akan saling menunjang satu sama lain.

3.2 Perencanaan Perangkat Keras

Dalam Tugas Akhir ini, perancangan dan pembuatan alat retribusi angkutan kota pada termional dengan sistem prabayar menggunakan RFID., dan menggunakan komponen-komponen lain sebagai komponen pendukung.

3.2.1 Blok Diagram Sistem



Gambar 3.1 Blok Diagram Sistem

Dari gambar blok diagram diatas, maka prinsip kerja dari Indikator Angkutan Kota ini adalah sebagai berikut :

- **Tag RFID** : Menyimpan kode ASCII dengan nilai tertentu Berfungsi sebagai proses identifikasi seseorang atau objek yang mampu menampilkan identitas kendaraan yang tersimpan di data *base computer.*
- **RFID Reader** : Menerima kode ASCII dari Tag RFID untuk dikirimkan ke mikrokontroller.
- **Mikrokontroller AT89S52** : Menterjemahkan data ASCII dari reader dan mengirimkannya ke PC. Selain itu menerima perintah dari PC untuk menggerakkan motor DC.
- **Driver Motor** : mengendalikan motor DC. Dengan membaca perintah dari mikrokontroller.
- **Motor DC** : Berfungsi sebagai penggerak mekanik naik turunnya pintu portal.

- **Limit Swith** : Rangkaian ini menggunakan 2 buah *limit switch* yang berfungsi untuk membatasi putaran motor DC agar tidak melebihi putaran yang kita inginkan
- **RS232** : Sebagai interface dari mikrokontroller ke PC atau sebalinya.
- **PC** : Sebagai database angkutan kota, memerintahkan mikrokontroller untuk menggerakkan pembatas dan mengolah data untuk dicetak melalui printer.

  Database tersebut berisi :

  - Nama terminal
  - Hari, tgl/bln/th dan waktu
  - Nama dan alamat pemilik kendaraan
  - Jenis angkutan kota (sesuai dengan jurusan kota)
  - Nomor polisi kendaraan
  - Besar retribusi
  - Sisa saldo
  - Berapa kali beroperasi dalam sehari.
- **Printer** : untuk mencetak data yang diperintahkan oleh PC.

  Data yang dicetak oleh printer :

  - Nama terminal
  - Hari, tgl/bln/th dan waktu
  - Jenis angkutan
  - Nomor polisi
  - Besar retribusi
  - Sisa saldo

- Keterangan

- Laporan yang bisa dihasilkan sistem :
  - Laporan dapat dijadikan data pembuatan statistik mengenai jumlah angkot yang beroperasi setiap harinya yang nantinya dapat digunakan untuk menentukan perlu tidaknya dilakukan penambahan armada angkot pada suatu jurusan seiring dengan perkembangan tata letak wilayah kota.
  - Laporan yang dihasilkan lebih rapi dan jelas.
  - Kemungkinan kesalahan penyimpanan data kecil sekali karena data yang disimpan tidak dilakukan secara manual.
  - Waktu yang dibutuhkan untuk mendata angkot lebih cepat karena data langsung dimasukkan ke database.

3.2.2 Prinsip Kerja

RFID reader memancarkan frequensi 125 Khz kemudian sinyal ini diterima oleh RFID tag, Tag akan terdeteksi dan mentransmisikan informasi pada Tag kepada reader. Informasi tersebut dikirimkan ke mikrokontroller untuk diterjemahkan, kemudian dari mikrokontroller dikirimkan ke PC untuk dicocokkan dengan database. Jika data tersebut valid maka PC akan mengupdate database dan mencetak hasilnya melalui printer. Selain itu PC memerintahkan mikrokontroller untuk menggerakkan motor DC.

3.2.3 RFID

RFID merupakan suatu alat yang dapat digunakan untuk mengidentifikasi suatu barang / benda, pada skripsi ini digunakan *RFID card* dan *RFID reader*.

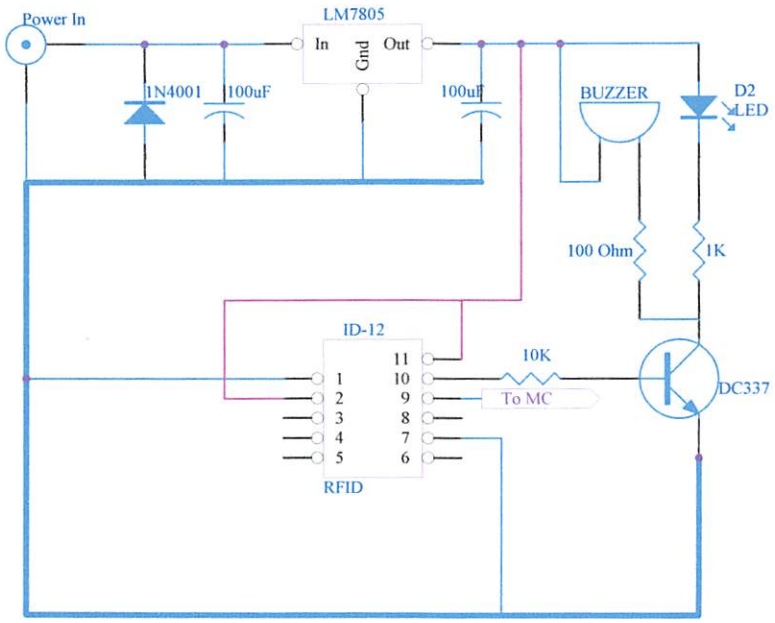
3.2.3.1 RFID Card

RFID card chip yang di dalamnya juga terdapat nomor identitas kartu atau nomor seri kartu yang nantinya nomor tersebut akan diambil oleh reader kartu saat chip dari kartu tersebut dibaca oleh reader kartu, dimana keluaran nomor seri tersebut sudah berupa ASCII dan itu tergantung dari konfigurasi rangkaian reader kartunya.

Gambar 3.2 RFID Card
Sumber: www.digiware.com, RFID

3.2.3.2 RFID Reader

RFID disini yang kita gunakan adalah jenis ID-12, ID-12 yang kita gunakan mempunyai jarak baca maksimal 12 cm. Sesuai dengan data sheet dari reader kartu ID-12 itu sendiri, untuk memperoleh keluaran yang berbentuk ASCII maka reader itu disusun seperti gambar 3.3 di bawah ini :



Gambar 3.3 Perencanaan rangkaian *Reader*

- Perencanaan Rangkaian Switching dengan transistor bipolar

1. Parameter input dan output :

   Input

   - Pulsa TTL dengan frekuensi 3,1KHz dari pin.8 Module ID 12. *Output*
   - Beban Buzzer dengan resistansi 100Ω tegangan 5 volt
   - Beban LED dengan spesifikasi I_{LED} = 4mA.
   - kedua beban didrive oleh satu transistor bipolar.

2. ***Perencanaan Kondisi Saturasi***

   - Perencanaan Parameter Output:
     - Agar buzzer tidak mudah rusak maka diberi tegangan bias sebesar 2,5 volt, karena buzzer dipicu oleh pulsa 3,1KHz sehingga terjadi pendisipasian daya ke beban.

$$I_{bz} = \frac{V_{bz}}{R_{bz}}$$

$$= \frac{2,5}{100} = 25\ mA$$

$$R_{bz} = \frac{V_s - V_{bz}}{I_{bz}}$$

$$= \frac{5-2,5}{25x100^{-3}} = 100\ \Omega$$

- $I_{LED} = 4mA$, karena tegangan inputnya sebesar 5 volt maka diperlukan resistor untuk mendropkan tegangan sebesar

$$R_{Led} = \frac{V}{I_{Led}}$$

$$= \frac{5}{4x10^{-3}} = 1,25\ K\Omega$$

- Sehingga Arus total sebesar

  $I_T = I_{Led} + I_{bz} = 4 + 25 = 29mA$

- Dalam pemilihan transistor dicari karakteristik I_{Cmin} = 29mA dengan Disipasi daya minimal P_D = 145mW.

- Perencanaan Parameter Input :

$$Ib = \frac{I_C}{Hfe}$$

$$= \frac{29}{60} = 0,48mA$$

$$Rb = \frac{V_{in} - V_{be}}{Ib}$$

$$= \frac{5 - 0,7}{0,49} = 8,89K\Omega$$

Dari gambar 3.3 rangkaian Reader di atas maka dapat kita ketahui pin kaki yang kita pakai adalah pin 1 (GND), pin 3 dan 4 merupakan antenna, pin 11 Vcc, pin 10 untuk keluaran buser, antara D0 dan D1 kita menggunakan D0 karena disini reader mengirim data serial ke mikrokontroler, kemudian pin 2 (reset bar) kita sambung dengan Vcc dan pin 5 (format selektor) dihubungkan ke ground karena untuk mendapatkan keluaran berupa ASCII saat reader kartu membaca data dari kartu.

3.2.4 Mikrokontroller AT89S52

Disini rangkaian mikrokontroller AT89S52 berfungsi sebagai pengelolah data dan pengendali alat, agar dapat melakukan prosesnya harus didukung oleh beberapa komponen tambahan, yaitu berupa rangkaian *clock*. Dan *reset*.

3.2.4.1 Perancangan Minimum Sistem AT89S52

Penggunaan port Mikrokontroller AT89S52 pada rancangan ini adalah sebagai berikut :

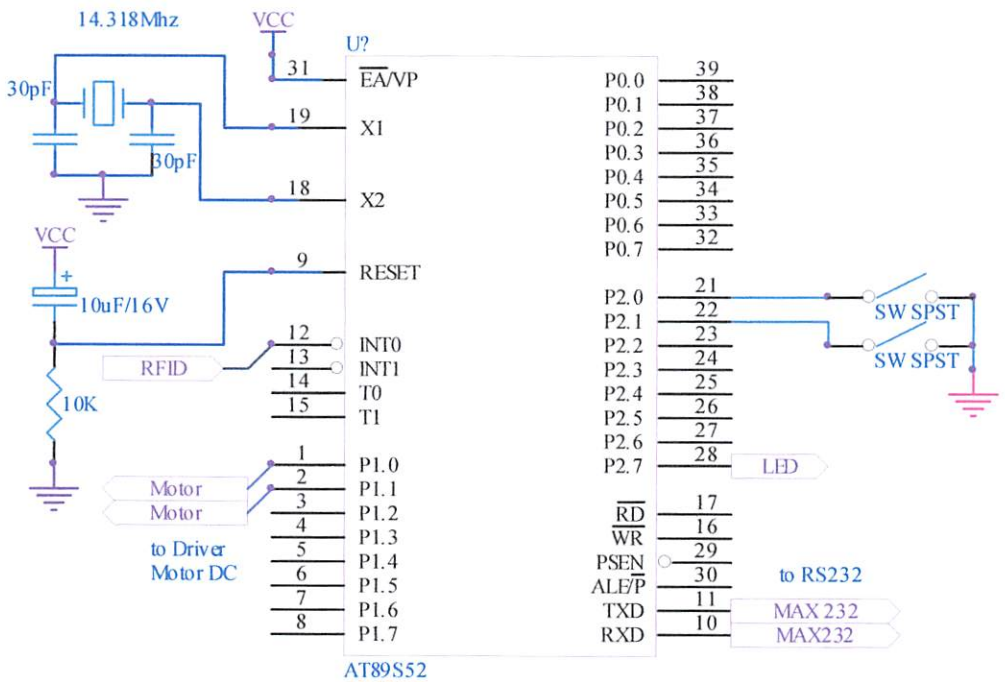
a. P1.0 sampai P1.1 : Menggerakkan Motor
b. P3.0 sampai P3.1 : RS 232
c. P2.7 : Led
d. P2.0 Sampai P2.1 : Dihubungakan ke Limit Swit
e. Pin 18 (XTAL 1) : Sebagai pembangkit oscilator (*clock)* XTAL 1
h. Pin 19 (XTAL 2) : Sebagai Pembangkit Oscilator (*clock*) XTAL 2
j. Pin 31 dan Pin 40 : Dihubungkan ke Vcc

k. Pin 9 : Berfungsi sebagai reset

l. Pin 20 : Dihubungkan ke Ground

m. Pin 3.2 : Dihubungkan ke Reader

Digunakannya mikrokontroller AT89S52, mengingat mikrokontroller ini telah menyediakan internal memori, maka tidak perlu menambah memori luar, sehingga bentuk fisik alat lebih kecil serta memudahkan dalam pemrograman. Dan mikrokontroller ini sangat mudah didapat dipasar dengan harga yang relatif murah.



Gambar 3.4 Minimum Sistem Mikrokontroler AT89S52

3.2.4.2 Rangkaian Clock Internal

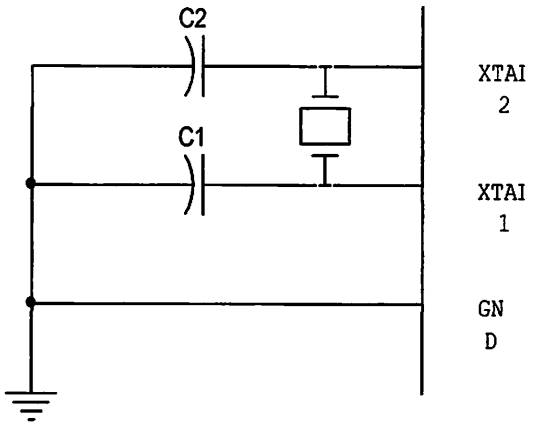
Mikrokontroller AT89S52 ini memiliki internal clock generator yang berfungsi sebagai sumber clock, tetapi masih diperlukan rangkaian tambahan

untuk membangkitkan clock tersebut. Rangkaian ini terdiri dari 2 buah kapasitor dan sebuah kristal dengan ketentuan:

C1 dan C2 = 20 pF – 40 pF untuk kristal

= 30 pF – 50 pF untuk keramik resonator

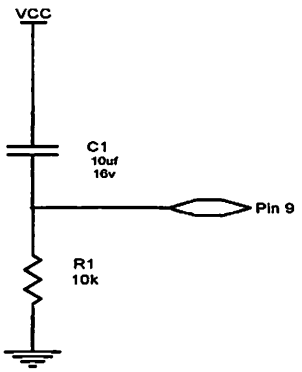
Dalam perencanaan rangkaian mikrokontroller ini digunakan kapasitor sebesar 30 pF.



Gambar 3.5 Rangkaian Clock Internal

3.2.4.3 Rangkaian Reset

Rangkaian Reset digunakan untuk mereset atau mengembalikan ke keadaan awal dari mikrokontroller AT89S52. Rangkaian ini dipergunakan untuk mereset mikrokontroller pada keadaan pertama kali saat power diaktifkan atau disebut Power On Reset. Untuk menjalankan reset maka pin reset (pin no.9) pada mikrokontroller diberi sinyal high (1), rangkaian reset ini dapat ditunjukkan pada gambar berikut:



Gambar 3.6 Rangkaian Reset

Rangkaian *reset* bertujuan agar mikrokontroller dapat menjalankan proses dari awal. Rangkaian *reset* untuk mikrokontroller dirancang agar mempunyai kemampuan *power on reset*, yaitu *reset* yang terjadi pada saat sistem dinyalakan untuk pertama kalinya.

Rangkaian *reset* terbentuk oleh komponen Resistor dan kapasitor yang sudah baku. Nilai Resistor yang dipakai adalah 10 KΩ dan kapasitor 10 μF. Besar nilai tahanan dan kapasitor pada rangkaian tersebut akan menentukan lama waktu pulsa *reset.*

Sedangkan untuk mencari frekuensi dari *reset* tersebut dengan mengunakan rumus sebagai berikut:

$$F_0 = \frac{1}{1{,}1 * R_1 * C_1}$$

Sehingga dengan komponen resistor dengan nilai 10 KΩ dan kapasitor dengan nilai 10 μF akan dihasilkan frekuensi:

$$F_0 = \frac{1}{1{,}1 * 10^3 * 10^{-6}}$$

$$F_0 = 9{,}09Hz$$

Maka periode $clock = \frac{1}{F}$

$$T = \frac{1}{9{,}09} = 0{,}11 \det ik$$

Dengan frekuensi yang dihasilkan serta periode *clock* yang telah diketahui maka akan memberikan siklus ke mikrokontroler untuk bekerja terus sampai *power on reset.*

3.2.5 Komunikasi Serial

Pada perancangan alat ini digunakan komunikasi serial untuk menghubungkan PC dengan alat .Pada port serial data dalam format serial sehingga data dapat dikirim dengan cepat.Untuk melakukan proses komunikasi data antara mikrokontroler dengan computer diperlukan syarat – syarat yang harus dipenuhi ,yaitu pengaturan *baud rate* serta pengaturan secara perangkat lunak. Jenis data yang akan dikirim adalah dalam bentuk data biner (bit per bit transfer) dengan satuan *baud rate* untuk kecepatan proses transfernya.

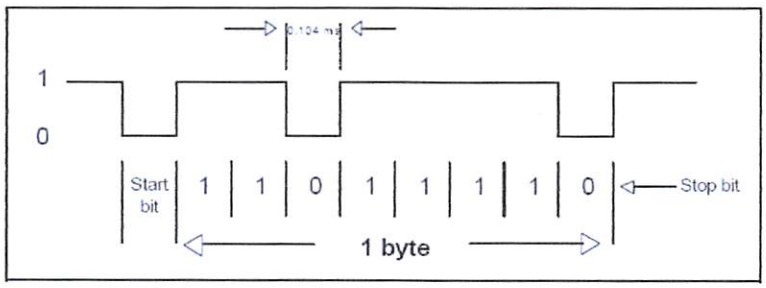
Serial control (SCON) merupakan register khusus pengontrol kerja port serial, diset untuk mentransmisikan 8 bit data UART (*Universal Asynchronous Reciever Transmitter)* yang merupakan standar komunikasi data dengan *baud rate* yang dapat diatur *(variable)* .Penentuan *baud rate* tersebut dilakukan dengan pengesetan

timer/counter 1 high bit (TH1) .Bila diinginkan *baud rate* 9600 bps maka :

$$\text{Baud Rate} = \frac{2^{SMOD} \text{ x Frekuensi Osilator}}{32 \text{ x } 12[256 - TH1]}$$

$$9600 = \frac{2^0 x 11{,}059.10^6}{32x12[256 - TH1]}$$

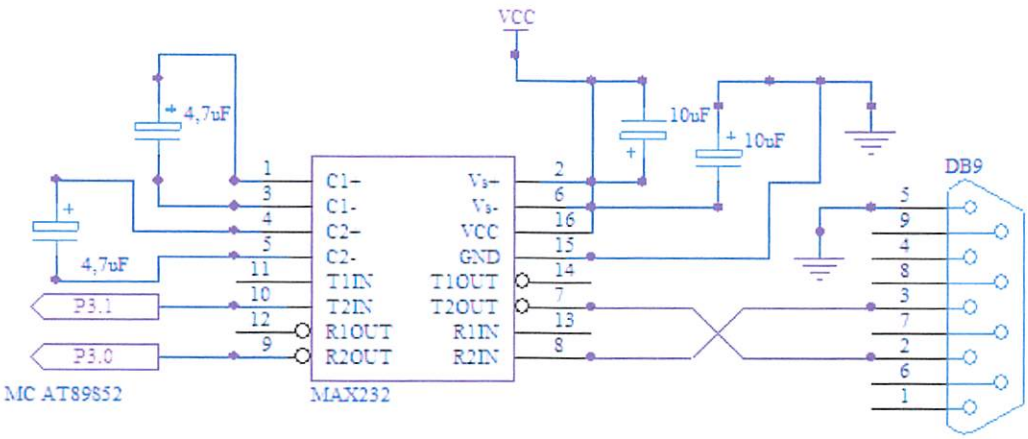
TH1 = 0FDh = 253 d



Gambar 3.7 Format Standar Transmisi Data Asinkron

Jika *baud rate* besarnya adalah 9600 bps bit data membutuhkan waktu 0,104 mili detik .Karena 1 karakter terdiri atas 10 bit atau 10 x 0,104 = 1,04 mili detik, kecepatan transfer adalah 1/1,04 mili detik atau 0,96 karakter/ mili detik.

Antarmuka serial menggunakan IC MAX 232, konfigurasi IC 232 seperti terlihat pada gambar dibawah. Pin nomor 9 berfungsi untuk mengirim data ketujuan. Pin nomor 10 berfungsi untuk meneruskan data yang diterima ketujuan.



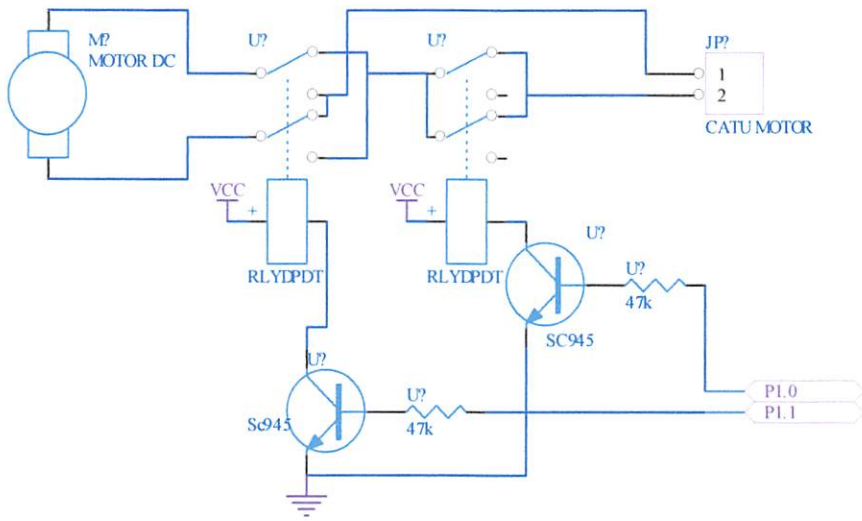
Gambar 3.8 Rangkaian RS 232

Saluran data pada *port* serial *PC* menggunakan standard RS232, dimana logic 0 (*low*) dinyatakan sebagai tegangan antara +3 *Volt* sampai +15 *Volt* dan logic 1 (*high*) dinyatakan sebagai tegangan antara -3 *Volt* sampai -15 *Volt*. Level tegangan ini tidak sesuai dengan level tegangan yang dipakai pada *port* seri AT89S52 yang menggunakan standard *TTL (Transistor Transistor Logic)*, yaitu level tegangan baku dalam rangkaian-rangkaian digital.

Dalam standar *TTL* logic 0 (*low*) dinyatakan sebagai tegangan antara 0 *Volt* sampai 0.8 *Volt*, dan logic 1 (*high*) dinyatakan sebagai tegangan antara 2.5 *Volt* sampai 5 *Volt*. Karena perbedaan tegangan tersebut, agar *port* seri PC tidak merusak *port* seri AT89S52 antara keduanya dipasangkan IC MAX232 sebagai penyesuai tegangan.

3.2.6 Perencanaan Rangkaian *Driver*

Rangkaian *driver* berfungsi untuk menggerakkan relay yang terhubung dengan motor dan palang (pembatas). Tegangan keluaran dari mikrokontroler dihubungkan dengan *driver* motor sehingga motor akan aktif dan dapat menggerakkan palang (pembatas). Selain itu tegangan keluaran dari mikrokontroler juga dihubungkan dengan rangkaian limit swit, rangkaian ini menggunakan 2 buah *limit switch* yang berfungsi untuk membatasi putaran motor DC agar tidak melebihi putaran yang kita inginkan (motor tidak bergerak terus). Rangkaian *driver* motor ditunjukkan dalam Gambar beriikut.



Gambar 3.9 Rangkaian *Driver* Motor

Perhitungan dari rangkaian *driver* motor pada Gambar 3.9 diatas terlebih dahulu harus dicari nilai *Rb*. Data yang diperlukan untuk mencari besar resisitansi *Rb* adalah sebagai berikut:

Data transistor SC945 yang diperoleh dari *datasheet* adalah :

- Besar pengukuran tahanan dalam relay($R_{relay} = R_c$) = 400Ω.
- V_{ce} saturasi = 0,3 volt.
- H_{fe} = 100
- V_{be} = 0,7 volt.

Dengan resistansi relay sebesar 400 Ω, tegangan catu sebesar $5V$, dan V_{ce} saturasi sebesar $0,3V$ besar arus I_{relay} adalah :

$$I_{relay} = \frac{Vcc - Vce}{R_{relay}}$$

$$= \frac{5v - 0,3v}{400\Omega} = 0,0117 \text{ A}$$

Dalam perancangan ini digunakan β 100 dengan pertimbangan transistor tetap mampu menggerakkan relay meskipun penguatan yang dipakai adalah minimum.

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

$$I_B = \frac{0{,}0117A}{100} = 0{,}117 \text{ mA}$$

Jika V_{bb} adalah tegangan keluaran dari mikrokontroler yaitu sebesar 5V dan $Vbe = 0{,}7\ V$ maka besar resistansi Rb adalah :

$$R_B = \frac{V_{bb} - V_{be}}{I_B}$$

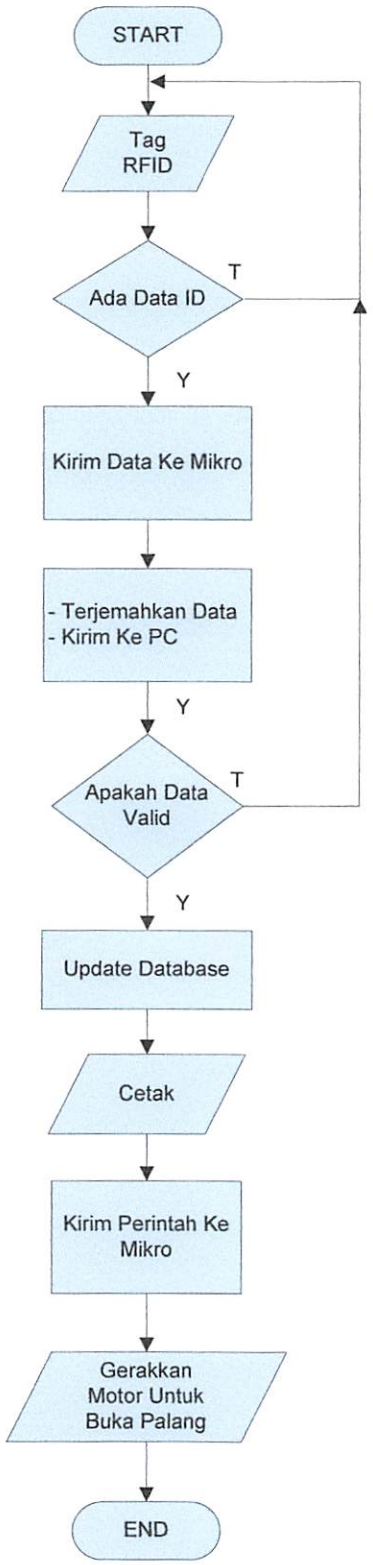
$$= \frac{5v - 0{,}7v}{0{,}117mA} = 36.7 \text{ K}\Omega$$

Maka diperoleh nilai $R_B = 36.7$ KΩ

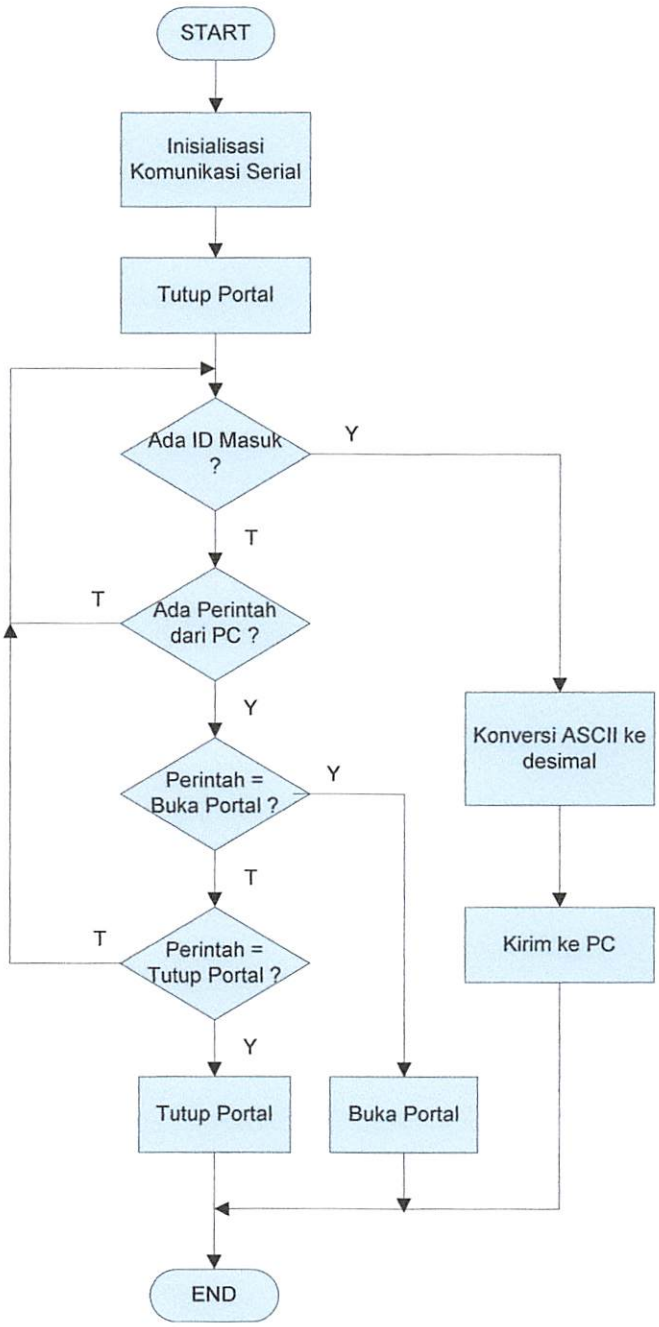
3.3 Perencanaan Perangkat Lunak

Setelah semua perangkat keras telah selesai dikerjakan pada tahap selanjutnya adalah pembuatan perangkat lunak (software) yang akan menangani sistem rangkaian. Pada perangkat lunak inilah kita dapat menentukan bagaimana sistem rangkaian ini akan bekerja, pada bagian inilah tata kerja rangkaian ditentukan.

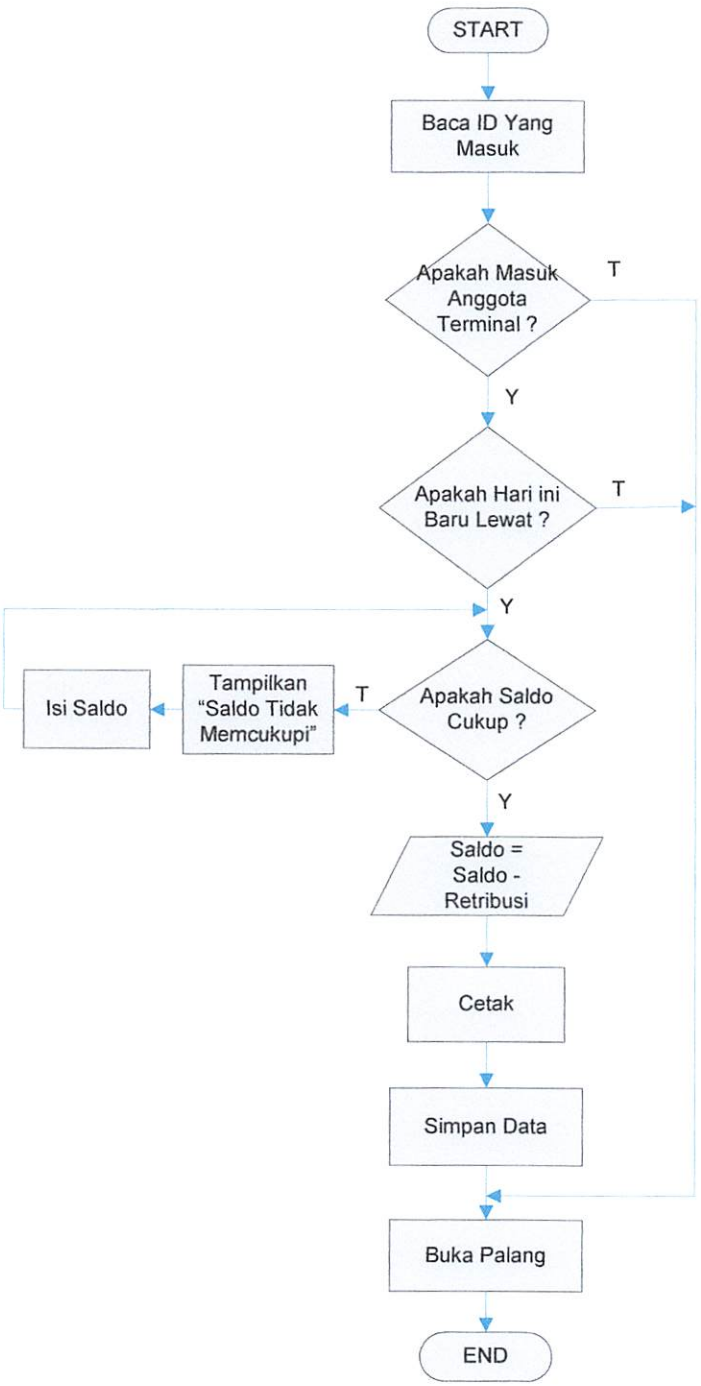
3.3.1 Perencanaan Diagram Alir (Flowchart) Keseluruhan Sistem



3.3.2 Flowchart Software Pada Mikrokontroler



3.3.3 Flowchart Software Pada Komputer



BAB IV

PENGUJIAN ALAT

Untuk mendapatkan hasil yang maksimal setelah melaksanakan perencanaan dan pembuatan alat, maka perlu dilakukan suatu pengujian terhadap alat yang telah dibuat. Pengujian ini bertujuan untuk mengetahui apakah alat yang telah dibuat telah dapat bekerja sesuai dengan perencanaan.

Bagian-bagian yang diuji dari peralatan ini adalah :

1. Rangkaian RFID
2. Rangkaian Driver
3. Pengujian Limit Switch
4. Keseluruhan Sistem

4.1 Pengujian Tag RFID dan Reader RFID

4.1.1 Tujuan

Untuk mengetahui apakah RFID *reader* dapat membaca Tag (kartu) RFID atau tidak.

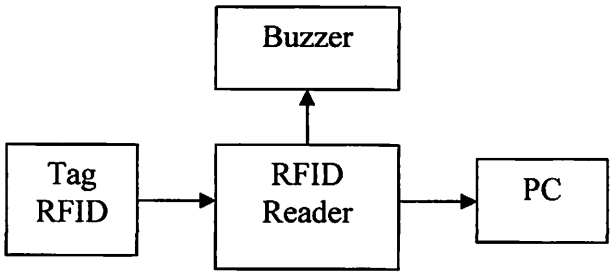
4.1.2 Peralatan Yang Dibutuhkan

1. Tag (kartu) RFID dan RFID *reader.*
2. *Buzzer.*
3. Personal Komputer (PC)

4.1.3 Prosedur Pengujian

1. Menyusun rangkaian seperti gambar 4.1
2. Menempelkan tag pada RFID *reader.*

3. Mengamati jarak antara *Tag RFID* Dengan *RFID Reader* .



Gambar 4.1 Diagram Blok Pengujian *Tag* Terhadap *reader* RFID

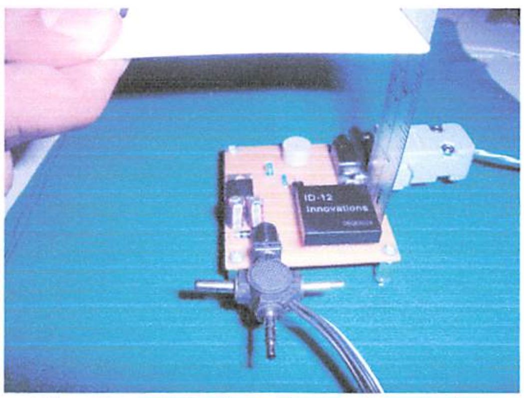
4.1.4 Hasil Pengujian dan Analisis

Hasil Pengujian menunjukkan bahwa RFID *reader* dapat membaca tag (kartu) RFID atau tag dapat diakses oleh *reader* sampai sejauh kurang lebih 7 cm. Dengan demikian RFID yang digunakan sebagai kartu identitas angkutan kota telah bekerja dengan baik.

Tabel 4.1 Pengujian RFID

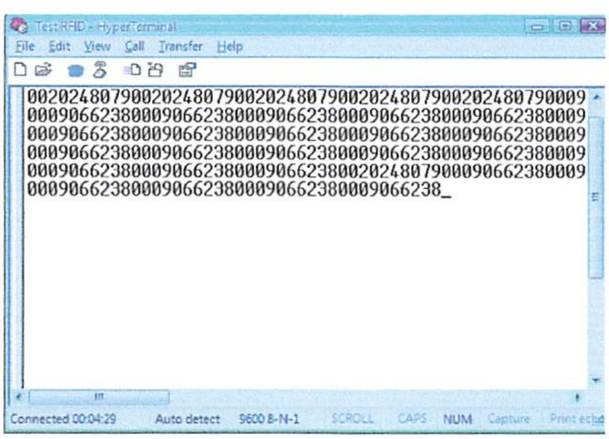
Jarak jangkauan	**Tingkat pengujian**	**Tingkat Keberhasilan**
1 cm	1 kali	Berhasil
	2 kali	Berhasil
2 cm	1 kali	Berhasil
	2 kali	Berhasil
3 cm	1 kali	Berhasil
	2 kali	Berhasil
4 cm	1 kali	Berhasil
	2 kali	Berhasil
5 cm	1 kali	Berhasil
	2 kali	Berhasil

6 cm	1 kali 2 kali	Berhasil Berhasil
7 cm	1 kali 2 kali	Berhasil Berhasil
8 cm	1 kali 2 kali	Tidak Berhasil Tidak Berhasil



Gambar 4.2 Foto Pengujian RFID

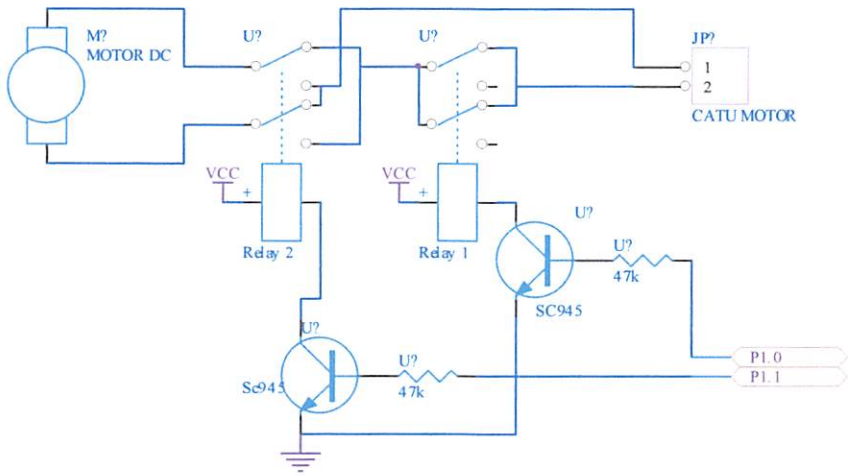
Dalam melakukan pengujian RFID pada PC, data diterima dan ditampilkan menggunakan Hyper Terminal. Hasil penerimaan data pada Hyper Terminal ditunjukkan pada gambar 4.3 berikut :



Gambar 4.3 Hasil Pembacaan RFID Reader Pada Hyper Terminal

4.2 Pengujian rangkaian *driver*

Blok pengujian rangkaian driver relay untuk motor ditunjukkan seperti dalam Gambar 4.4.



Gambar 4.4 Pengujian Rangkaian *Driver*

4.2.1 Tujuan

Pengujian ini bertujuan untuk mengetahui apakah motor bekerja dengan baik sesuai dengan yang direncanakan.

4.2.2 Langkah pengujiannya adalah sebagai berikut :

1. Menyusun rangkaian seperti gambar 4.4 diatas.
2. Memberikan catu daya pada rangkaian *driver* motor.
3. Mengamati kondisi relay dan putaran motor.
4. Hasil Pengujian rangkaian driver relay ditunjukkan dalam Tabel 4.2

4.2.3 Hasil Pengujian

Tabel 4.2 Hasil Pengujian Rangkaian *Driver* relay

Port 1.0	Port 1.1	Kondisi Relay 1	Kondisi Relay 2	Kondisi Motor
0,02	0,02	Relay on	Relay on	Kanan
0,02	4,58	Relay on	Relay on	Kiri
4,58	0,02	Relay off	Relay on	Off
4,58	4,58	Relay off	Relay on	Off

4.2.4 Analisis hasil pengujian

Dari Tabel 4.2 terlihat bahwa jika driver relay 1 berlogika high ("4,58") maka kondisi relay 1 off, demikian sebaliknya jika driver relay 1 berlogika low ("0,02") maka kondisi relay 1 on, hal ini menunjukkan bahwa relay 1 berfungsi sebagai pemutus atau penghubung catu daya motor. Dan jika driver relay 2 berlogika high atau low ("4,58"/"0,02") maka kondisi relay 2 tetap on, hanya merubah tegangan antara ('–') dan ('+'), hal ini menunjukkan bahwa relay 2 berfungsi sebagai pengatur arah putar motor.

4.3 Pengujian *Limit Switch*

4.3.1 Tujuan

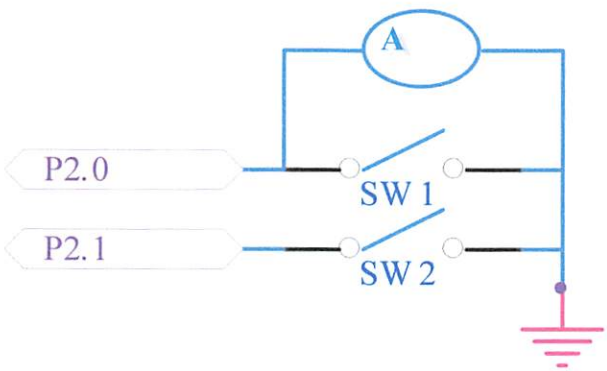
Untuk mengetahui rangkaian *limit switch* ini apakah dapat berfungsi sebagai penghubung atau pemutus logika *low* atau "0" sebagai inputan mikrokontroller.

4.3.2 Peralatan yang digunakan

1. Multimeter digital.
2. Rangkaian *limit switch.*

4.3.3 Prosedur Pengujian

1. Membuat rangkaian seperti pada gambar 4.5.
2. Pengukuran dilakukan dalam dua kondisi yaitu saat tidak ada penekanan dan ada penekanan.
3. Memasukkan hasil pengukuran pada tabel 4.3.



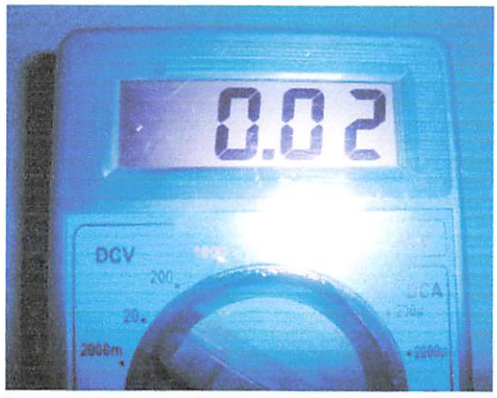
Gambar 4.5 Pengujian *Limit Switch*

4.3.4 Hasil Pengujian

Tabel 4-3 Hasil Pengujian Rangkaian *Limit Switch*

Kondisi	*Limit Switch*	
	Switch	Tegangan (volt)
Ada Penekanan	***Close***	**0,02 volt**
Tidak Ada Penekanan	***Open***	**4,58 volt**

Gambar 4.6 Foto Pengujian Tegangan Switch Pada Saat Tidak Ada Penekanan



Gambar 4.7 Foto Pengujian Tegangan Switch Pada Saat Ada Penekanan

4.3.5 Analisa Hasil Pengujian

Dari hasil pengujian rangkaian *limit switch* pada tabel diatas, maka dapat diketahui bahwa pada saat ada penekanan saklar dikondisikan sebagai logika *low* atau"0" sehingga inputan mikrokontroller tidak ada, sebaliknya pada saat kondisi tidak ada penekanan maka saklar akan menutup, sehingga logika *high* atau "1" dapat terhubung sebagai inputan mikrokontroller.

4.4 Pengujian Keseluruhan Sistem

4.4.1 Tujuan

Untuk mengetahui kinerja sistem secara keseluruhan berdasarkan perancangan yang telah dibuat, maka dilakukan pengujian sistem secara keseluruhan.

4.4.2 Peralatan yang digunakan

1. Personal Komputer (PC).
2. Kabel serial.
3. Tag RFID.
4. Rangkaian keseluruhan sistem.
5. Mekanik Miniatur.

4.4.3 Prosedur Pengujian

1. Menggabungkan seluruh rangkaian dengan PC.
2. Menghidupkan catu daya.
3. Membuka program Delphi yang telah dibuat.
4. Cek pada *form* tampilan utama apakah seluruh rangkaian dalam status *connect* atau *not connect*. dengan cara mendekatkan tag RFID pada reader.
5. Menekan tombol *buka palang* dan mengamati apakah palang membuka atau tidak.
6. Menekan tombol *tutup palang* dan mengamati apakah palang menutup atau tidak.

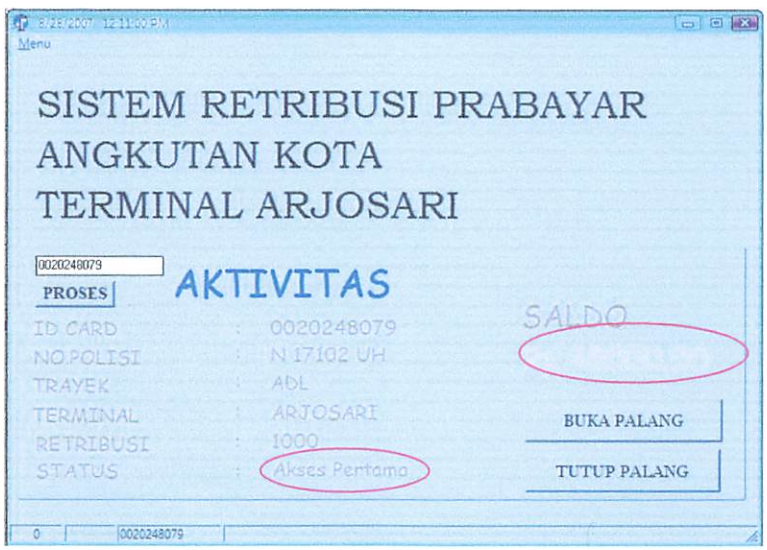
4.4.4 Hasil Pengujian Dan Analisa

Pengujian keseluruhan system dilakukan dengan mengunakan 2 RFID Tag yang berbeda. Setiap RFID Tag akan dibaca oleh reader sebanyak 5 kali.

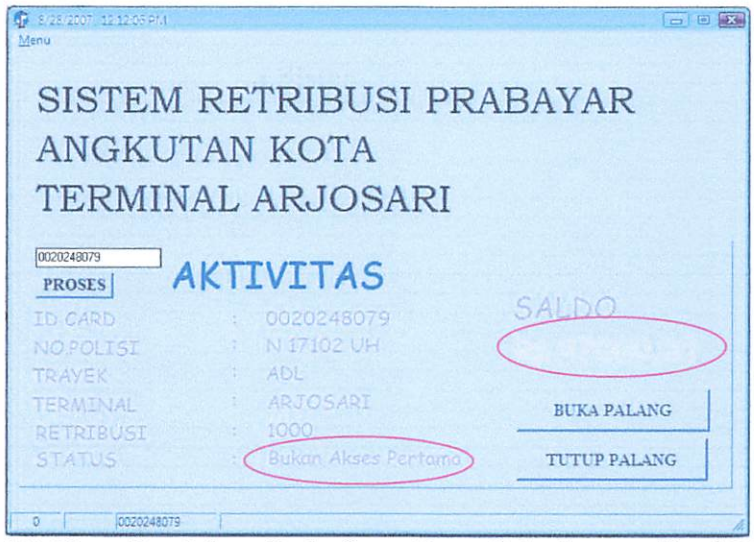
- Pengujian pertama Menggunakan Tag RFID dengan kode 0020248079

Tabel 4.4 Hasil Pengujian Pertama Keseluruha Sistem

No.	ID	Saldo
1	0020248079	Dikurangi Rp. 1000
2	0020248079	Tetap
3	0020248079	Tetap
4	0020248079	Tetap
5	0020248079	Tetap



Gambar 4.8 Tampilan Menu Pada Saat Akses Pertama Kali

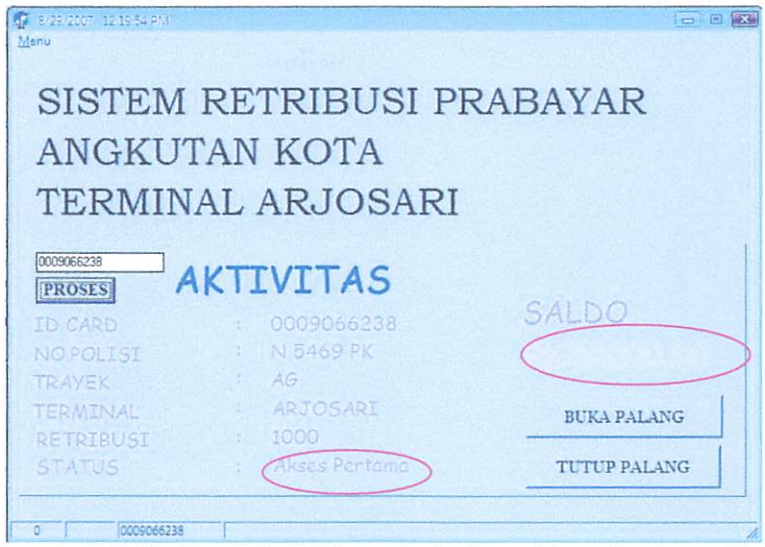


Gambar 4.9 Tampilan Menu Pada Saat Akses Selanjutnya dihari Yang Sama

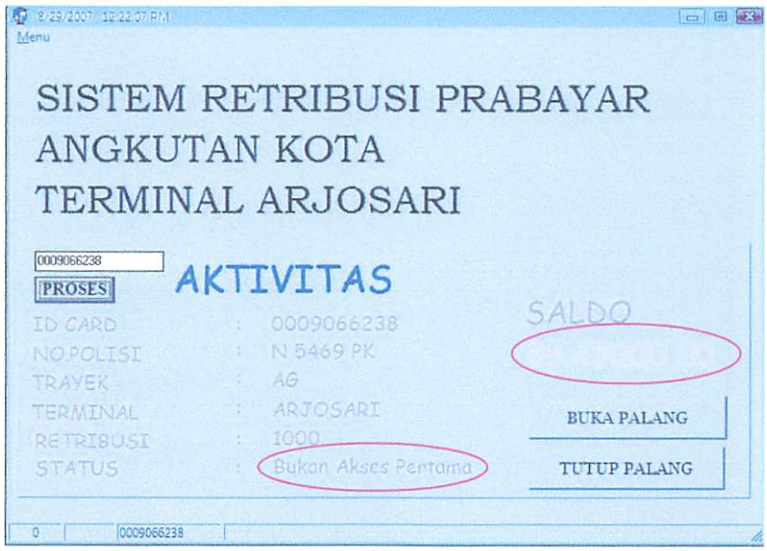
- Pengujian kedua menggunakan Tag RFID dengan kode 0009066238

Tabel 4.5 Hasil Pengujian Kedua Keseluruha Sistem

No.	ID	Saldo
1	0009066238	Dikurangi Rp. 1000
2	0009066238	Tetap
3	0009066238	Tetap
4	0009066238	Tetap
5	0009066238	Tetap



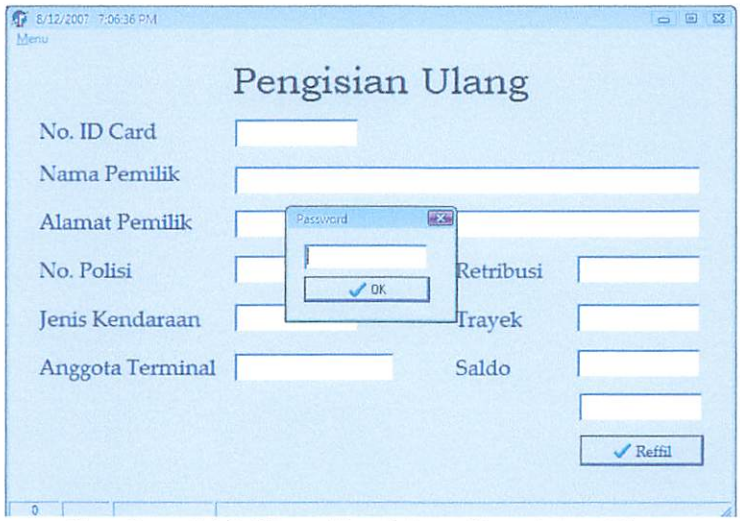
Gambar 4.10 Tampilan Menu Pada Saat Akses Pertama Kali



Gambar 4.11 Tampilan Menu Pada Saat Akses Selanjutnya dihari Yang Sama

Pada waktu akses pertama computer akan mencetak data tersebut melalui printer dan akan mengurangi saldo sesuai dengan retribusi yang telah ditentukan, dan jika akses selanjutnya program akan menampilkan '*bukan akses pertama*' jadi data tida di cetak dan saldo tidak dikurangi, karena retribusi dilakukan sekali dalam sehari.

Dari hasil pengujian keseluruhan sistem, maka dapat disimpulkan bahwa Sistem Retribusi Angkutan Kota Pada Terminal dengan Sistem Prabayar Menggunakan RFID ini dapat bekerja dengan baik sesuai dengan perencanaan dan perancangan.

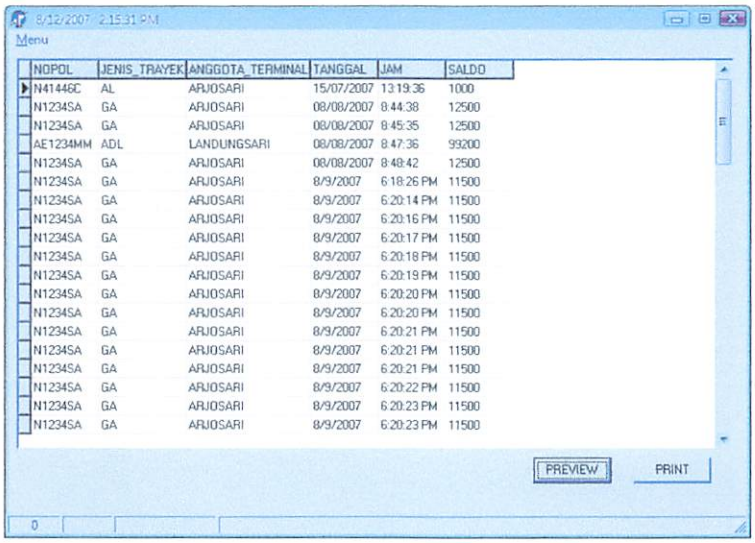


Gambar 4.12 Tampilan Menu Pengisian Ulang

Tampilan diatas ditunjukkan apabila kita mengisi ulang saldo yang nantinya digunakan sebagai retribusi. Sebelumnya kita harus memasukan password terlebih dahulu yang kemudian dilanjutkan dengan memasukkan no ID card dan memasukkan jumlah saldo yang akan ditambahkan.

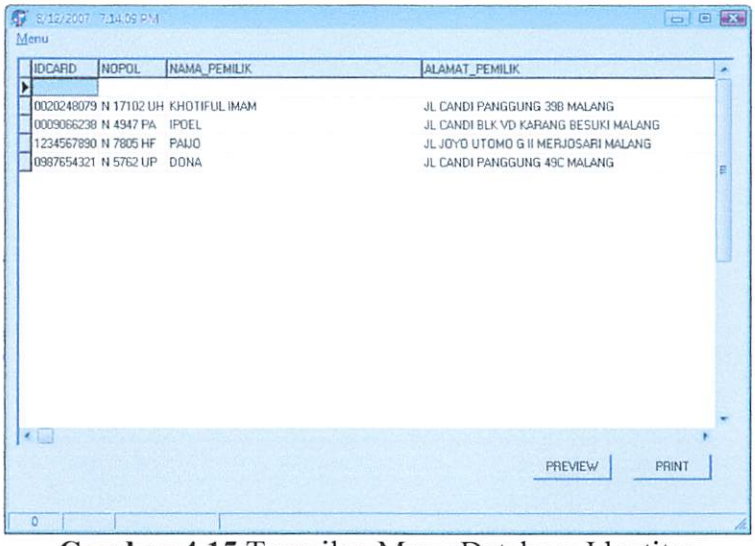
Gambar 4.13 Tampilan Menu Masukan Data Baru

Tampilan diatas muncul jika kita memilih menu lalu meng-clik masukan baru. untuk memesukkan data angkutan kota yang belum terdaftar yang nantinya akan disimpan dalam database.



Gambar 4.14 Tampilan Menu Database Aktivitas

Tampilan diatas adalah tampilan database aktivitas angkutan yang beroperasi setiap harinya (tanggal/bulan/tahun dan jam berapa angkutan tersebut beroperasi).



Gambar 4.15 Tampilan Menu Database Identitas

Gambar 4.15 diatas merupakan tampilan database identitas, Apabila kita ingin melihat data-data angkutan kota yang beroperasi diterminal tesebut,

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan dari hasil pengujian alat maka dapat diambil kesimpulan sebagai berikut :

1. RFID reader dapat membaca tag pada jarak kurang lebih 7 cm, sedangkan pada jarak 8 cm tag RFID tidak dapat terbaca oleh RFID reader.
2. Pada saat akses pertama angkutan kota keluar dari terminal saldo retribusi akan berkurang, tetapi pada saat akses selanjutnya angkutan kota keluar dari terminal saldo retribusi tidak berkurang karena retribusi dilakukan sekali dalam satu hari.
3. Palang pintu tidak terbuka secara otomatis, hal tersebut untuk menghindari terjadinya penggunaan kartu oleh orang lain, karena kartu prabayar ini disesuaikan dengan nomor polisi angkutan kota tersebut.

5.2. Saran

Sistem yang dirancang ini merupakan konsep dari pembuatan pos penarikan restribusi baik yang digunakan pada terminal angkot maupun pada perkumpulan dari pemilik tiap angkot sehingga untuk mengembangkan alat maupun perluasan aplikasi selanjutnya diberikan saran-saran sebagai berikut :

1. Untuk mempermudah akses pembacaan RFID dengan jarak yang lebih jauh, maka RFID reader bisa ditambahkan antena external.

2. Alat ini juga bisa ditambahkan sensor yang berfungsi untuk menutup palang pada saat angkot telah keluar dari terminal
3. Alat akan lebih sempurna apabila diberikan waktu tiba pada pos dan jeda waktu untuk tiba kembali pada pos tersebut yang telah ditentukan agar tidak ada angkot yang memperpendek rute jurusannya.

Daftar Pustaka

[1]. Woollard, Barry, *Elektronika Praktis*, 2002.

[2]. Malvino, *Prinsip-Prinsip Elektronika*, 2002.

[3]. Wahana komputer, *Pemrograman Borlan Delphi 7.0,* 2006

[4]. Abdul Kadir, *Pemrograman Database dengan Dephi 7 Menggunakan access,* 2004

[5]. Catatan Kuliah Sistem Instrumentasi Elektronika.

[6]. Datasheet IC AT89S52, www.atmel.com, ATMEL Corporation.

[7]. Datasheet RFID www.digi-ware.com.

[8]. DB9 Serial Connector Pin Assignments, www.comtrol.com.

[9]. RS-232 Standart Configuration, www.loop-back.com.

[10]. Datasheet IC MAX 232, www.maxim-ic.com, 2004.

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

FORM BIMBINGAN SKRIPSI

Nama : KHOTIFUL IMAM
NIM : 02.17.102
Masa Bimbingan : 22 Mei 2007 s/d 22 Nopember 2007
Judul : Perancangan dan Pembuatan Alat Retribusi Angkutan Kota Pada Terminal Dengan Sistem Prabayar Menggunakan RFID

NO	Tanggal	Uraian	Paraf
1	27/08 07	BAB II Revisi Sumber Gambar.	
2	27/08 07	BAB III Revisi perhitungan	
3	08/08 07	BAB II-III acc	
4	08/08 07	BAB IV Revisi analisa driver motor	
5	13/08 07	BAB IV acc	
6	28/08 07	BAB V Revisi Kesimpulan + Saran	
7	31/08 07	BAB V acc	
8	31/08 07	ACC KOMPRE	
9			
10			

Malang, 2007
Dosen Pembimbing

(Joseph Dedy Irawan, ST, MT)
NIP : 132315178

Form S-4a

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

FORM BIMBINGAN SKRIPSI

Nama : KHOTIFUL IMAM
NIM : 02.17.102
Masa Bimbingan : 22 Mei 2007 s/d 22 Nopember 2007
Judul : Perancangan dan Pembuatan Alat Retribusi Angkutan Kota Pada Terminal Dengan Sistem Prabayar Menggunakan RFID

NO	Tanggal	Uraian	Paraf
1	9/8 '07	Revisi bab II (analisa HW)	
2	15/8 '07	Revisi bab V (kesimp + saran)	
3	29/8 '07	Bab V ok.	
4			
5			
6			
7			
8			
9			
10			

Malang, 1 - 9 - 2007
Dosen Pembimbing

(Sotyohadi, ST, MSc)

Form S-4a

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Karanglo KM. 2 Malang

LEMBAR PERBAIKAN SKRIPSI

Nama : Khotiful Imam
NIM : 02.17.102
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika
Masa Bimbingan : 22 Mei 2007 s/d 22 Nopember 2007
Judul Skripsi : Perancangan Dan Pembuatan Alat Retribusi Angkutan Kota Pada Terminal Dengan Sistem Prabayar Menggunakan RFID

Hari/Tgl Skripsi : Senin, 3 September 2007

No.	Materi Perbaikan	Paraf
1.	Jelaskan fungsi SW1 dan SW2 dalam hubunganya MC dan RFID Reder	
2.	Apa fungsi P2.0 dan P2.1 dari MC	
3	Jelaskan format pengiriman data dari MC ke komputer	
4	Jelaskan format data dari RFID ke MC	
5	Gambarkan dan jelaskan fungsi kaki dari DB9 serial komputer	
6	Jelaskan bagaimana RFID Card mendapatkan catu tegangan (sumber tegangannya mana)	

Diperiksa / Disetujui

Penguji

DR. Cahyo Crysdian, Msc)
NIP. 1030400412

Mengetahui

Dosen Pembimbing I

Joseph Dedy Irawan, ST, MT
NIP. 132315178

Dosen Pembimbing II

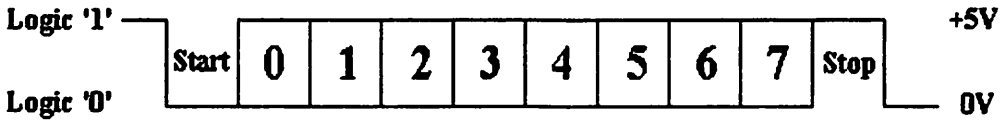
Sotrohadi, ST, MSc

1. Fungsi dari SW1 dan SW2 adalah digunakan untuk masukan bagi MCU sebagai tanda bahwa palang pintu sudah terbuka penuh atau tertutup penuh yang kemudian MCU mematikan motor agar putaran motor tidak melebihi putaran yang di inginkan.

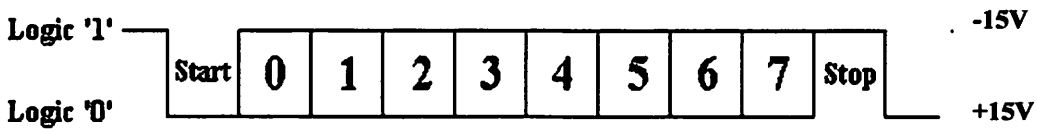
2. P2.0 dan P2.1 pada mikrokontroler AT89S52 digunakan sebagai input dari limit swicth1 & limit swicth2 seperti pada penjelasan no 1.

   - Jika limit swicth1 ditekan maka P2.0 akan berlogika "0" jika tidak P2.0 berlogika "1".
   - Jika limit swicth2 ditekan maka P2.1 akan berlogika "0" jika tidak P2.1 berlogika "1"

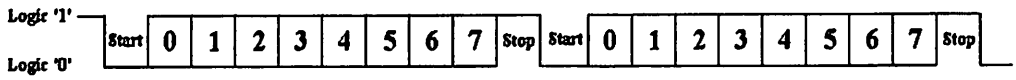
3. Format pengiriman data dari MCU ke komputer adalah UART (universal Asinchronous Receiver Transmiter)
   Format Sinyal Serial Asinkron level TTL adalah :



   Data diawali sinyal start bit (yang selalu nol) diikuti 8 data dan diakhiri dengan stop bit.
   Setelah melalui IC MAX232 level converter maka sinyalnya menjadi serial asinkron RS232 sebagai berikut :

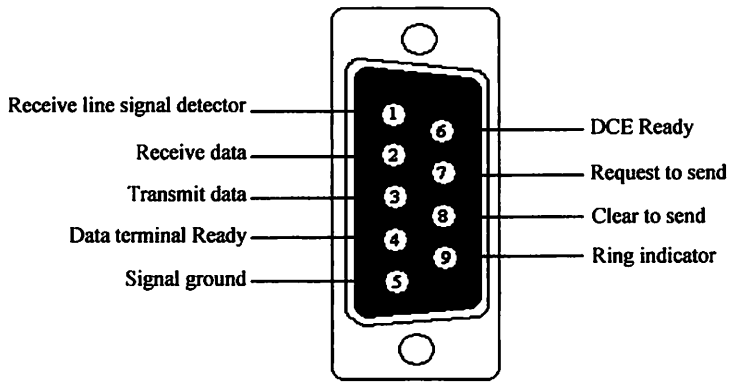


4. Format pengiriman data dari RFID reader ke MCU adalah Asinchronous
   Data diawali sinyal start bit (yang selalu nol) diikuti 8 data dan diakhiri dengan stop bit.



ASCII digit pertama ASCII digit kedua.....

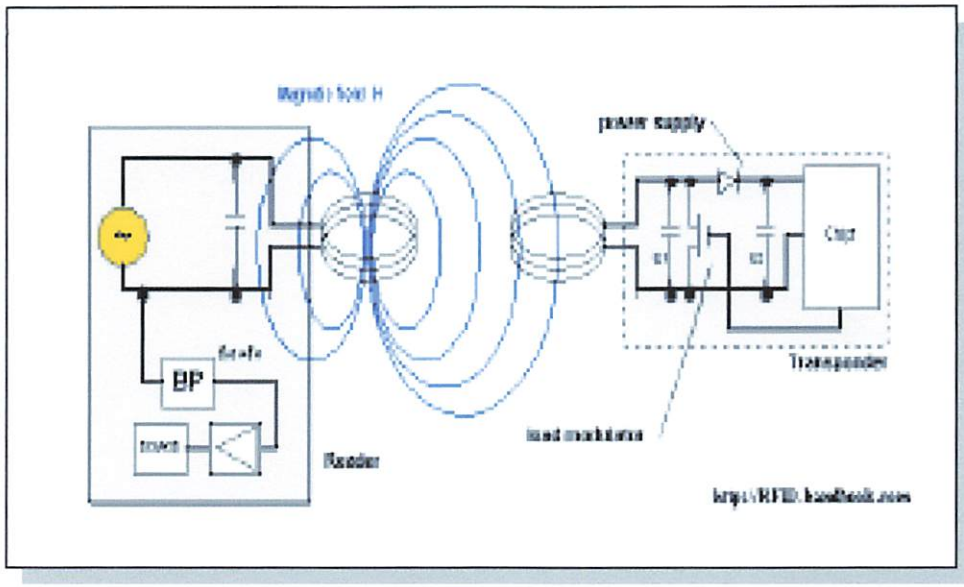
5. Gambar dan fungsi kaki dari DB9 serial komputer

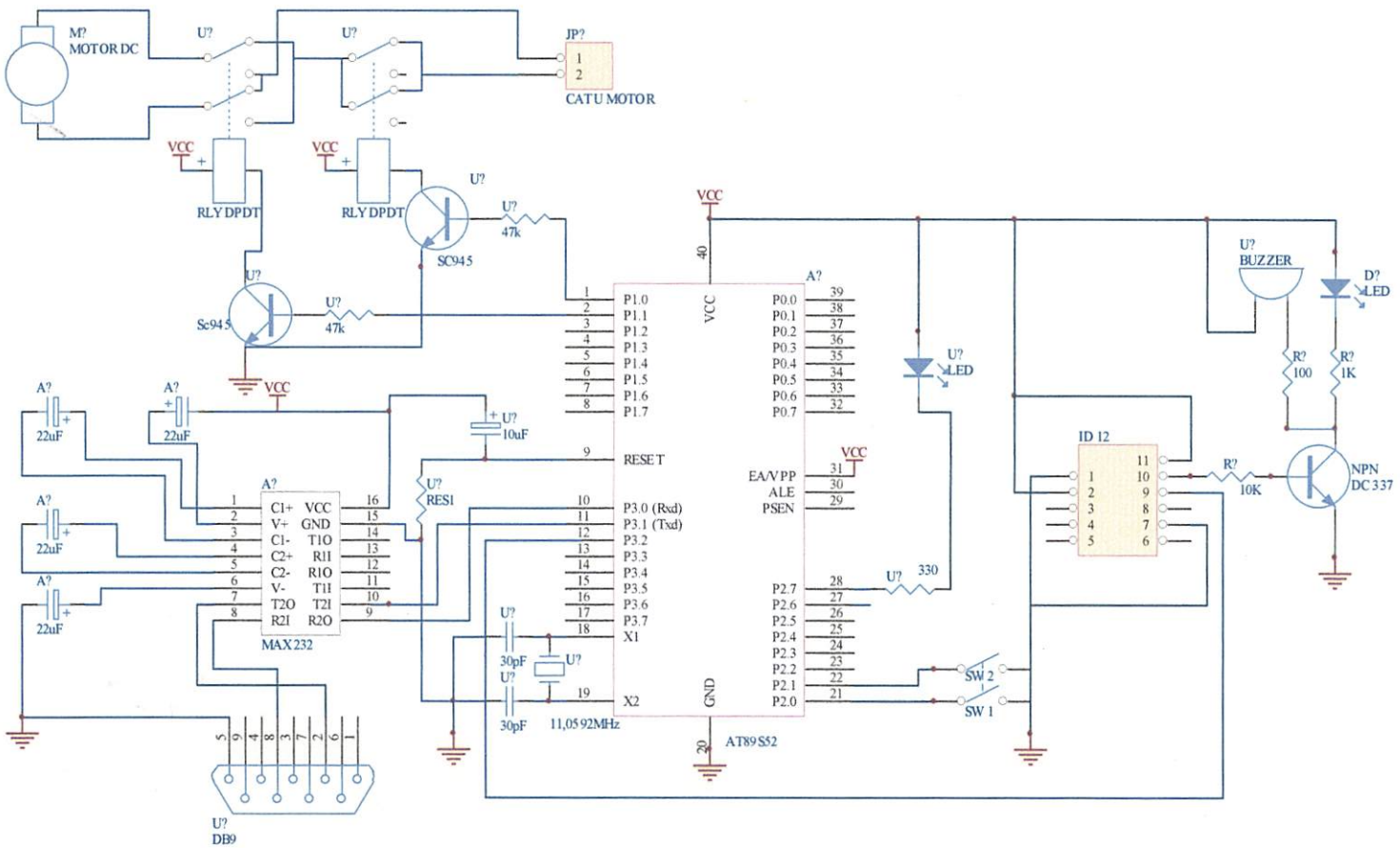


DB-9	NAMA SINYAL	FUNGSI
1	Receive line signal detektor	Dengan saluran ini DCE memberitahukan DTE bahwa pada terminal masukan ada data masuk
2	RD, Receive Data	Untuk menerima data
3	TD, Transmit Data	Untuk mengirimkan data
4	DTR, Data Terminal Ready	Pada saluran ini DTE memberitahukan kesiapan terminalnya
5	SG, signal Ground	Saluran ground
6	DSR, Data Set Ready	Sinyal aktif pada saluran ini menunjukkan bahwa DCE sudah siap
7	RTS, Request to Send	Dengan saluran ini DCE diminta mengirim data oleh DTE
8	CTS, Clear To Send	Dengan saluran ini DCE memberitahukan bahwa DTE boleh mulai mengirim data
9	RI, Ring indicator	Pada saluran ini DCE memberitahu ke DTE bahwa sebuah stasiun menghendaki hubungan dengannya

6. RFID card mendapatkan catu tegangan :

RFID reader memancarkan gelombang elektromagnetik dan didalam tag RFID terdapat rangkaian pengkonversi dari gelombang elektromagnetik menjadi suatu sinyal power DC untuk catu daya tag tersebut, ketika tag ini melalui medan elektromagnetik yang dihasilkan oleh RFID reader dan rangkaian tersebut akan bekerja dengan menghasilkan daya untuk mengirim data nomor ID yang ada pada single chip tag tersebut ke RFID reader.





Title		
Size: A4	Number	Revision
Date: 30-Aug-2007		Sheet of
File: J:\Rangkeseluruhan SCH_1.DDB		Drawn By:

```
#include  <at89x51.h>
#include  "Pending.c"

#define  off            1
#define  on             0

#define  palang_membuka 1
#define  palang_menutup 0

#define  limitswitch_tutup    P2_0
#define  limitswitch_buka     P2_1

#define  motor_direction      P1_0
#define  motor_power          P1_1


#define  rx2   P3.2

unsigned char     rx_buf[15],hasil,ID_count,i,perintah_PC,dser;
unsigned long     IDCARD;

/*=================== Prototype fungsi
====================================*/
void init_serial()
{
   TMOD     =0x20;
   TH1      =253;
   SCON     =0x50;
   TR1   =1;
   EA=1;
   ES=1;
   PS=1  ;
}

unsigned char getchar()
{
 while(!RI);
 RI=0;
 return SBUF;
}

void putchar(unsigned char dt)
{
      SBUF  = dt;
      while(!TI){;}
      TI          = 0;
}

void buka()
{
  while(limitswitch_buka)
  {
   motor_direction   =palang_membuka;
   motor_power       =on;
  }
  motor_power       =off;
}

void tutup()
{
   motor_direction   =palang_menutup;
   delay_ms(500);
   motor_power       =on;
   while(limitswitch_tutup)
   {
      motor_direction   =palang_menutup;
```

```
      motor_power       =on;
   }
   motor_power       =off;
}

void it_Serial () interrupt 4
{
      RI                = 0   ;
      dser  = SBUF;
      if (dser=='B')
   {
      ES=0;
      buka();
      ES=1;
   }
      else if (dser=='T')
   {
      ES=0;
      tutup();
      ES=1;
   }

}

unsigned char ascii2hex(char asciinya)
{
  if (asciinya <= '9') return asciinya&0x0F;
  else if (asciinya=='A') return 0x0A;
  else if (asciinya=='B') return 0x0B;
  else if (asciinya=='C') return 0x0C;
  else if (asciinya=='D') return 0x0D;
  else if (asciinya=='E') return 0x0E;
  else if (asciinya=='F') return 0x0F;
  return 0;
}


unsigned char getchar1()
{
_asm
loop:    jb    rx2,loop
         mov   r0,#22
liip:    djnz  r0,liip
         mov   r1,#8
laap:    mov   r0,#45
leep:    djnz  r0,leep
         mov   c,rx2
         rrc   a
         djnz  r1,laap
         mov   r0,#45
leep1:   djnz  r0,leep1
         mov   _hasil,acc
_endasm;
return hasil;
}

/*================ A W A L  P R O G R A M  U T A M A
=======================*/
void main(void)
{
init_serial();
motor_power=off;
tutup();

while(1)
{
```

```
   ID_count=0;
   do
   {
      rx_buf[ID_count]=getchar1();
      ID_count++;
   }
   while (ID_count<15);//tunggu sampai ada 15 karakter dr RF ID
   IDCARD=0;
   IDCARD=IDCARD | ascii2hex(rx_buf[3]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[4]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[5]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[6]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[7]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[8]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[9]);
   IDCARD=IDCARD << 4;
   IDCARD=IDCARD | ascii2hex(rx_buf[10]);


   ES=0;
   putchar(IDCARD/1000000000|0x30);
   putchar(IDCARD/100000000%10|0x30);
   putchar(IDCARD/10000000%10|0x30);
   putchar(IDCARD/1000000%10|0x30);
   putchar(IDCARD/100000%10|0x30);
   putchar(IDCARD/10000%10|0x30);
   putchar(IDCARD/1000%10|0x30);
   putchar(IDCARD/100%10|0x30);
   putchar(IDCARD/10%10|0x30);
   putchar(IDCARD%10|0x30);
   ES=1;
}
}
/*================ A K H I R  P R O G R A M  U T A M A
====================*/
```

===Program PC===

```
unit pass;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Buttons;

type
  TForm3 = class(TForm)
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses Terminal1;

{$R *.dfm}

procedure TForm3.BitBtn1Click(Sender: TObject);
begin
if Edit1.Text='alit' then
begin
   form1.Edit9.Visible:=true;
   form1.BitBtn2.Visible:=true;
   form1.BitBtn1.Visible:=False;
   form1.BitBtnCari.Visible:=False;
   form1.Label4.Caption:='Pengisian Ulang';
   Form1.lbKey.Visible:=false;
end;
form3.Hide;
form1.enabled:=true;
end;

procedure TForm3.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
form3.Hide;
form1.enabled:=true;
end;

end.
```

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Buttons;

type
  TForm2 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Terminal1;

{$R *.dfm}

procedure TForm2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
form2.Hide;
form1.Enabled:=true;
end;

procedure TForm2.BitBtn1Click(Sender: TObject);
begin
terminal_current:=UpperCase( Form2.Edit1.Text);
Form1.Label3.Caption := 'TERMINAL '+UpperCase( Form2.Edit1.Text);
form2.Hide;
form1.Enabled:=true;
end;

end.
```

```
rminal1;

ce


ws, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
gs, Printers,Menus, DB, DBTables, ExtCtrls, StdCtrls, CPort, ComCtrls,
ns, Grids, DBGrids;


1 = class(TForm)
ry1: TQuery;
nMenu1: TMainMenu;
u1: TMenuItem;
u1: TMenuItem;
abase1: TMenuItem;
uar1: TMenuItem;
ebook1: TNotebook;
el1: TLabel;
el2: TLabel;
el3: TLabel;
inal1: TMenuItem;
er1: TTimer;
er2: TTimer;
an: TTimer;
Port1: TComPort;
tusBar1: TStatusBar;
el4: TLabel;
er1: TMenuItem;
el5: TLabel;
el6: TLabel;
el7: TLabel;
el8: TLabel;
el9: TLabel;
el11: TLabel;
t1: TEdit;
t2: TEdit;
t3: TEdit;
t4: TEdit;
t5: TEdit;
t7: TEdit;
Btn1: TBitBtn;
el12: TLabel;
t8: TEdit;
le1: TTable;
t9: TEdit;
fil1: TMenuItem;
el10: TLabel;
t6: TEdit;
el13: TLabel;
t10: TEdit;
Btn2: TBitBtn;
aSource1: TDataSource;
rid1: TDBGrid;
```

```
el1: TPanel;
el14: TLabel;
el15: TLabel;
el16: TLabel;
el17: TLabel;
el18: TLabel;
el19: TLabel;
el20: TLabel;
el21: TLabel;
el22: TLabel;
el23: TLabel;
t11: TEdit;
ton1: TButton;
el31: TLabel;
el32: TLabel;
el33: TLabel;
el24: TLabel;
el25: TLabel;
el2: TPanel;
ton2: TButton;
t12: TEdit;
el26: TLabel;
el27: TLabel;
el28: TLabel;
ey: TLabel;
i1: TMenuItem;
BtnCari: TBitBtn;
ton4: TButton;
ton5: TButton;
ton3: TButton;
ton6: TButton;
el29: TLabel;
el30: TLabel;
el34: TLabel;
el35: TLabel;
el36: TLabel;
el37: TLabel;
abaseIdentitas1: TMenuItem;
Update: TButton;
Hapus: TButton;
cedure Keluar1Click(Sender: TObject);
cedure Timer1Timer(Sender: TObject);
cedure erminal1Click(Sender: TObject);
cedure Timer2Timer(Sender: TObject);
cedure ComPort1RxChar(Sender: TObject; Count: Integer);
cedure FormCreate(Sender: TObject);
cedure TScanTimer(Sender: TObject);
cedure Baru1Click(Sender: TObject);
cedure Cover1Click(Sender: TObject);
cedure Reffil1Click(Sender: TObject);
cedure BitBtn2Click(Sender: TObject);
cedure BitBtn1Click(Sender: TObject);
cedure Database1Click(Sender: TObject);
cedure Button1Click(Sender: TObject);
```

```
cedure Button2Click(Sender: TObject);
cedure Cari1Click(Sender: TObject);
cedure BitBtnCariClick(Sender: TObject);
cedure Button4Click(Sender: TObject);
cedure Button5Click(Sender: TObject);
cedure Button3Click(Sender: TObject);
cedure Button6Click(Sender: TObject);
cedure DatabaseIdentitas1Click(Sender: TObject);
cedure BtnUpdateClick(Sender: TObject);
cedure BtnHapusClick(Sender: TObject);
te
ate declarations }
ction cekserial():boolean;

c
lic declarations }
 procedure print_struk();



: TForm1;
nal_current,dataserial:string;
ag:Boolean;
OPOL,tanggal_sekarang,jam_sekarang,
enis_trayek,tmp_anggota_terminal,tmp_ID,
etribusi,tmp_saldo,tmp_saldo_akhir:string;
ntation

it2, pass, UnitLaporan, lap;

fm}

n TForm1.cekserial():boolean;


re TForm1.print struk();

nya : textfile;

gnPrn(Filenya);
ite(Filenya);
eln(Filenya,'=====================================');
eln(Filenya,'Struk Kontrol Angkot                  ');
eln(Filenya,'=====================================');
eln(Filenya,' No. ID                    : '+tmp_ID);
eln(Filenya,' No. Polisi                : '+tmp_NOPOL);
eln(Filenya,' Trayek                    : '+tmp_jenis_trayek);
eln(Filenya,' Terminal Retribusi : '+tmp_anggota_terminal);
eln(Filenya,' Retribusi                 : Rp.'+tmp_retribusi+',00');
eln(Filenya,' Saldo                     : Rp.'+tmp_saldo_akhir+',00');
eln(Filenya,'=====================================');
eln(Filenya,' TANGGAL: '+ tanggal_sekarang );
eln(Filenya,' JAM: '+ jam_sekarang );
em.CloseFile(Filenya);
```

```
re TForm1.Keluar1Click(Sender: TObject);

tion.Terminate;


re TForm1.Timer1Timer(Sender: TObject);

1.Caption:= DateToStr(date)+'    '+TimeToStr(time);


re TForm1.erminal1Click(Sender: TObject);

how;
nabled:=false;


re TForm1.Timer2Timer(Sender: TObject);

l_current:='ARJOSARI';
Enabled:=False;


re TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
tring;

rt1.ReadStr(s,Count);
_flag then

 dataserial:= dataserial+s;
 StatusBar1.Panels[2].Text:=dataserial;
 //Label26.Caption:= IntToStr(StrToInt('$'+dataserial));
 Label26.Caption:= dataserial;
 if length(dataserial)>=10 then
 begin
   edit11.Text:= Label26.Caption;
   Button1.Click;
   dataserial:= '';
 end;



re TForm1.FormCreate(Sender: TObject);

ial:='';
k1.PageIndex:=0;
1.Open;
.Caption:='';
.Caption:='';
.Caption:='';
.Caption:='';
.Caption:='';
.Caption:='';
```

```
:=true;


re TForm1.TScanTimer(Sender: TObject);

sBar1.Panels[0].Text:=inttostr(length(dataserial));
.Enabled:=false;



.Enabled:=True;


re TForm1.Baru1Click(Sender: TObject);

te.Visible:=false;
s.Visible:=false;
nabled:=True;
Visible:=False;
.Visible:=False;
.Visible:=True;
isible:=False;
.Visible:=False;
ari.Visible:=False;
ext:='';
ext:='';
ext:='';
ext:='';
ext:='';
ext:='';
ext:='';
ext:='';
ext:='';
Text:='';
Caption:='Masukan Baru';
k1.PageIndex:=1;
isible :=false;
etFocus;


re TForm1.Cover1Click(Sender: TObject);

k1.PageIndex:=0;


re TForm1.Reffil1Click(Sender: TObject);

.Edit1.Text:='';
.show;
.Enabled:=False;
ook1.PageIndex:=1;
```

```
re TForm1.BitBtn2Click(Sender: TObject);

aldo,saldo_tambahan :integer;
aldo:string;


1.Close;
1.DatabaseName := GetCurrentDir;
1.TableName := 'tb_ID';
1.Close; Query1.SQL.Clear;
1.SQL.Add('SELECT * FROM tb_ID WHERE tb_ID.IDCARD = "'+Edit1.Text+'"');
1.ExecSQL;
1.Open;
ery1.RecordCount>0 then

p_saldo:=StrToInt(Query1.FieldValues['SALDO']);
do_tambahan:=StrToInt(edit9.Text);
p_saldo:=IntToStr(itmp_saldo+saldo_tambahan);
ry1.Close; Query1.SQL.Clear;
ry1.SQL.Add('UPDATE tb_ID SET saldo="'+stmp_saldo+'" WHERE IDCARD = "'+Edit1.Text+

ry1.ExecSQL;
ry1.Close; Query1.SQL.Clear;
ry1.SQL.Add('SELECT * FROM tb_ID WHERE tb_ID.IDCARD = "'+Edit1.Text+'"');
ry1.ExecSQL;
ry1.Open;
el13.Visible:=true;
t10.Visible:=true;
t1.Text  := Query1.FieldValues['idcard'];
t2.Text  := Query1.FieldValues['nama_pemilik'];
t3.Text  := Query1.FieldValues['alamat_pemilik'];
t4.Text  := Query1.FieldValues['nopol'];
t5.Text  := Query1.FieldValues['jenis_kendaraan'];
t6.Text  := Query1.FieldValues['jenis_trayek'];
t7.Text  := Query1.FieldValues['retribusi'];
t8.Text  := Query1.FieldValues['anggota_terminal'];
t10.Text := 'Rp.'+Query1.FieldValues['saldo']+',00';



lication.MessageBox (' ID Card Belum terdaftar....  ',
 'Informasi', MB_OK or MB_ICONINFORMATION);

n2.Visible:=false;
.Visible:=false;


re TForm1.BitBtn1Click(Sender: TObject);

1.Close;
1.DatabaseName := GetCurrentDir;
1.TableName := 'tb_ID';
1.Close; Query1.SQL.Clear;
1.SQL.Add('SELECT * FROM tb_ID WHERE tb_ID.IDCARD = "'+Edit1.Text+'"');
```

```
1.ExecSQL;
1.Open;
ery1.RecordCount>0 then

lication.MessageBox (' ID Card Sudah Terdaftar....    ',
 'Informasi', MB_OK or MB ICONINFORMATION);



(Edit1.Text='') OR
(Edit2.Text='') OR
(Edit3.Text='') OR
(Edit4.Text='') OR
(Edit5.Text='') OR
(Edit6.Text='') OR
(Edit7.Text='') OR
(Edit8.Text='') then
egin
pplication.MessageBox (' Semua harus Diisi  ',
 'Informasi', MB_OK or MB_ICONINFORMATION);
nd
lse
egin
Query1.Close; Query1.SQL.Clear;
Query1.SQL.Add('INSERT INTO tb_ID(IDCARD,NAMA_PEMILIK,ALAMAT_PEMILIK,NOPOL,'+
               'JENIS_KENDARAAN,ANGGOTA_TERMINAL,JENIS_TRAYEK,RETRIBUSI,SALDO)'+
               'VALUES("'+UpperCase(edit1.Text)+'","'+UpperCase(edit2.Text)+
               '","'+UpperCase(edit3.Text)+'","'+UpperCase(edit4.Text)+
               '","'+UpperCase(edit5.Text)+'","'+UpperCase(edit8.Text)+
               '","'+UpperCase(edit6.Text)+'","'+edit7.Text+'","0")');
Query1.ExecSQL;

nd;



re TForm1.Database1Click(Sender: TObject);

ook1.PageIndex:=2;
1.Close;
1.DatabaseName := GetCurrentDir;
1.TableName := 'tb_LAPOR';
1.Close; Query1.SQL.Clear;
1.SQL.Add('SELECT * FROM tb_LAPOR ');
1.ExecSQL;
1.Open;


re TForm1.Button1Click(Sender: TObject);

.Caption:=Edit11.Text;
Close;Query1.SQL.Clear;
SQL.Add('SELECT * FROM tb_ID WHERE IDCARD="'+Edit11.Text+'" ');
ExecSQL;
```

```
Open;
y1.RecordCount=0 then

sBar1.Panels[3].Text:='ID CARD belum Terdaftar';
34.Font.Color    :=clRed;
34.Caption       :='ID CARD belum Terdaftar';
22.Caption       :='';
23.Caption       :='';
33.Caption       :='';
37.Caption       :='';
25.Caption       :='Rp.0,00';



ng data data yang diperlukan dari tabel ID
OPOL             :=Query1.FieldValues['NOPOL'];
enis_trayek      :=Query1.FieldValues['jenis_trayek'];
nggota_terminal  :=Query1.FieldValues['anggota_terminal'];
etribusi         :=Query1.FieldValues['retribusi'];
aldo             :=Query1.FieldValues['saldo'];
al_sekarang      :=DateToStr(Date);
ekarang          :=TimeToStr(Time);
D                :=Query1.FieldValues['IDCARD'];
22.Caption       :=tmp_NOPOL;
23.Caption       :=tmp_jenis_trayek;
33.Caption       :=tmp_anggota_terminal;
37.Caption       :=tmp_retribusi;

rToInt(tmp_saldo) < strtoint(tmp_retribusi) then

 Label34.Caption:='Saldo Tidak mencukupi';
 Label25.Caption:='Rp.'+tmp_saldo+',00';
 Label34.Font.Color:=clRed;
 Label25.Font.Color:=clRed;



 Label25.Font.Color:=clSilver;
 Label25.Caption:='Rp.'+tmp_saldo+',00';
 Query1.Close;
 Query1.SQL.Clear;
 Query1.SQL.Add('SELECT * FROM tb_LAPOR WHERE '+
                'NOPOL="'+tmp_NOPOL+'" AND TANGGAL="'+tanggal_sekarang+'"');
 Query1.ExecSQL;
 Query1.Open;

 if Query1.RecordCount=0 then
 begin
         Label34.Font.Color:=clOlive;
         Label34.Caption:='Akses Pertama';

         if  tmp_anggota_terminal=terminal_current then
         begin
           tmp_saldo_akhir:=IntToStr(StrToInt(tmp_saldo)-StrToInt(tmp_retribusi));
```

```
          print_struk;
        end
        else
        begin
          Label34.Font.Color:=clFuchsia;
          Label34.Caption:='Bukan Wajib Retribusi';
          tmp_saldo_akhir:=IntToStr(StrToInt(tmp_saldo)-0);
        end;
 end
 else
 begin
        if  tmp_anggota_terminal=terminal_current then
        begin
          Label34.Font.Color:=clOlive;
          Label34.Caption:='Bukan Akses Pertama';
        end
        else
        begin
          Label34.Font.Color:=clFuchsia;
          Label34.Caption:='Bukan Wajib Retribusi';
        end;
        tmp_saldo_akhir:=IntToStr(StrToInt(tmp_saldo)-0);
 end;
 Table1.Close;
 Table1.DatabaseName := GetCurrentDir;
 Table1.TableName := 'tb_LAPOR';
 Query1.Close;Query1.SQL.Clear;
 Query1.SQL.Add('INSERT INTO tb_LAPOR
JENIS_TRAYEK,ANGGOTA_TERMINAL,TANGGAL,JAM,SALDO) '+
                'VALUES ("'+tmp_NOPOL+'","'+tmp_JENIS_TRAYEK+'","'+
GOTA_TERMINAL+
                '","'+tanggal_sekarang+'","'+jam_sekarang+'","'+tmp_saldo_akhir+

 Query1.ExecSQL;

 Query1.Close;Query1.SQL.Clear;
 Query1.SQL.Add('UPDATE tb_ID SET saldo="'+tmp_saldo_akhir+'" WHERE NOPOL="'+
OL+'"');
 Query1.ExecSQL;
 dataserial:='';




re TForm1.Cari1Click(Sender: TObject);

Caption:='Cari Data Kendaraan';
Alignment:=taCenter;
k1.PageIndex:=1;
isible :=True;
etFocus;
```

```
isible:=false;
Visible:=true;
.Visible:=true;
ari.Visible:=True;
te.Visible:=true;
s.Visible:=true;



re TForm1.BitBtnCariClick(Sender: TObject);

t4.Text ='' then

lication.MessageBox (' Isian NO. Polisi harus diisi...    ',
 'Informasi', MB_OK or MB_ICONINFORMATION);
t4.SetFocus;



it1.Enabled:=False;
ery1.Close; Query1.SQL.Clear;
ery1.SQL.Add('SELECT * FROM tb_ID where NOPOL="'+UpperCase(Edit4.Text)+'"');
ery1.ExecSQL;
ery1.Open;
 Query1.RecordCount>0 then
gin
 Edit1.Text:=Query1.FieldValues['IDCARD'];
 Edit2.Text:=Query1.FieldValues['NAMA_PEMILIK'];
 Edit3.Text:=Query1.FieldValues['ALAMAT_PEMILIK'];
 Edit4.Text:=Query1.FieldValues['NOPOL'];
 Edit5.Text:=Query1.FieldValues['JENIS_KENDARAAN'];
 Edit6.Text:=Query1.FieldValues['JENIS_TRAYEK'];
 Edit7.Text:=Query1.FieldValues['RETRIBUSI'];
 Edit8.Text:=Query1.FieldValues['ANGGOTA_TERMINAL'];
 Edit10.Text:=Query1.FieldValues['SALDO'];
d
se
gin
lication.MessageBox (' Data kendaraan dengan No. Polisi tersebut tidak ditemukan  ',
 'Informasi', MB_OK or MB_ICONINFORMATION);
d;



re TForm1.Button4Click(Sender: TObject);

an.Query1.Close;
an.Query1.Open;
an.Print;


re TForm1.Button5Click(Sender: TObject);

an.Query1.Close;
```

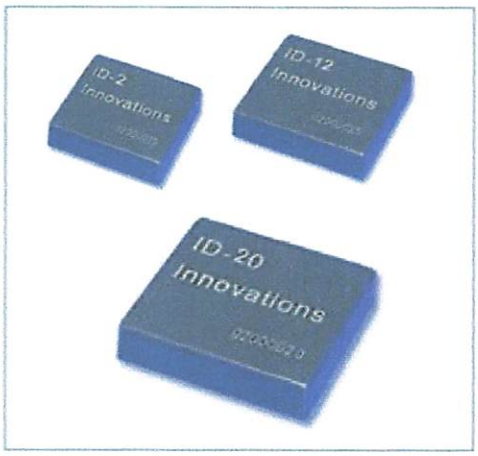
```
an.Query1.Open;
an.Preview;


re TForm1.Button3Click(Sender: TObject);

:=false;
1.WriteStr('B');
0);
:=true;
ial:='';


re TForm1.Button6Click(Sender: TObject);

:=false;
1.WriteStr('T');
0);
:=true;
ial:='';


re TForm1.DatabaseIdentitas1Click(Sender: TObject);

ook1.PageIndex:=2;
1.Close;
1.DatabaseName := GetCurrentDir;
1.TableName := 'tb_ID';
1.Close; Query1.SQL.Clear;
1.SQL.Add('SELECT * FROM tb_ID ');
1.ExecSQL;
1.Open;



re TForm1.BtnUpdateClick(Sender: TObject);

y1.Close;Query1.SQL.Clear;
y1.SQL.Add('DELETE FROM tb_ID WHERE IDCARD="'+Edit1.Text+'"');
y1.ExecSQL;
y1.Close; Query1.SQL.Clear;
y1.SQL.Add('INSERT INTO tb_ID(IDCARD,NAMA_PEMILIK,ALAMAT_PEMILIK,NOPOL,'+
           'JENIS_KENDARAAN,ANGGOTA_TERMINAL,JENIS_TRAYEK,RETRIBUSI,SALDO)'+
           'VALUES("'+UpperCase(edit1.Text)+'","'+UpperCase(edit2.Text)+
           '","'+UpperCase(edit3.Text)+'","'+UpperCase(edit4.Text)+
           '","'+UpperCase(edit5.Text)+'","'+UpperCase(edit8.Text)+
           '","'+UpperCase(edit6.Text)+'","'+edit7.Text+'","0")');
y1.ExecSQL;


re TForm1.BtnHapusClick(Sender: TObject);

t1.Text <>'' then
```

```
/1.Close;Query1.SQL.Clear;
/1.SQL.Add('DELETE FROM tb_ID WHERE IDCARD="'+Edit1.Text+'"');
/1.ExecSQL;
.cation.MessageBox (' Data sudah terhapus  ',
 'Informasi', MB_OK or MB_ICONINFORMATION);
1.Text:='';
2.Text:='';
3.Text:='';
4.Text:='';
5.Text:='';
6.Text:='';
7.Text:='';
8.Text:='';
9.Text:='';
10.Text:='';
```

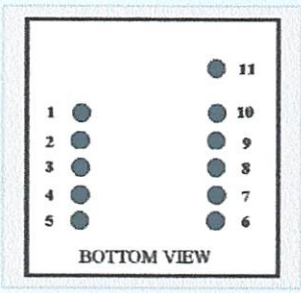
ID SERIES DATASHEET Feb 10 , 2004

ID-2 / ID-12 / ID-20

The ID2. ID12 and ID20 are similar to the ID0, ID10 and ID15 MK(ii) series devices, but they have extra pins which allow Magnetic Emulation output to be included in the functionality. The ID-12 and ID-20 come with internal antennas, and have read ranges of 12+ cm and 16+ cm, respectively. With an external antenna, the ID-2 can deliver read ranges of up to 25 cm. All three readers support ASCII, Wiegand26 and Magnetic ABA Track2 data formats.



ID2 / ID12 / ID20 PIN-OUT



1. GND
2. RES (Reset Bar)
3. ANT (Antenna)
4. ANT (Antenna)
5. CP
6. Future
7. +/- (Format Selector)
8. D1 (Data Pin 1)
9. D0 (Data Pin 0)
10. LED (LED / Beeper)
11. +5V

Operational and Physical Characteristics

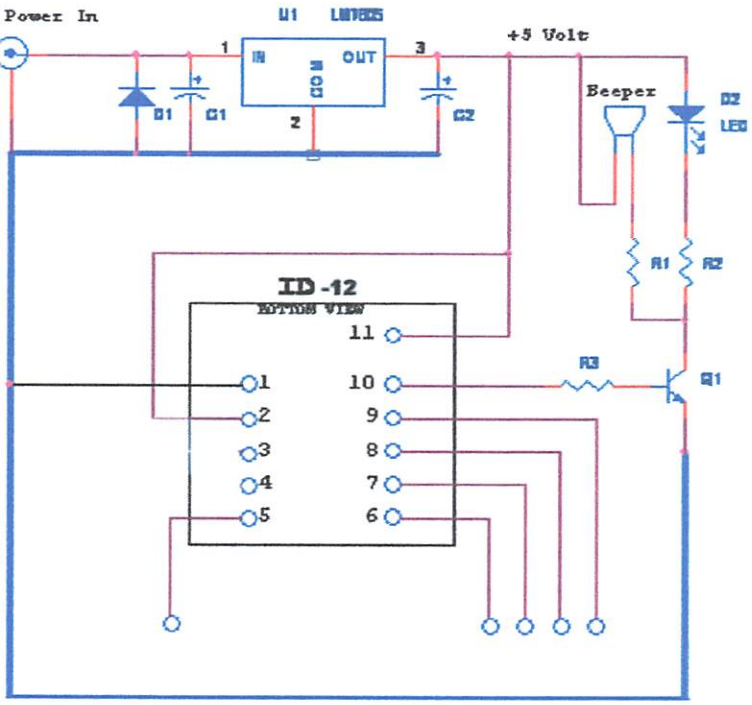
Parameters	ID-2	ID-12	ID-20
Read Range	N/A (no internal antenna)	12+ cm	16+ cm
Dimensions	21 mm x 19 mm x 6 mm	26 mm x 25 mm x 7 mm	40 mm x 40 mm x 9 mm
Frequency	125 kHz	125 kHz	125 kHz
Card Format	EM 4001 or compatible	EM 4001 or compatible	EM 4001 or compatible
Encoding	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64
Power Requirement	5 VDC @ 13mA nominal	5 VDC @ 30mA nominal	5 VDC @ 65mA nominal
I/O Output Current	+/-200mA PK	-	-
Voltage Supply Range	+4.6V through +5.4V	+4.6V through +5.4V	+4.6V through +5.4V

Pin Description & Output Data Formats

Pin No.	Description	ASCII	Magnet Emulation	Wiegand26
Pin 1	Zero Volts and Tuning Capacitor Ground	GND 0V	GND 0V	GND 0V
Pin 2	Strap to +5V	Reset Bar	Reset Bar	Reset Bar
Pin 3	To External Antenna and Tuning Capacitor	Antenna	Antenna	Antenna
Pin 4	To External Antenna	Antenna	Antenna	Antenna
Pin 5	Card Present	No function	Card Present	No function
Pin 6	Future	Future	Future	Future
Pin 7	Format Selector (+/-)	Strap to GND	Strap to Pin 10	Strap to +5V
Pin 8	Data 1	CMOS	Clock	One Output
Pin 9	Data 0	TTL Data (inverted)	Data	Zero Output
Pin 10	3.1 kHz Logic	Beeper / LED	Beeper / LED	Beeper / LED
Pin 11	DC Voltage Supply	+5V	+5V	+5V

Advanced Digital Reader Technology
----Better by Design

Circuit Diagram for the ID-12



COMPONENT LIST
R1 = 100R
R2 = 1K
R3 = 10K
C1 = 100uF 16V
C2 = 100uF 10V
Beeper = 2.7-3.5KHz 100R
D1 = 1N4001
D2 = GREEN LED
U1 = LM7805
Q1 = DC337 (NPN)
ID2 = ID Innovations ID2

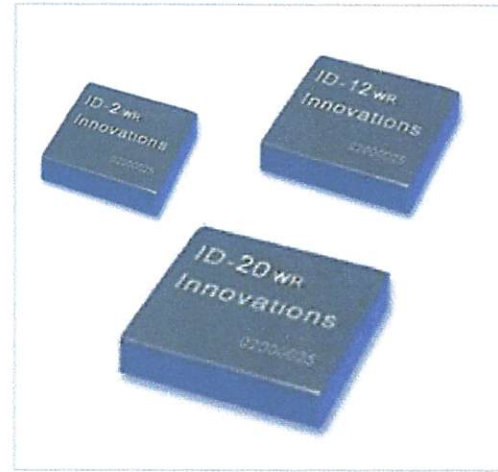
* Please Note the ID2 has an internal tuning capacitor of 1.5nF and this makes the total tuning capacity = 2.5nF

The 3.1Khz Beeper Logic is centered for most Beepers in range 2.7-3.5Khz

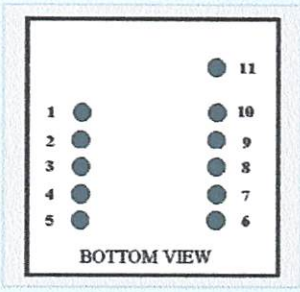
Advanced Digital Reader Technology
----Better by Design

ID-2RW. ID-12RW Brief Data

The ID2-RW, ID12-RW and ID15-RW are a new series of Read/Write modules for the Temec Q5 tag. It has full functionality including password. They contain built-in algorithms to assist customers programming the popular Sokymat Unique type tag. Password protection is allowed. Control is via a host computer using a simple terminal program such as hyper terminal or Qmodem.



ID2 / ID12 / ID20 PIN-OUT



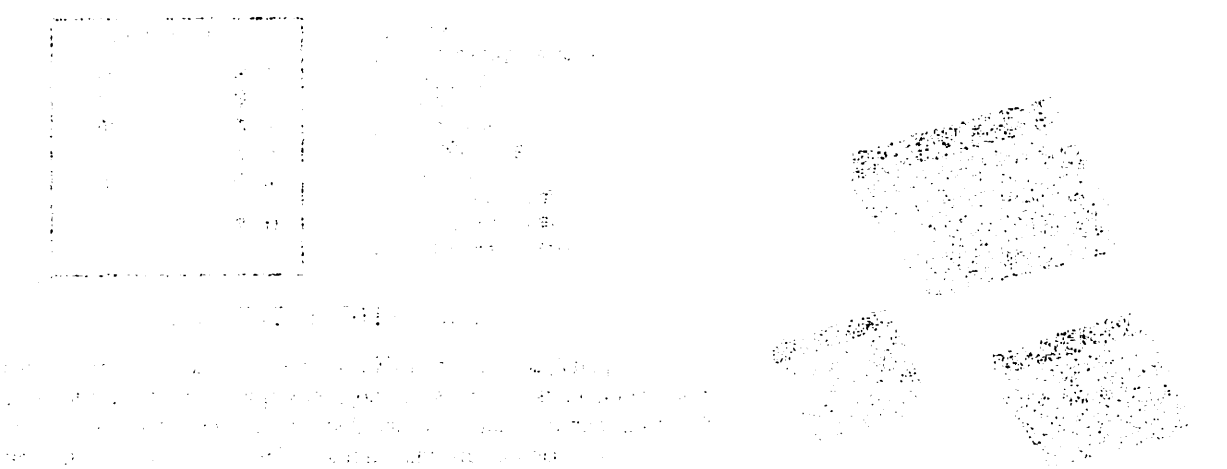
1	GND
2	RES (Reset Bar)
3	ANT (Antenna)
4	ANT (Antenna)
5	Future
6	Program LED
7	ASCII in
8	Future
9	ASCII Out
10	Read (LED / Beeper)
11	+5V

Operational and Physical Characteristics

Parameters	ID-2RW	ID-12RW	ID-20RW
Read Range	N/A (no internal antenna)	12+ cm (Unique Format)	15+ cm (Unique Format)
Dimensions	21 mm x 19 mm x 6 mm	26 mm x 25 mm x 7 mm	40 mm x 40 mm x 9 mm
Frequency	125 kHz	125 kHz	125 kHz
Card Format	Temec Q5555	Temec Q5555	Temec Q5555
Read Encoding	Manchester modulus 64	Manchester modulus 64	Manchester modulus 64
Power Requirement	5 VDC @ 13mA nominal	5 VDC @ 30mA nominal	5 VDC @ 50mA nominal
I/O Output Current	+/-200mA PK	-	-
Voltage Supply Range	+4.6V through +5.4V	+4.6V through +5.4V	+4.6V through +5.4V
Coil Detail	L = 0.6mH - 1.5mH, Q = 15-30	-	-

Description

A host computer is required to send the commands to the module. A simple terminal program such as Qmodem or Hyper-terminal can be used to send commands to the module. The blocks are individually programmable. If you have ever found that the Q5 can be a bit 'Twitchy' to program this programmer module is your solution. The command interface is simple to use and easily understood. The programmer also has two types of internal reader. One of these is provided to read Sokymat 'Unique' type tag configuration.



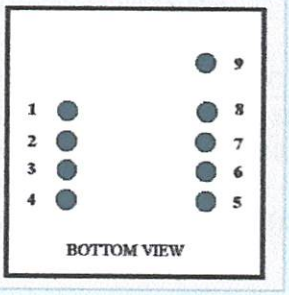
Low Cost Short-Range Proximity Readers

ID-0 / ID-10 / ID-15 MK(ii) Series

Maintenance only. Consider using the ID-2, ID-12 and the ID-20 that have improved performance.

The ID Series short-range readers come in three different sizes and read ranges. Both the ID-10 and ID-15 come with internal antennas, and have read ranges of 12+ cm and 16+ cm, respectively. With an external antenna, the ID-0 Mk(ii) can deliver read ranges of up to 25 cm. All three readers support ASCII and Wiegand26 data formats.

ID0 / ID10 / ID15 PIN-OUT



1. GND
2. RES (Reset Bar)
3. ANT (Antenna)
4. ANT (Antenna)
5. +/- (Format Selector)
6. D1 (Data Pin 1)
7. D0 (Data Pin 0)
8. LED (LED / Beeper)
9. +5V

Operational and Physical Characteristics

Parameters	ID-0	ID-10	ID-15
Read Range	N/A (no internal antenna)	12+ cm	15+ cm
Dimensions	21 mm x 19 mm x 6 mm	26 mm x 25 mm x 7 mm	40 mm x 40 mm x 9 mm
Frequency	125 kHz	125 kHz	125 kHz
Card Format	EM 4001 or compatible	EM 4001 or compatible	EM 4001 or compatible
Encoding	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64
Power Requirement	5 VDC @ 13mA nominal	5 VDC @ 30mA nominal	5 VDC @ 50mA nominal
I/O Output Current	+/-200mA PK	-	-
Voltage Supply Range	+4.6V through +5.4V	+4.6V through +5.4V	+4.6V through +5.4V

Pin Description & Output Data Formats

Pin No.	Description	ASCII	Wiegand26
Pin 1	Zero Volts and Tuning Capacitor Ground	GND 0V	GND 0V
Pin 2	Strap to +5V	Reset Bar	Reset Bar
Pin 3	To External Antenna and Tuning Capacitor	Antenna	Antenna
Pin 4	To External Antenna	Antenna	Antenna
Pin 5	Format Selector (+/-)	Strap to GND	Strap to +5V
Pin 6	Data 1	CMOS	One Output
Pin 7	Data 0	TTL Data (inverted)	Zero Output
Pin 8	3.1 kHz Logic	Beeper / LED	Beeper / LED
Pin 9	DC Voltage Supply	+5V	+5V

DATA FORMATS

Output Data Structure – ASCII

STX (02h)	DATA (10 ASCII)	CHECK SUM (2 ASCII)	CR	LF	ETX (03h)

[The 1byte (2 ASCII characters) Check sum is the "Exclusive OR" of the 5 hex bytes (10 ASCII) Data characters.]

Output Data Structure – Wiegand26

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
P	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	P
Even parity (E)													Odd parity (O)												

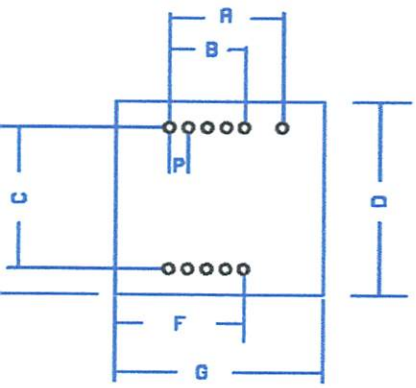
P = Parity start bit and stop bit

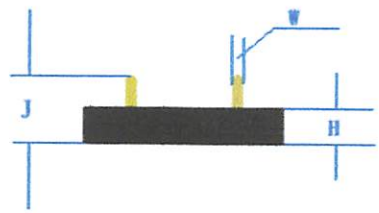
Output Data Magnetic ABA Track2

10 Leading Zeros	SS	Data	ES	LCR	10 Ending Zeros

[SS is the Start Character of 11010, ES is the end character of 11111, LRC is the Longitudinal Redundancy Check.]

Dimensions (Top View) (mm)





	ID-0/ID-2			ID-10/ID-12			ID-15/ID-20		
	Nom.	Min.	Max.	Nom.	Min.	Max.	Nom.	Min.	Max
A	12.0	11.6	12.4	12.0	11.6	12.4	12.0	11.6	12.4
B	8.0	7.6	8.4	8.0	7.6	8.4	8.0	7.6	8.4
C	15.0	14.6	15.4	15.0	14.6	15.4	15.0	14.6	15.4
D	20.5	20.0	21.5	25.3	24.9	25.9	40.3	40.0	41.0
E	18.5	18.0	19.2	20.3	19.8	20.9	27.8	27.5	28.5
F	14.0	13.0	14.8	16.3	15.8	16.9	22.2	21.9	23.1
G	22.0	21.6	22.4	26.4	26.1	27.1	38.5	38.2	39.2
P	2.0	1.8	2.2	2.0	1.8	2.2	2.0	1.8	2.2
H	5.92	5.85	6.6	6.0	5.8	6.6	6.8	6.7	7.0
J	9.85	9.0	10.5	9.9	9.40	10.5	9.85	9.4	10.6
W	0.66	0.62	0.67	0.66	0.62	0.67	0.66	0.62	0.67

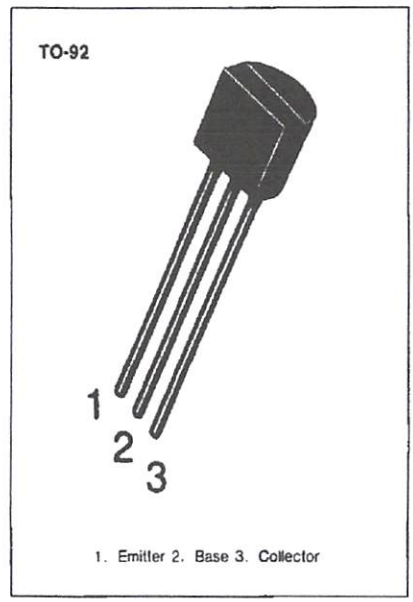
Note – measurements do not include any burring of edges.

ransistors
SC945

USHA (INDIA) LTD

AUDIO FREQUENCY AMPLIFIER
HIGH FREQUENCY OSC.

- Complement to KSA733
- Collector-Base Voltage V_{CBO}=60V
- High Current Gain Bandwidth Product f_T=300MHz (Typ)



ABSOLUTE MAXIMUM RATINGS (T_a=25°C)

Characteristic	Symbol	Rating	Unit
Collector-Base Voltage	V_{CBO}	60	V
Collector-Emitter Voltage	V_{CEO}	50	V
Emitter-Base Voltage	V_{EBO}	5	V
Collector Current	I_C	150	mA
Collector Dissipation	P_C	250	mW
Junction Temperature	Tj	150	°C
Storage Temperature	Tstg	−55~150	°C

ELECTRICAL CHARACTERISTICS (T_a=25°C)

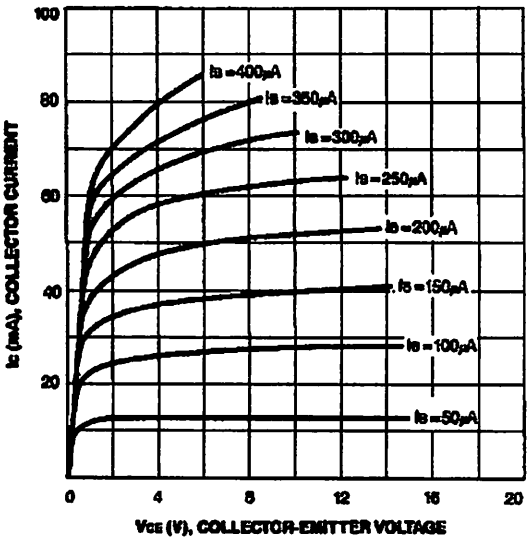
Characteristic	Symbol	Test Conditions	Min	Typ	Max	Unit
Collector-Base Breakdown Voltage	BV_{CBO}	I_C=100μA, I_E=0	60			V
Collector-Emitter Breakdown Voltage	BV_{CEO}	I_C=10mA, I_B=0	50			V
Emitter-Base Breakdown Voltage	BV_{EBO}	I_E=10μA, I_C=0	5			V
Collector Cut-off Current	I_{CBO}	V_{CB}=40V, I_E=0			0.1	μA
Emitter Cut-off Current	I_{EBO}	V_{EB}=3V, I_C=0			0.1	μA
DC Current Gain	h_{FE}	V_{CE}=6V, I_C=1.0mA	40		700	
Collector-Emitter Saturation Voltage	V_{CE} (sat)	I_C=100mA, I_B=10mA		0.15	0.3	V
Current-Gain-Bandwidth Product	f_T	V_{CE}=6V, I_C=10mA		300		MHz
Output Capacitance	Cob	V_{CB}=6V, I_E=0 f=1MHz		2.5		pF
Noise Figure	NF	V_{CE}=6V, I_E=−0.5mA f=1KHz, R_S=500Ω		4.0		dB

h_{FE} CLASSIFICATION

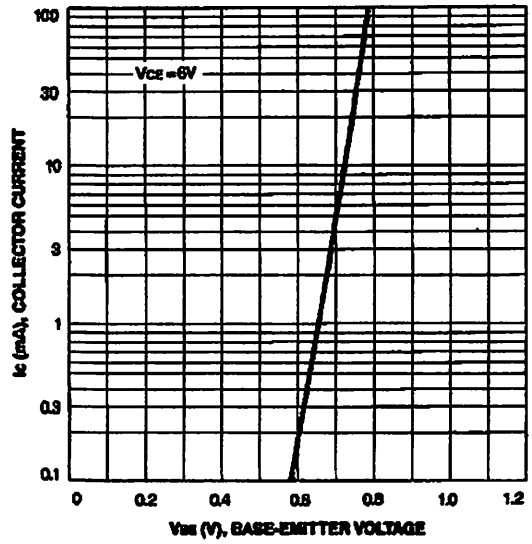
Classification	R	O	Y	G	L
h_{FE}	40-80	70-140	120-240	200-400	350-700

ISO 14001 CERTIFIED
AN ISO 9002 COMPANY

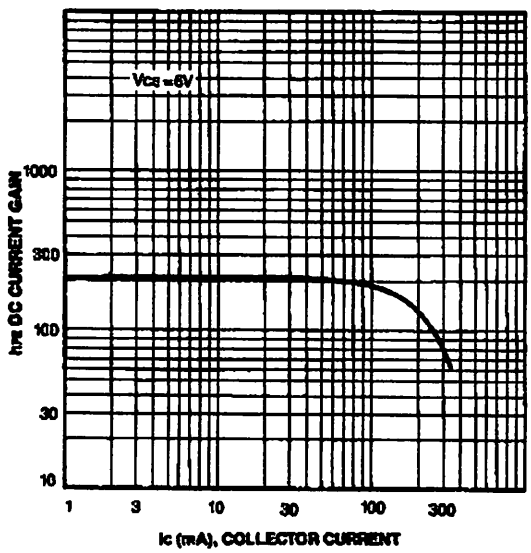
STATIC CHARACTERISTIC



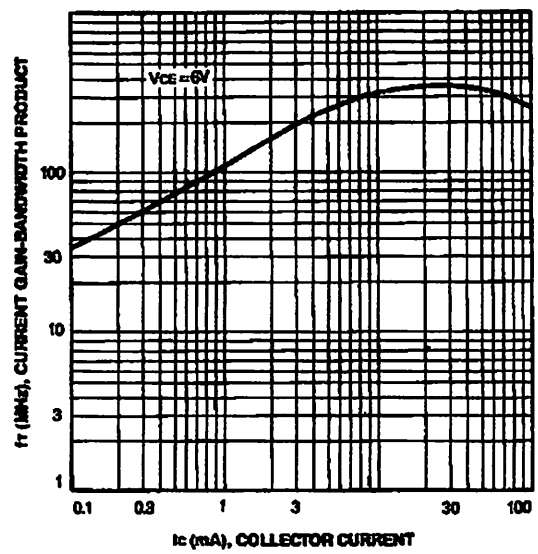
TRANSFER CHARACTERISTIC



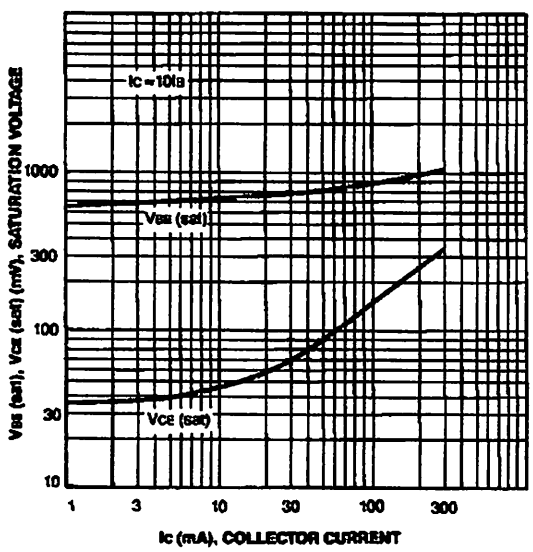
DC CURRENT GAIN



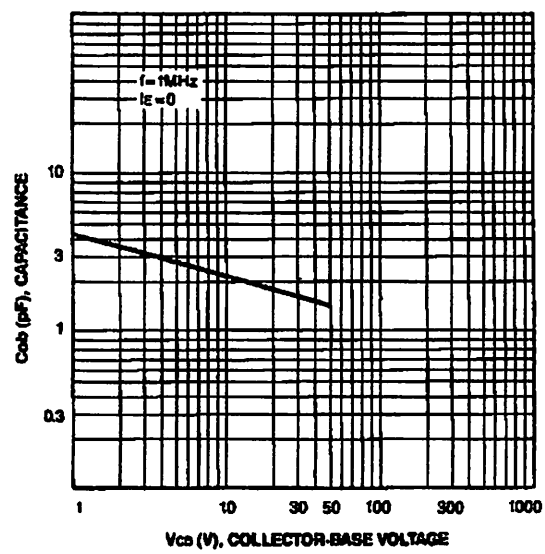
CURRENT GAIN BANDWIDTH PRODUCT



BASE-EMITTER SATURATION VOLTAGE
COLLECTOR-EMITTER SATURATION VOLTAGE



OUTPUT CAPACITANCE



NEC

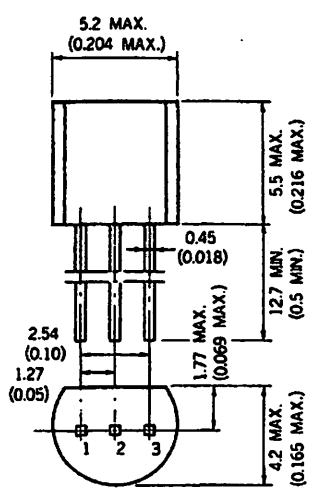
NPN SILICON TRANSISTOR
2SC945

DESCRIPTION The 2SC945 is designed for use in driver stage of AF amplifier and low speed switching.

FEATURES

- High Voltage — LV_{CEO} : 50 V MIN.
- Excellent h_{FE} Linearity
  h_{FE1} (0.1 mA)/h_{FE2} (1.0 mA) : 0.92 TYP.

PACKAGE DIMENSIONS
in millimeters (inches)



1. EMITTER
2. COLLECTOR
3. BASE

EIAJ : SC 43B
JEDEC : TO 92
IEC : PA33

ABSOLUTE MAXIMUM RATINGS

Maximum Temperatures

- Storage Temperature . . . . . −55 to +125 °C
- Junction Temperature . . . . . +125 °C Maximum

Maximum Power Dissipation (Ta = 25 °C)

- Total Power Dissipation . . . . . 250 mW

Maximum Voltages and Currents (Ta = 25 °C)

- V_{CBO} Collector to Base Voltage . . . . . 60 V
- V_{CEO} Collector to Emitter Voltage . . . . . 50 V
- V_{EBO} Emitter to Base Voltage . . . . . 5.0 V
- I_C Collector Current . . . . . 100 mA
- I_B Base Current . . . . . 20 mA

ELECTRICAL CHARACTERISTICS (Ta = 25 °C)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
h_{FE1}	DC Current Gain	50	185			V_{CE}=6.0 V, I_C=0.1 mA
h_{FE2}	DC Current Gain	90	200	600		V_{CE}=6.0 V, I_C=1.0 mA
NF	Noise Figure		0.8	15	dB	V_{CE}=6.0 V, I_C=0.1 mA, R_G=2.0 kΩ, f=1.0 kHz
f_T	Gain Bandwidth Product	150	250	450	MHz	V_{CE}=6.0 V, I_E=−10 mA
C_{ob}	Collector to Base Capacitance		3.0	4.0	pF	V_{CB}=6.0 V, I_E=0, f=1.0 MHz
I_{CBO}	Collector Cutoff Current			100	nA	V_{CB}=60 V, I_E=0
I_{EBO}	Emitter Cutoff Current			100	nA	V_{EB}=5.0 V, I_C=0
V_{BE}	Base to Emitter Voltage	0.55	0.62	0.65	V	V_{CE}=6.0 V, I_C=1.0 mA
$V_{CE(sat)}$	Collector Saturation Voltage		0.15	0.3	V	I_C=100 mA, I_B=10 mA
$V_{BE(sat)}$	Base Saturation Voltage		0.86	1.0	V	I_C=100 mA, I_B=10 mA

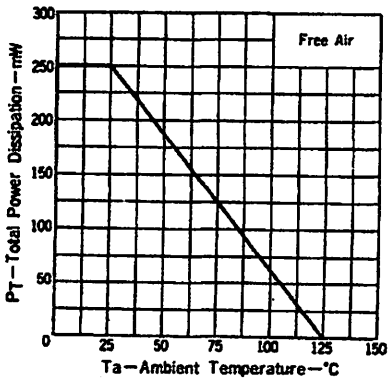
Classification of h_{FE2}

Rank	R	Q	P	K
Range	90 − 180	135 − 270	200 − 400	300 − 600

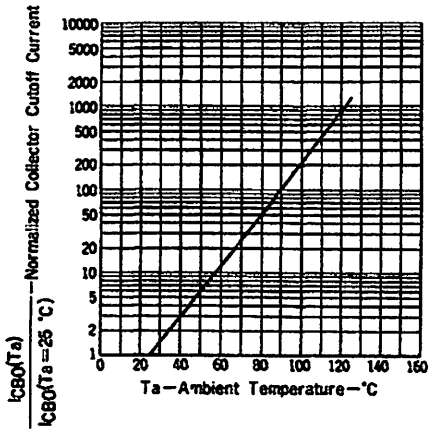
h_{FE2} Test Conditions : V_{CE} = 6.0 V, I_C = 1.0 mA

TYPICAL CHARACTERISTICS (Ta = 25 °C unless otherwise noted)

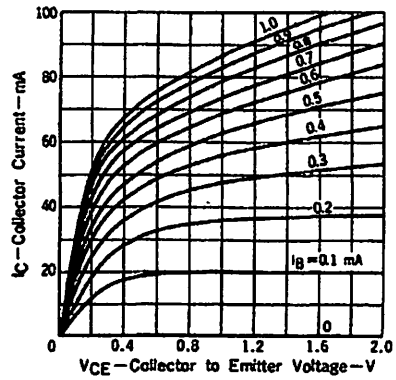
TOTAL POWER DISSIPATION vs. AMBIENT TEMPERATURE



NORMALIZED COLLECTOR CUTOFF CURRENT vs. AMBIENT TEMPERATURE

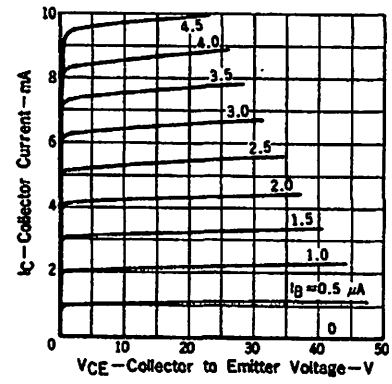


COLLECTOR CURRENT vs. COLLECTOR TO EMITTER VOLTAGE

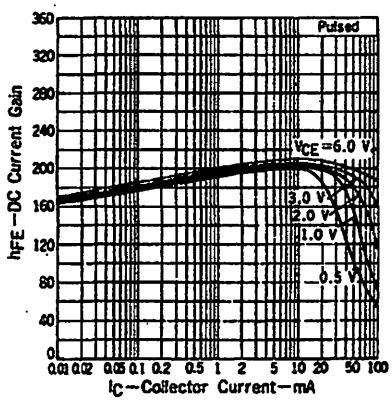


4-3

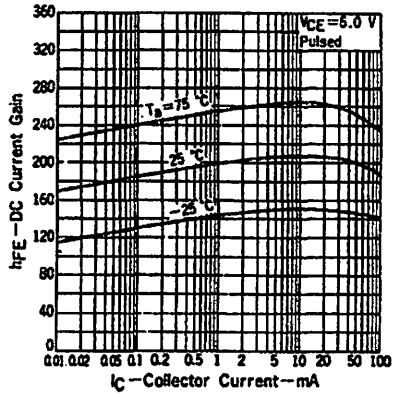
COLLECTOR CURRENT vs. COLLECTOR TO EMITTER VOLTAGE



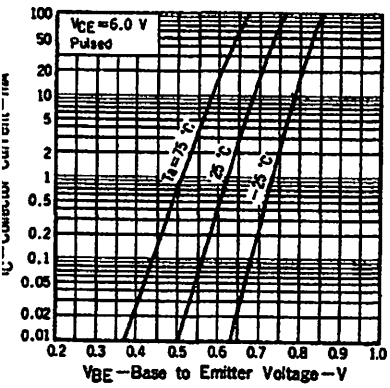
DC CURRENT GAIN vs. COLLECTOR CURRENT



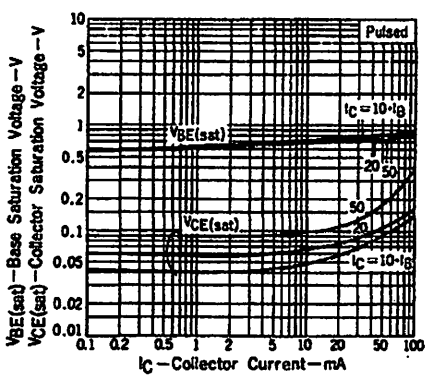
DC CURRENT GAIN vs. COLLECTOR CURRENT



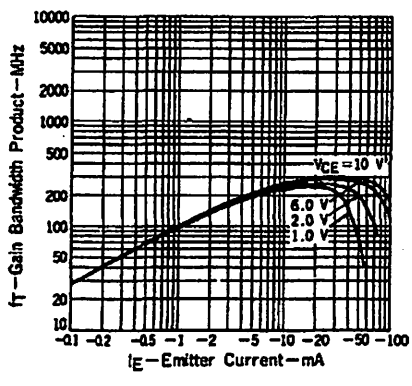
COLLECTOR CURRENT vs. BASE TO EMITTER VOLTAGE



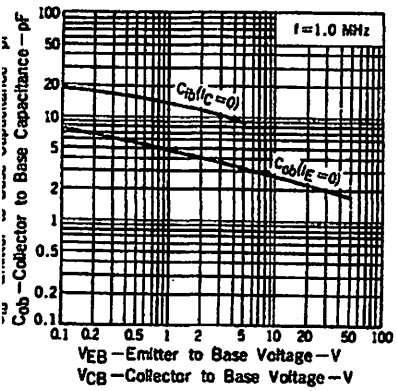
COLLECTOR AND BASE SATURATION VOLTAGE vs. COLLECTOR CURRENT



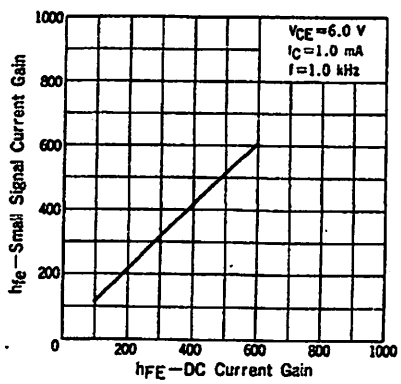
GAIN BANDWIDTH PRODUCT vs. EMITTER CURRENT



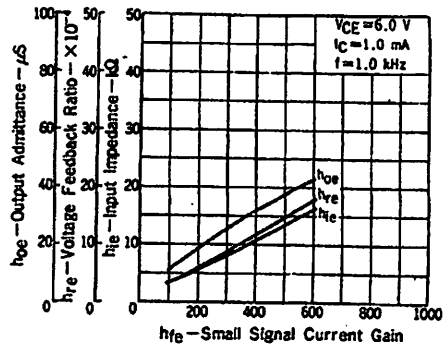
MITTER TO BASE AND COLLECTOR TO
ASE CAPACITANCE vs. REVERSE VOLTAGE



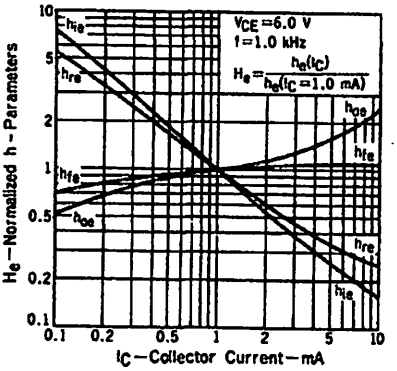
SMALL SIGNAL CURRENT GAIN
vs. DC CURRENT GAIN



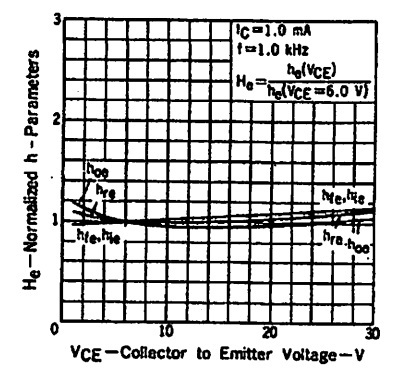
INPUT IMPEDANCE, VOLTAGE FEEDBACK
RATIO AND OUTPUT ADMITTANCE
vs. SMALL SIGNAL CURRENT GAIN



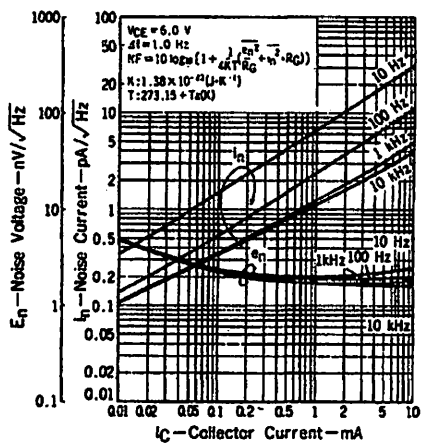
NORMALIZED h-PARAMETERS
vs. COLLECTOR CURRENT



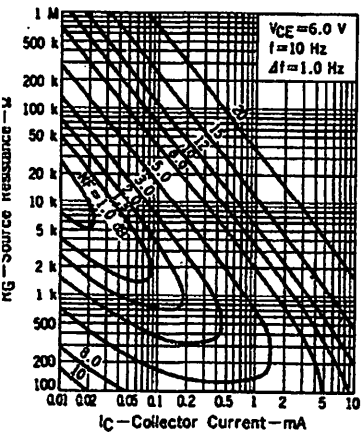
NORMALIZED h-PARAMETERS
vs. COLLECTOR TO EMITTER VOLTAGE



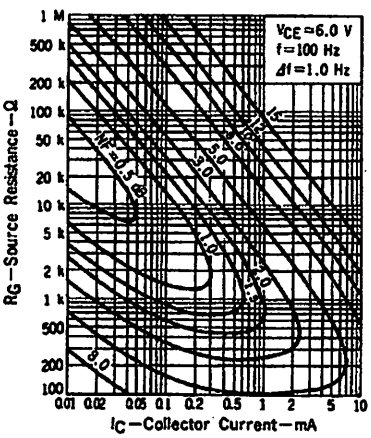
En AND In vs. COLLECTOR CURRENT



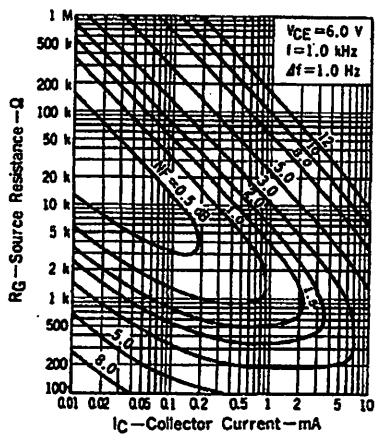
NOISE FIGURE MAP 1



NOISE FIGURE MAP 2



NOISE FIGURE MAP 3



atures

- mpatible with MCS®-51 Products
- Bytes of In-System Programmable (ISP) Flash Memory
- Endurance: 1000 Write/Erase Cycles
- V to 5.5V Operating Range
- lly Static Operation: 0 Hz to 33 MHz
- ree-level Program Memory Lock
- 6 x 8-bit Internal RAM
- Programmable I/O Lines
- ree 16-bit Timer/Counters
- ght Interrupt Sources
- il Duplex UART Serial Channel
- w-power Idle and Power-down Modes
- errupt Recovery from Power-down Mode
- tchdog Timer
- al Data Pointer
- wer-off Flag
- st Programming Time
- xible ISP Programming (Byte and Page Mode)
- een (Pb/Halide-free) Packaging Option

Description

AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K s of in-system programmable Flash memory. The device is manufactured using el's high-density nonvolatile memory technology and is compatible with the indus- tandard 80C51 instruction set and pinout. The on-chip Flash allows the program ory to be reprogrammed in-system or by a conventional nonvolatile memory pro- mer. By combining a versatile 8-bit CPU with in-system programmable Flash on nolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a y-flexible and cost-effective solution to many embedded control applications.

AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes AM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a ector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, clock circuitry. In addition, the AT89S52 is designed with static logic for operation n to zero frequency and supports two software selectable power saving modes. dle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and rupt system to continue functioning. The Power-down mode saves the RAM con- but freezes the oscillator, disabling all other chip functions until the next interrupt rdware reset.

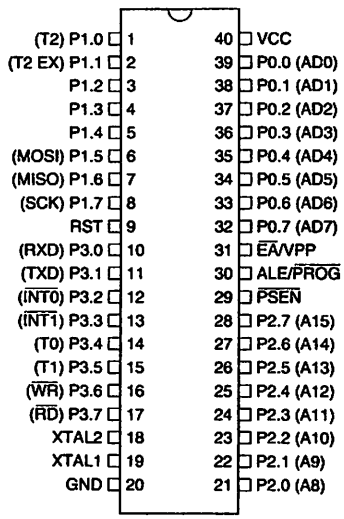
8-bit Microcontroller with 8K Bytes In-System Programmable Flash

AT89S52

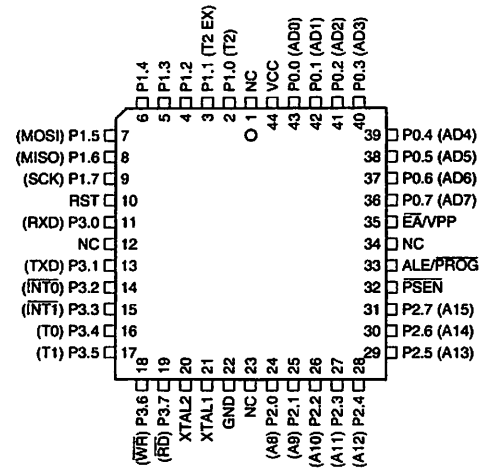
1919C–MICRO–3/05

Pin Configurations

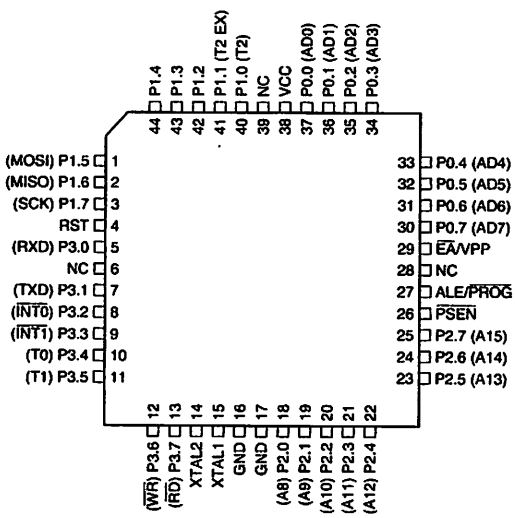
40-lead PDIP



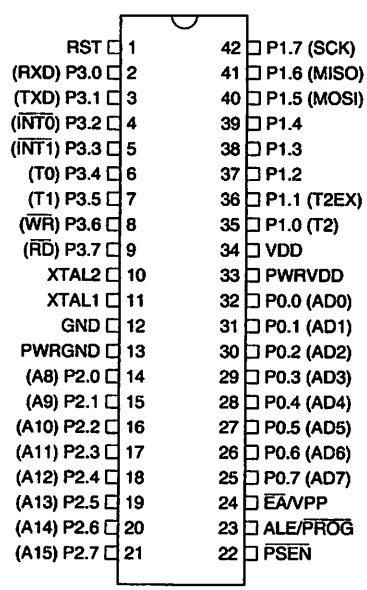
2.3 44-lead PLCC



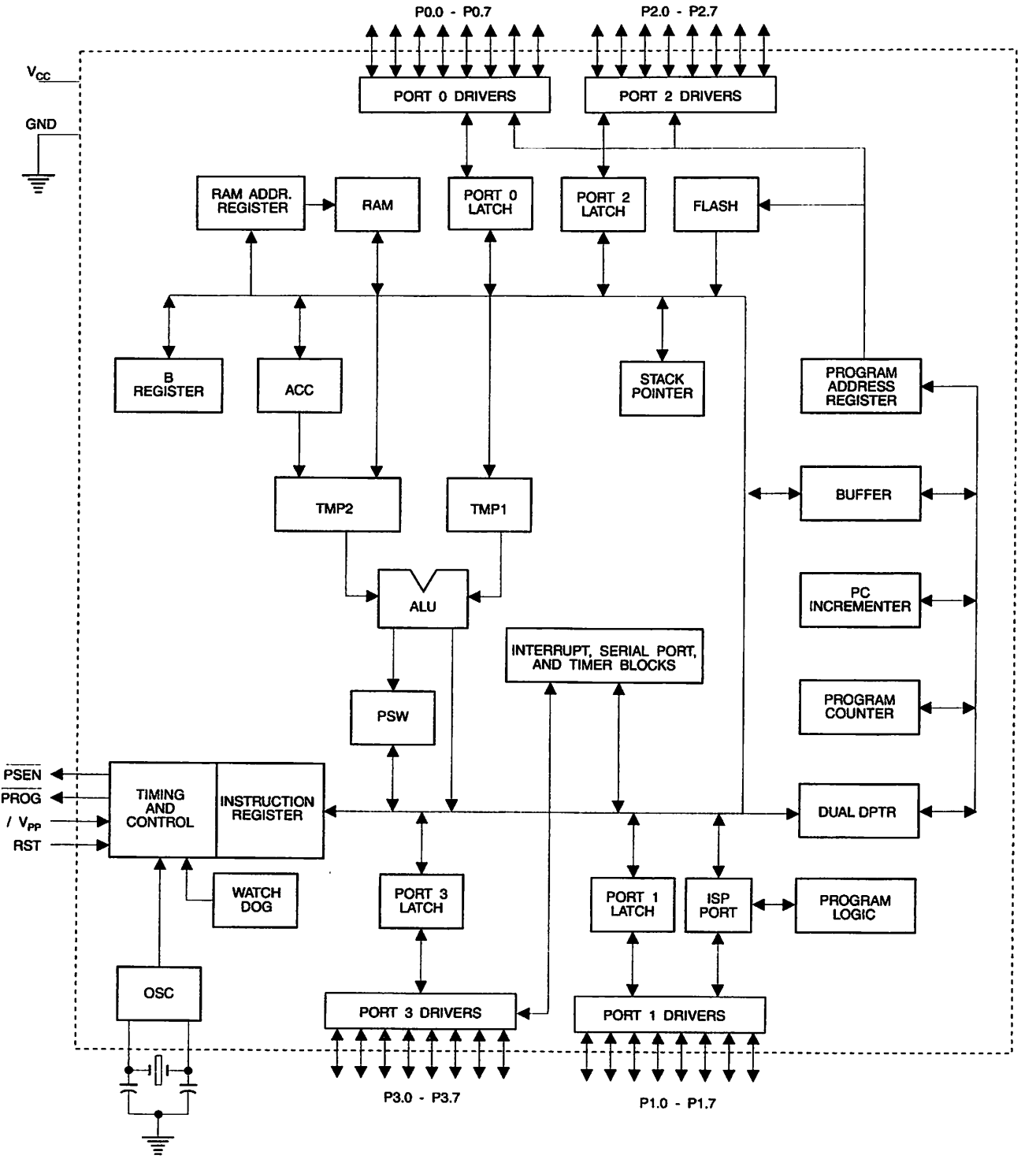
44-lead TQFP



2.4 42-lead PDIP



Block Diagram



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

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

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. **External pull-ups are required during program verification.**

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

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

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pull-ups.

Port 3 receives some control signals for Flash programming and verification.

Port 3 also serves the functions of various special features of the AT89S52, as shown in the following table.

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

RST

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

ALE/$\overline{PROG}$

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

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

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

$\overline{\text{PSEN}}$

Program Store Enable ($\overline{\text{PSEN}}$) is the read strobe to external program memory.

When the AT89S52 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{\text{EA}}$/VPP

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

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming.

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 5-1.

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

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers: Control and status bits are contained in registers T2CON (shown in Table 5-2) and T2MOD (shown in Table 10-2) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers: The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.

e 5-1. AT89S52 SFR Map and Reset Values

8H								0FFH
0H	B 00000000							0F7H
8H								0EFH
0H	ACC 00000000							0E7H
8H								0DFH
0H	PSW 00000000							0D7H
8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		0CFH
0H								0C7H
8H	IP XX000000							0BFH
0H	P3 11111111							0B7H
8H	IE 0X000000							0AFH
0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX	0A7H
8H	SCON 00000000	SBUF XXXXXXXX						9FH
0H	P1 11111111							97H
8H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0	8FH
0H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	PCON 0XXX0000	87H

e 5-2. T2CON – Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

nbol	**Function**
	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
F2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
LK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
LK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
R	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

e 5-3. AUXR: Auxiliary Register

(R Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

	–	–	–	WDIDLE	DISRTO	–	–	DISALE
Bit	7	6	5	4	3	2	1	0

Reserved for future expansion

ALE Disable/Enable ALE

DISALE Operating Mode

0 ALE is emitted at a constant rate of 1/6 the oscillator frequency

1 ALE is active only during a MOVX or MOVC instruction

RTO Disable/Enable Reset out

DISRTO

0 Reset pin is driven High after WDT times out

1 Reset pin is input only

IDLE Disable/Enable WDT in IDLE mode

WDIDLE

0 WDT continues to count in IDLE mode

1 WDT halts counting in IDLE mode

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

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

e 5-4. AUXR1: Auxiliary Register 1

R1 Address = A2H Reset Value = XXXXXXX0B

Not Bit Addressable

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

Reserved for future expansion

Data Pointer Register Select

DPS

0 Selects DPTR Registers DP0L, DP0H

1 Selects DPTR Registers DP1L, DP1H

Memory Organization

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

Program Memory

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

On the AT89S52, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions which use direct addressing access the SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Watchdog Timer (One-time Enabled with Reset-out)

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

Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When

WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC = 1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

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

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

UART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers' operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 5-2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 10-1. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 10-1. Timer 2 Operating Modes

RCLK +TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

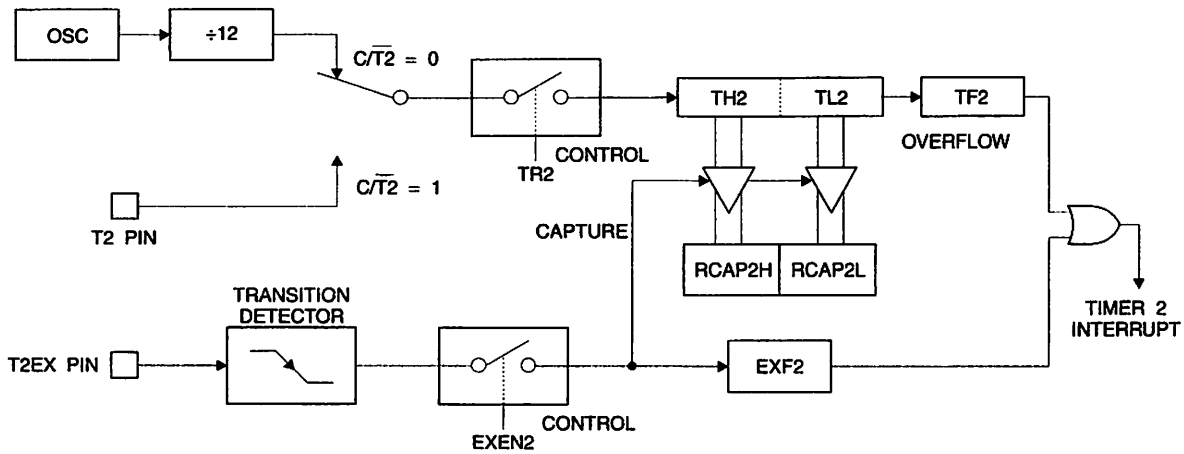
Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 10-1.

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 10-2). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

re 10-1. Timer in Capture Mode



e 10-2. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

	–	–	–	–	–	–	T2OE	DCEN
Bit	7	6	5	4	3	2	1	0

ıbol	**Function**
	Not implemented, reserved for future
E	Timer 2 Output Enable bit
N	When set, this bit allows Timer 2 to be configured as an up/down counter

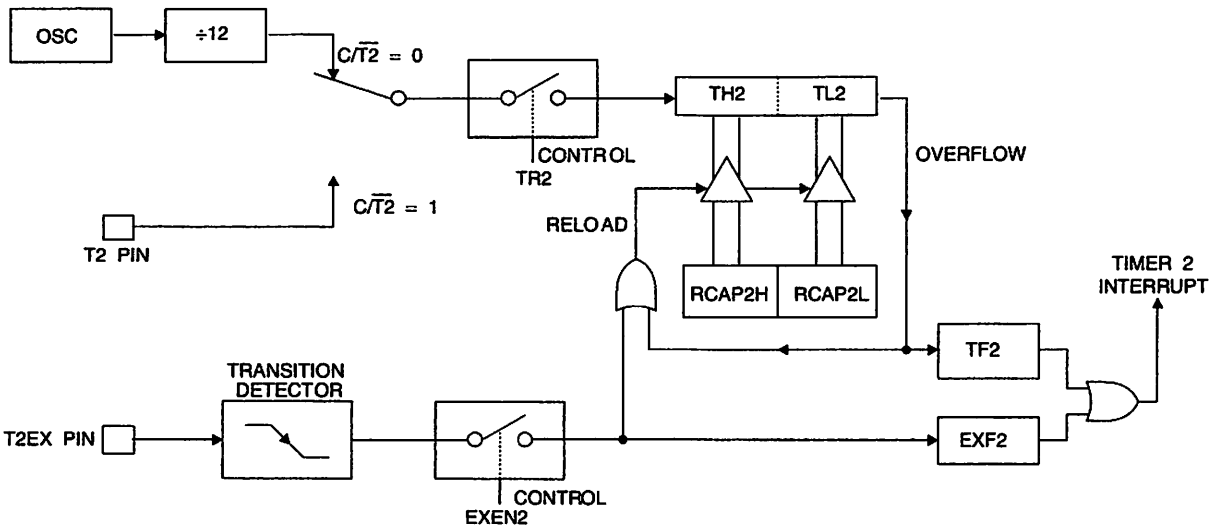
Figure 10-2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 10-2. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

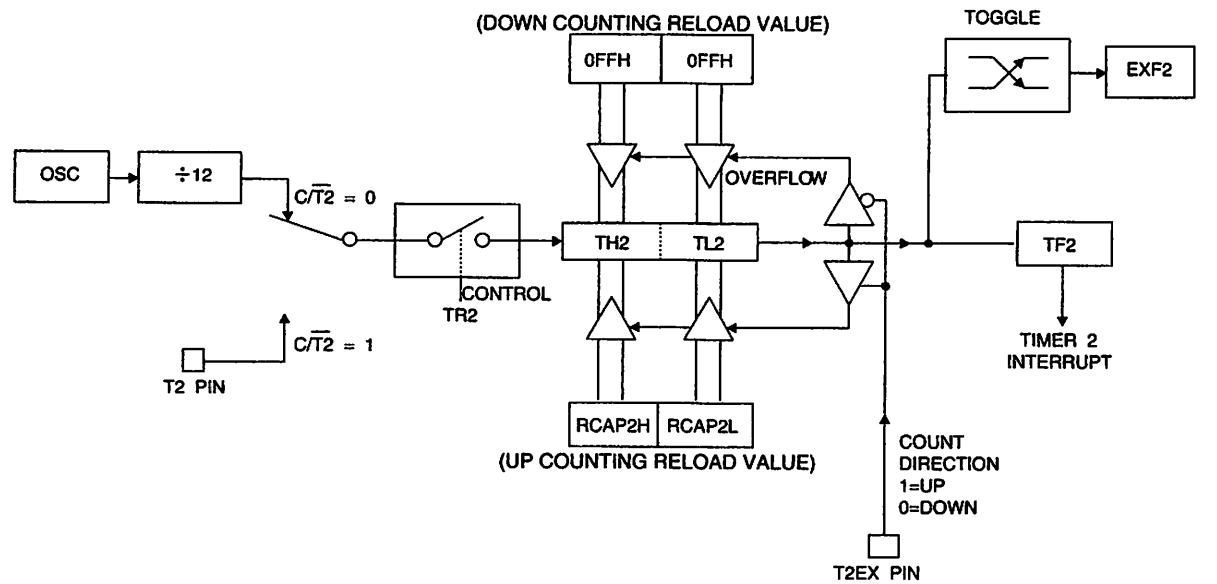
A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

re 10-2. Timer 2 Auto Reload Mode (DCEN = 0)



re 10-3. Timer 2 Auto Reload Mode (DCEN = 1)



Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 5-2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 11-1.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation ($C/\overline{T2} = 0$). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

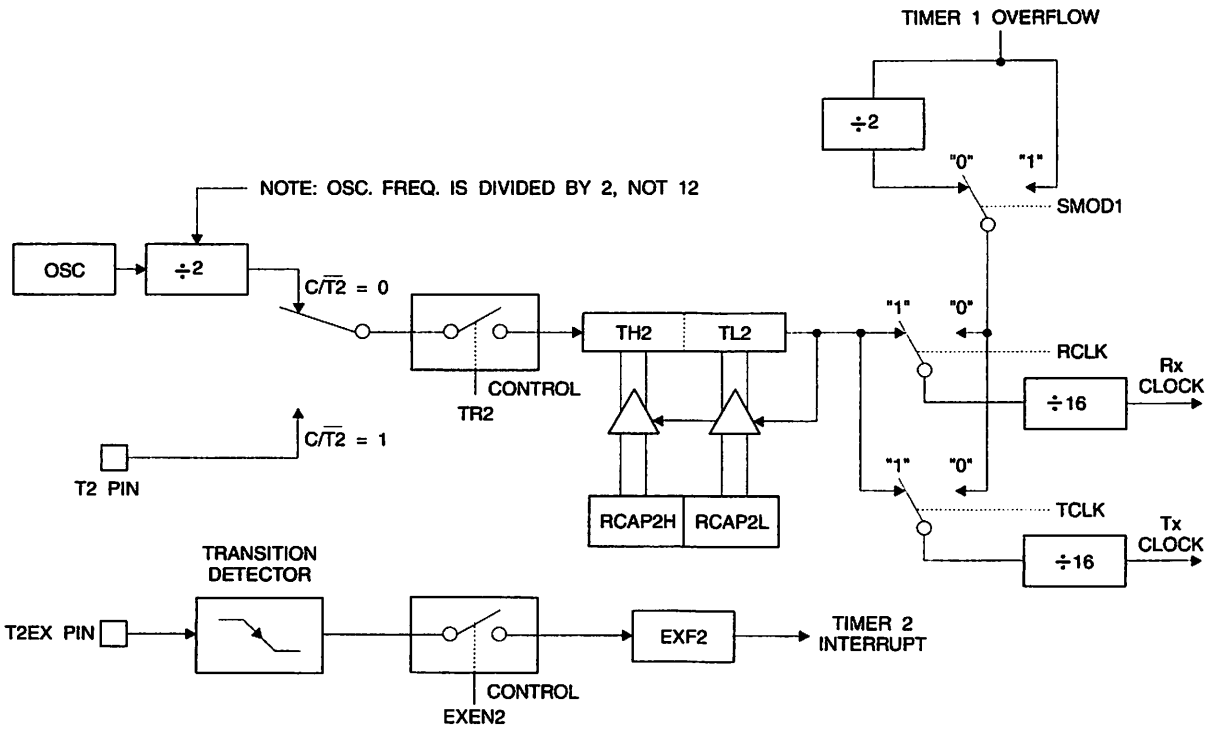
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536\text{-RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 11-1. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus, when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

re 11-1. Timer 2 in Baud Rate Generator Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 12-1. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz (for a 16-MHz operating frequency).

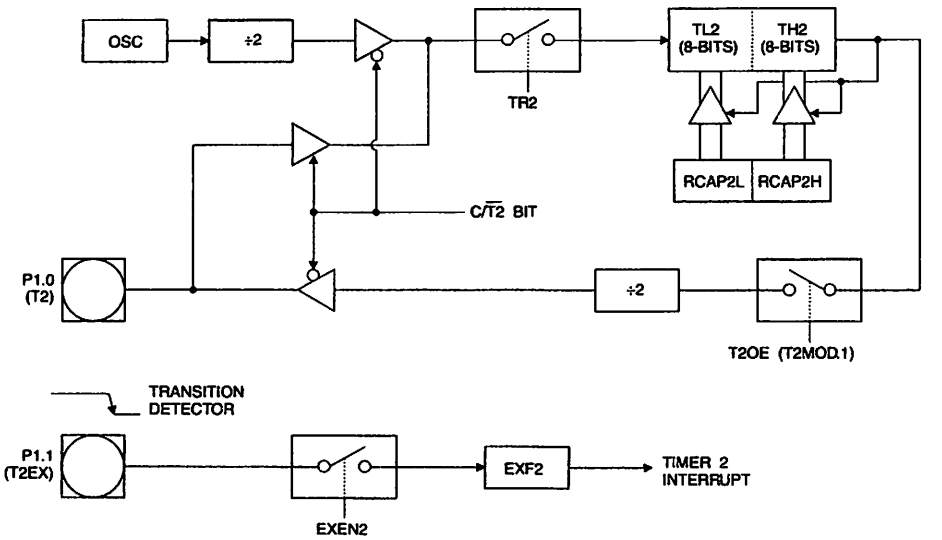
To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536-(\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

re 12-1. Timer 2 in Clock-Out Mode



Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 13-1.

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

Note that Table 13-1 shows that bit position IE.6 is unimplemented. User software should not write a 1 to this bit position, since it may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

e 13-1. Interrupt Enable (IE) Register

MSB)							(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0

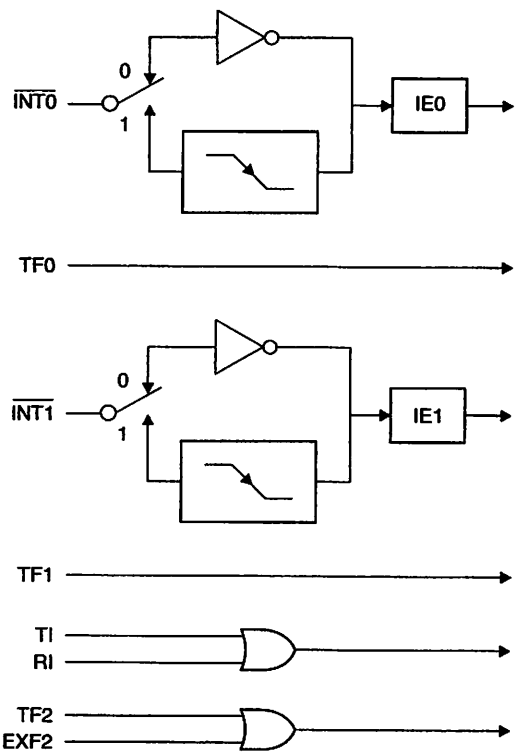
Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

ıbol	Position	Function
	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Timer 2 interrupt enable bit.
	IE.4	Serial Port interrupt enable bit.
	IE.3	Timer 1 interrupt enable bit.
	IE.2	External interrupt 1 enable bit.
	IE.1	Timer 0 interrupt enable bit.
	IE.0	External interrupt 0 enable bit.

r software should never write 1s to reserved bits, because they may be used in future AT89 products.

'e 13-1. Interrupt Sources



Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 16-1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 16-2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

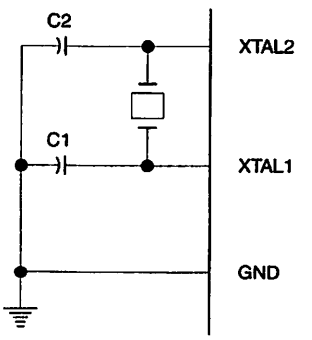
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

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

Power-down Mode

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

Figure 16-1. Oscillator Connections



Note: 1. C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 16-2. External Clock Drive Configuration

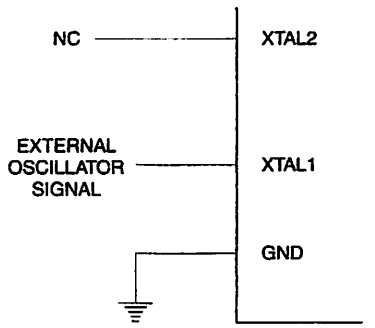


Table 16-1. Status of External Pins During Idle and Power-down Modes

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

Program Memory Lock Bits

The AT89S52 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in Table 17-1.

Table 17-1. Lock Bit Protection Modes

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

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

Programming the Flash – Parallel Mode

The AT89S52 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S52 code memory array is programmed byte-by-byte.

Programming Algorithm: Before programming the AT89S52, the address, data, and control signals should be set up according to the "Flash Programming Modes" (Table 22-1) and Figure 22-1 and Figure 22-2. To program the AT89S52, take the following steps:

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

$\overline{Data}$ Polling: The AT89S52 features $\overline{Data}$ Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

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

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

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

(000H) = 1EH indicates manufactured by Atmel
(100H) = 52H indicates AT89S52
(200H) = 06H

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

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

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

Programming the Flash – Serial Mode

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

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

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

Serial Programming Algorithm

To program and verify the AT89S52 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:
   a. Apply power between VCC and GND pins.
   b. Set RST pin to "H".

If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 33 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.

2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 16.
3. The Code array is programmed one byte at a time in either the Byte or Page mode. The write cycle is self-timed and typically takes less than 0.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal device operation.

Power-off sequence (if needed):

1. Set XTAL1 to "L" (if a crystal is not used).
2. Set RST to "L".
3. Turn V_{CC} power off.

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

Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 24-1.

Programming Interface – Parallel Mode

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

Most major worldwide programming vendors offer support for the Atmel AT89 microcontroller series. Please contact your local programming vendor for the appropriate software revision.

22-1. Flash Programming Modes

	V_{CC}	RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.4-0 Address	P1.7-0 Address
Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A12-8	A7-0
Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A12-8	A7-0
Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
Lock Bits 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	X 0000	00H
Device ID	5V	H	L	H	H	L	L	L	L	L	52H	X 0001	00H
Device ID	5V	H	L	H	H	L	L	L	L	L	06H	X 0010	00H

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

Figure 22-1. Programming the Flash Memory (Parallel Mode)

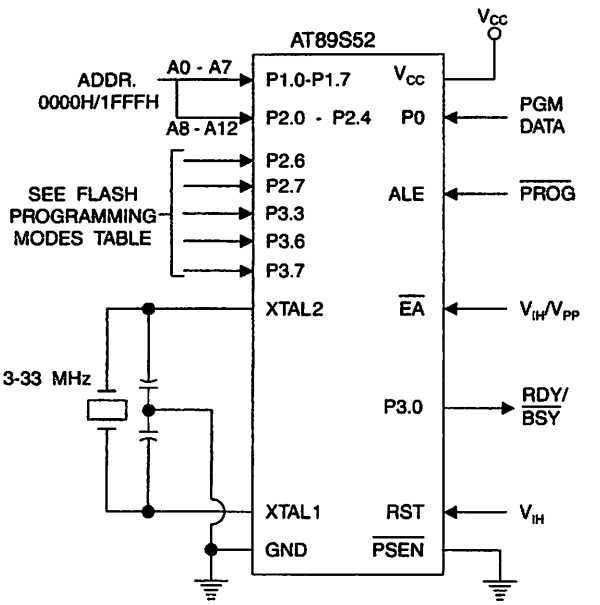
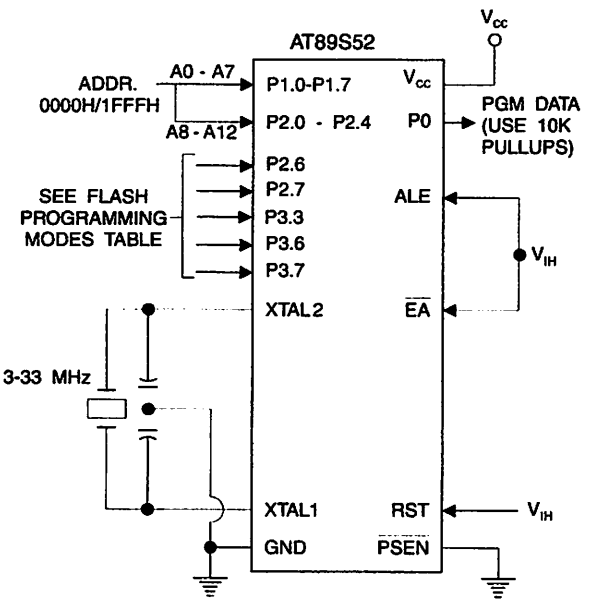


Figure 22-2. Verifying the Flash Memory (Parallel Mode)



Flash Programming and Verification Characteristics (Parallel Mode)

20°C to 30°C, V_{CC} = 4.5 to 5.5V

bol	Parameter	Min	Max	Units
	Programming Supply Voltage	11.5	12.5	V
	Programming Supply Current		10	mA
	V_{CC} Supply Current		30	mA
CL	Oscillator Frequency	3	33	MHz
	Address Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
	Address Hold After $\overline{PROG}$	48 t_{CLCL}		
	Data Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
	Data Hold After $\overline{PROG}$	48 t_{CLCL}		
	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48 t_{CLCL}		
	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
	V_{PP} Hold After $\overline{PROG}$	10		µs
	$\overline{PROG}$ Width	0.2	1	µs
	Address to Data Valid		48 t_{CLCL}	
	$\overline{ENABLE}$ Low to Data Valid		48 t_{CLCL}	
	Data Float After $\overline{ENABLE}$	0	48 t_{CLCL}	
	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
	Byte Write Cycle Time		50	µs

e 23-1. Flash Programming and Verification Waveforms – Parallel Mode

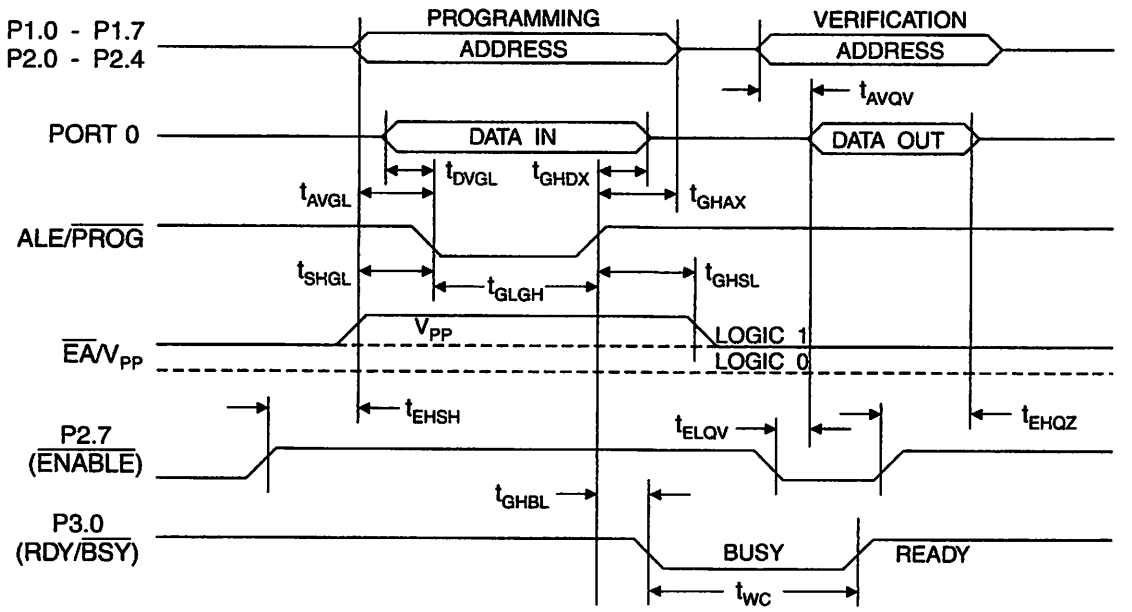
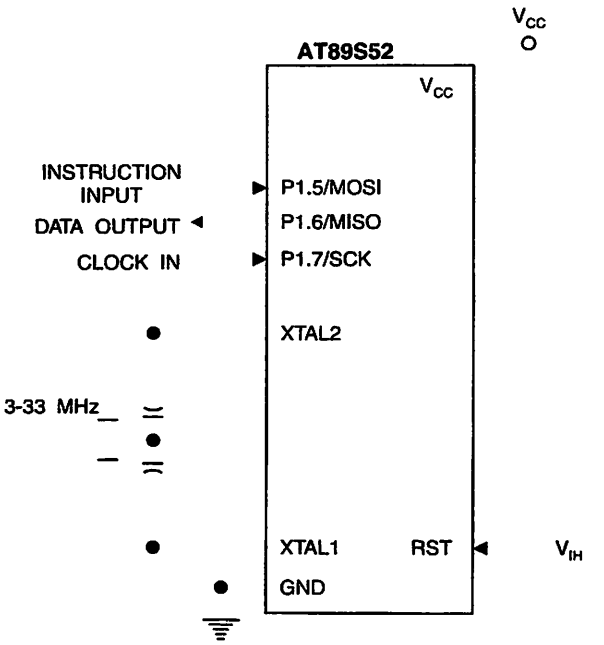
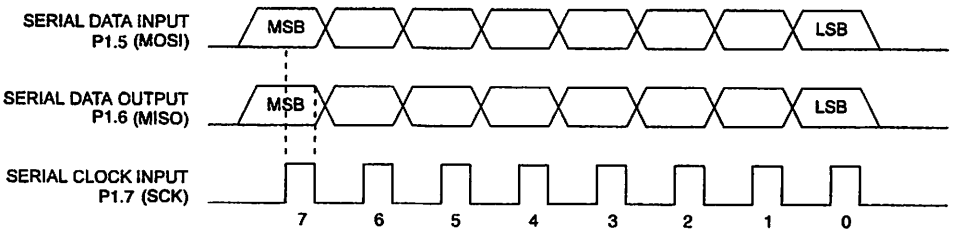


Figure 23-2. Flash Memory Serial Downloading



Flash Programming and Verification Waveforms – Serial Mode

Figure 24-1. Serial Programming Waveforms



24-1. Serial Programming Instruction Set

	Instruction Format				
uction	Byte 1	Byte 2	Byte 3	Byte 4	Operation
ramming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output on MISO)	Enable Serial Programming while RST is high
Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
d Program Memory e Mode)	0010 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Read data from Program memory in the byte mode
e Program Memory e Mode)	0100 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Write data to Program memory in the byte mode
e Lock Bits[1]	1010 1100	1110 00 B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (1).
d Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xxx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a "1")
d Signature Bytes	0010 1000	xxx A12 A11 A10 A9 A8	A7 xxx xxx0	Signature Byte	Read Signature Byte
d Program Memory e Mode)	0011 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
e Program Memory e Mode)	0101 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

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

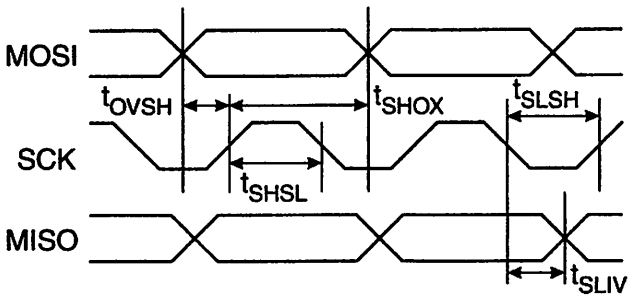
   } Each of the lock bit modes needs to be activated sequentially before Mode 4 can be executed.

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

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

Serial Programming Characteristics

e 25-1. Serial Programming Timing



25-1. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 4.0 - 5.5V (Unless Otherwise Noted)

bol	Parameter	Min	Typ	Max	Units
CL	Oscillator Frequency	3		33	MHz
	Oscillator Period	30			ns
	SCK Pulse Width High	8 t_{CLCL}			ns
	SCK Pulse Width Low	8 t_{CLCL}			ns
	MOSI Setup to SCK High	t_{CLCL}			ns
	MOSI Hold after SCK High	2 t_{CLCL}			ns
	SCK Low to MISO Valid	10	16	32	ns
E	Chip Erase Instruction Cycle Time			500	ms
	Serial Byte Write Cycle Time			64 t_{CLCL} + 400	µs

Absolute Maximum Ratings*

ating Temperature	-55°C to +125°C
ige Temperature	-65°C to +150°C
ige on Any Pin Respect to Ground	-1.0V to +7.0V
mum Operating Voltage	6.6V
Output Current	15.0 mA

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

DC Characteristics

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

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

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

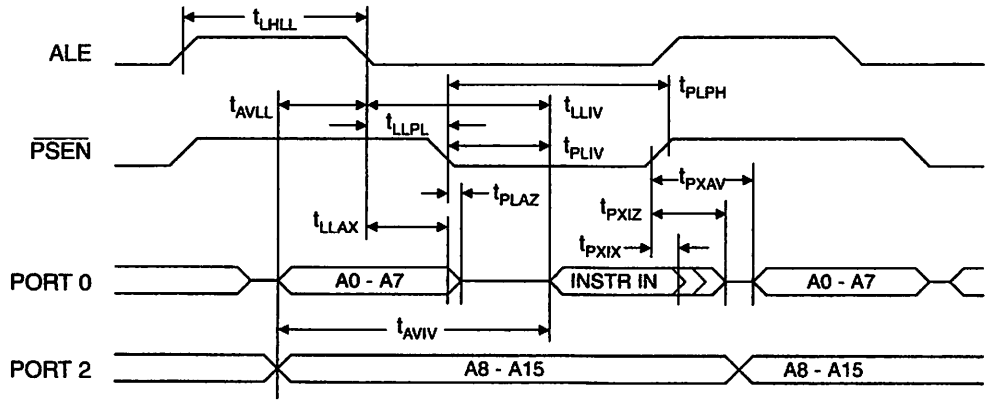
AC Characteristics

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

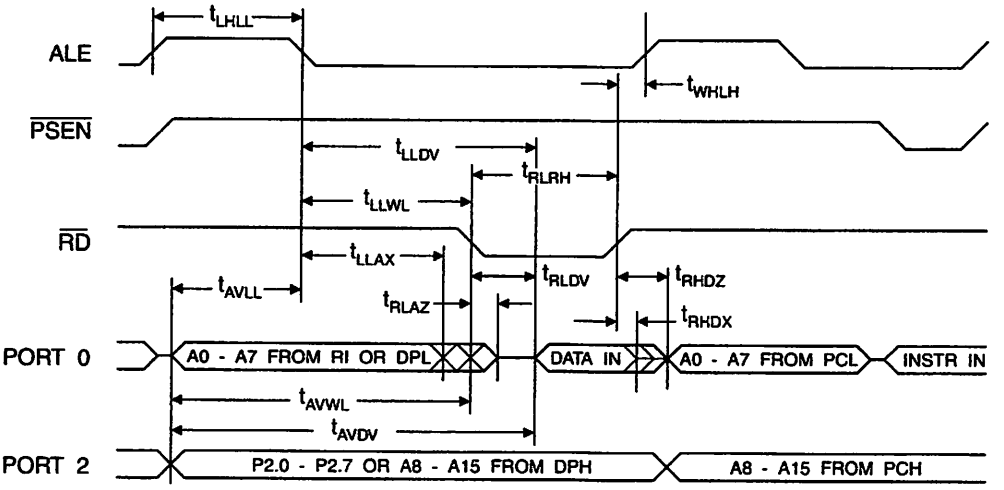
External Program and Data Memory Characteristics

bol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
CL	Oscillator Frequency			0	33	MHz
	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
	Address Valid to ALE Low	43		t_{CLCL}-25		ns
	Address Hold After ALE Low	48		t_{CLCL}-25		ns
	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-25		ns
	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-45		ns
	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-60	ns
	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-25	ns
	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
	Address to Valid Instruction In		312		$5t_{CLCL}$-80	ns
	$\overline{PSEN}$ Low to Address Float		10		10	ns
	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
	Data Hold After $\overline{RD}$	0		0		ns
	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-30		ns
	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-130		ns
	Data Hold After $\overline{WR}$	33		t_{CLCL}-25		ns
	$\overline{RD}$ Low to Address Float		0		0	ns
	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-25	t_{CLCL}+25	ns

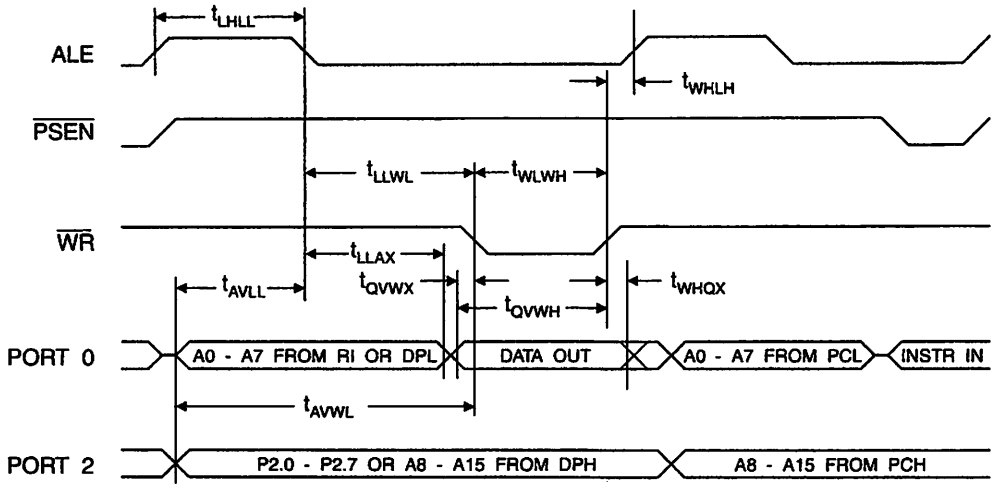
External Program Memory Read Cycle



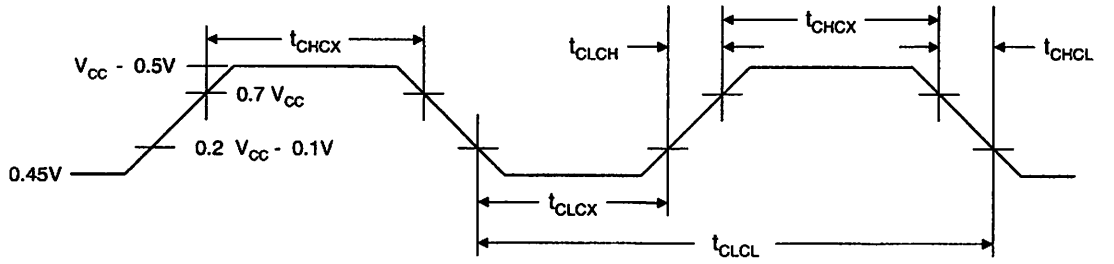
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

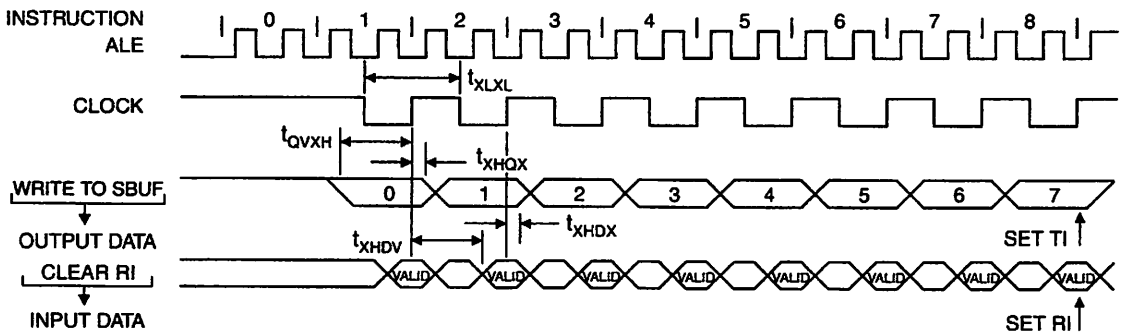
bol	Parameter	Min	Max	Units
:L	Oscillator Frequency	0	33	MHz
	Clock Period	30		ns
	High Time	12		ns
	Low Time	12		ns
	Rise Time		5	ns
	Fall Time		5	ns

Serial Port Timing: Shift Register Mode Test Conditions

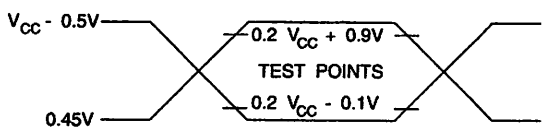
alues in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

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

Shift Register Mode Timing Waveforms

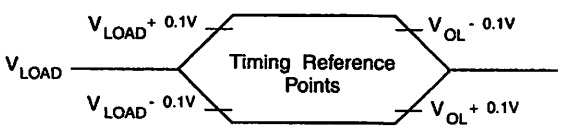


AC Testing Input/Output Waveforms[1]



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

Float Waveforms[1]



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

Ordering Information

Standard Package

eed Hz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S52-24AC	44A	Commercial (0°C to 70°C)
		AT89S52-24JC	44J	
		AT89S52-24PC	40P6	
		AT89S52-24SC	42PS6	
		AT89S52-24AI	44A	Industrial (-40°C to 85°C)
		AT89S52-24JI	44J	
		AT89S52-24PI	40P6	
		AT89S52-24SI	42PS6	
33	4.5V to 5.5V	AT89S52-33AC	44A	Commercial (0°C to 70°C)
		AT89S52-33JC	44J	
		AT89S52-33PC	40P6	
		AT89S52-33SC	42PS6	

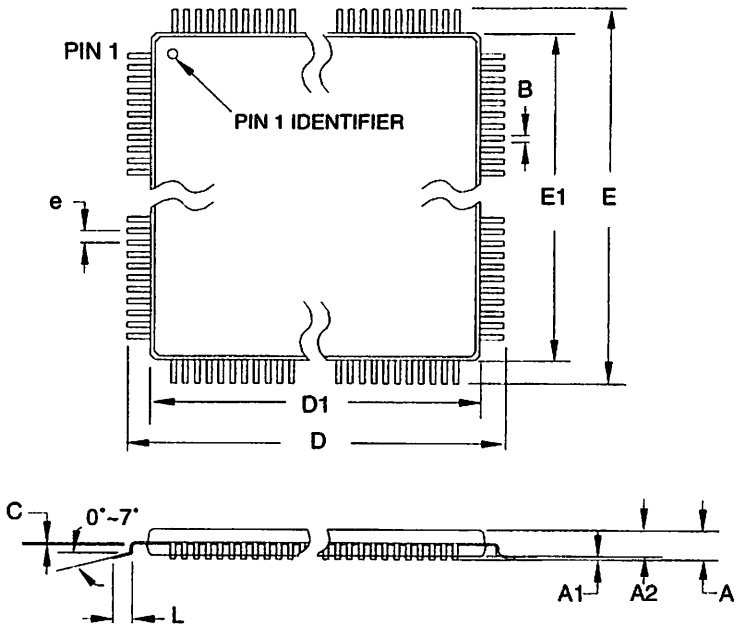
Green Package Option (Pb/Halide-free)

eed Hz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S52-24AU	44A	Industrial (-40°C to 85°C)
		AT89S52-24JU	44J	
		AT89S52-24PU	40P6	

Package Type	
	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
	44-lead, Plastic J-leaded Chip Carrier (PLCC)
	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
6	42-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

Packaging Information

44A – TQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

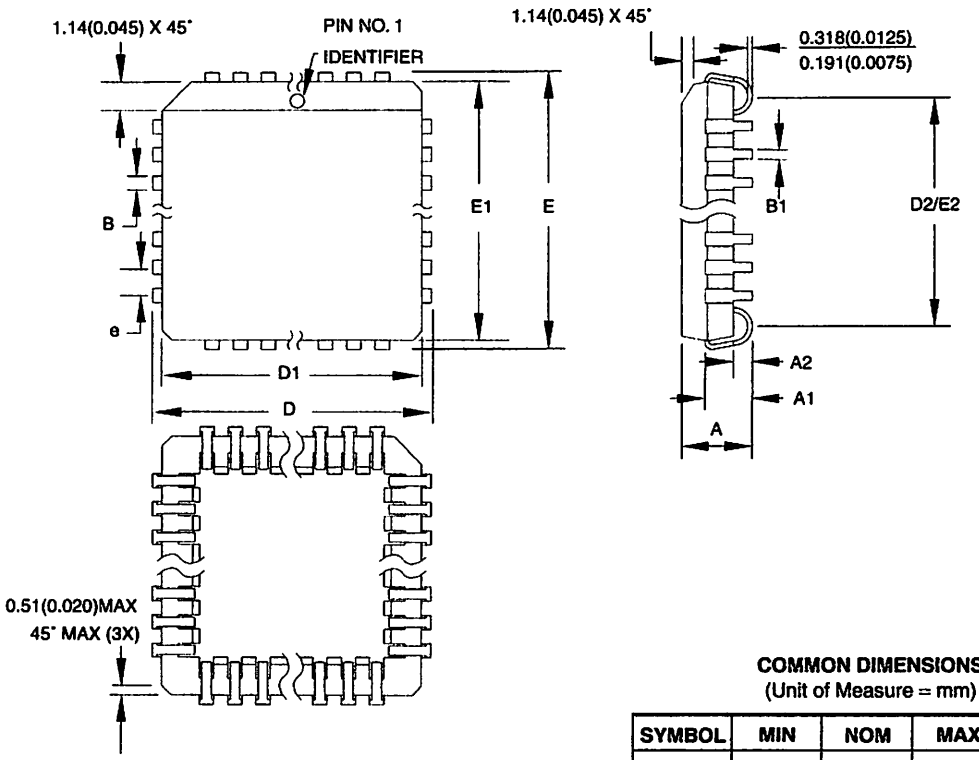
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

lotes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

44J – PLCC



COMMON DIMENSIONS
(Unit of Measure = mm)

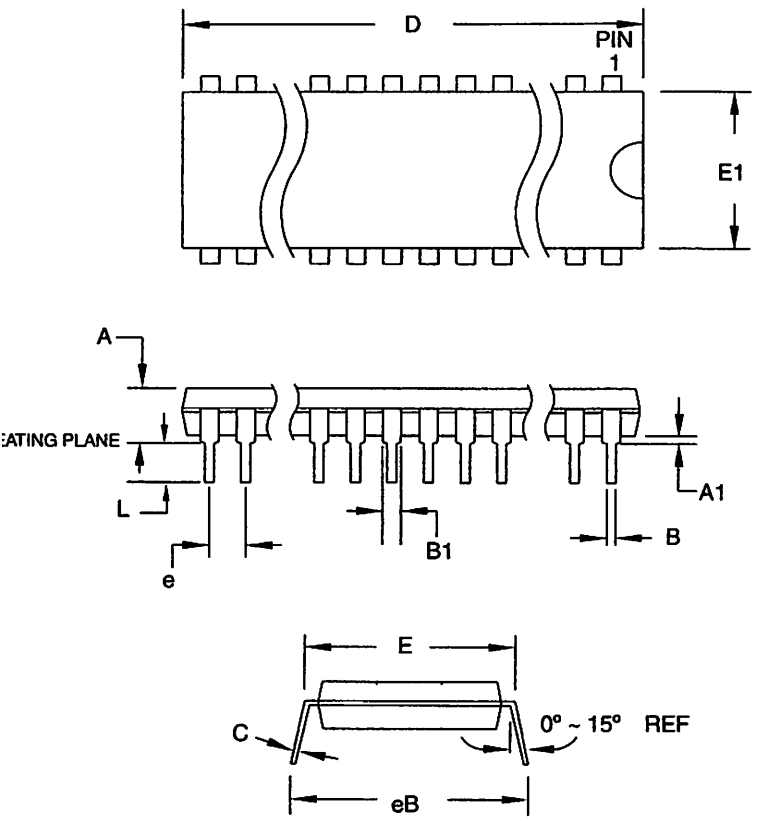
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

10/04/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B

40P6 – PDIP



COMMON DIMENSIONS
(Unit of Measure = mm)

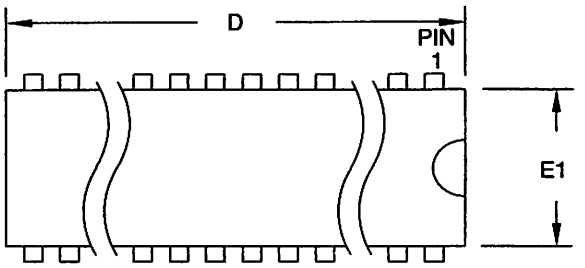
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

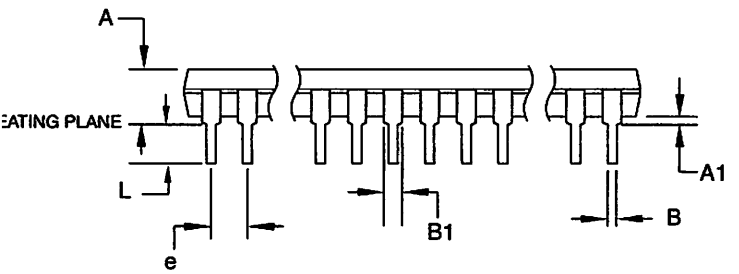
Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

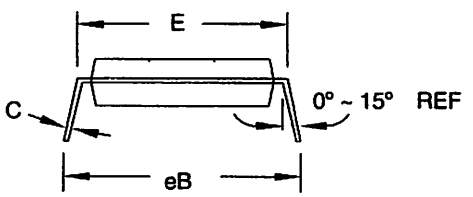
09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

42PS6 – PDIP







COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.83	
A1	0.51	–	–	
D	36.70	–	36.96	Note 2
E	15.24	–	15.88	
E1	13.46	–	13.97	Note 2
B	0.38	–	0.56	
B1	0.76	–	1.27	
L	3.05	–	3.43	
C	0.20	–	0.30	
eB	–	–	18.55	
e	1.78 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

11/6/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**42PS6**, 42-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	42PS6	A

ɔduction

The Serial Port is harder to interface than the Parallel Port. In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port. (SPP)

So what are the advantages of using serial data transfer rather than parallel?

1. Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables than they are for parallel.

2. You don't need as many wires than parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.

3. Infra Red devices have proven quite popular recently. You may of seen many electronic diaries and palmtop computers which have infra red capabilities build in. However could you imagine transmitting 8 bits of data at the one time across the room and being able to (from the devices point of view) decipher which bits are which? Therefore serial transmission is used where one bit is sent at a time. IrDA-1 (The first infra red specifications) was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to 3/16th of a RS232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.

4. Microcontroller's have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method (You may also require a Strobe).

t One : Hardware (PC's)

Hardware Properties

Devices which use serial cables for their communication are split into two categories. These are DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipment are devices such as your modem, TA adapter, plotter etc while Data Terminal Equipment is your Computer or Terminal.

The electrical specifications of the serial port is contained in the EIA (Electronics Industry Association) RS232C standard. It states many parameters such as -

1. A "Space" (logic 0) will be between +3 and +25 Volts.
2. A "Mark" (Logic 1) will be between -3 and -25 Volts.
3. The region between +3 and -3 volts is undefined.
4. An open circuit voltage should never exceed 25 volts. (In Reference to GND)
5. A short circuit current should not exceed 500mA. The driver should be able to handle this without damage. (Take note of this one!)

Above is no where near a complete list of the EIA standard. Line Capacitance, Maximum Baud Rates etc are also included. For more information please consult the EIA RS232-E standard. It is interesting to note however, that the RS232C standard specifies a maximum baud rate of 20,000 BPS!, which is rather slow by today's standards. Revised standards, EIA-232D & EIA-232E were released, in 1987 & 1991 respectively.

Serial Ports come in two "sizes", There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus you will require a female connector on your device. Below is a table of pin connections for the 9 pin and 25 pin D-Type connectors.

erial Pinouts (D25 and D9 Connectors)

D-Type-25 Pin No.	D-Type-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send

Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

Table 1 : D Type 9 Pin and D Type 25 Pin Connectors

Pin Functions

Abbreviation	Full Name	Function
TD	Transmit Data	Serial Data Output (TXD)
RD	Receive Data	Serial Data Input (RXD)
CTS	Clear to Send	This line indicates that the Modem is ready to exchange data.
DCD	Data Carrier Detect	When the modem detects a "Carrier" from the modem at the other end of the phone line, this Line becomes active.
DSR	Data Set Ready	This tells the UART that the modem is ready to establish a link.
DTR	Data Terminal Ready	This is the opposite to DSR. This tells the Modem that the UART is ready to link.
RTS	Request To Send	This line informs the Modem that the UART is ready to exchange data.
RI	Ring Indicator	Goes active when modem detects a ringing signal from the PSTN.

Null Modems

A Null Modem is used to connect two DTE's together. This is commonly used as a cheap way to network games or to transfer files between computers using Zmodem Protocol, Xmodem Protocol etc. This can also be used with many Microprocessor Development Systems.

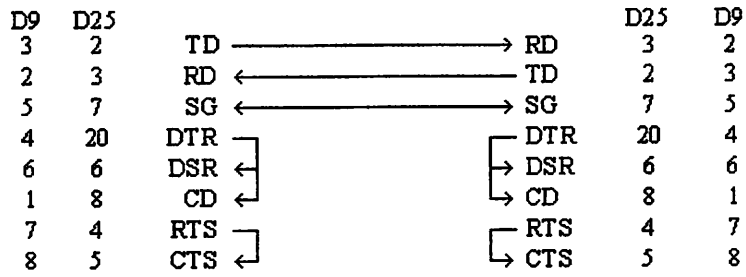


Figure 1 : Null Modem Wiring Diagram

Above is my preferred method of wiring a Null Modem. It only requires 3 wires (TD, RD & SG) to be wired straight through thus is more cost effective to use with long cable runs. The theory of operation is reasonably easy. The aim is to make to computer think it is talking to a modem rather than another computer. Any data transmitted from the first computer must be received by the second thus TD is connected to RD. The second computer must have the same set-up thus RD is connected to TD. Signal Ground (SG) must also be connected so both grounds are common to each computer.

The Data Terminal Ready is looped back to Data Set Ready and Carrier Detect on both computers. When the Data Terminal Ready is asserted active, then the Data Set Ready and Carrier Detect immediately become active. At this point the computer thinks the Virtual Modem to which it is connected is ready and has detected the carrier of the other modem.

All left to worry about now is the Request to Send and Clear To Send. As both computers communicate together at the same speed, flow control is not needed thus these two lines are also linked together on each computer. When the computer wishes to send data, it asserts the Request to Send high and as it's hooked together with the Clear to Send, It immediately gets a reply that it is ok to send and does so.

Notice that the ring indicator is not connected to anything of each end. This line is only used to tell the computer that there is a ringing signal on the phone line. As we don't have a modem connected to the phone line this is left disconnected.

oopBack Plug

LoopBack Plug

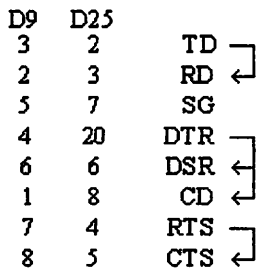


Figure 2 : Loopback Plug Wiring Diagram

This loopback plug can come in extremely handy when writing Serial / RS232 Communications Programs. It has the receive and transmit lines connected together, so that anything transmitted out of the Serial Port is immediately received by the same port. If you connect this to a Serial Port an load a Terminal Program, anything you type will be immediately displayed on the screen. This can be used with the examples later in this tutorial.

Please note that this is not intended for use with Diagnostic Programs and thus will probably not work. For these programs you require a differently wired Loop Back plug which may vary from program to program.

)TE / DCE Speeds

We have already talked briefly about DTE & DCE. A typical Data Terminal Device is a computer and a typical Data Communications Device is a Modem. Often people will talk about DTE to DCE or DCE to DCE speeds. DTE to DCE is the speed between your modem and computer, sometimes referred to as your terminal speed. This should run at faster speeds than the DCE to DCE speed. DCE to DCE is the link between modems, sometimes called the line speed.

Most people today will have 28.8K or 33.6K modems. Therefore we should expect the DCE to DCE speed to be either 28.8K or 33.6K. Considering the high speed of the modem we should expect the DTE to DCE speed to be about 115,200 BPS.(Maximum Speed of the 16550a UART) This is where some people often fall into a trap. The communications program which they use have settings for DCE to DTE speeds. However they see 9.6 KBPS, 14.4 KBPS etc and think it is your modem speed.

Today's Modems should have Data Compression build into them. This is very much like PK-ZIP but the software in your modem compresses and decompresses the data. When set up correctly you can expect compression ratios of 1:4 or even higher. 1 to 4 compression would be typical of a text file. If we were transferring that text file at 28.8K (DCE-DCE), then when the modem compresses it you are actually transferring 115.2 KBPS between computers and thus have a DCE-DTE speed of 115.2 KBPS. Thus this is why the DCE-DTE should be much higher than your modem's connection speed.

Some modem manufacturers quote a maximum compression ratio as 1:8. Lets say for example its on a new 33.6 KBPS modem then we may get a maximum 268,800 BPS transfer between modem and UART. If you only have a 16550a which can do 115,200 BPS tops, then you would be missing out on a extra bit of performance. Buying a 16C650 should fix your problem with a maximum transfer rate of 230,400 BPS.

However don't abuse your modem if you don't get these rates. These are MAXIMUM compression ratios. In some instances if you try to send a already compressed file, your modem can spend more time trying the compress it, thus you get a transmission speed less than your modem's connection speed. If this occurs try turning off your data compression. This should be fixed on newer modems. Some files compress easier than others thus any file which compresses easier is naturally going to have a higher compression ratio.

'low Control

So if our DTE to DCE speed is several times faster than our DCE to DCE speed the PC can send data to your modem at 115,200 BPS. Sooner or later data is going to get lost as buffers overflow, thus flow control is used. Flow control has two basic varieties, Hardware or Software.

Software flow control, sometimes expressed as Xon/Xoff uses two characters Xon and Xoff. Xon is normally indicated by the ASCII 17 character where as the ASCII 19 character is used for Xoff. The modem will only have a small buffer so when the computer fills it up the modem sends a Xoff character to tell the computer to stop sending data. Once the modem has room for more data it then sends a Xon character and the computer sends more data. This type of flow control has the advantage that it doesn't require any more wires as the characters are sent via the TD/RD lines. However on slow links each character requires 10 bits which can slow communications down.

Hardware flow control is also known as RTS/CTS flow control. It uses two wires in your serial cable rather than extra characters transmitted in your data lines. Thus hardware flow control will not

slow down transmission times like Xon-Xoff does. When the computer wishes to send data it takes active the Request to Send line. If the modem has room for this data, then the modem will reply by taking active the Clear to Send line and the computer starts sending data. If the modem does not have the room then it will not send a Clear to Send.

The UART (8250 and Compatibles)

UART stands for Universal Asynchronous Receiver / Transmitter. Its the little box of tricks found on your serial card which plays the little games with your modem or other connected devices. Most cards will have the UART's integrated into other chips which may also control your parallel port, games port, floppy or hard disk drives and are typically surface mount devices. The 8250 series, which includes the 16450, 16550, 16650, & 16750 UARTS are the most commonly found type in your PC. Later we will look at other types which can be used in your homemade devices and projects.

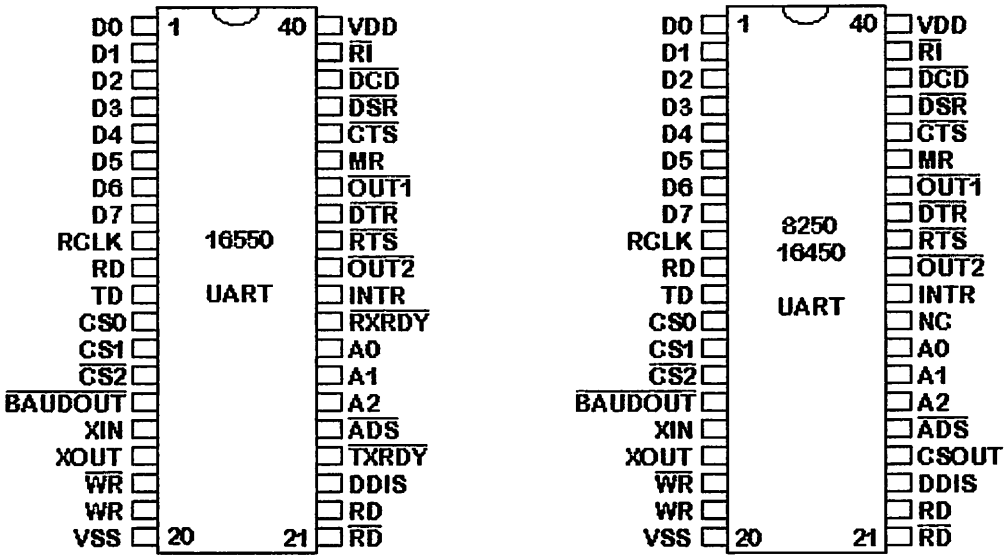


Figure 3 : Pin Diagrams for 16550, 16450 & 8250 UARTs

The 16550 is chip compatible with the 8250 & 16450. The only two differences are pins 24 & 29. On the 8250 Pin 24 was chip select out which functioned only as a indicator to if the chip was active or not. Pin 29 was not connected on the 8250/16450 UARTs. The 16550 introduced two new pins in their place. These are Transmit Ready and Receive Ready which can be implemented with DMA (Direct Memory Access). These Pins have two different modes of operation. Mode 0 supports single transfer DMA where as Mode 1 supports Multi-transfer DMA.

Mode 0 is also called the 16450 mode. This mode is selected when the FIFO buffers are disabled via Bit 0 of the FIFO Control Register or When the FIFO buffers are enabled but DMA Mode Select = 0. (Bit 3 of FCR) In this mode RXRDY is active low when at least one character (Byte) is present in the Receiver Buffer. RXRDY will go inactive high when no more characters are left in the Receiver Buffer. TXRDY will be active low when there are no characters in the Transmit Buffer. It will go inactive high after the first character / byte is loaded into the Transmit Buffer.

Mode 1 is when the FIFO buffers are active and the DMA Mode Select = 1. In Mode 1, RXRDY will go active low when the trigger level is reached or when 16550 Time Out occurs and will return to inactive state when no more characters are left in the FIFO. TXRDY will be active when no characters are present in the Transmit Buffer and will go inactive when the FIFO Transmit Buffer is completely Full.

All the UARTs pins are TTL compatible. That includes TD, RD, RI, DCD, DSR, CTS, DTR and RTS which all interface into your serial plug, typically a D-type connector. Therefore RS232 Level Converters (which we talk about in detail later) are used. These are commonly the DS1489 Receiver and the DS1488 as the PC has +12 and -12 volt rails which can be used by these devices. The RS232 Converters will convert the TTL signal into RS232 Logic Levels.

Pin No.	Name	Notes
Pin 1:8	D0:D7	Data Bus
Pin 9	RCLK	Receiver Clock Input. The frequency of this input should equal the receivers baud rate * 16
Pin 10	RD	Receive Data
Pin 11	TD	Transmit Data
Pin 12	CS0	Chip Select 0 - Active High
Pin 13	CS1	Chip Select 1 - Active High
Pin 14	nCS2	Chip Select 2 - Active Low
Pin 15	nBAUDOUT	Baud Output - Output from Programmable Baud Rate Generator. Frequency = (Baud Rate x 16)
Pin 16	XIN	External Crystal Input - Used for Baud Rate Generator Oscillator
Pin 17	XOUT	External Crystal Output
Pin 18	nWR	Write Line - Inverted
Pin 19	WR	Write Line - Not Inverted
Pin 20	VSS	Connected to Common Ground
Pin 21	RD	Read Line - Inverted
Pin 22	nRD	Read Line - Not Inverted
Pin 23	DDIS	Driver Disable. This pin goes low when CPU is reading from UART. Can be connected to Bus Transceiver in case of high capacity data bus.
Pin 24	nTXRDY	Transmit Ready
Pin 25	nADS	Address Strobe. Used if signals are not stable during read or write cycle
Pin 26	A2	Address Bit 2
Pin 27	A1	Address Bit 1
Pin 28	A0	Address Bit 0

Pin 29	nRXRDY	Receive Ready
Pin 30	INTR	Interrupt Output
Pin 31	nOUT2	User Output 2
Pin 32	nRTS	Request to Send
Pin 33	nDTR	Data Terminal Ready
Pin 34	nOUT1	User Output 1
Pin 35	MR	Master Reset
Pin 36	nCTS	Clear To Send
Pin 37	nDSR	Data Set Ready
Pin 38	nDCD	Data Carrier Detect
Pin 39	nRI	Ring Indicator
Pin 40	VDD	+ 5 Volts

Table 2 : Pin Assignments for 16550A UART

The UART requires a Clock to run. If you look at your serial card a common crystal found is either a 1.8432 MHZ or a 18.432 MHZ Crystal. The crystal in connected to the XIN-XOUT pins of the UART using a few extra components which help the crystal to start oscillating. This clock will be used for the Programmable Baud Rate Generator which directly interfaces into the transmit timing circuits but not directly into the receiver timing circuits. For this an external connection mast be made from pin 15 (BaudOut) to pin 9 (Receiver clock in.) Note that the clock signal will be at Baudrate * 16.

If you are serious about pursuing the 16550 UART used in your PC further, then would suggest downloading a copy of the PC16550D data sheet from National Semiconductor, (http://www.natsemi.com) Data sheets are available in .PDF format so you will need Adobe Acrobat Reader to read these. Texas Instruments (http://www.ti.com) has released the 16750 UART which has 64 Byte FIFO's. Data Sheets for the TL16C750 are available from the Texas Instruments Site.

ypes of UARTS (For PC's)

8250 First UART in this series. It contains no scratch register. The 8250A was an improved version of the 8250 which operates faster on the bus side.

8250A This UART is faster than the 8250 on the bus side. Looks exactly the same to software than 16450.

8250B Very similar to that of the 8250 UART.

16450 Used in AT's (Improved bus speed over 8250's). Operates comfortably at 38.4KBPS. Still quite common today.

16550	This was the first generation of buffered UART. It has a 16 byte buffer, however it doesn't work and is replaced with the 16550A.
16550A	Is the most common UART use for high speed communications eg 14.4K & 28.8K Modems. They made sure the FIFO buffers worked on this UART.
16650	Very recent breed of UART. Contains a 32 byte FIFO, Programmable X-On / X-Off characters and supports power management.
16750	Produced by Texas Instruments. Contains a 64 byte FIFO.

t Two : Serial Port's Registers (PC's)

ort Addresses & IRQ's

Name	Address	IRQ
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

Table 3 : Standard Port Addresses

Above is the standard port addresses. These should work for most P.C's. If you just happen to be lucky enough to own a IBM P/S2 which has a micro-channel bus, then expect a different set of addresses and IRQ's. Just like the LPT ports, the base addresses for the COM ports can be read from the BIOS Data Area.

Start Address	Function
0000:0400	COM1's Base Address
0000:0402	COM2's Base Address
0000:0404	COM3's Base Address
0000:0406	COM4's Base Address

Table 4 - COM Port Addresses in the BIOS Data Area;

The above table shows the address at which we can find the Communications (COM) ports addresses in the BIOS Data Area. Each address will take up 2 bytes. The following sample program in C, shows how you can read these locations to obtain the addresses of your communications ports.

```
#include <stdio.h>
#include <dos.h>

void main(void)
{
 unsigned int far *ptraddr;  /* Pointer to location of Port Addresses */
 unsigned int address;       /* Address of Port */
 int a;

 ptraddr=(unsigned int far *)0x00000400;

 for (a = 0; a <  4; a++)
   {
    address = *ptraddr;
    if (address == 0)
                printf("No port found for COM%d \n",a+1);
    else
                printf("Address assigned to COM%d is %Xh\n",a+1,address);
    *ptraddr++;
   }
}
```

able of Registers

Base Address	DLAB	Read/Write	Abr.	Register Name
+ 0	=0	Write	-	Transmitter Holding Buffer
	=0	Read	-	Receiver Buffer
	=1	Read/Write	-	Divisor Latch Low Byte
+ 1	=0	Read/Write	IER	Interrupt Enable Register
	=1	Read/Write	-	Divisor Latch High Byte
+ 2	-	Read	IIR	Interrupt Identification Register
	-	Write	FCR	FIFO Control Register
+ 3	-	Read/Write	LCR	Line Control Register
+ 4	-	Read/Write	MCR	Modem Control Register
+ 5	-	Read	LSR	Line Status Register
+ 6	-	Read	MSR	Modem Status Register
+ 7	-	Read/Write	-	Scratch Register

Table 5 : Table of Registers

)LAB ?

You will have noticed in the table of registers that there is a DLAB column. When DLAB is set to '0' or '1' some of the registers change. This is how the UART is able to have 12 registers (including the scratch register) through only 8 port addresses. DLAB stands for Divisor Latch Access Bit. When DLAB is set to '1' via the line control register, two registers become available from which you can set your speed of communications measured in bits per second.

The UART will have a crystal which should oscillate around 1.8432 MHZ. The UART incorporates a divide by 16 counter which simply divides the incoming clock signal by 16. Assuming we had the 1.8432 MHZ clock signal, that would leave us with a maximum, 115,200 hertz signal making the UART capable of transmitting and receiving at 115,200 Bits Per Second (BPS). That would be fine for some of the faster modems and devices which can handle that speed, but others just wouldn't communicate at all. Therefore the UART is fitted with a Programmable Baud Rate Generator which is controlled by two registers.

Lets say for example we only wanted to communicate at 2400 BPS. We worked out that we would have to divide 115,200 by 48 to get a workable 2400 Hertz Clock. The "Divisor", in this case 48, is stored in the two registers controlled by the "Divisor Latch Access Bit". This divisor can be any number which can be stored in 16 bits (ie 0 to 65535). The UART only has a 8 bit data bus, thus this is where the two registers are used. The first register (Base + 0) when DLAB = 1 stores the "Divisor latch low byte" where as the second register (base + 1 when DLAB = 1) stores the "Divisor latch high byte."

Below is a table of some more common speeds and their divisor latch high bytes & low bytes. Note that all the divisors are shown in Hexadecimal.

Speed (BPS)	Divisor (Dec)	Divisor Latch High Byte	Divisor Latch Low Byte
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

Table 6 : Table of Commonly Used Baudrate Divisors

nterrupt Enable Register (IER)

Bit	Notes
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Enables Low Power Mode (16750)
Bit 4	Enables Sleep Mode (16750)
Bit 3	Enable Modem Status Interrupt
Bit 2	Enable Receiver Line Status Interrupt
Bit 1	Enable Transmitter Holding Register Empty Interrupt
Bit 0	Enable Received Data Available Interrupt

Table 7 : Interrupt Enable Register

The Interrupt Enable Register could possibly be one of the easiest registers on a UART to understand. Setting Bit 0 high enables the Received Data Available Interrupt which generates an interrupt when the receiving register/FIFO contains data to be read by the CPU.

Bit 1 enables Transmit Holding Register Empty Interrupt. This interrupts the CPU when the transmitter buffer is empty. Bit 2 enables the receiver line status interrupt. The UART will interrupt when the receiver line status changes. Likewise for bit 3 which enables the modem status interrupt. Bits 4 to 7 are the easy ones. They are simply reserved. (If only everything was that easy!)

nterrupt Identification Register (IIR)

Bit	Notes		
Bits 6 : 7	Bit 6	Bit 7	
	0	0	No FIFO
	0	1	FIFO Enabled but Unusable
	1	1	FIFO Enabled
Bit 5	64 Byte Fifo Enabled (16750 only)		
Bit 4	Reserved		

Bit 3	0	Reserved on 8250, 16450	
	1	16550 Time-out Interrupt Pending	
Bits 1 : 2	Bit 2	Bit 1	
	0	0	Modem Status Interrupt
	0	1	Transmitter Holding Register Empty Interrupt
	1	0	Received Data Available Interrupt
	1	1	Receiver Line Status Interrupt
Bit 0	0	Interrupt Pending	
	1	No Interrupt Pending	

Table 8 : Interrupt Identification Register

The interrupt identification register is a read only register. Bits 6 and 7 give status on the FIFO Buffer. When both bits are '0' no FIFO buffers are active. This should be the only result you will get from a 8250 or 16450. If bit 7 is active but bit 6 is not active then the UART has it's buffers enabled but are unusable. This occurs on the 16550 UART where a bug in the FIFO buffer made the FIFO's unusable. If both bits are '1' then the FIFO buffers are enabled and fully operational.

Bits 4 and 5 are reserved. Bit 3 shows the status of the time-out interrupt on a 16550 or higher.

Lets jump to Bit 0 which shows whether an interrupt has occurred. If an interrupt has occurred it's status will shown by bits 1 and 2. These interrupts work on a priority status. The Line Status Interrupt has the highest Priority, followed by the Data Available Interrupt, then the Transmit Register Empty Interrupt and then the Modem Status Interrupt which has the lowest priority.

irst In / First Out Control Register (FCR)

Bit	**Notes**		
Bits 6 : 7	Bit 7	Bit 6	Interrupt Trigger Level
	0	0	1 Byte
	0	1	4 Bytes
	1	0	8 Bytes
	1	1	14 Bytes
Bit 5	Enable 64 Byte FIFO (16750 only)		
Bit 4	Reserved		

Bit 3	DMA Mode Select. Change status of RXRDY & TXRDY pins from mode 1 to mode 2.
Bit 2	Clear Transmit FIFO
Bit 1	Clear Receive FIFO
Bit 0	Enable FIFO's

Table 9 : FIFO Control Register

The FIFO register is a write only register. This register is used to control the FIFO (First In / First Out) buffers which are found on 16550's and higher.

Bit 0 enables the operation of the receive and transmit FIFO's. Writing a '0' to this bit will disable the operation of transmit and receive FIFO's, thus you will loose all data stored in these FIFO buffers.

Bit's 1 and 2 control the clearing of the transmit or receive FIFO's. Bit 1 is responsible for the receive buffer while bit 2 is responsible for the transmit buffer. Setting these bits to 1 will only clear the contents of the FIFO and will not affect the shift registers. These two bits are self resetting, thus you don't need to set the bits to '0' when finished.

Bit 3 enables the DMA mode select which is found on 16550 UARTs and higher. More on this later. Bits 4 and 5 are those easy type again, Reserved.

Bits 6 and 7 are used to set the triggering level on the Receive FIFO. For example if bit 7 was set to '1' and bit 6 was set to '0' then the trigger level is set to 8 bytes. When there is 8 bytes of data in the receive FIFO then the Received Data Available interrupt is set. See (IIR)

ine Control Register (LCR)

Bit	**Notes**			
Bit 7	1	Divisor Latch Access Bit		
	0	Access to Receiver buffer, Transmitter buffer & Interrupt Enable Register		
Bit 6	Set Break Enable			
Bits 3 : 5	Bit 5	Bit 4	Bit 3	Parity Select
	X	X	0	No Parity
	0	0	1	Odd Parity
	0	1	1	Even Parity
	1	0	1	High Parity (Sticky)
	1	1	1	Low Parity (Sticky)

Bit 2	Length of Stop Bit		
	0	One Stop Bit	
	1	2 Stop bits for words of length 6,7 or 8 bits or 1.5 Stop Bits for Word lengths of 5 bits.	
Bits 0 : 1	Bit 1	Bit 0	Word Length
	0	0	5 Bits
	0	1	6 Bits
	1	0	7 Bits
	1	1	8 Bits

Table 10 : Line Control Register

The Line Control register sets the basic parameters for communication. Bit 7 is the Divisor Latch Access Bit or DLAB for short. We have already talked about what it does. (See DLAB?) Bit 6 Sets break enable. When active, the TD line goes into "Spacing" state which causes a break in the receiving UART. Setting this bit to '0' Disables the Break.

Bits 3,4 and 5 select parity. If you study the 3 bits, you will find that bit 3 controls parity. That is, if it is set to '0' then no parity is used, but if it is set to '1' then parity is used. Jumping to bit 5, we can see that it controls sticky parity. Sticky parity is simply when the parity bit is always transmitted and checked as a '1' or '0'. This has very little success in checking for errors as if the first 4 bits contain errors but the sticky parity bit contains the appropriately set bit, then a parity error will not result. Sticky high parity is the use of a '1' for the parity bit, while the opposite, sticky low parity is the use of a '0' for the parity bit.

If bit 5 controls sticky parity, then turning this bit off must produce normal parity provided bit 3 is still set to '1'. Odd parity is when the parity bit is transmitted as a '1' or '0' so that there is a odd number of 1's. Even parity must then be the parity bit produces and even number of 1's. This provides better error checking but still is not perfect, thus CRC-32 is often used for software error correction. If one bit happens to be inverted with even or odd parity set, then a parity error will occur, however if two bits are flipped in such a way that it produces the correct parity bit then an parity error will no occur.

Bit 2 sets the length of the stop bits. Setting this bit to '0' will produce one stop bit, however setting it to '1' will produce either 1.5 or 2 stop bits depending upon the word length. Note that the receiver only checks the first stop bit.

Bits 0 and 1 set the word length. This should be pretty straight forward. A word length of 8 bits is most commonly used today.

lodem Control Register (MCR)

Bit	Notes
Bit 7	Reserved
Bit 6	Reserved
Bit 5	Autoflow Control Enabled (16750 only)
Bit 4	LoopBack Mode
Bit 3	Aux Output 2
Bit 2	Aux Output 1
Bit 1	Force Request to Send
Bit 0	Force Data Terminal Ready

Table 11 : Modem Control Register

The Modem Control Register is a Read/Write Register. Bits 5,6 and 7 are reserved. Bit 4 activates the loopback mode. In Loopback mode the transmitter serial output is placed into marking state. The receiver serial input is disconnected. The transmitter out is looped back to the receiver in. DSR, CTS, RI & DCD are disconnected. DTR, RTS, OUT1 & OUT2 are connected to the modem control inputs. The modem control output pins are then place in an inactive state. In this mode any data which is placed in the transmitter registers for output is received by the receiver circuitry on the same chip and is available at the receiver buffer. This can be used to test the UARTs operation.

Aux Output 2 maybe connected to external circuitry which controls the UART-CPU interrupt process. ıx Output 1 is normally disconnected, but on some cards is used to switch between a 1.8432MHZ crystal a 4MHZ crystal which is used for MIDI. Bits 0 and 1 simply control their relevant data lines. For example tting bit 1 to '1' makes the request to send line active.

ine Status Register (LSR)

Bit	Notes
Bit 7	Error in Received FIFO
Bit 6	Empty Data Holding Registers
Bit 5	Empty Transmitter Holding Register
Bit 4	Break Interrupt
Bit 3	Framing Error
Bit 2	Parity Error
Bit 1	Overrun Error
Bit 0	Data Ready

Table 12 : Line Status Register

The line status register is a read only register. Bit 7 is the error in received FIFO bit. This bit is high when at least one break, parity or framing error has occurred on a byte which is contained in the FIFO.

When bit 6 is set, both the transmitter holding register and the shift register are empty. The UART's holding register holds the next byte of data to be sent in parallel fashion. The shift register is used to convert the byte to serial, so that it can be transmitted over one line. When bit 5 is set, only the transmitter holding register is empty. So what's the difference between the two? When bit 6, the transmitter holding and shift registers are empty, no serial conversions are taking place so there should be no activity on the transmit data line. When bit 5 is set, the transmitter holding register is empty, thus another byte can be sent to the data port, but a serial conversion using the shift register may be taking place.

The break interrupt (Bit 4) occurs when the received data line is held in a logic state '0' (Space) for more than the time it takes to send a full word. That includes the time for the start bit, data bits, parity bits and stop bits.

A framing error (Bit 3) occurs when the last bit is not a stop bit. This may occur due to a timing error. You will most commonly encounter a framing error when using a null modem linking two computers or a protocol analyzer when the speed at which the data is being sent is different to that of what you have the UART set to receive it at.

A overrun error normally occurs when your program can't read from the port fast enough. If you don't get an incoming byte out of the register fast enough, and another byte just happens to be received, then the last byte will be lost and a overrun error will result.

Bit 0 shows data ready, which means that a byte has been received by the UART and is at the receiver buffer ready to be read.

Modem Status Register (MSR)

Bit	Notes
Bit 7	Carrier Detect
Bit 6	Ring Indicator
Bit 5	Data Set Ready
Bit 4	Clear To Send
Bit 3	Delta Data Carrier Detect
Bit 2	Trailing Edge Ring Indicator
Bit 1	Delta Data Set Ready
Bit 0	Delta Clear to Send

Table 13 : Modem Status Register

Bit 0 of the modem status register shows delta clear to send, delta meaning a change in, thus delta clear to send means that there was a change in the clear to send line, since the last read of this register. This is the same for bits 1 and 3. Bit 1 shows a change in the Data Set Ready line where as Bit 3 shows a change in the Data Carrier Detect line. Bit 2 is the Trailing Edge Ring Indicator which indicates that there was a transformation from low to high state on the Ring Indicator line.

Bits 4 to 7 show the current state of the data lines when read. Bit 7 shows Carrier Detect, Bit 6 shows Ring Indicator, Bit 5 shows Data Set Ready & Bit 4 shows the status of the Clear To Send line.

Scratch Register

The scratch register is not used for communications but rather used as a place to leave a byte of data. The only real use it has is to determine whether the UART is a 8250/8250B or a 8250A/16450 and even that is not very practical today as the 8250/8250B was never designed for AT's and can't hack the bus speed.