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

SKRIPSI

RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA ABSENSI DOSEN DAN MAHASISWA DI KAMPUS BERBASIS MIKROKONTROLLER AT89S51

Disusun Oleh :
YOGA ANGGORO
00.17.232

MARET 2006

LEMBAR PERSETUJUAN

RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA
ABSENSI DOSEN DAN MAHASISWA
BERBASIS MIKROKONTROLLER AT89S51

SKRIPSI

Disusun dan diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :

YOGA ANGGORO
00.17.232

Diperiksa dan disetujui :

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. Muhammad Luqman, MT)	(M. Ibrahim Ashari, ST)
NIP.P 131 793 370	NIP.P 1 030 100 358

Mengetahui :
Ketua Jurusan Teknik Elektro S-1

(Ir. F. Yudi Limpraptono, MT.)
NIP. 1 039 500 274

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL
2006

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	:	YOGA ANGGORO
NIM	:	00.17.232.
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika
Judul Skripsi	:	

"RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA
ABSENSI DOSEN DAN MAHASISWA
BERBASIS MIKROKONTROLLER AT89S51"

Dipertahankan Dihadapan Team Penguji Skripsi Jenjang Strata Satu (S-1), pada :

Hari	:	Selasa
Tanggal	:	21 Maret 2006
Dengan nilai	:	88,9 (A)

Panitia Ujian Skripsi:

(Ir. Mochtar Asroni, MSME)
Ketua

(Ir.F.Yudi Limpraptono, MT)
Sekretaris

Anggota Penguji:

(Ir. Widodo Pudji M, MT)
Penguji I

(Ir. Mimien Mustikawati)
Penguji II

LEMBAR PERSEMBAHAN

ALHAMDU LILLAAHI RABBIL 'AALAMIIN

Akir dari perjanan kuliah S-1 ku, Atas Rahmat dan hidayah ALLAH S.W.T. saya dapat mentelesaikan Tugas Akir ini dengan sempurna. Sujud Sukur saya Ya ALLAH S.W.T.

Semua ini saya Persembahkan buat PAPA dan MAMA
Buat adik ku yang maniZz.. NENO
Buat adik ku yang Lucu Dimas
Terimakasih buat keluarga tercinta ku, yang selalu mendoakan ku,
Yang selalu mendukung aku

Buat kelurga besar ku TERIMAKASIH

OyA....Chika ku TERSAYANG yang membantu ku selama aku kuliah
Yang mengajari aku banyak hal
Yang membuat aku lebih dewasa
Sesuatu yang spesial buuUanget, Kamu buat ku

TRIMAKASIH JUGA BUAT TIM SUSSES KU:
SPESIAL GAK PAKE TELOR TONY INDRAJAYA, ST
KOKOK YANG KONYOL, AWIE YANG MISTERIUS, OGET DENGAN IDE EXRIMNYA, KACAK + KELIK SI KEMBAR JENIUS, DONY PRIA GAK ROKOK'AN, OVEK SIIDUNG GAJAH, PAY PEJUANG CINTA, IDUY PECINTA SEJATI, ACHOO YANG LATAH, ROFAT PLAY BOY, ANSA GRUP YANG SELALU GAK KEHABISAN AKAL, TERAKIR MBK RURY KARMA CEWEK JADI TERAKIR...HEHE

Temen-temen yang belum wisuda...cepet nyusul ya...
Sukses buat semuanya ya...sampai ketemu dilain waktu....perjalanan masih panjang...perjuangan yang takkan berhenti...sampai disini

wasalam
YOGA ANGGORO, ST
Hehe....

ABSTRAKSI

RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA ABSENSI DOSEN DAN MAHASISWA DI KAMPUS BERBASIS MIKROKONTROLLER AT89S51

(Yoga Anggoro, 0017232, Jurusan Teknik Elektro S-1, Konsentrasi Teknik Elektronika, 70 halaman)

(Dosen Pembimbing : Ir. M. Luqman, MT dan M. Ibrahim Ashari, ST.)

Kata kunci : Mikrokontroller, RS 485,RS 232, Sistem Absensi.

Pesatnya perkembangan teknologi berkembang seiring perkembangan zaman yang kian canggih. Sehingga peralatan elektronika menjadi piranti yang banyak digunakan diberbagai bidang khususnya pada kerumitan yang tidak dapat dilakukan oleh manusia dengan cepat, maka dengan teknologi elektronika didapatkan kemudahan dan kenyamanan dalam. sistem pengelolaan data absensi dosen dan mahasiswa di kampus (Politeknik, Universitas, Sekolah Tinggi) sangat beragam. Namun biasanya memiliki kesamaan tertentu, diantaranya: adanya proses pencatatan kehadiran dosen dan mahasiswa berupa presensi. Pemberian surat peringatan bagi mahasiswa yang jumlah kehadirannya kurang, penghitungan honor dosen sesuai jumlah jam mengajar kelas ekstensi (sore) dan lain sebagainya yang pada umumnya dilakukan secara manual

Sistem pengolahan data absensi dosen dan mahasiswa di kampus ini terdiri atas minimum sistem AT89S51 yang digunakan sebagai pengontrol utama dari keseluruhan sistem,RS 485, PC, yang digunakan untuk pengaplikasian dari keseluruhan kerja alat pengukur curah hujan

Diharapkan dengan Sistem pengolahan data absensi, dapat mempermudah proses pencatatan dan pengolahan data kehadiran dosen dan mahasiswa di kampus.

KATA PENGANTAR

Alhamdulillah, dengan memanjatkan puji syukur kehadirat Allah SWT, yang telah memberikan rahmat, hidayah serta karunia-Nya, tak lupa sholawat dan salam kepada nabi Muhammad SAW. dan keluarganya, akhirnya penyusun dapat dapat menyelesaikan skripsi ini yang berjudul "Rancang Bangun Alat Dan Sistem Pengolah Data Absensi Dosen Dan Mahasiswa Di Kampus Berbasis Mikrokontroller AT89S51", laporan skripsi ini merupakan salah satu persyaratan kelulusan Program Strata 1 Teknik Elektro/Konsentrasi Elektronika Institut Teknologi Nasional Malang.

Keberhasilan penyusunan laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terimakasih kepada:

1. Bapak dan Ibu tercinta yang selalu memberikan kasih sayang, do'a restu, dukungan dan bantuan secara moril maupun materiil kepada penyusun, dengan segala dukungan orang tua dapat membesarkan hati penyusun meyelesaikan skripsi ini.
2. Bapak Dr. Ir. Abraham Lomi, MSEE. selaku Rektor Institut Teknologi Nasional Malang.
3. Bapak Ir. Mochtar Asroni, MSME. selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang
4. Bapak Ir. F. Yudi Limpraptono, MT. selaku Ketua Jurusan Teknik Elektro S-1 Institut Teknologi Nasional Malang
5. Bapak Ir. M. Luqman, MT selaku Dosen Pembimbing I

6. Bapak M. Ibrahim Ashari, ST. selaku Dosen Pembimbing II
7. Teman-teman yang telah memberikan motifasi serta bantuan baik berupa tenaga maupun fikiran dalam proses penyelesaian skripsi ini *(special thank's to Tony Indrajaya).*

Penyusun telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam penyelesaian laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang bersifat membangun demi kesempurnaan laporan ini.

Malang, Maret 2006

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Teknologi elektronika sekarang ini makin pesat berkembang seiring perkembangan zaman yang kian hari semakin canggih. Sehingga peralatan elektronika menjadi piranti yang sangat banyak digunakan diberbagai bidang khususnya pada piranti dengan kerumitan yang tidak dapat dilakukan oleh manusia dengan cepat dalam prosesnya, maka dengan teknologi elektronika didapatkan kemudahan dan kenyamanan dalam melakukan suatu pekerjaan.

Sistem pengelolaan data absensi dosen dan mahasiswa di kampus (Politeknik, Universitas, Sekolah Tinggi) sangat beragam. Namun biasanya memiliki kesamaan tertentu, diantaranya: adanya proses pencatatan kehadiran dosen dan mahasiswa berupa presensi. Pemberian surat peringatan bagi mahasiswa yang jumlah kehadirannya kurang, penghitungan honor dosen sesuai jumlah jam mengajar kelas ekstensi (sore) dan lain sebagainya. Menurut informasi yang kami peroleh bahwa semua pekerjaan tersebut umumnya masih dilakukan secara manual dan ditulis diatas kertas. Cara seperti ini kami nilai kurang efektif, kurang efisien dan menghabiskan kertas sebab cara manual sangat menuntut usaha keras dalam mengolah data untuk dapat dihasilkan keluaran/hasil tertentu. Belum lagi jika terjadi kekeliruan akibat kelelahan atau human error.

Mensikapi hal tersebut, terkait dengan penelitian skripsi, saya bermaksud merancang alat dan pengolah data absensi dosen dan mahasiswa di kampus

menggunakan mikrokontroller dan computer. Alat yang dimaksud adalah terminal tempat memasukan data primer seperti kehadiran dosen dan mahasiswa. Sedangkan pengolah data berupa program yang mampu menghasilkan keluaran seperti yang diharapkan.

I.2. Rumusan Masalah

Masalah yang mungkin timbul dalam perancangan dan pembuatan alat meliputi:

1. Bagaimana membuat terminal-terminal data tempat memasukan data primer yang mampu terhubung dengan computer secara bersamaan.
2. Bagaimana menerapkan mikrokontroller sebagai alat yang dapat berkomunikasi dengan computer menggunakan jaringan standart RS-485
3. Data primer apa saja yang akan dikelola.
4. Bagaimana cara mengolah data primer sehingga menghasilkan keluaran yang diinginkan.
5. Data keluaran berupa apa saja.

I.3. Batasan Masalah

Penulisan Skripsi ini hanya terbatas pada:

1. Terminal data primer dibuat menggunakan MCS51 dengan standart komunikasi RS485 dan program computer dibuat menggunakan Delphi 7.

2. Data keluaran dibatasi hanya berupa jumlah alpha, ijin dan sakit masing-masing mahasiswa.
3. Tidak membahas arsitektur computer secara mendetail

1.4. Tujuan

Tujuan dari penulisan skripsi ini adalah untuk merancang dan membuat alat pengolah data absensi dosen dan mahasiswa di kampus berbasis Mikrokontroller AT89S51.

I.5. Metodologi Penulisan

Metode yang digunakan dalam penulisan tugas akhir ini adalah

1. Studi Pustaka

   Memperoleh data dengan cara membaca dan mempelajari buku literature yang berhubungan dengan penyusunan skripsi ini.

2. Studi Lapangan

   Memperoleh data dengan cara praktek secara langsung untuk menunjang pembuatan alat.

3. Pengolahan Data

   Mengolah data dengan jalan membuat analisa dan menarik kesimpulan dari hasil pengujian yang ada.

1.6. Sistematika Penulisan

Sistematika penulisan dari laporan ini adalah sebagai berikut :

BAB I **PENDAHULUAN**

Berisi latar belakang permasalahan, batasan masalah, metodologi dan sistematika penulisan.

BAB II **TEORI PENUNJANG**

Berisi landasan teori yang berhubungan dengan pembuatan alat

BAB III **PERENCANAAN DAN PEMBUATAN ALAT**

Meliputi tentang penjelasan dan tata cara perencanaan dan pembuatan alat.

BAB IV **PENGUJIAN ALAT**

Meliputi proses pengujian alat yang terdiri dari peralatan yang digunakan, langkah kerja dan analisa hasil pengujian.

BAB V **PENUTUP**

Meliputi kesimpulan dan saran.

BAB II

LANDASAN TEORI

2.1. Sistem Absensi

Sistem absensi yang diterapkan pada tiap-tiap perguruan tinggi bermacam-macam dan berbeda-beda. Ada yang menerapkan sistem absensi secara manual, menggunakan RFID, menggunakan sensor sidik jari dan lain sebagainya.

Sistem absensi manual biasanya dengan menggunakan buku absen yang diisi dengan tanda tangan mahasiswa dan dosen. Sistem absensi manual ini sangat mungkin dan sangat mudah untuk dicurangi, misalnya mahasiswa titip absen atau dosen mengisi daftar hadir tapi tidak mengajar.

Penggunaan sistem absensi dengan memanfaatkan RFID mungkin hanya terbatas pada dosen saja. Jika diterapkan untuk absen mahasiswa, maka dibutuhkan kartu dan *reader* yang banyak dan tentunya biaya yang dibutuhkan juga tidak sedikit. Dan bisa jadi, kartu hilang atau rusak sehingga tidak dapat digunakan.

Sensor sidik jari mungkin sangat jarang digunakan untuk sistem absensi karena disamping harganya mahal pengolahan datanya juga rumit. Kampus harus mengetahui semua data sidik jari yang akan diabsen, tentunya hal itu tidaklah mudah. Kalaupun ada yang menggunakan sensor sidik jari, mungkin hanya untuk dosen karena jumlah dosen sedikit dibanding jumlah mahasiswa.

2.2. Mikrokontroler AT89S51

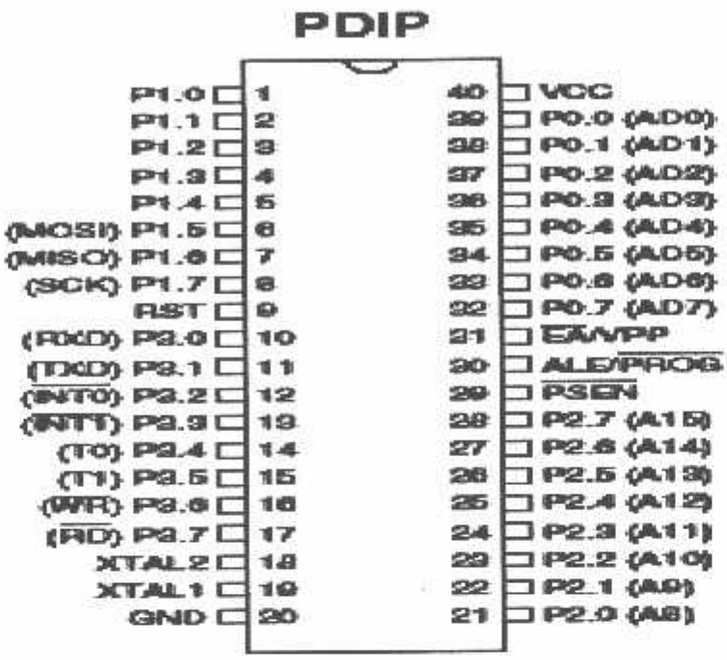
2.2.1. Arsitektur Perangkat Keras

Mikrokontroller AT89S51 adalah mikrokontroller ATMEL yang kompatibel penuh dengan mikrokontroler keluarga MCS – 51, membutuhkan daya rendah, memiliki performance yang tinggi dan merupakan mikrokomputer 8 bit yang dilengkapi 4Kbyte EEPROM *(Electrical Erasable and Programmable Read Only Memory)* dan 128 Byte RAM internal. Program memori yang dapat diprogram ulang dalam sistem atau menggunakan programmer *Nonvolatile* Memory konvensional. Dalam sistem mikrokontroller terdapat dua hal yang mendasar, yaitu: perangkat lunak dan perangkat keras yang keduanya saling terkait dan mendukung.

Secara umum Mikrokontroller AT89S51 memiliki :

- CPU 8 bit termasuk keluarga MCS-51
- 4 Kb Flash memory
- 128 byte Internal RAM
  - 4 bank regester, masing – masing berisi 8 register.
  - 16 byte yang dapat dialamati pada bit level.
  - 80 byte *general purpose memory data.*
- 32 buah Port I/O, tersusun atas P0 – P3,masing – masing 8 bit.
- 2 Timer/ counter 16 bit
- 2 Serial Port Full Duplex
- Kecepatan pelaksanaan intruksi per siklus 1 us pada frekuensi clock 12 Mhz

- 2 DPTR (*Data Pointer*)
- Watchdog Timer
- Fleksibel ISP Programming



Gambar 2-1
Konfigurasi Pin Mikrokontroler AT89S51
(Sumber : Data sheet AT89S51)

Berikut ini adalah penjelasan dari masing-masing pin mikrokontroler AT89S51:

- 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 AT89S51 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.

- Pin 18 dan 19

  X1 (*XTAL*1) dan X2 (*XTAL*2) : 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 dan keluaran dua arah. Setiap salurannya mampu melayani 4 masukan. Port 2 mengeluarkan alamat bagian high (A8-A15), selama pengambilan instruksi dari memori program eksternal dan pengambilan data dari memori data eksternal yang menggunakan mode pengalamatan 16-bit.

- Pin 29

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

- Pin 30

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

- Pin 31

  $\overline{EA}$/VP : *Eksternal Acces.* $\overline{EA}$ dihubungkan dengan VSS untuk memungkinkan pengambilan instruksi pada memori program eksternal yang berlokasi 0000_H sampai $FFFF_H$. Jika diinginkan menggunakan program internal, maka $\overline{EA}$ dihubungkan VCC.

- Pin 32 sampai 39

  Port 0 : Port 0 terdiri dari 8 saluran masukan dan keluaran dua arah. Setiap saluran mampu melayani 8 masukan. Port 0 merupakan saluran alamat bagian low (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 10µs.

Tabel 2-1. Keluarga MCS 51

Tipe	Tipe tanpa EPROM	Tipe ber EPROM	Kapasitas ROM	Kapasitas RAM	Port I/O	Pewaktu
8031	8031	-	4K	128	4	2
8051AH	8031AH	8751H	4K	128	4	2
8052AH	8032AH	8752BH	8K	256	4	3
80C52BH	80C31BH	87C51	4K	128	4	2
80C52	80C32	-	8K	256	4	3
83C51FA	80C51BH	87C51FA	8K	256	4	3
83C51FB	80C51FB	87C51FB	16K	256	4	3
83C152	80C152	-	8K	256	5	3
89S52	-	89S52	8K	256	4	3

(Sumber: Elex Media Komputindo. Bereksperimen dengan Mikrokontroler 8031)

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 AT89S51 merupakan padanan dari mikrokontroler 8051. Tabel tersebut memperlihatkan sebagian dari keluarga MCS-51.

2.2.2. Organisasi Memori

Organisasi memori pada mikrokontroler AT89S51 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 AT89S51 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 Vss (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 /PSEN (*Program Store Enable*).

Selain memori program mikrokontroler AT89S51 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 $\overline{/RD}$, sedangkan untuk menulis data digunakan sinyal $\overline{/WR}$.

2.2.3. Register Fungsi Khusus

Register fungsi khusus (*Special Function Register*) terletak pada 64 *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 2-2.

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 2-2. 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 byte	-	-
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	Interupt 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	$9B_H$
SBUF	Serial data buffer	Independen	99_H
PCON	Power control	HMOS $0XXXXXXX_B$	87_H
		CHMOS $0XXX0000_B$	

(Sumber: Elex Media Komputindo. Bereksperimen dengan Mikrokontroler 8031)

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 diinisialisasikan 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*).

2.2.4. Port Masukan dan Keluaran

Mikrokontroler AT89S51 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 di multipleks dengan saluran alamat rendah untuk mengakses memori eksternal, baik memori program maupun memori data. Port 2 mengeluarkan bagian alamat high 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 berikut.

Tabel 2-3.Fungsi Khusus Port 3

Nama Penyemat	Fungsi Khusus
Port 3.0	RxD (port asukan 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 eksternal)
Port 3.7	/RD (sinyal baca memori data eksternal)

(Sumber: Atmel. 2001,1-5)

2.2.5. Sistem Interupsi

Mikrokontroler AT89S51 mempunyai dua sumber interupsi eksternal dan sumber interupsi internal yang dapat diprogram agar sensitive 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.

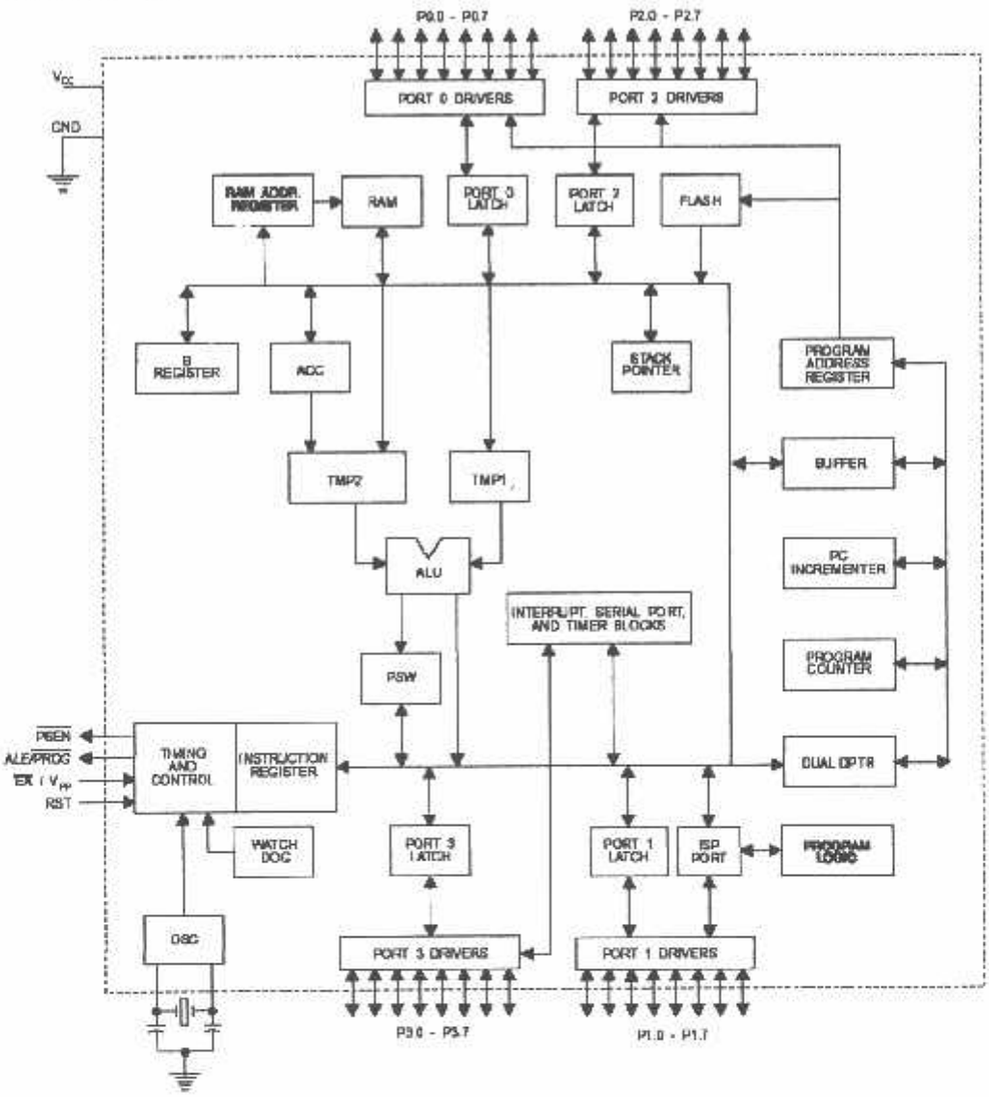
Tabel 2-4. Tingkatan Prioritas Interupsi

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	RI dan TI	0023_H
6	TF2 dan EXF2	$002B_H$

(Sumber: Intel. 1994: 3-25)

Hirarki tingkatan prioritas interupsi dapat dilihat dalam tabel diatas. 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).

Block Diagram



Gambar 2-2
Blok Diagram Mikrokontroler AT89S51
(Sumber : ATMELDatasheet Book)

2.3. *Liquid Crystal Display* (LCD)

LCD atau *Liquid Crystal Display* merupakan sarana tampilan yang terdiri dari tumpukan tipis antar sel dari dua lembar kaca yang pinggirnya tertutup rapat. Diantara kedua lapisan tersebut diberi kristal cair yang tembus cahaya. Permukaan

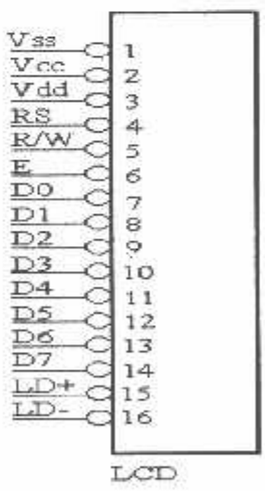
luar masing – masing keping kaca memiliki lapisan penghantar tembus cahaya. Sel memiliki ketebalan kira – kira $1x10^5$ m yang diisi dengan kristal cair. Pada modul LCD TM162 keluaran Seiko Instrument, LCD memiliki 2 baris tampilan dengan 16 karakter setiap barisnya.

Tabel 2-5. Nama dan Fungsi Penyemat Pensinyalan pada Modul LCD

Nama	Fungsi
D0 – D7	Saluran data yang berisi perintah dan data LCD
Enable	Sinyal pengaktif komponen '1' untuk mengaktifkan '0' untuk tidak memilihnya
R/W	Selektor baca tulis '1' untuk membaca '0' untuk menulis
RS	Pemilih register '0' untuk register inisialisasi (hanya tulis / write only) '1' untuk register data (baca dan tulis)
VLC	Pengendali terang redupnya cahaya LCD
VCC	Catu daya positif (5 V)
VSS	Catu negatif (ground)

Sumber : Datasheet Seiko Instrument LCD, 1987

Konfigurasi kaki pada modul LCD dapat dijelaskan pada Gambar 2.3. di bawah ini :



Gambar 2-3 Konfigurasi Kaki LCD
Sumber : Datasheet Seiko Instrument LCD, 1987

Adapun karateristik dari LCD TM162 adalah sebagai berikut:

- 16x2 karakter dengan 5x7 dot matriks.
- ROM generator karakter dengan 192 tipe karakter
- RAM generator karakter dengan 8 tipe karakter (untuk program write).
- 80 x 8 bit RAM data display dengan 80 karakter maksimal
- Dapat diantarmukakan dengan MPU 4 atau 8 bit
- RAM data dan RAM generator karakter dapat dibaca dari MPU
- Rangkain osilator terpadu
- Catu daya tunggal 5V
- Reset otomatis terpadu
- Temperatur antara 0^{0}C sampai 50^{0}C

Adapun untuk penampilan karakter yang ada dilakukan dengan cara memberikan kode karakter untuk tiap-tiap karakter yang diinginkan pada bus data dan menggunakan sinyal control E,RS, dan R/$\overline{W}$.

2.4. Komunikasi Serial

Dalam dunia komunikasi ada dua cara pemindahan data yaitu secara pararel dan serial. Perbedaan dua cara tersebut terletak pada jumlah bit yang dipindahkan. Setiap data merupakan kelompok dari bit-bit tersebut, dikenal dengan istilah *byte* yang merupakan kelompok yang terdiri dari 8 bit, serta *word* yaitu kelompok yang terdiri dari 16 bit.

Pada komunikasi serial terjadi pemindahan data satu bit pada satuan waktu, sedangkan pada komunikasi pararel terjadi pemindahan data secara berurutan dari sekelompok bit pada satuan waktu.

Ditinjau dari arah komunikasi data, dikenal ada tiga cara yaitu:

1. Komunikasi *simplex* adalah sistem komunikasi data yang arah perpindahan datanya satu arah saja.
2. Komunikasi *half duplex* adalah sistem komunikasi data yang arah perpindahan datanya dua arah, namun proses pemindahannya tidak bersamaan.
3. Komunikasi *full duplex* adalah sistem komunikasi yang arah perpindahan datanya dua arah secara bersamaan.

Yang dimaksud dengan komunikasi secara serial adalah komunikasi dengan memanfaatkan hanya satu saluran sinyal untuk pengiriman dan penerimaan data. Untuk data digital yang merupakan serangkain bit maka data dikirimkan bit per bit. Pengiriman akan dimulai dari LSB (*least Significant Bit*), dan diakhiri dengan MSB (*Most Significant Bit*). Setiap karakter yang dikirimkan disusun sesuai dengan suatu urutan bit tertentu.

2.4.1. Metode Transmisi Serial

Ada dua metode transmisi serial yang digunakan yaitu:

2. Komunikasi Serial *Sinkron* (*Syncronous Serial Communication*) adalah clock dikirimkan bersama-sama dengan data serial.
3. Komunikasi Serial *Asinkron* (*Asynchronous Serial Communication*) adalah clock tidak dikirimkan bersama data serial, tetapi dibangkitkan sendiri-sendiri baik pada sisi pengirim (*transmitter*) maupun pada sisi penerima (*receiver*).

Pada IBM PC kompatibel port serialnya termasuk jenis asinkron. Komunikasi data serial ini dikerjakan oleh UART (*Universal Asynchronous Receiver/Transmitter*). IC UART dibuat khusus untuk mengubah data pararel menjadi data serial dan menerima data serial yang kemudian diubah kembali menjadi data pararel.

Pada UART, kecepatan pengiriman data (*baud rate*) dan fase clock pada sisi transmitter dan pada sisi receiver harus sinkron. Untuk itu diperlukan sinkronisasi antara transmitter dan receiver. Hal ini dilakukan oleh bit 'Start' dan bit 'stop'. Ketika saluran transmisi dalam keadaan idle, output UART adalah dalam keadaan logika '1'. Ketika transmitter ingin mengirimkan data, output UART akan diset lebih dulu ke logika '0' untuk waktu satu bit. Sinyal ini pada receiver akan dikenali sebagai sinyal 'start' yang digunakan untuk mensinkronkan fase clocknya sehingga dengan fase clock transmitter. Selanjutnya data akan dikirimkan secara serial dari bit paling rendah (bit 0) sampai bit tertinggi. Selanjutnya, akan dikirim sinyal 'stop' sebagai akhir dari pengiriman data serial(Catur Edi dan Retna,2004).

2.5. Sistem Komunikasi RS 232

Sistem komunikasi RS 232 merupakan sistem komunikasi serial standart pada setiap komputer. Ada 2 jenis keluaran yang tersedia pada komputer yaitu COM 1 (DB 9) dan COM 2 (DB 25). Komunikasi serial pada COM 1 untuk mouse dan COM 2 disediakan untuk komunikasi data.

2.5.1. Sinyal-Sinyal pada Interface RS 232

Sinyal yang dikeluarkan oleh komputer secara DCE maupun DTE adalah :

1. RLSD (*Received Line Signal Detect*)

Dengan saluran ini DCE memberitahukan ke DTE bahwa pada terminal masukan ada data yang masuk.

2. RxD (*Received Data*)

Sinyal yang digunakan DTE untuk menerima data dari DCE.

3. TxD (*Transmitted Data*)

Sinyal yang digunakan DTE untuk mengirimkan data ke DCE.

4. DTR (*Data Terminal Ready*)

Sinyal yang digunakan oleh PC untuk memberitahu kepada DCE bahwa PC siap menerima data dari DCE.

5. Signal Ground (GND)

Saluran untuk ground.

6. DSR (*Data Set Ready*)

Sinyal aktif pada saluran ini menunjukan bahwa DTE sudah siap.

7. RTS (*Request To Send*)

Sinyal ini digunakan oleh PC Untuk memberitahu kepada DCE bahwa PC akan mengirim data. Apabila DCE dalam kondisi siap, maka modem akan memberikan sinyal CTS

8. CTS (*Clear To Send*)

Sinyal yang dikirimkan DCE untuk memberitahu kepada PC bahwa data data boleh dikirimkan.

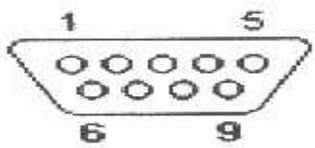
9. RI (*Ring Indikator*)

Pada saluran ini DCE memberitahu ke DTE bahwa stasiun menghendaki hubungan dengannya.

2.5.2. Interface EIA 232

2.5.2.1. Konfigurasi Pin

Sebuah komputer dapat berkomunikasi melalui *interface port* I/O serial yang disebut interface RS 232. Terdapat 9 pin dikenal dengan DB 9 yang terdapat pada bagian belakang komputer.



Gambar 2-4. Konektor DB 9
(Sumber : Interfacing port pararel dan port serial dengan visual basic)

Fungsi dari pin-pin yang ada untuk konektor DB 9 adalah sebagai berikut :

Tabel 2-6. Konfigurasi Pin DB 9

EIA 232 (RS 232) Function	PIN
Data Carrier Detect (DCD)	1
Received Data (RxD)	2
Transmitted Data (TxD)	3
Data Terminal Ready (DTR)	4
Sinyal Ground (GND)	5
Data Set Ready (DSR)	6
Request To Send (RTS)	7
Clear To Send (CTS)	8
Ring Indicator (RI)	9

Sumber : B & B Electronics Mfg Co, USA, 2001, halaman 5

Jalur data (TxD dan RxD) digunakan untuk transport data, TxD adalah jalur output pada komputer, data dikirim dari pin ini sedangkan RxD adalah penerima untuk komputer, data yang dating akan diterima olch pin ini. Pin keempat adalah output (RTS) dimana sebuah sinyal akan diberikan pada alat yang dihubungkan dengan maksud meminta kiriman data. CTS adalah sinyal masukan yang menunggu sinyal dari alat yang terhubung. Ketika alat tersebut menerima sinyal RTS dan bisa menerima data maka ia akan mengirimkan sinyal balik yang merupakan CTS. DTR adalah sinyal keluaran yang memberi tanda bahwa ada alat yang terhubung dan akan mengirim data. DSR merupakan sinyal input yang mana jika alat yang terhubung menerima sinyal balik kemudian diterima sebagai sinyal DSR.

2.5.2.2. Sinyal Transmisi Data

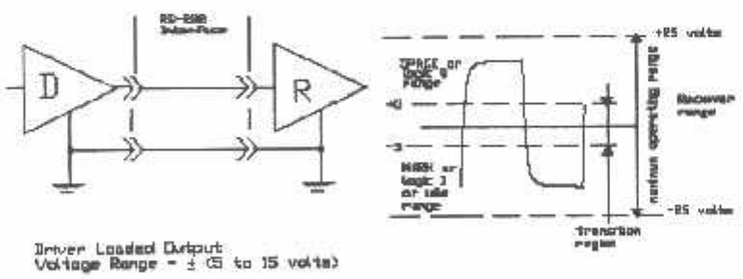
RS 232 dibuat untuk interface antara peralatan terminal data dan peralatan komunikasi data, dengan menggunakan data biner serial sebagai data yang ditransmisikan. Serial interface RS 232 memberi ketentuan logic level sebagai berikut:

1. Logic 1 disebut *'mark'* terletak antara –3 volt hingga –15 volt.

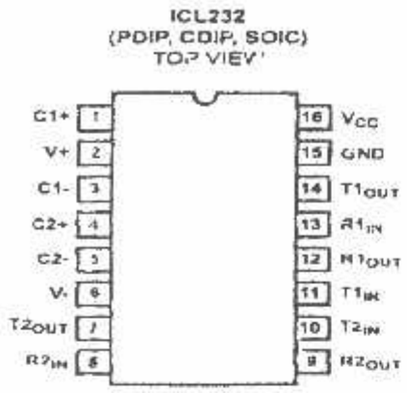
2. Logic 0 disebut *'space'* terletak antara +3 volt hingga +15 volt.

3. Daerah tegangan antara –3 volt hingga +3 volt adalah invalid level, yaitu daerah tegangan yang tidak memiliki logic sehingga daerah itu harus dihindari. Suatu saluran data RS 232 yang memberi keadaan tegangan ini berarti ada kesalahan. Demikian pula pada saluran pada daerah lebih dari –15 volt daerah lebih positif dari +15 volt.

Pada saat pengiriman data, sebelum mengirim data bit 0 harus diawali dengan start bit dahulu kemudian baru mengirim bit 0. Setelah mengirim bit 7 masih harus diakhiri dengan stop bit, jumlah stop bit dapat 1 bit, atau 2 stop bit atau 8 bit yang dibatasi oleh start bit dan stop bit disebut 1 frame.



Gambar 2-5. Rangkaian Interface RS 232
(Sumber: B&B Electronics Manufacturing Company)



Gambar 2-6 IC RS 232
(Sumber : Data Sheet Harris Semiconductor)

2.6. Sistem Komunikasi RS 485

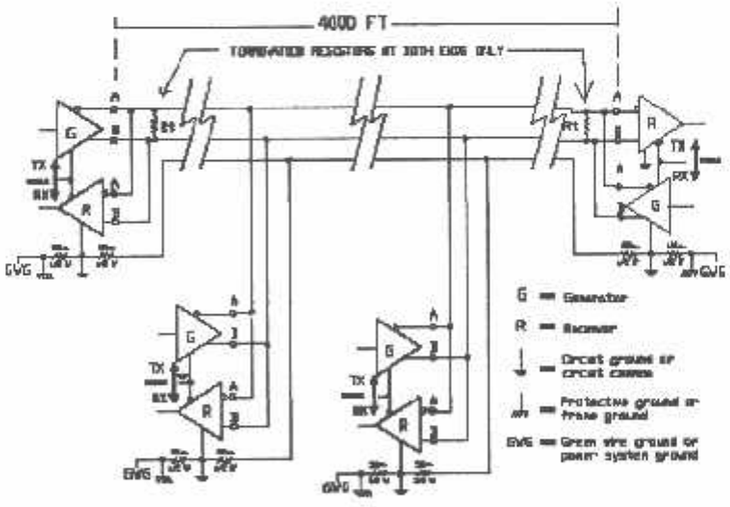
Standard RS 485 diterapkan oleh *Electronic Industry Association*(EIA) *dan Telecommunication Industry Association* (TIA) pada tahun 1983. Standard RS 485 hanya membicarakan karakteristik sinyal dalam transmisi data secara Balanced Digital Multipoint System. Jadi jauh lebih sederhana di banding dengan standard RS 232 yang mencakup ketentuan tentang karakteristik sinyal, macam-macam sinyal dan konektor yang dipakai, serta konfigurasi sinyal pada kaki- kaki di konektor dan juga penentuan tata cara pertukaran informasi antara komputer dan alat-alat pelengkapnya.

RS 485 bisa dipakai untuk saluran samapai sejauh 4000 *feet* dan kecepatan lebih dari 1 Megabit/ detik. Pada RS 485 menggunakan saluran ganda (*Differential* atau *Unbalanced Transmission*). Transmisi saluran ganda (*Differential* atau *Unbalanced Transmission*) memakai satu pasang kabel untuk mengirim satu sinyal, informasi logika ditafsirkan dari beda tegangan antara 2 utas kabel saluran. Tegangan pada kedua utas kabel saluran selalu berlawanan,

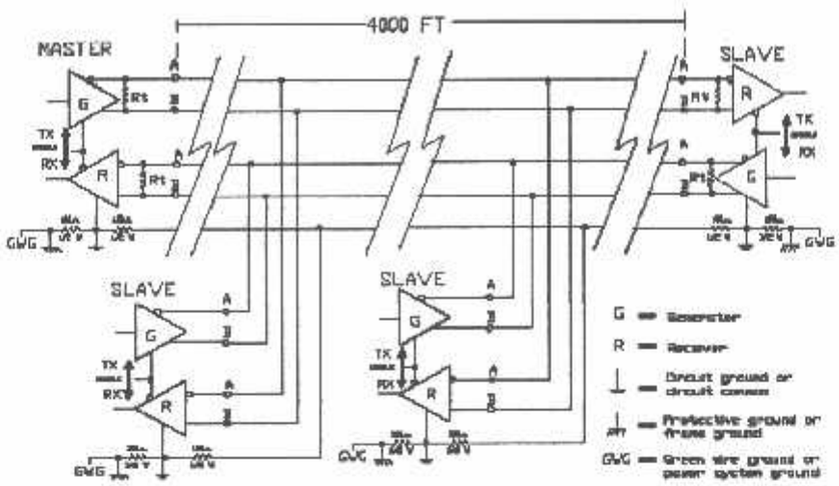
saat satu kabel bertegangan tinggi maka kabel yang lain bertegangan rendah, demikian pula sebaliknya. Rangkaian penerima sinyal membandingkan tegangan kedua kabel saluran, level logika pada bagian output ditentukan oleh kabel mana yang lebih positif.

Meskipun demikian, Saluran ganda tidak dipakai untuk transmisi yang memerlukan banyak saluran, mengingat RS 485 memakai kabel jauh lebih banyak sehingga mahal. Untuk penghematan kabel, bahkan saluran ganda sering dipakai untuk saluran *Half Duplex*, yaitu saluran dua arah secara bergantian yang hanya menggunakan satu pasang kabel, bisa dipakai untuk menghubungkan line generator dan banyak line receiver menjadi satu, sistem ini disebut sebagai *komunikasi multidrop (Multipoint Communication)*.

Meskipun balanced data transmission lebih rumit, tapi mempunyai sifat yang sangat kebal terhadap gangguan listrik, sehingga bias dipakai untuk menyalurkan data lebih jauh dengan kecepatan lebih tinggi.



Gambar 2-7 Tipe RS 485 Multidrop 2 saluran
(Sumber : B&B Electronics Manufacturing Company)



Gambar 2-8 Tipe RS 485 Multidrop 4 Saluran
(Sumber : B&B Electronics Manufacturing Company)

2.7. Borland Delphi 7.0

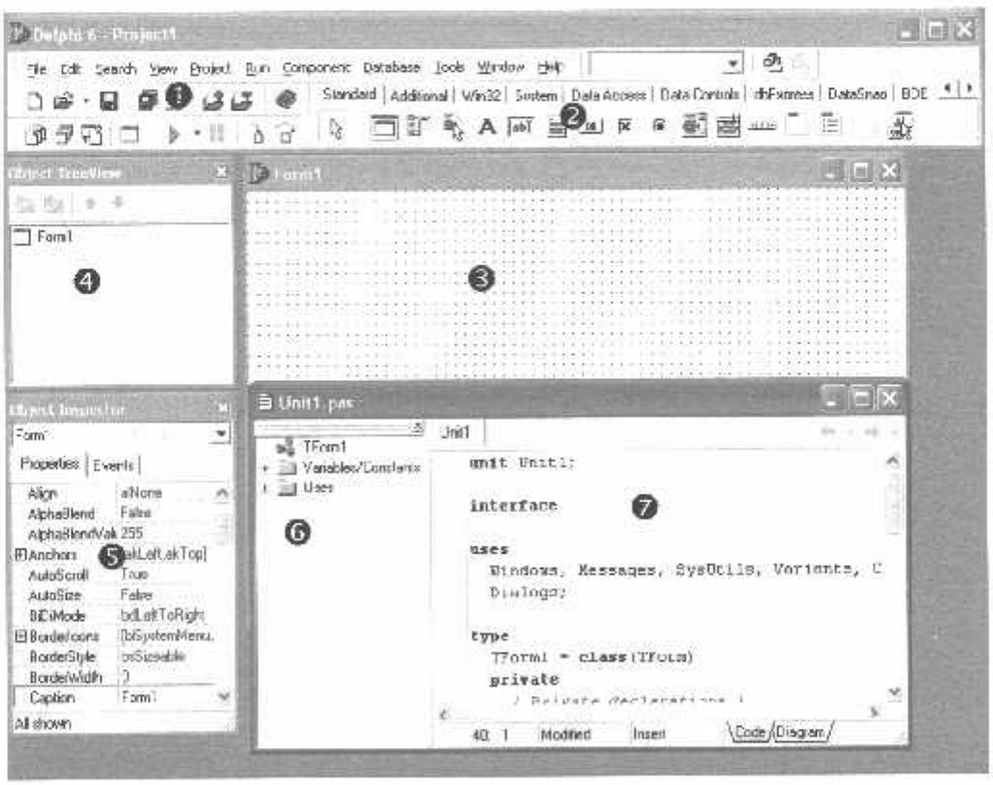
Delphi adalah salah satu pemrograman visual di lingkungan sistem operasi Microsoft Windows, berbasis bahasa PASCAL, sehingga bila kita telah mengetahui konsep dasar dan aturan-aturan yang berlaku dalam pemrograman PASCAL maka untuk masuk ke Delphi akan lebih mudah. Untuk mempermudah pemrogram dalam membuat program aplikasi, delphi menyediakan fasilitas pemrograman yang sangat lengkap. Fasilitas pemrograman tersebut dibagi dalam dua kelompok, yaitu object dan bahasa pemrograman.

Secara ringkas, object adalah suatu komponen yang mempunyai bentuk fisik dan biasanya dapat dilihat (visual). Object biasanya dipakai untuk melakukan tugas tertentu dan mempunyai batasan-batasan tertentu. Sedangkan bahasa pemrograman secara singkat dapat disebut sebagai sekumpulan teks yang mempunyai arti tertentu dan disusun dengan aturan tertentu serta untuk

menjalankan tugas tertentu. Delphi menggunakan struktur bahasa pemrograman Object Pascal. Gabungan dari object dan bahasa pemrograman ini sering disebut sebagai bahasa pemrograman berorientasi object atau Object Oriented Programming (OOP)

2.7.1. IDE Delphi

Lingkungan pengembangan terpadu atau *Integrated Development Environment* (IDE) Dalam program Delphi terbagi menjadi enam bagian utama, yaitu *Main Window, ToolBar, Component palette, Form Designer, Code Editor* dan *Object Inspector*. Untuk lebih jelasnya terlihat pada gambar di bawah ini :



Gambar 2-9 Lembar Kerja Delphi
(Sumber : Pengembangan Aplikasi Client/Server Dengan Borland Delphi)

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

Jendela IDE Delphi 7.0 mempunyai perangkat-perangkat yang dapat dipergunakan untuk memudahkan seorang programmer dalam membuat program. Perangkat-perangkat tersebut seperti terlihat pada gambar 2-8 diantaranya adalah:

❶ *Toolbar*

Dengan *toolbar* dapat melakukan beberapa operasi pada menu utama dengan sebuah klik tunggal.Setiap tombol pada *toolbar* mempunyai sebuah *tooltip* yang berisi informasi mengenai fungsi dari tombol tersebut.

❷ *Component Palette*

Component Pallete merupakan bagian yang digunakan untuk meletakkan berbagai komponen yang sesuai dengan kategorinya. Pada bidang ini semua komponen yang merupakan bawaan dari Delphi, baik berupa komponen visual maupun komponen nonvisual. Komponen-komponen tersebut berguna untuk mendesain user interface (antarmuka pemakai) dari program yang sedang dibuat. Borland Delphi sendiri memungkinkan untuk menambahkan komponen dari luar, baik yang dibuat sendiri ataupun komponen dari pihak ketiga (third party).

❸ *Form Designer*

Merupakan sebuah bidang jendela (window) yang masih kosong. Pada bidang ini dapat menempatkan komponen-komponen visual dan nonvisual untuk mendesain *user interface* program. Ketika menjalankan Borland

Delphi 7, secara otomatis *Form Designer* akan memanggil sebuah form yang bernama Form1.

❹ *Object treeView*

Merupakan sebuah diagram pohon yang menggambarkan hubungan logis antara komponen visual dan non visual yang terletak pada form, data module atau frame. Semua obyek yang anda pakai pada form, data module atau frame akan muncul pada *obyek TreeView*.

❺ *Object Inspector*

Dengan perangkat ini dapat mengubah *property* dan *event* pada setiap *object* atau komponen. Object atau komponen yang satu dengan yang lain mempunyai properti dan event yang berbeda. Jika menempatkan sebuah komponen pada suatu Form, komponen tersebut akan berisi nilai *default* dari Delphi. Misalnya komponen TButton akan berisi nilai 25 dan75 untuk properti height dan width. Nilai-nilai properti tersebut dapat diubah kemudian. Baik pada saat perancangan *interface* program (*design time*) maupun pada saat program berjalan (*run time*) dengan menggunakan kode program.

❻ *Code Explorer*

Pada jendela Code Explorer ini akan ditampilkan sebuah type, variable, dan routine yang didefinisikan pada unit. Selain itu juga ditampilkan semua unit yang digunakan yang terletak pada klausa uses. Untuk type yang kompleks seperti kelas, code exsplorer akan menampilkan semua informasi termasuk daftar *field, properties* dan *method.*

❼ *Code Editor*

Pada bidang ini dapat menuliskan kode-kode program dan logika program dalam bahasa Delphi untuk mengatur jalannya program. Antara *Form Designer* dan *Code Editor* merupakan dua bagian yang berkaitan, tidak bisa hanya mendesain *user interface* pada *Form Designer* dengan melupakan penulisan kode program pada bagian *Code Editor*.

2.7.2. Komponen Delphi

Pada Delphi terdapat beberapa komponen, yaitu :

1. Project

Project adalah sekumpulan form, unit dan beberapa hal lain dalam program aplikasi (singkatnya, project adalah program aplikasi itu sendiri). File utama project disimpan dalam file berakhiran .dpr (*Delphi Project*). Pada saat dijalankan, file project ini selalu dikompilasi menjadi file yang dapat dilaksanakan, berakhiran .exe atau .dll. Selain itu, pada saat membuat project, maka secara otomatis juga akan dibuatkan oleh Delphi sejumlah file yang diperlukan project tersebut.

2. Form

Form adalah suatu object yang dipakai sebagai tempat bekerja program aplikasi. Form berbentuk jendela dan dapat dibayangkan sebagai kertas atau meja kerja yang dapat digambari atau diletakkan object-object lain di atasnya. Pada saat membuat project baru, maka akan otomatis tersedia satu form dan diberi nama Form1. Dalam satu project dapat menggunakan lebih dari satu form. Dalam form

terdapat *grid* (garis titik-titik) yang sangat berguna untuk membantu pengaturan tata letak object yang digambarkan ke dalam form.

3. Unit

Unit adalah modul kode program. Satu project memungkinkan mempunyai satu unit atau lebih. Setiap kali dibuat satu form, maka otomatis dibuat satu unit. Unit yang berhubungan dengan form biasanya dipakai untuk mengatur dan mengendalikan segala sesuatu yang berhubungan dengan form dan berinteraksi dengan komponen lain.Unit dapat berisi kumpulan *function* atau *procedure* yang dipakai project. *Function* dan *procedure* adalah satu atau lebih baris program yang dipakai untuk menjalankan tugas tertentu.

4. Program

Program dibangun dari satu unit atau lebih. Program akan mengatur form dan unit serta menjalankannya. Secara umum sebuah program mempunyai struktur sebagai berikut :

1. *Heading Program*, yaitu yang menunjukkan nama program tersebut.
2. *Pernyataan Uses*, yang berisi daftar unit yang dipakai program.
3. *Blok deklarasi* dan pernyataan, yaitu bagian yang berisi deklarasi dan pernyataan program yang dilaksanakan pada saat program dijalankan. Bagian ini harus diakhiri dengan pernyataan end diikuti tanda titik. Bagian pada butir 1 dan 2 diperlukan pada saat kompilasi, bukan pada saat pelaksanaan program.

5. Property

Property digunakan untuk mendefinisikan atribut atau setting suatu object. Suatu object biasanya mempunyai beberapa *property* yang dapat diatur langsung dari tab properties dalam jendela Object Inspector maupun melalui kode program.

Setting *property* akan menentukan cara kerja dari object yang bersangkutan saat project dijalankan, misalnya menentukan warna object, ukuran object, font yang dipakai, database yang diambil dan lain-lain.

6. Event

Event adalah peristiwa atau kejadian yang diterima oleh suatu object, misalnya klik, drag, tunjuk dan lain-lain. *Event* yang diterima object akan memicu Delphi untuk memeriksa apakah ada kode program yang didefinisikan dalam event tersebut. Jika ada maka Delphi akan menjalankannya.

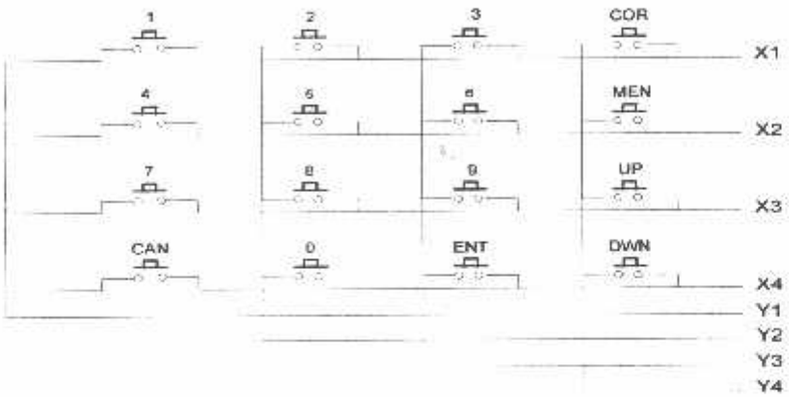
7. *Method*

Method adalah prosedur atau perintah yang melakat pada suatu object. Cara penulisan *method* sama dengan *property*, bedanya adalah *property* dipakai untuk menampung dan mengambil suatu nilai, sedangkan method untuk melakukan suatu aksi.

2.8. Keypad

Keypad digunakan sebagai sarana untuk memasukkan data ke minimum system. Keypad ini berfungsi untuk menampilkan data yang dimasukkan ke LCD. Keypad yang digunakan adalah keypad dengan 4 baris dan 4 kolom sehingga membentuk matriks 4 x 4. Pada keypad jenis ini, metode *scanning* tidak perlu

digunakan. Untuk pengambilan data dari keypad dapat digunakan metode polling biasa. Pada kondisi tidak ada penekanan keypad akan berlogika 1 pada setiap bitnya dan sebaliknya apabila salah satu dari keypad ditekan,baris dan kolom yang berhubungan akan terhubung ke ground sehingga baris dan kolom tersebut akan berlogika 0. Gambar matriks keypad 4 x 4 terlihat pada gambar 2.9.



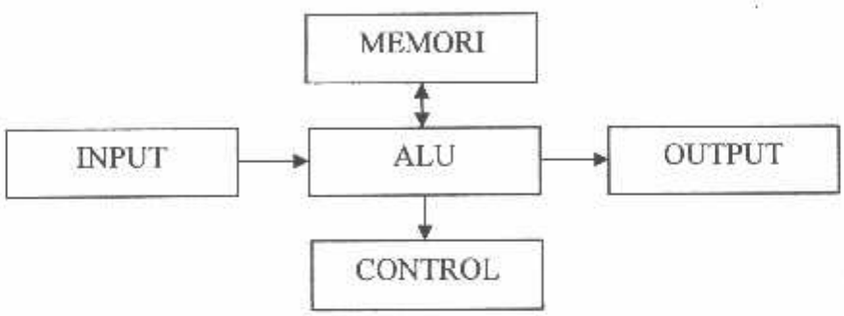
Gambar 2-10 Gambar Keypad
(Sumber: Teknik Antarmuka dan Pemograman Mikrokontroller AT89C51)

2.9. *Personal Computer*

Setiap komputer (PC) memiliki 5 bagian pokok agar dapat bekerja dengan sempurna, antara lain adalah *Arithmatic Logic Unit* (*ALU*), unit pengendali (*Control Unit*),Unit Memori (*Memori*),unit masukan (*Input*) dan unit keluaran (*Output*). Bagian-bagian ini dirangkai dalam blok diagram, seperti yang terlihat pada gambar 2-13.

Blok masukan menerima data untuk dikirim kedalam blok ALU agar diolah berdasarkan instruksi dari blok kendali, apakah akan diproses dengan operasi matematika atau logika. Hasil proses tersebut kemudian dikirim ke unit

keluaran untuk ditampilkan atau untuk mengaktifkan peralatan lain. Seluruh data dan hasilolahan tersebut disimpan sementara didalam unit memori selama proses berlangsung.



Gambar 2-11
Blok Diagram Komputer (PC)

(Sumber: C. Robert Brenner, IBM PC Trableshooting & Repair Guide, 1986, Hal 9)

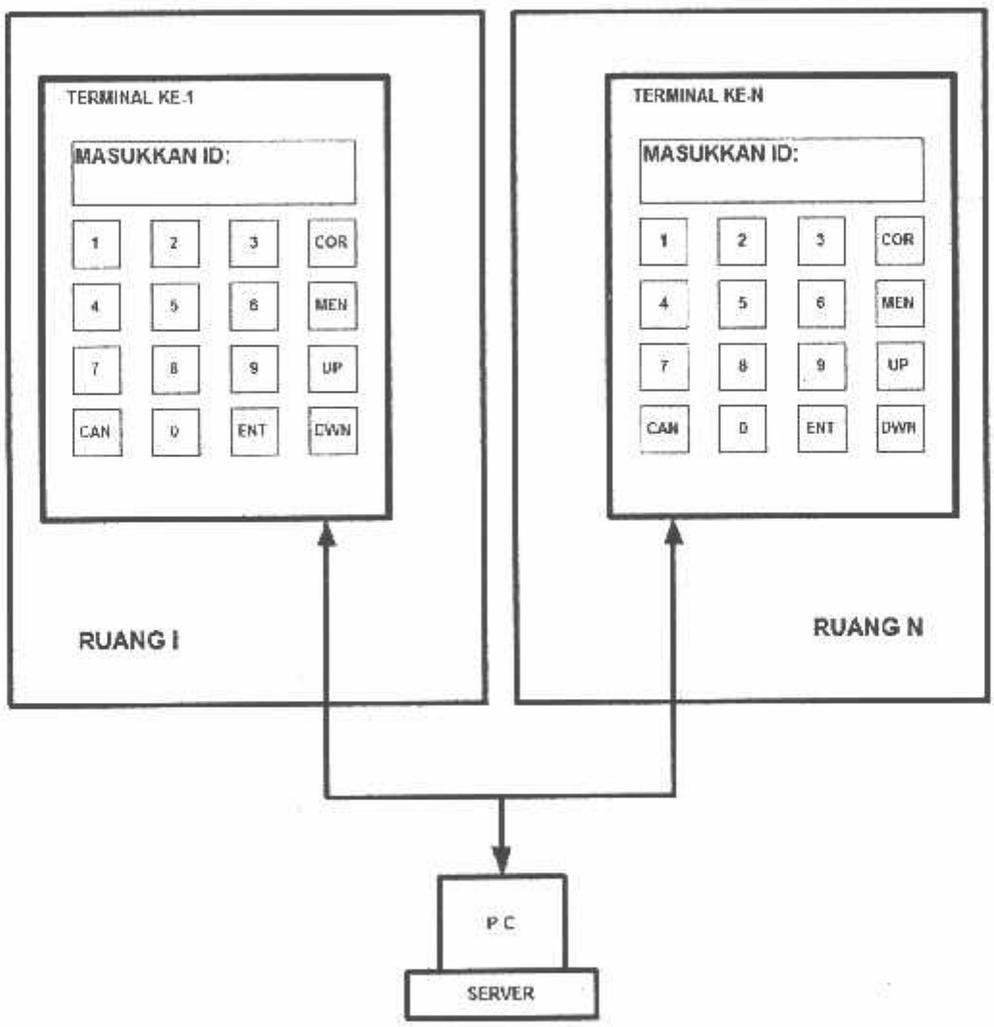
Ciri khas yang paling mendasar dari sebuah komputer (PC) bila dibandingkan dengan sebuah mesin hitung adalah ia memiliki kemampuan untuk diprogram ulang (*Reliabilitas-Programmable*), sesuai dengan keinginan programmer atau operator. Karena didalam sebuah komputer (PC) terdapat fasilitas untuk berkomunikasi secara langsung antara mikroprosesor dengan peralatan luar (*Eksternal*). Fasilitas tersebut adalah Port Printer (LPT) untuk komunikasi paralel dan PS untuk komunikasi serial.

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

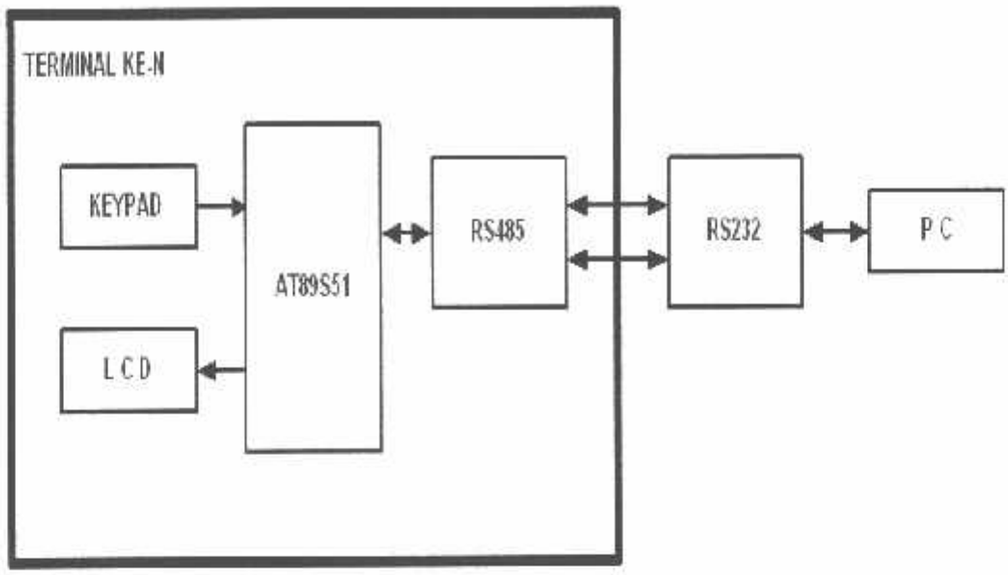
3.1. Pendahuluan

Perencanaan dan pembuatan alat ini menggunakan server berupa komputer dan memanfaatkan RS 485 sebagai komunikasi multiclient untuk meminta data dari server.



Gambar 3-1 Perancangan Secara Fisik

Dengan blok diagram seperti terlihat dibawah ini.



Gambar 3-2 Blok Diagram Sistem

Keterangan blok diagram:

- Menggunakan Microcontroller AT89S51 sebagai pengontrol masukan atau keluaran.
- LCD (*Liquid Crystal Display*) berfungsi sebagai tampilan data informasi dan perintah yang telah diolah oleh komputer.
- Keypad berfungsi untuk memasukkan data yang berupa ID dosen, *password,* pemilihan mata kuliah dan absen mahasiswa.
- RS 485 berfungsi sebagai komunikasi data *multipoint* dengan spesifikasi dalam perencanaan dan pembuatan alat ini dengan 2 *slave* sebagai *client* dan sebuah *master* yang terkoneksikan dengan komputer yang telah distandarkan dengan RS 232.

- RS 232 berfungsi sebagai konverter tegangan dengan level tegangan RS 232.

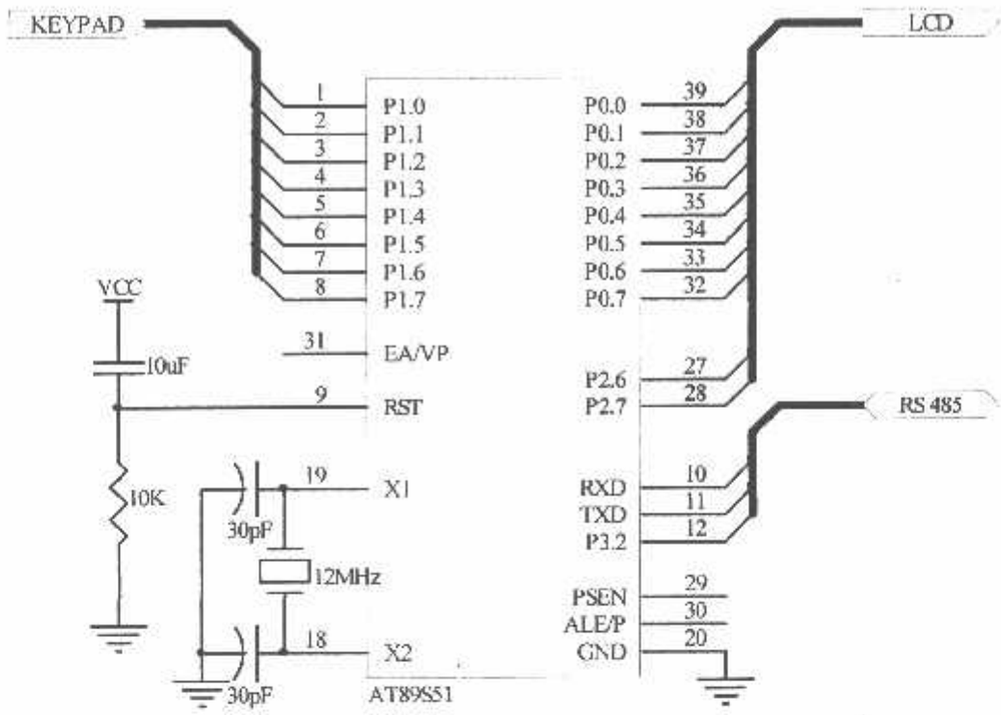
Dalam perencanaan ini dilakukan bertahap blok demi blok untuk memudahkan dalam menganalisis setiap bagiannya maupun dalam sistem. Perencanaan dan pembuatan alat ini terdiri atas dua perencanaan utama, yaitu perencanaan *hardware* dan perencanaan *software*.

3.2. Perencanaan dan Pembuatan Perangkat Keras *(Hardware)*

3.2.1. Sistem Mikrokontroler AT89S51

AT89S51 merupakan sebuah chip tunggal sebagai pengaturan keluar masuknya data. Karena dalam sistem ini menggunakan komunikasi data serial, sehingga harus ada suatu komponen yang dapat mengubah data serial menjadi data pararel untuk diproses oleh mikrokontroller yang bekerja dalam data pararel. Ada IC khusus yang dapat menangani hal tersebut yaitu IC UART (*Universal Asyncronous Receiver Transmitter*). Dalam berbagai macam mikrokontroller ada yang dilengkapi UART diantaranya yaitu AT89S51. Alasan menggunakan produksi Atmel dengan tipe AT89S51 adalah karena mudah diperoleh dipasaran dengan harga yang relatif murah, mudah dan praktis dalam pemrograman karena mempunyai program memori tipe EEPROM. AT89S51 merupakan memori dengan teknologi nonvolatile memori, artinya isi memori tersebut dapat diisi ulang ataupun dihapus berulang kali. MCU ini merupakan rangkaian slave yang berfungsi sebagai penerima perintah- perintah dari PC.

Rangkaian sistem mikrokontroller AT89S51 sebagaimana diperlihatkan dalam gambar dibawah ini.



Gambar 3-3 Rangkain Sistem Mikrokontroller

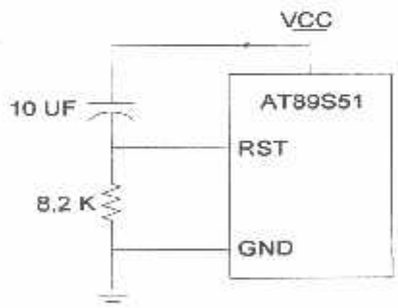
Fungsi MCU yang digunakan sebagai *slave* aplikasi MCU yang mengontrol masukan dan keluaran data dalam sistem absensi dosen dan mahasiswa adalah sebagai berikut :

1. Port 1
   - P1.0-P1.7 digunakan sebagai inputan keypad.
2. Port 0
   - P0.0-P0.7 digunakan sebagai tampilan data dari proses komputer.
3. Port 2
   - P2.6 untuk jalur RS pada LCD 16x2.
   - P2.2 untuk jalur E pada LCD 16x2.

4. Pin 10 (RXD) untuk jalur RO pada RS 485 berfungsi sebagai penerimaan data dari proses pengolahan PC.
5. Pin 11 (TXD) untuk jalur GI pada RS 485 berfungsi sebagai pengiriman data dari proses pengolahan PC.
6. Pin 12 (INT0) untuk jalur GE pada RS 485.
7. Pin 9 (RESET), reset aktif tinggi yang terhubung dengan rangkaian power on reset dan jika diaktifkan akan mereset mikrokontroller AT89S51.
8. Pin 18 (XTAL 2) untuk clock pada mikrokontroller AT89S51.
9. Pin 19 (XTAL 1) untuk clock pada mikrokontroller AT89S51.
10. Pin 20 (GND) digunakan sebagai ground Mikrokontroller.

3.2.1.1. Perencanaan Rangkaian Reset

Untuk mereset mikrokontroller AT89S51, maka pin RST diberi logika tinggi selama sekurangnya dua siklus mesin (24 periode osilator). Untuk membangkitkan sinyal reset kapasitor dihubungkan dengan VCC dan sebuah resistor yang dihubungkan ke ground. Rangkaian reset ditunjukkan dalam gambar dibawah ini sebagai berikut:



Gambar 3-4 Perencanaan Rangkain Reset

Karena kristal yang digunakan mempunyai frekuensi sebesar 11,0592MHz, maka satu periode membutuhkan waktu sebesar:

$$T = \frac{1}{fxtal} = \frac{1}{11{,}0592MHz} S = 9{,}042X10^{-8}\,s \quad \text{.....(3-1)}$$

Sehingga waktu minimal logika tinggi yang dibutuhkan untuk mereset mikrokontroller adalah:

Reset (min) = T x periode yang dibutuhkan.....(3-2)

$= 9{,}042 \times 10^{-8} \times 24 = 2{,}17\mu s$

Jadi mikrokontroller membutuhkan waktu minimal 2,17μs untuk mereset. Waktu minimal inilah yang dijadikan pedoman untuk menentukan nilai R dan C. Dari penentuan nilai R = 8,2 kΩ dan C = 10 μF, maka:

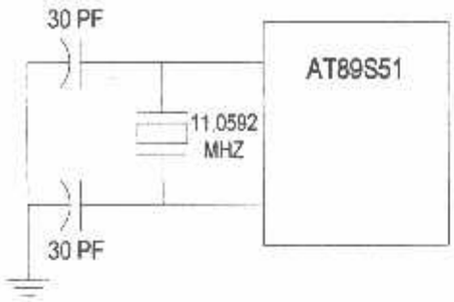
$t = 0{,}357\ R.C = 0{,}357 \times 8200\Omega \times 10.10^{-6} = 29{,}274$ ms

Jadi dengan nilai komponen R = 8,2 kΩ dan C = 10 μF dapat memenuhi syarat minimal untuk waktu yang dibutuhkan oleh mikrokontroller.

3.2.1.2. Perencanaan Clock

Kecepatan proses yang diperlukan oleh mikrokontroller ditentukan oleh sumber clock yang mengendalikan mikrokontroller tersebut. Sistem yang dirancang ini menggunakan osilator internal yang telah tersedia dalam chip AT89S51. Untuk mengendalikan frekuensi osilatornya cukup dengan menghubungkan kristal dalam pin 19 (X_1) dan pin 18 (X_2) serta dua buah kapasitor ke ground.

Besarnya kapasitansinya disesuaikan dengan spesifikasi dalam lembar data AT89S51 yaitu 30 pF. Kristal yang digunakan adalah 11,0592 MHz. Gambar 3.3 memperlihatkan gambar clock yang direncanakan.

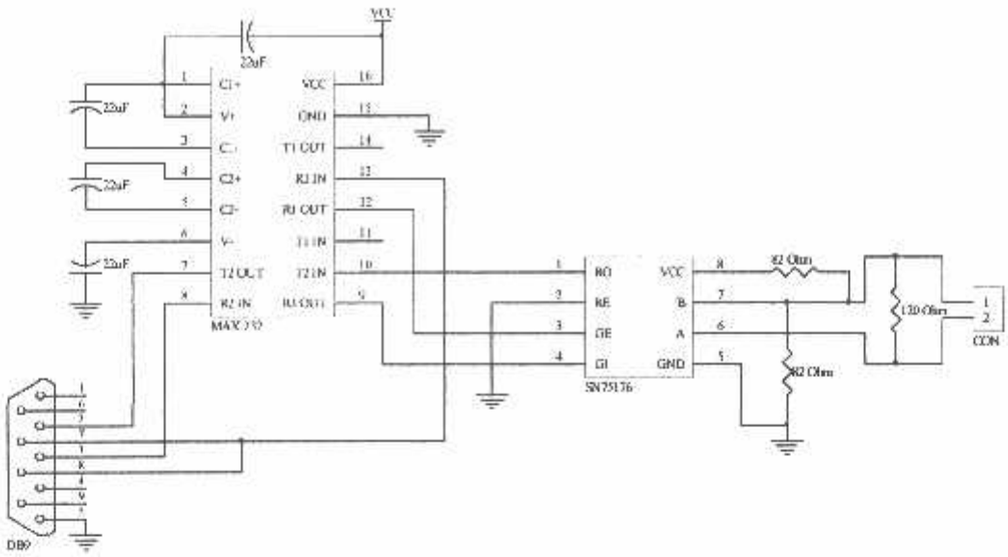


Gambar 3-5 Perencanaan Rangkain Clock

3.2.2. Konverter RS-232 ke RS 485

Interface antara device dengan PC (*Personal Computer*) yang masih berbeda level tegangannya diperlukan pengaturan tegangan antarmukanya sebagai konversi level tegangan dari level TTL ke PC. Karena port serial pada computer mempunyai level tegangan RS 232 sedangkan mikrokontroller menghasilkan level logika TTL. Pada mikrokontroller terdapat saluran untuk transfer data secara serial melalui fungsi khusus port 3 yaitu pin P3.0 dan P3.1.

Dalam komputer hanya tersedia port komunikasi serial RS-232 sehingga untuk mengubah tegangan RS-232 menjadi RS-485 diperlukan sebuah rangkaian yang mengubah level tegangannya. Rangkaian ini dirancang atas dua buah IC yaitu MAX232 dan MAX485. Gambar 3.4 memperlihatkan rangkaian pengubah RS-232 menjadi RS-485. IC MAX232 berfungsi untuk mengubah level tegangan RS-232 menjadi level tegangan RS-232 menjadi level tegangan TTL.



Gambar 3-6
Perencanaan Rangkain Konverter RS 232 Ke RS 485

IC MAX 232 memiliki dua *charge-pumps* internal yang mengkonversi tegangan +5V menjadi −10V (tanpa beban) untuk operasi antarmuka RS-232. C_1 berfungsi sebagai converter tegangan masukan +5V menjadi tegangan +10V yang tegangan keluarannya distabilkan oleh kapasitor C_3. Sedangkan kapasitor C_2 berfungsi untuk pembalik (*invert*) tegangan masukan +10V menjadi -10V dengan tegangan keluarannya distabilkan oleh kapasitor C_4.

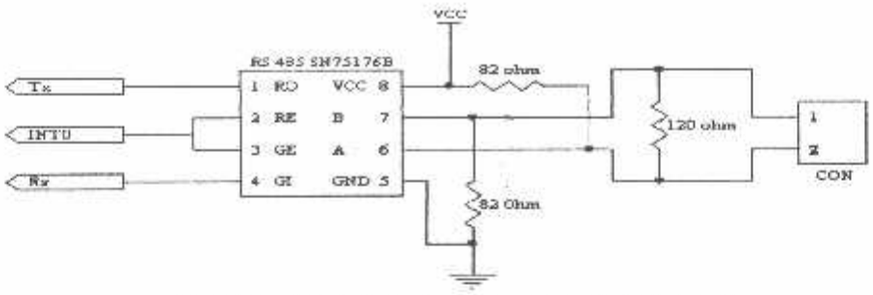
Untuk menghubungkan dengan komputer, maka dari keluaran MAX232 menuju port serial dihubungkan ke konektor tipe DB9 yang sesuai dengan standart komunikasi serial RS232 pada saluran COM1. Pin yang digunakan pada DB 9 yaitu pin 15 di hubungkan dengan ground, pin 2 (*Receiver* Data) dihubungkan dengan pin 7 (T2 out) dari RS232 dan pin 3 (*Transmitter* Data) dihubungkan dengan pin 8 (R2in) RS 232, ini dilakukan supaya pengiriman data dari mikrokontroller dapat diterima di penerima dari komputer dan sebaliknya, pin

7 (*Request To Send*) dihubungkan dengan pin 13 (T1in) untuk pengaturan permintaan untuk mengirim atau tidak. Pin 8 (*Clear To Send*) di hubungkan dengan RTS berfungsi sebagai sinyal yang dikirimkan DCE untuk memberitahu kepada PC bahwa data data boleh dikirimkan. Pin 5 di hubungkan dengan ground sesuai dengan data sheet DB 9. Pengaturan komunikasi datanya diatur pada perangkat lunaknya.

Sedangkan IC MAX485 digunakan untuk mengubah level tegangan TTL menjadi tegangan yang standar dengan RS-485. Dalam penggunaan IC MAX485, untuk pin RE dihubungsingkat yang berfungsi sebagai penerimaan data. Jika pin GE diberi logika 1, maka akan berfungsi sebagai pengirim data *(transmitter)*, sebaliknya jika pin RE diberi logika rendah maka akan berfungsi sebagai penerima data (*receiver*). Pemberian logika ini melalui perangkat lunaknya.

3.2.3. Rangkaian RS-485

Rangkaian RS-485 ini ditempatkan pada setiap ruangan kelas. Sesuai spesifikasi RS-485, rangkaian ini diletakkan dengan jarak maksimal sampai 1200m dari rangkaian converter RS-232 ke RS-485. Apabila beda tegangan dalam masukan V_{AB} lebih dari +200mV (V_A lebih positif dari V_B), maka dalam port penerima (*receiver*) RS-485 akan berlogika 1, sebaliknya jika beda tegangan V_{AB} kurang dari -200mV (V_A lebih negative dari V_B), maka port penerima (*receiver*) akan berlogika 0. Gambar rangkaian RS-485 seperti ditunjukkan dalam gambar 3.6.



Gambar 3-7 Perencanaan Rangkain RS 485

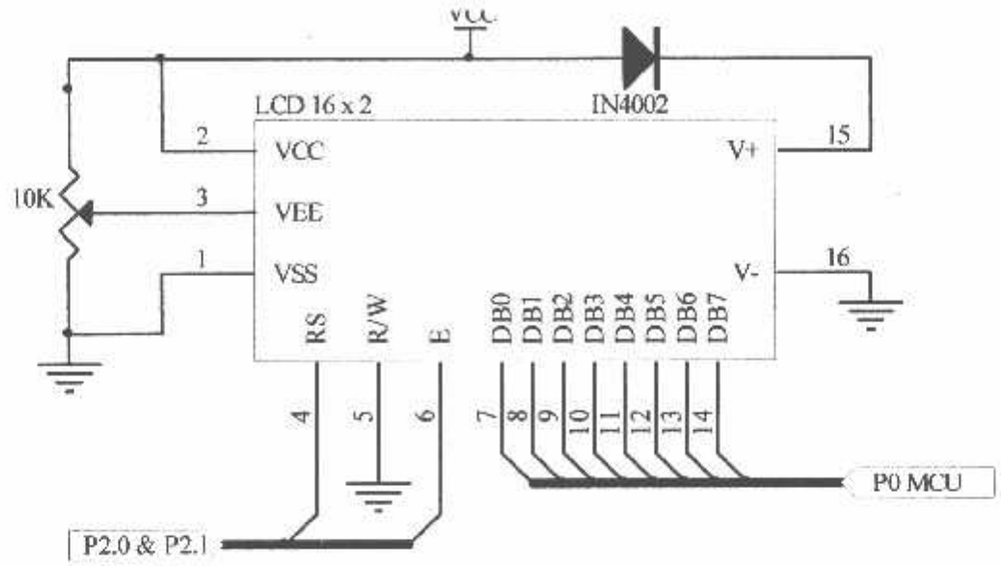
Dalam IC MAX485 pada sisi *master* dan *slave* ujung yang terjauh terdapat resistor dengan nilai 120Ω. Resistor tersebut dimaksud untuk mengurangi gelombang pantul dalam saluran, yang sering terjadi pada transmisi dengan kecepatan tinggi. Resistor ini berguna untuk menjaga impedansi karakteistik saluran tetap *matching*. Pada saat pergantian aktivitas *Line Generator Master* dan *Slave*, bisa terjadi satu saat secara bersamaan semua *Line Generator* tidak aktip, akibatnya saluran menjadi mengambang dan keadaan logika dari saluran tidak menentu. Untuk mencegah terjadinya hal tersebut, pada saluran ditambahkan 2 buah resistor masing-masing bernilai 82 Ohm, resistor yang terhubung ke A dihubungkan ke +5 Volt dan resistor yang terhubung ke B dihubungkan ke ground, dengan cara begini kalau semua Line Generator tidak aktif bisa dipastikan saluran dalam keadaan '1'.

Rangkaian RS-485 ini menggunakan IC MAX485 yang sama dengan rangkaian dalam converter RS-232 ke RS-485. Untuk pin RE dihubungsingkat yang berfungsi sebagai penerimaan data. Jika pin GE diberi logika 1, maka akan berfungsi sebagai pengirim data, sebaliknya jika pin RE diberi logika rendah maka akan berfungsi sebagai penerima data (*receiver*). Pemberian logika ini melalui perangkat lunaknya.

3.2.4. LCD (Liquid Crystal Display)

Sebagai tampilan perintah dan informasi. Disini menggunakan LCD M1632 (16 x 2), karena karakter yang ditampilkan dirancang tidak lebih dari 16 karakter per barisnya atau tidak lebih dari 32 karakter. Misalnya menampilkan tulisan "MASUKKAN ID:", "MASUKKAN PASSWORD:" atau menampilkan nama mahasiswa, dirancang tidak lebih dari 32 karakter.

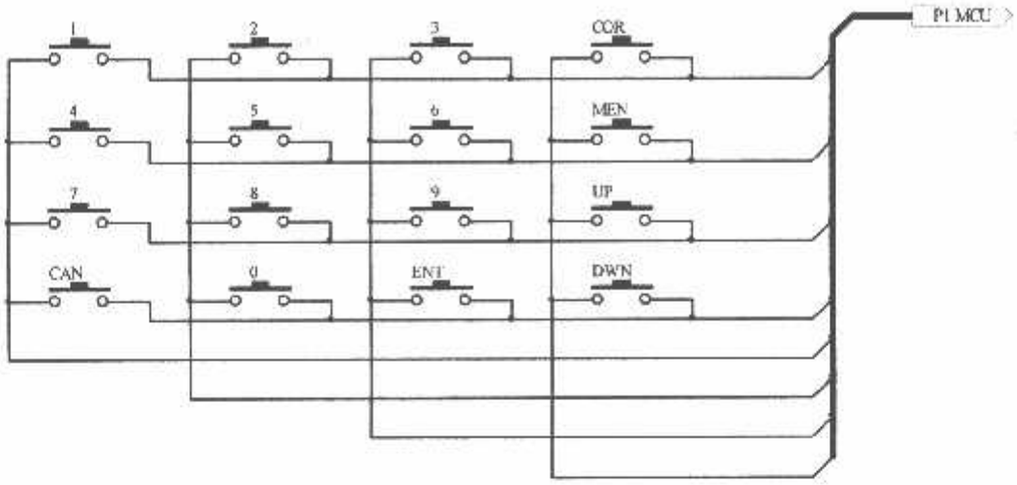
Sinyal – sinyal yang dipergunakan oleh LCD adalah data bus, RS, R/W dan E. Sinyal E dihubungkan ke P2.1 untuk mengaktifkan LCD. LCD akan aktif jika Mikrokontroller memberikan instruksi tulis pada alamat LCD. Sedangkan P2.0 dipergunakan untuk memberikan sinyal RS yang membedakan data yang diberikan pada LCD. Sinyal RS diberikan ke LCD untuk membedakan sinyal antara instruksi program atau instruksi penulisan data. Untuk pin R/W akan berlogika low (0) apabila dihubungkan dengan ground maka LCD difungsikan hanya untuk menuliskan program atau data ke display.Untuk mengambil data dari Mikrokontroller maka Pin – Pin data dihubungkan dengan Port 0 yang merupakan Pin – pin data dari Mikrokontroller.



Gambar 3-8 Perencanaan Rangkain LCD

3.2.5 Keypad

Keypad digunakan untuk memasukkan kode dosen, *password* dosen, pilihan mata kuliah dan kehadiran mahasiswa. Keypad yang digunakan adalah keypad berukuran 4 x 4 (4 bagian kolom dan 4 bagian baris) yang terdiri dari saklar angka 0 sampai 9 sebagai penunjuk angka kode dosen, pilihan mata kuliah dan password dosen. Dan karakter *Enter, Cancel, Up, Down, Cord* dan *Men* yang digunakan sebagai proses menaikkan atau menurunkan *cursor* tiap baris pada LCD jika karakter yang ditampilkan lebih dari 32 karakter, pembatalan kesalahan penulisan, pembatalan proses inisialisasi dan keluar dari proses inisialisasi. Jumlah data yang dikirim oleh keypad pada mikrokontroler tergantung dari jumlah digit ID dosen dan *password* dosen, yaitu untuk ID dosen sebanyak 3 digit sedangkan untuk *password* dosen sebanyak 4 digit. Gambar 3.7 memperlihatkan rangkain keypad matrik 4 x 4 yang terhubung dengan mikrokontroler.



Gambar 3-9 Perencanaan Rangkain Keypad Matrik 4x4

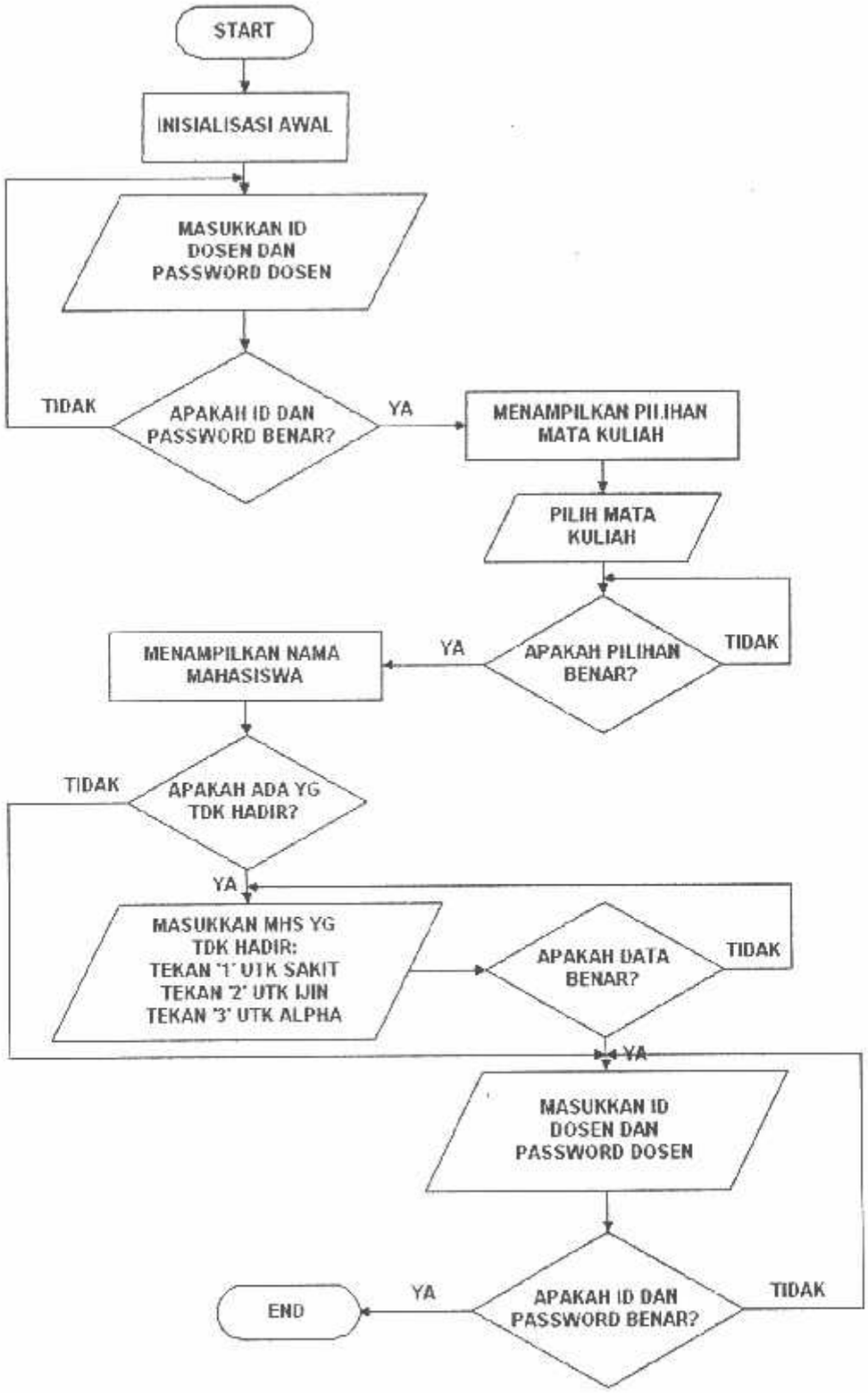
3.3. Perencanaan Software

Komputer disini digunakan sebagai pengontrol pengiriman dan penerimaan data ID, *password*, pemilihan mata kuliah dan nama mahasiswa yang tidak hadir, yang selanjutnya akan mengolah permintaan dari *slave* mikrokontroller yang telah masuk dalam *data base* komputer. Komunikasi yang terjadi antara PC dan *slave* adalah komunikasi serial *half duplex*.

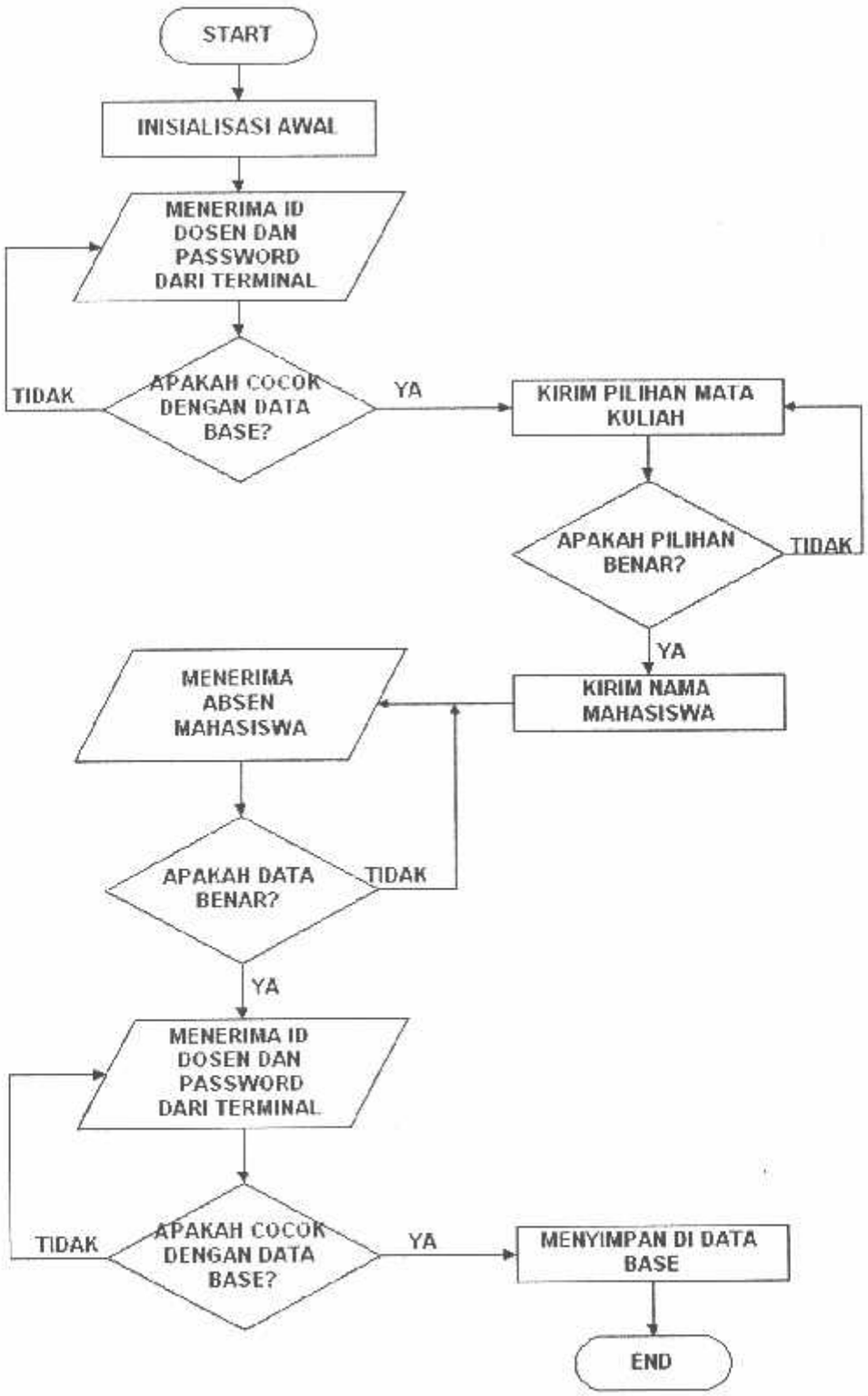
Langkah-langkah perencanaan *software:*

1. Perintah memasukkan ID dosen yang akan mengajar.
2. Konfirmasi ID dosen dengan *data base*.
3. Perintah memasukkan *password* dosen.
4. Konfirmasi *password* dosen dengan *database*.
5. Menampilkan pilihan mata kuliah.
6. Konfirmasi pilihan mata kuliah dengan *database*.
7. Menampilkan nama mahasiswa yang mengambil mata kuliah yang dipilih.
8. Jika ada mahasiswa yang tidak hadir, informasikan dengan keterangan sakit, ijin atau alpha.
9. Simpan semua data yang masuk pada *database*.

FLOWCHART MIKROKONTROLLER



FLOWCHART PADA PC



BAB IV

PENGUJIAN ALAT

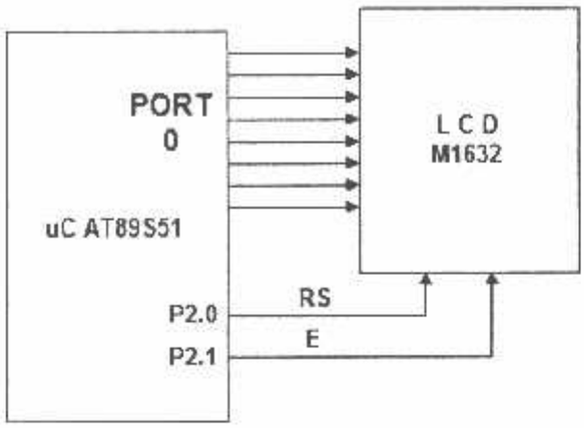
Pada bab ini membahas tentang pengukuran dan pengujian alat yang dirancang, dimana meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*). Untuk mengetahui sistem yang dirancang sesuai dengan fungsi yang diharapkan, dilakukan pengujian terhadap sistem aplikasi tersebut baik secara kesceluruhan atau sub sistem. Berikut penjelasan mengenai prosedur pengukuran dan data hasil pengujian.

4.1. Pengujian LCD

- Tujuan Pengujian:

  Untuk mengetahui apakah LCD dapat bekerja dengan baik, yaitu menampilkan karakter sesuai yang diinginkan berupa huruf dan angka.

- Peralatan yang digunakan
  - Mikrokontroler AT89S51.
  - LCD M1632.
  - Catu Daya.
- Prosedur Pengujian
  - Membuat rangkaian sebagai berikut:



Gambar 4-1 Blok Diagram Pengujian LCD

- Membuat program pada mikrokontroler untuk menampilkan karakter pada LCD dengan *flowchart* sebagai berikut:

Gambar 4-2 Flowchart Pengujian LCD

- Mengamati tampilan pada LCD.

➡ Hasil Pengujian

Hasil pengujian yang diperoleh adalah LCD dapat menampilkan karakter tersebut pada kedua baris tanpa ada kesalahan. Pada baris pertama "Absensi" dan pada baris kedua "Mahasiswa ITN" seperti yang terlihat pada gambar.

Gambar 4-3 Tampilan LCD pada Pengujian

- Analisis Hasil Pengujian

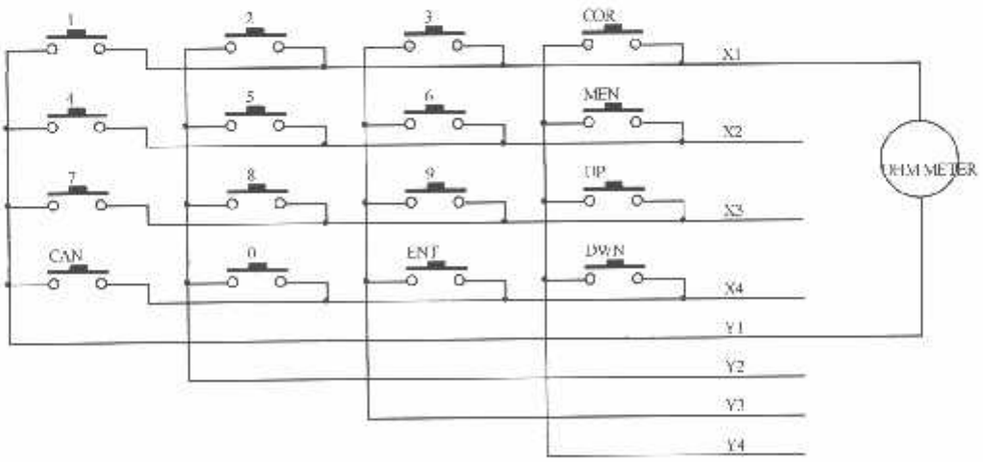
Dengan demikian rangkain interface modul LCD dapat berfungsi dengan baik dan dapat diakses oleh program.

4.2. Pengujian Keypad

- Tujuan Pengujian

Agar dapat diketahui apakah keypad bisa digunakan sebagai input.

- Peralatan yang digunakan
  - Keypad
  - Ohm Meter.
- Prosedur pengujian
  - Merangkai keypad dan Ohm Meter seperti dibawah ini:



Gambar 4-4 Rangkaian Pengujian Keypad

- Menekan tombol keypad satu persatu. Pada saat menekan satu tombol, hubungkan satu *probe* Ohm Meter pada kaki X dan satu lagi pada kaki Y.
- Mengamati hasil pengujian pada Ohm Meter.

➡ Hasil Pengujian

Tabel 4-1 Hasil Pengujian Keypad

TOMBOL	DATA DARI OHM METER							
	X1	X2	X3	X4	Y1	Y2	Y3	Y4
1	0	1	1	1	0	1	1	1
2	0	1	1	1	1	0	1	1
3	0	1	1	1	1	1	0	1
4	1	0	1	1	0	1	1	1
5	1	0	1	1	1	0	1	1
6	1	0	1	1	1	1	0	1
7	1	1	0	1	0	1	1	1
8	1	1	0	1	1	0	1	1
9	1	1	0	1	1	1	0	1
CAN	1	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1	1
ENT	1	1	1	0	1	1	0	1
COR	0	1	1	1	1	1	1	0
MEN	1	0	1	1	1	1	1	0
UP	1	1	0	1	1	1	1	0
DWN	1	1	1	0	1	1	1	0

➡ Analisis Hasil Pengujian

Keypad bekerja dengan baik dan sesuai dengan yang diharapkan.

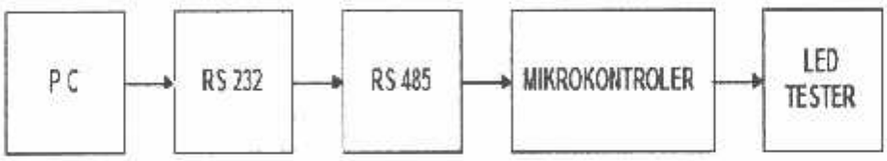
4.3. Pengujian Komunikasi Serial

4.3.1. Komputer Sebagai Pengirim dan Mikrokontroller Sebagai Penerima

➡ Tujuan:

Untuk mengetahui apakah komunikasi serial antara PC dan mikrokontroler dapat berjalan.

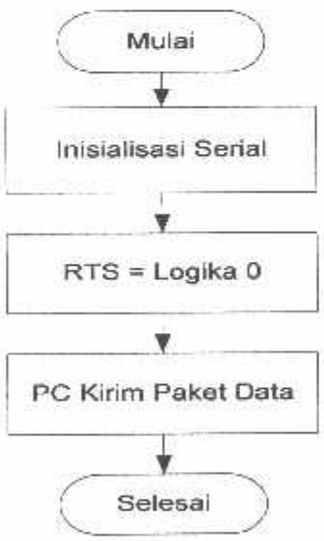
- Peralatan yang digunakan:
  - Power supply 5 volt.
  - Mikrokontroler.
  - PC.
  - 8 LED.
- Langkah-langkah pengujian:
  - Membuat rangkaian seperti pada gambar dibawah ini:



Gambar 4-5

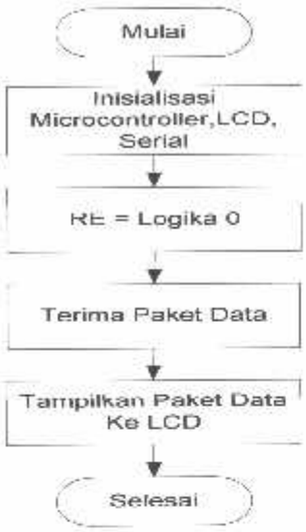
Blok Diagram Pengujian Komunikasi Serial dari PC ke Mikrokontroler

  - Membuat Program pada PC untuk mengirim data seperti pada flowchart dibawah ini:



Gambar 4-6 Flowchart Pengujian Komputer Sebagai Pengirim

- Membuat Program pada mikrokontroler untuk menerima data seperti pada flowchart dibawah ini:



Gambar 4-7
Flowchart Pengujian Mikrokontroller Sebagai Penerima

➧ Hasil Pengujian

Gambar 4-8
Tampilan Komunikasi Serial dari PC ke Mikrokontroler

Tabel 4-2
Hasil Pengujian Komunikasi serial dari PC ke Mikrokontroler

DATA INPUT PC	LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
01010101	0	1	0	1	0	1	0	1
10001000	1	0	0	0	1	0	0	0
00110011	0	0	1	1	0	0	1	1

➡ Analisa Pengujian

Data yang dikirimkan oleh PC dapat diterima dengan baik oleh Mikrokontroler. Ini berarti komunikasi serial antara PC dan mikrokontroler berjalan lancar.

4.3.2. Mikrokontroller Sebagai Pengirim dan Komputer Sebagai Penerima

➡ Tujuan:

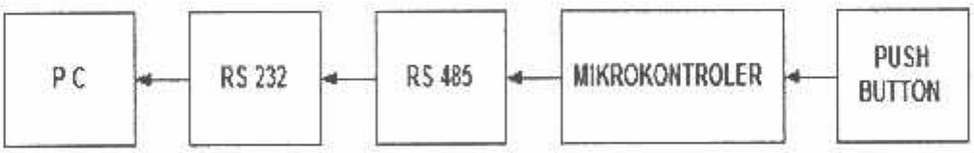
Untuk mengetahui apakah komunikasi serial antara PC dan mikrokontroler dapat berjalan.

➡ Peralatan yang digunakan:

- Power supply 5 volt.
- Mikrokontroler.
- PC.
- 8 Push Button.

➡ Langkah-langkah pengujian:

- Membuat rangkaian seperti pada gambar dibawah ini:



Gambar 4-9

Blok Diagram Pengujian Komunikasi Serial dari Mikrokontroler ke PC

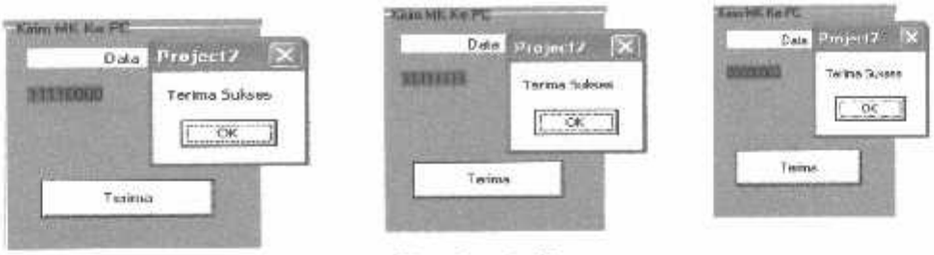
- Membuat Program pada mikrokontroler untuk mengirim data seperti pada flowchart dibawah ini:

Gambar 4-10
Flowchart Pengujian Mikrokontroller Sebagai Pengirim

- Membuat Program pada PC untuk menerima data seperti pada flowchart dibawah ini:

Gambar 4-11
Flowchart Pengujian Komputer Sebagai Penerima

- Hasil pengujian:



Gambar 4-12
Tampilan Komunikasi Serial dari Mikrokontroler ke PC

Tabel 4-3
Hasil Pengujian Komunikasi serial dari Mikrokontroler ke PC

DATA INPUT PADA P2								TAMPILAN PADA PC
PB1	PB2	PB3	PB4	PB5	PB6	PB7	PB8	
1	1	1	1	0	0	0	0	11110000
0	0	0	0	0	0	0	0	00000000
1	1	1	1	1	1	1	1	11111111

- Analisa Pengujian

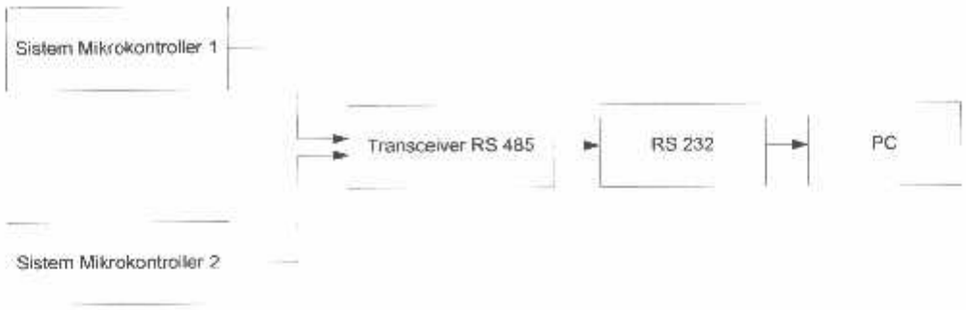
Data yang dikirimkan oleh mikrokontroler dapat diterima dengan baik oleh PC. Ini berarti komunikasi serial antara mikrokontroler dan PC berjalan lancar.

4.4. Pengujian Secara Keseluruhan

- Tujuan Pengujian

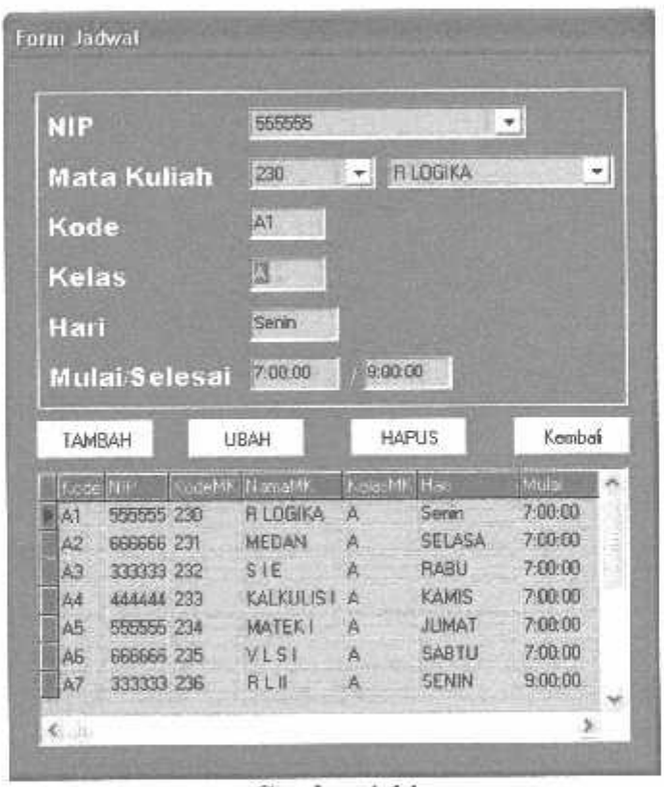
Untuk membuktikan bahwa sistem dapat bekerja dengan baik sesuai dengan perencanaan.

- Peralatan Yang digunakan
  - PC
  - Keseluruhan *Hardware*.
- Prosedur Pengujian
  - Menginstalasi sebagaimana dibawah



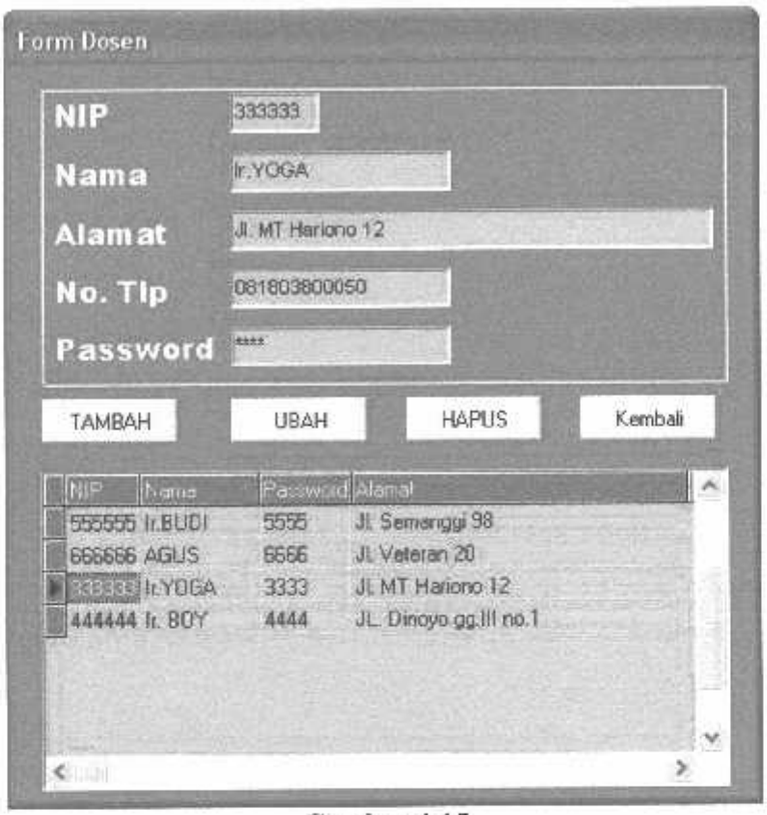
Gambar 4-13 Blok Diagram Pengujian Sistem Keseluruhan

- Memasukkan data pada *data base*.
  - Memasukkan jadwal kuliah pada form dibawah



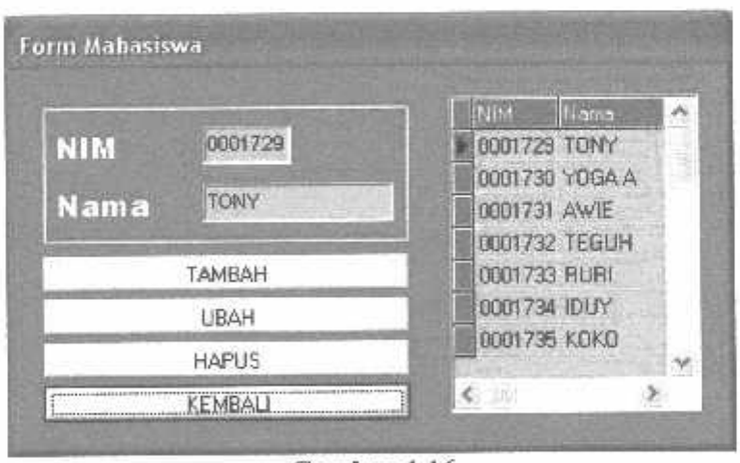
Gambar 4-14
Tampilan PC Saat Pengujian Memasukkan Jadwal Mata Kuliah

- Memasukkan NIP (ID) dosen, nama dosen, *password* dosen dan nama mata kuliah sepeti pada gambar dibawah.



Gambar 4-15
Tampilan PC Saat Pengujian Memasukkan Nama Dosen

- Memasukkan nama mahsiswa berdasarkan mata kuliah.

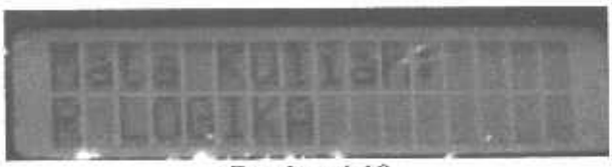


Gambar 4-16
Tampilan PC Saat Pengujian Memasukkan Nama Mahasiswa

- Memasukkan ID dosen, *password* dosen dari *keypad*

Gambar 4-17
Tampilan LCD Saat Pengujian Memasukkan ID dan *Password*

- Memilih mata kuliah dengan menekan 'ENT' pada mata kuliah yang ditampilkan LCD.

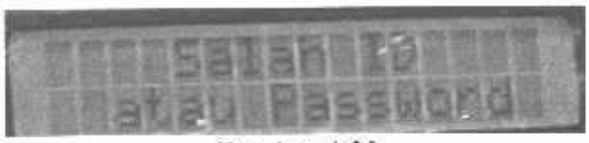


Gambar 4-18
Tampilan LCD Saat Pengujian Memilih Mata Kuliah

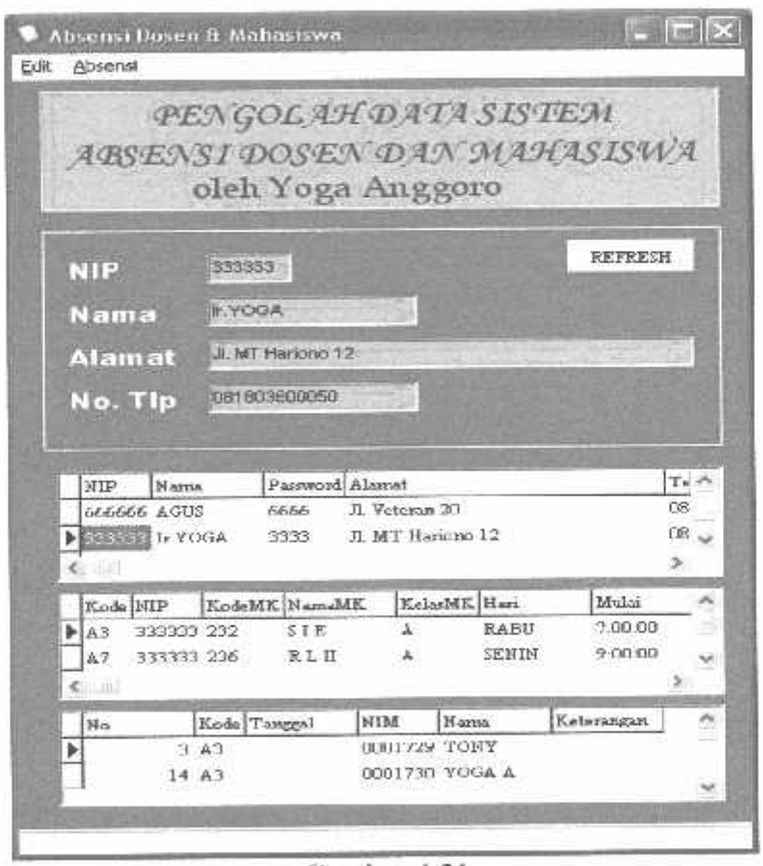
- Memasukkan mahasiswa yang tidak hadir dengan cara:
  - Memilih nama mahasiswa.
  - Tekan '1' untuk sakit
  - Tekan '2' untuk ijin
  - Tekan '3' untuk alpha
  - Tekan '4' untuk clear
- Masukkan kembali ID dosen dan *password* jika semua mahasiswa hadir dan keluar dari proses.
- Mengamati hasil pengujian.

- Hasil Pengujian

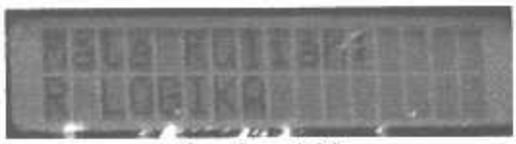
Gambar 4-19
Tampilan LCD Hasil Pengujian Memasukkan ID dan *Password*



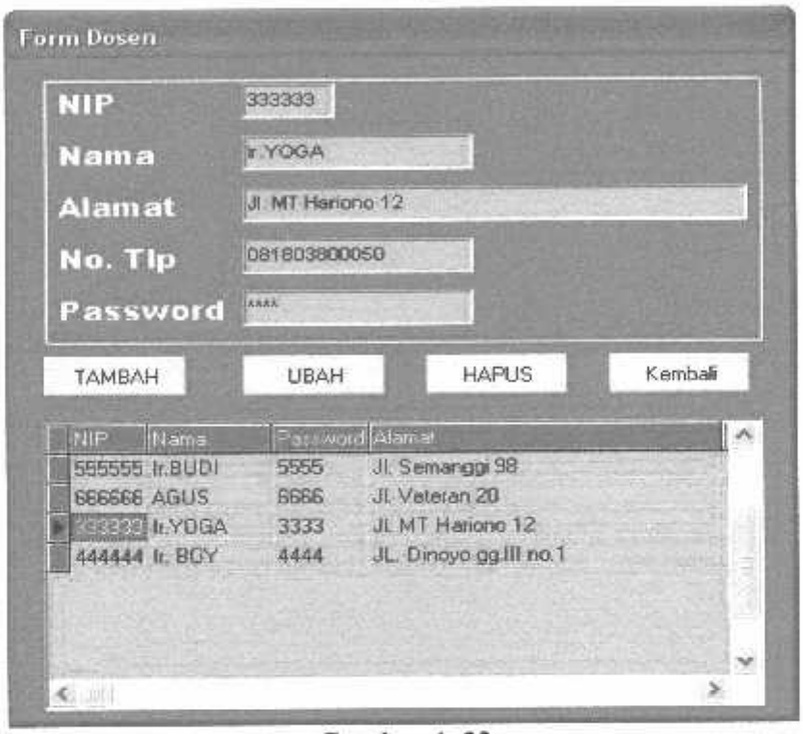
Gambar 4-20
Tampilan LCD Jika Terjadi Kesalahan Memasukkan ID atau *Password*



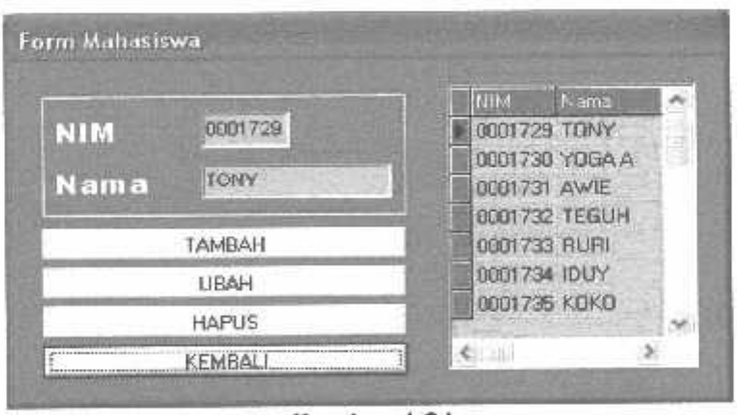
Gambar 4-21
Tampilan pada PC Saat Memasukkan ID dan *Password*



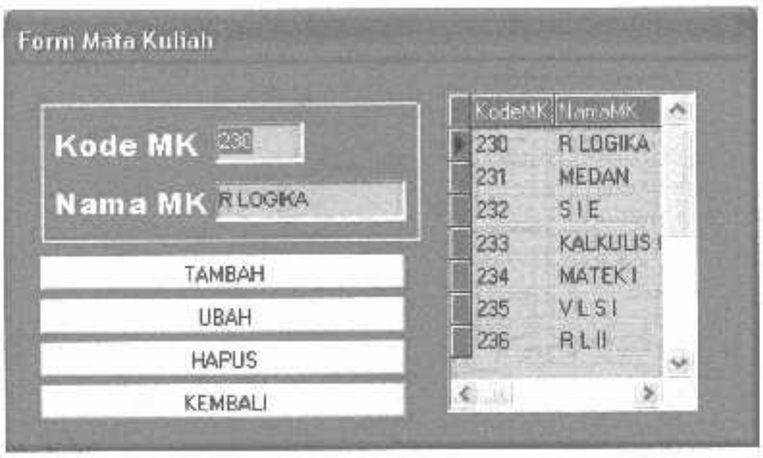
Gambar 4-22
Tampilan LCD Hasil Pengujian Pemilihan Mata Kuliah



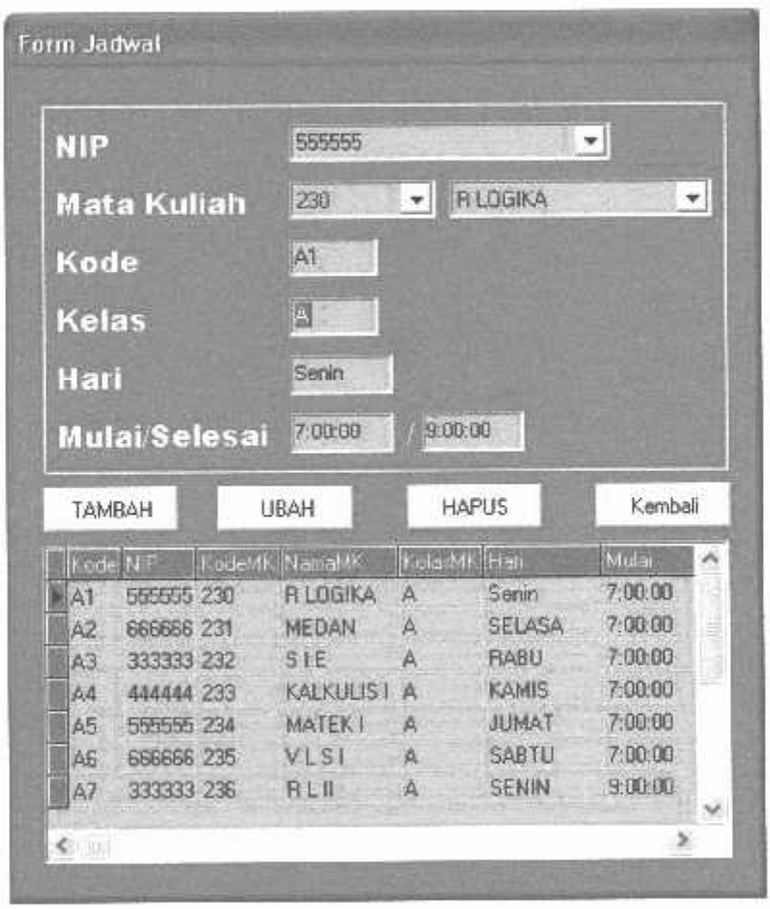
Gambar 4- 23
Tampilan Penginputan Dosen



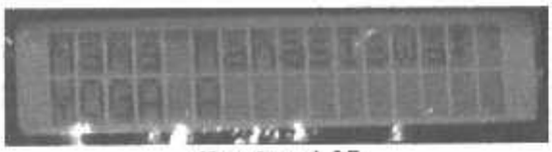
Gambar 4-24
Tampilan Penginputan Mahasiswa



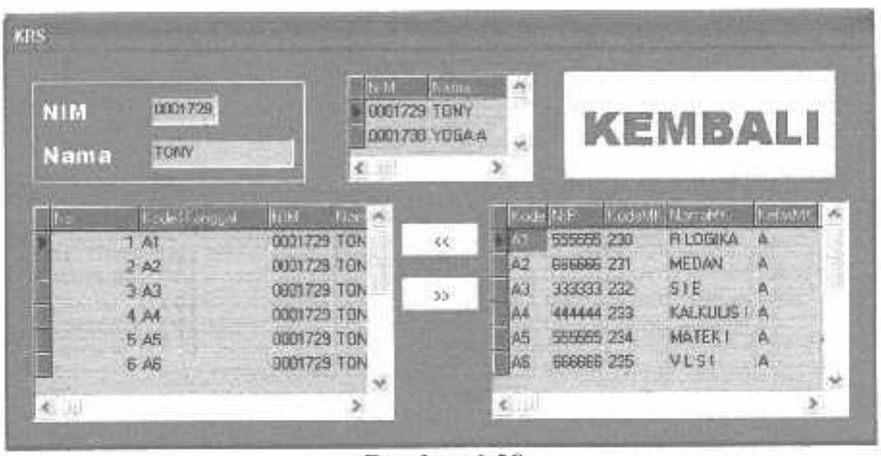
Gambar 4-25
Tampilan Penginputan Matakuliah



Gambar 4-26
Tampilan Jadwal Mata Kuliah pada PC



Gambar 4-27
Tampilan Nama Mahasiswa pada LCD



Gambar 4-28
Tampilan Nama Mahasiswa Berdasarkan Mata Kuliah

Gambar 4-29
Tampilan Absensi Mahasiswa Satu Pertemuan

Gambar 4-30
Tampilan Absensi Dosen

➧ Analisis Hasil Pengujian

Software dapat bekerja sesuai perancangan, yaitu dengan mengirim, menerima semua data dan menyimpan pada *data base*.

BAB V

PENUTUP

5.1. Kesimpulan

1. Alat ini bisa mempermudah sistem absensi dosen dan mahasiswa pada perguruan tinggi serta mempermudah perekapan data karena terpusat pada satu PC.
2. Dari hasil pengujian, jarak antara PC dan terminal adalah 10 M.
3. Alat ini terdiri dari keypad sebagai input, LCD sebagai display serta PC sebagai pengolah data dan penyimpan semua data.
4. Pengiriman dan penerimaan data tidak dapat dilakukan secara bersama-sama karena menggunakan mode *half duplex.*.
5. Bahasa pemrograman Borland Delphi 7.0 yang digunakan pada alat ini bisa dioperasikan pada PC dengan spesifikasi minimal:
   - INTEL P3
   - MICROSOFT WINDOWS 98

5.2. Saran

1. Sistem ini dapat dikembangkan dengan ditambahkan data-data lain yang berhubungan dengan perguruan tinggi.
2. Untuk jarak lebih jauh dari 1200 m dapat digunakan *repeater bus differential* SN 75178B.

3. Pengembangan *software* pada komputer agar dapat terintegrasi dengan sistem informasi yang telah ada.

Daftar Pustaka

1. Ainun, Moch dan Aistardi. 1997. *Bereksperimen Dengan Mikrokontroller 8031*. PT Elex Media Komputindo. Jakarta
2. Andi Nalwan, Paulus. 2003. *Teknik Antar Muka dan Pemograman*. PT Elex Media Komputindo. Jakarta
3. Eko Putra, Agfianto. 2002. *Belajar Mikrokontroller AT 89C51/52/55*. Gava Media. Yogyakarta
4. LCD module user manual M1632, Original 1985 dan Completrly revised 1987
5. AT89S51 *Data Sheet*. www.atmel.com
6. SN75176 *Data Sheet*. www.national.com
7. MAX 232 *Data Sheet*.
8. MAX 485 *Data Sheet*.

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

LEMBAR BIMBINGAN SKRIPSI

1. Nama	:	YOGA ANGGORO
2. NIM	:	00.17.232.
3. Jurusan	:	Teknik Elektro S-1
4. Konsentrasi	:	Teknik Elektronika
5. Judul Skripsi	:	

"RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA ABSENSI DOSEN DAN MAHASISWA BERBASIS MIKROKONTROLLER AT89S51"

6. Tanggal Pengajuan Skripsi	:	15 Febuari 2006
7. Selesai menulis skripsi	:	21 Maret 2006
8. Dosen Pembimbing I	:	Ir. Muhammad Luqman, MT
9. Dosen Pembimbing II	:	M. Ibrahim Ashari, ST
10. Telah Dievaluasi Dengan Nilai	:	97

Diperiksa dan Disetujui:

Dosen Pembimbing I

(Ir. Muhammad Luqman, MT)
NIP. P 131 793 370

Dosen Pembimbing II

(M.Ibrahim Ashari, ST)
NIP.P 103 0100358

Ketua Jurusan Teknik Elektro S-1

(Ir.F.Yudi Limpraptono, MT)
NIP.P. 103 950 0274

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No. 2
MALANG

No. 006852

UNTUK MAHASISWA

SURAT PUAS TUGAS / PRAKTIKUM

Nomor : ITN 02 / SKR / XLIII / SPT / 2006

Menerangkan bahwa mahasiswa :

Nama : YOGA ANGGORO NIM : 00.17.232
Jurusan : Teknik ELEKRO MESIN S 1
Angkatan : 2000 ( )
Telah menyelesaikan tugas : Skripsi :

Semester : XII (DUA BELAS) ( )
Dengan Hasil : 97 (sembilan puluh tujuh)

Malang, 19 - Maret - 2006

Dosen / Asisten ybs.

IR. Mohammad Luqman, MS.

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No. 2
MALANG

No. 006851

UNTUK MAHASISWA

SURAT PUAS TUGAS / PRAKTIKUM

Nomor : ITN 01 / SKR / XLIII / SPT / 2006

Menerangkan bahwa mahasiswa :

Nama : YOGA ANGGORO NIM : 00.17.232
Jurusan : Teknik ELEKTROMESIN S 1
Angkatan : 2000 ( )
Telah menyelesaikan tugas :

Semester : XII (DUA BELAS) ( )
Dengan Hasil : 97 (sembilan puluh tujuh)

Malang, 20 MARET 2006

Dosen / Asisten ybs.

M. Ibrahim Ashari, ST

INSTITUT TEKNOLOGI NASIONAL
Kampus I : Jl. Bendungan Sigura-gura No 2
Kampus II : Jl. Raya Karanglo, Km 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Yoga Anggoro
Nim : 00.17.232
Masa Bimbingan : 15 Febuari 2006 s/d 15 Juli 2006
Judul : Rancang Bangun Alat dan Sistem Pengolah Data Absensi Dosen dan Mahasiswa Berbasis Mikrokontroller AT89S51

NO	Tanggal	Uraian	Paraf Pembimbing
1	1- Maret	bab 1 & bab 2 Acc. bab 3 diperbaiki.	
2	3/maret/2006	Bab 1 - bab 3 Acc. Mhl. seminar diperbaiki.	
3	4/ /2006. /Maret	bab 4 → diuji sistem secara komprehensip. makalah harus diperbaiki.	
4	7/ - 2006 /maret	bab 4 dan makalah seminar Acc	
5	19. maret - 2006	Demo Alat Acc. hasil = Baik	
6		Acc utk Comprehensip	
7			
8			
9			
10			

Malang, 19 - Maret - 2006
Dosen Pembimbing

(Ir. M. Luqman, MT)

From. S-4a

INSTITUT TEKNOLOGI NASIONAL
Kampus I : Jl. Bendungan Sigura-gura No 2
Kampus II : Jl. Raya Karanglo, Km 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Yoga Anggoro
Nim : 00.17.232
Masa Bimbingan : 15 Febuari 2006 s/d 15 Juli 2006
Judul : Rancang Bangun Alat dan Sistem Pengolah Data Absensi Dosen dan Mahasiswa Berbasis Mikrokontroller AT89S51

NO	Tanggal	Uraian	Paraf Pembimbing
1	1 maret 06	acc · bab I .	
2	3 maret 2006	acc bab II	
3	4 maret '06	acc bab III	
4	7 maret '06	acc bab IV	
5	15 maret '06	acc bab V	
6			
7			
8			
9			
10			

Malang, 20 mar 2006
Dosen Pembimbing

(M. Ibrahim Ashari, ST)

From. S-4a

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

FORMULIR PERBAIKAN SKRIPSI

Nama	:	YOGA ANGGORO
NIM	:	00.17.232.
Masa Bimbingan	:	15 Febuari 2006 15 Juli 2006
Judul	:	

"RANCANG BANGUN ALAT DAN SISTEM PENGOLAH DATA
ABSENSI DOSEN DAN MAHASISWA
BERBASIS MIKROKONTROLLER AT89S51"

Tanggal	Uraian	Paraf
21 – 3 – 2006	-	-

Disetujui:

Penguji I

(Ir. Widodo Pudji M, MT)

Penguji II

(Ir. Mimien Mustikawati)

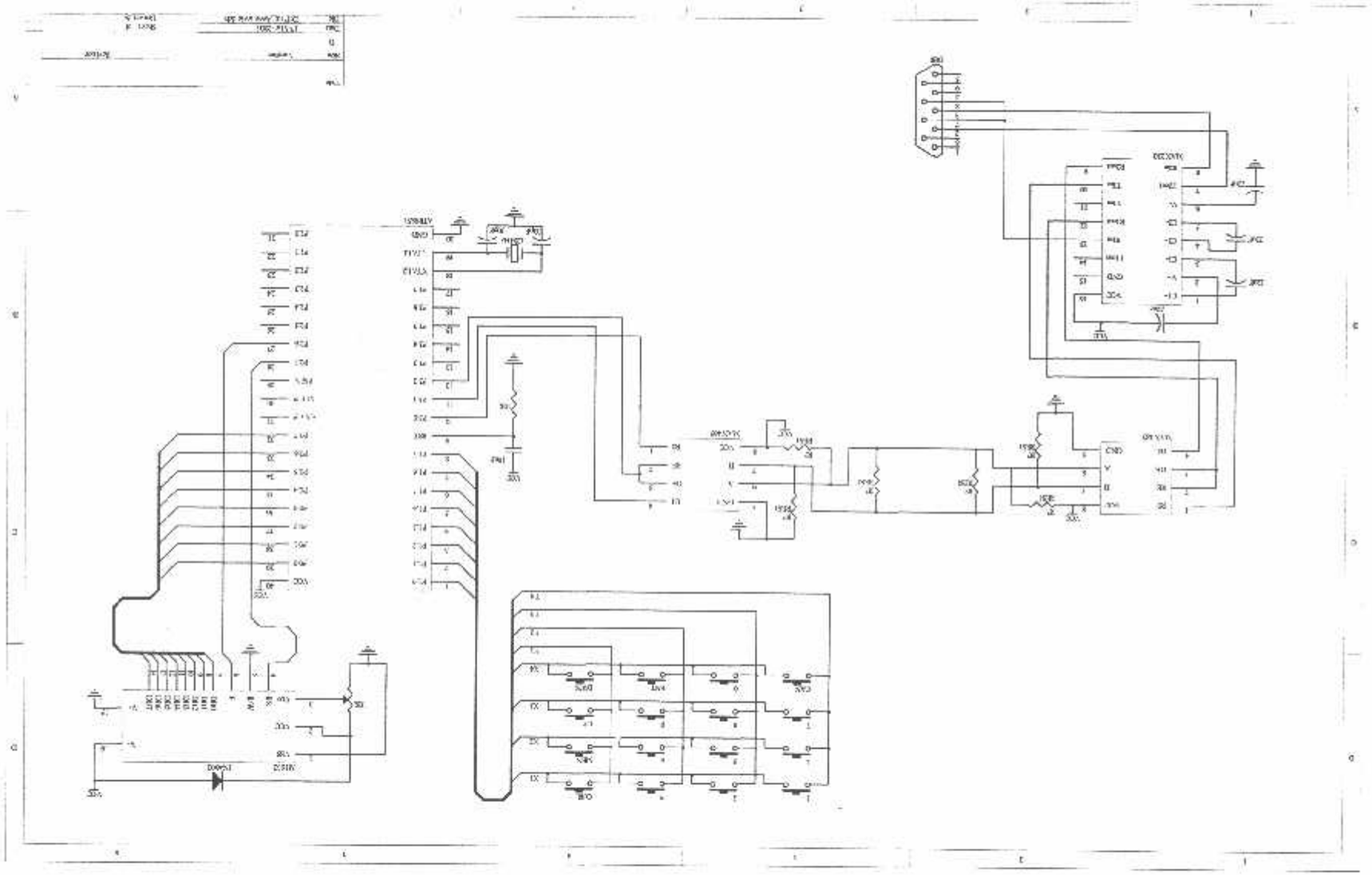
Mengetahui,

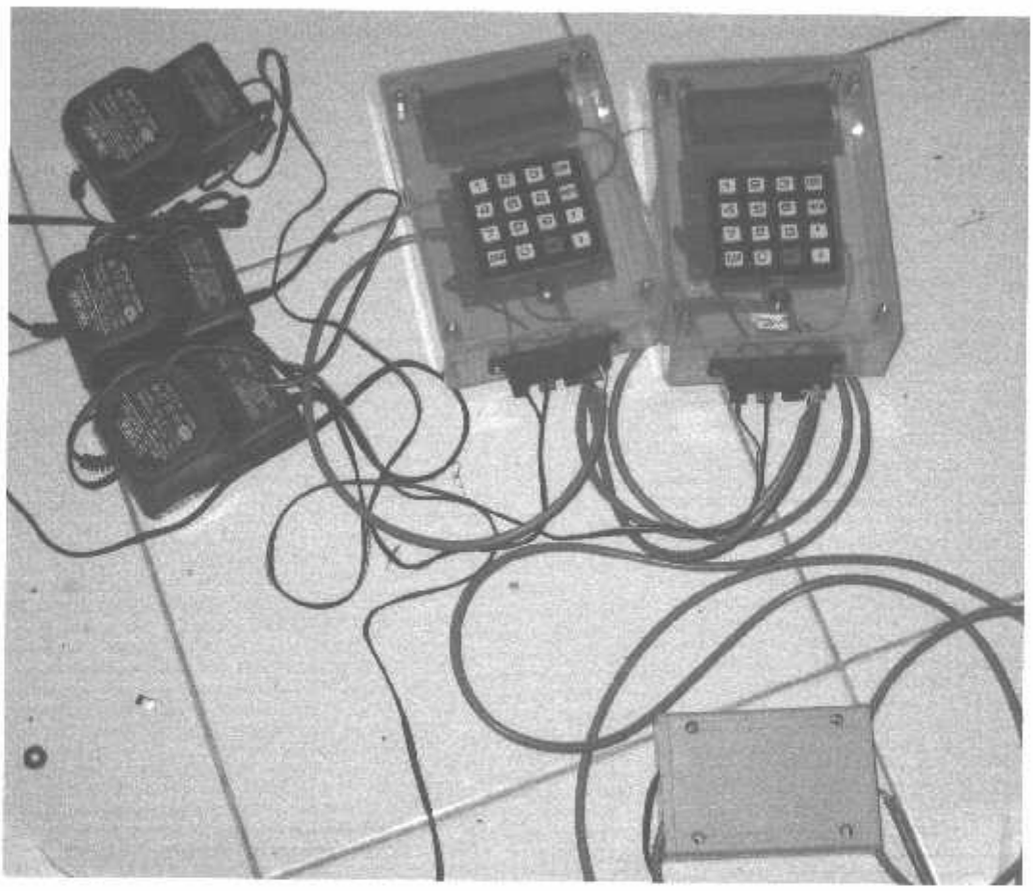
Dosen Pembimbing I

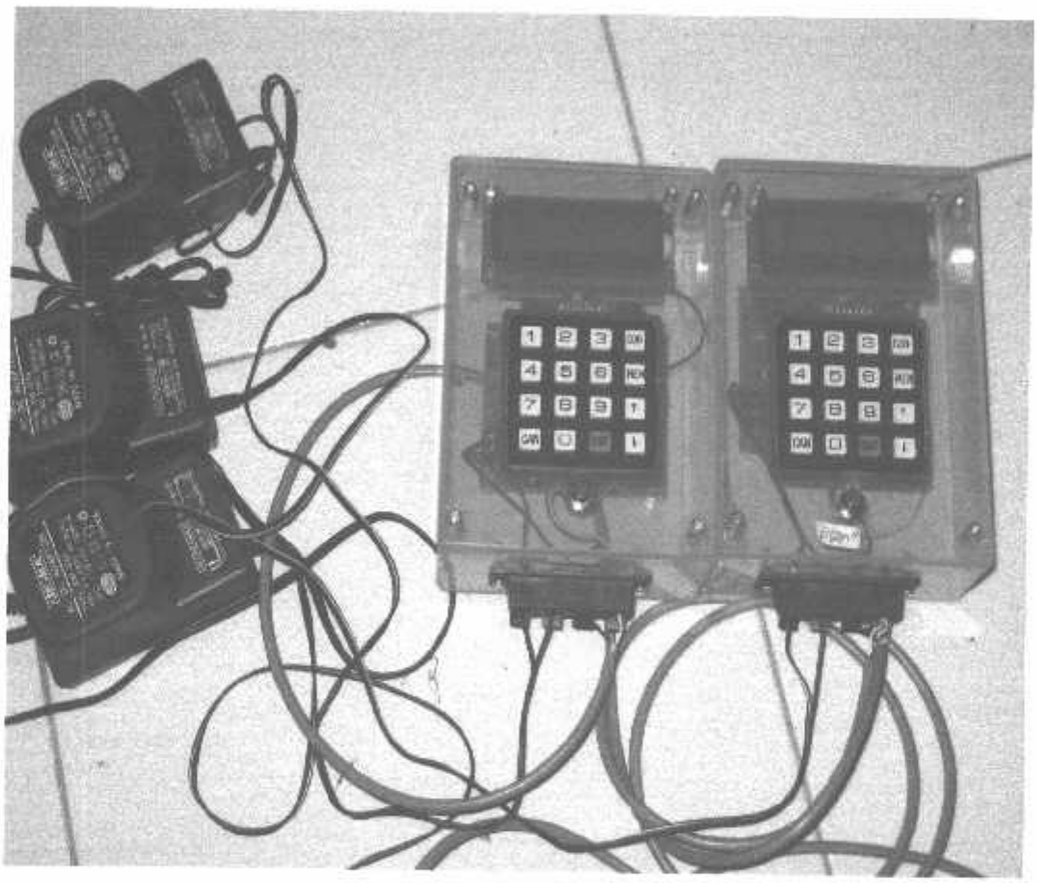
(Ir. Muhammad Luqman, MT)
NIP. P 131 793 370

Dosen Pembimbing II

(M.Ibrahim Ashari, ST)
NIP.P 103 0100358







```
dispclear     equ   00000001b
funcset       equ   00111000b
entrmod       equ   00000110b
dispon        equ   00001100b
cusor         equ   00001110b
blink         equ   00001101b
E           equ   p1.1
RS            equ   p1.0
con_lcd       equ   p1.2
data_lcd      equ   p1.3
tambah        equ    20h
control       equ    p3.2
pos           equ    '1'

buffer1       equ    30h
buffer2       equ    32h
buffer3       equ    33h

        org    0h
        jmp    mulai

;===program utama
mulai:
        mov tmod,#20h
        mov    TMOD,#20h
        mov    TH1,#230 ;1200
        setb   TR1
        mov    SCON,#50h
        lcall initlcd
yoga:   clr control
        lcall tawal
        lcall dly1000ms
        lcall dly1000ms
        lcall mid
        lcall kirimid
tan:
        lcall rx485
        cjne a,#'$',tan
        lcall rx485
        cjne a,#pos,tan
        lcall rx485
        cjne a,#'9',benaryoga
        lcall salahmas
        lcall dly1000ms
        lcall dly1000ms
        sjmp yoga
```

```
benaryoga:
        cjne a,#'8',benaryoga1
        lcall logout
        lcall dly1000ms
        lcall dly1000ms
        sjmp yoga
benaryoga1:
        lcall busek
        mov r5,#1
        mov dptr,#tmtk
        lcall cetak1
        mov r5,#1
        lcall baris2
tanda0:
        lcall rx485
        cjne a,#'$',tanda0
        lcall rx485
        cjne a,#pos,tanda0

        lcall rx485
        cjne a,#'Z',ben0
benloop:
        lcall key
        lcall cegatmtk
        cjne a,#'E',benloop1
        lcall kirawal
        MOV A,#'E'
        lcall tx485
        ljmp benyoga0
benloop1:
        mov buffer1,a
        lcall kirawal
        mov a,buffer1
        lcall tx485
        ljmp benaryoga1 ;tanda0
ben0:
        lcall dataout
        ljmp beno0
benyoga0:
        ;lcall kirawal

        lcall busek
        mov r5,#1
        mov dptr,#tnama
        lcall cetak1
        mov r5,#1
```

```
        lcall baris2
tanda1:
        lcall rx485
        cjne a,#'$',tanda1
        lcall rx485
        cjne a,#pos,tanda1
benarsela0:
        lcall rx485
        cjne a,#'Z',benarsela
        lcall key
        cjne a,#'E',benaryoga0

;==================perbedaan

        lcall kirawal
        MOV A,#'D'
        lcall tx485
        ljmp mulai
benarsela:
        lcall dataout
        ljmp benarsela0
benaryoga0:
        mov buffer1,a
        lcall kirawal
        mov a,buffer1
        lcall tx485
        ljmp benyoga0 ;tanda1



;===subprogram

logout: mov r5,#1
        mov dptr,#tlogout1
        lcall cetak1
        mov r5,#1
        mov dptr,#tlogout2
        lcall cetak2
        ret

cegatmtk:
        cjne a,#'C',cegatmtk1
        mov a,#'G'
        ret
cegatmtk1:
        cjne a,#'F',cegatmtk2
```

```
        mov a,#'H'
cegatmtk2:
        ret

tawal:  lcall busek
        mov r5,#1
        mov dptr,#tawal1
        lcall cetak1
        mov r5,#1
        mov dptr,#tawal2
        lcall cetak2
        ret

salahmas:
        lcall busek
        mov r5,#1
        mov dptr,#tsalahmas1
        lcall cetak1
        mov r5,#1
        mov dptr,#tsalahmas2
        lcall cetak2
        ret

kirawal:
        call tunggu
        lcall dly100ms
        mov a,#pos
        lcall tx485
        mov a,#','
        lcall tx485
        ret

tunggu:
        lcall rx485
        cjne a,#'$',tunggu
tunggu0:
        lcall rx485
        cjne a,#pos,tunggu
tunggu1:
        lcall rx485
        cjne a,#'5',tunggu
        ret

kirimid:
        lcall kirawal
        mov r0,#40h
```

```
kirimid3:
        mov a,@r0
        cjnc a,#'Z',kirimid2
        sjmp kirimidak
kirimid2:
        lcall tx485
        inc r0
        sjmp kirimid3
kirimidak
        ret
rx485:
        clr  control
        jnb     ri,$
        clr     ri
        mov  a,sbuf
        ret

tx485:  setb control
        mov     SBUF,A
        jnb     ti,$
        clr     ti
        ;call   dly100ms
        clr  control
        ret

mid:    lcall busek
        mov r5,#1
        mov dptr,#tmid1
        lcall cetak1
        mov buffer3,#6
        mov r0,#40h
mid1:   lcall key
        mov @r0,a
        lcall dataout
        inc r0
        djnz buffer3,mid1
        mov @r0,#','
        inc r0
mid2:   lcall key
        cjne a,#'E',mid2
        mov r5,#1
        mov dptr,#tmid2
        lcall cetak2
        mov buffer3,#4
mid3:   lcall key
        mov @r0,a
```

```
        mov a,#'*'
        lcall dataout
        inc r0
        djnz buffer3,mid3
mid4:   lcall key
        cjne a,#'E',mid4
        mov @r0,#'Z'
        ret

key:
    mov p2,#0feh
    jb p2.4,key1
        mov a,#31h
        jnb p2.4,$
        ljmp keyakhir
key1:
    jb p2.5,key2
        mov a,#32h
        jnb p2.5,$
        ljmp keyakhir
key2:
    jb p2.6,key3
        mov a,#33h
        jnb p2.6,$
        ljmp keyakhir
key3:
    jb p2.7,key4
        mov a,#41h
        jnb p2.7,$
        ljmp keyakhir
key4:
    mov p2,#11111101b
    jb p2.4,key5
    mov a,#34h
        jnb p2.4,$
        ljmp keyakhir
key5:
    jb p2.5,key6
        mov a,#35h
        jnb p2.5,$
        ljmp keyakhir
key6:
    jb p2.6,key7
        mov a,#36h
        jnb p2.6,$
        ljmp keyakhir
```

```
key7:
      jb p2.7,key8
         mov a,#42h
         jnb p2.7,$
         ljmp keyakhir
key8:
      mov p2,#11111011b
      jb p2.4,key9
         mov a,#37h
         jnb p2.4,$
         ljmp keyakhir
key9:
      jb p2.5,key10
         mov a,#38h
         jnb p2.5,$
         ljmp keyakhir
key10:
      jb p2.6,key11
         mov a,#39h
         jnb p2.6,$
         ljmp keyakhir
key11:
      jb p2.7,key12
         mov a,#'C'
         jnb p2.7,$
         ljmp keyakhir

key12:
         mov p2,#11110111b
      jb p2.4,key13
         mov a,#'L'
         jnb p2.4,$
         ljmp keyakhir
key13:
      jb p2.5,key14
         mov a,#30h
         jnb p2.5,$
         ljmp keyakhir
key14:
      jb p2.6,key15
         mov a,#'E'
         jnb p2.6,$
         ljmp keyakhir
key15:
      jb p2.7,key16
         mov a,#'F';20h
```

```
        jnb p2.7,$
        ljmp keyakhir1

key16:

    ljmp key
keyakhir:
keyakhir1:

    ret

kirimser:
        clr     ES
        setb    control
        mov     sbuf,A          ; kirim lewat serial comm.
        jnb     ti,$
        clr     ti
        setb    ES
        clr     control
        ret

detik5: lcall dly1000ms
        lcall dly1000ms
        lcall dly1000ms
        lcall dly1000ms
        lcall dly1000ms
        ret

busek:  mov r5,#1
        mov dptr,#kos
        lcall cetak1
        mov r5,#1
        mov dptr,#kos
        lcall cetak2
        ret


;=========routine lcd
baris2:
    mov a,r5
    add a,#0c0h ;11000000b
    sjmp posisisub
baris1:
    mov a,r5
    add a,#80h; 10000000b
posisisub:
```

```
        dec  a
        lcall controlout
        ret
cetak2:
        lcall baris2
        sjmp ansa
cetak1:
        lcall baris1
ansa:
        sjmp outstring
loop:
        lcall dataout
        inc dptr
outstring:
        clr a
        movc a,@a+dptr
        cjne a,#'$',loop
        ret
controlout:
        push dph
        push dpl
        clr rs
        sjmp out
dataout:
        push dph
        push dpl
        setb rs
out:
        setb e
        lcall geser
        mov r6,#250
        djnz r6,$
        pop dpl
        pop dph
        clr e
        ret

geser:
        clr con_lcd
        mov   b,#8
a1:
        rrc a
        mov data_lcd,c
        nop
        nop
        nop
```

```
        nop
        setb con_lcd
        nop
        nop
        nop
        nop
        clr con_lcd
        djnz b,a1
        ret
dela:
        mov r6,#00h
dlylcdlp:
        mov r7,#00h
        djnz r7,$
        djnz r6,dlylcdlp
        ret
ldelay:
ldelay2:
        mov r3,#00h
ldelay1:
        lcall tdclay
        djnz r3,ldelay1
        djnz r2, ldelay2
        ret
tdelay:
        mov r7,#00h
        djnz r7,$
        ret
initlcd:
        mov  a,#dispclear
        lcall controlout
        lcall dela
        mov a,#funcset
        lcall controlout
        mov a,#dispon
        lcall controlout
        mov a,#entrmod
        lcall controlout
        ret

;=================================
;rutin delay
;---------------------------------
dly1000mS:
        mov    r2,#10
UlDel:
```

```
        lcall   dly100mS
      djnz    r2,UlDel
      ret
dly100mS:
        mov r6,#200
      ljmp    waktu
dly10mS:
        mov  r6,#20
      ljmp    waktu

; Timer 0,5 mS
; R6 = perkalian
waktu:
        mov     r7,#247
      djnz    r7,$
      djnz    r6,Waktu
      ret
;=========================

kos:
        db '                    $'
hapus1:
        db '              $'
hapus2:
        db '              $'
tmid1:
        db 'ID:$'
tmid2:
        db 'Password:$'
tsalahmas1:
        db '   Salah ID    $'
tsalahmas2:
        db '  atau Password $'
tawal1:
        db '    Absensi    $'
tawal2:
        db ' Mahasiswa ITN  $'
tlogout1:
        db '    LOG OUT     $'
tlogout2:
        db '              $'
tnama:
        db 'Nama Mahasiswa: $'
tmtk:
        db 'Mata Kuliah:    $'
        end
```

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, Menus, CPort, DBTables, Mask, DBCtrls, StdCtrls, Buttons,
  ComCtrls, Grids, DBGrids, ExtCtrls;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Button1: TButton;
    Edit1: TEdit;
    Memo1: TMemo;
    Edit2: TEdit;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    ENIP: TDBEdit;
    ENama: TDBEdit;
    EAlamat: TDBEdit;
    ETlp: TDBEdit;
    Memo2: TMemo;
    TblIDDosen: TTable;
    TblIDMahasiswa: TTable;
    TbMK: TTable;
    TbMahasiswa: TTable;
    TbDosen: TTable;
    MainMenu1: TMainMenu;
    Edit3: TMenuItem;
    IDDosen1: TMenuItem;
    IDMahasiswa1: TMenuItem;
    MataKuliah1: TMenuItem;
    Absensi1: TMenuItem;
    Mahasiswa1: TMenuItem;
    Dosen1: TMenuItem;
    DSIDDosen: TDataSource;
    StatusBar1: TStatusBar;
    ComPort1: TComPort;
    E1: TEdit;
```

```
    Button2: TButton;
    E2: TEdit;
    E3: TEdit;
    E4: TEdit;
    E5: TEdit;
    E6: TEdit;
    Etot1: TEdit;
    Etot2: TEdit;
    Panel1: TPanel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    KRS1: TMenuItem;
    DBGrid1: TDBGrid;
    DBGrid2: TDBGrid;
    DSMK: TDataSource;
    DSMahasiswa: TDataSource;
    DBGrid3: TDBGrid;
    TbDosenNotDetil: TTable;
    TbPresensiMhs: TTable;
    DBGrid4: TDBGrid;
    DSPresensiMhs: TDataSource;
    BKIRIM: TButton;
    LKELAS: TLabel;
    Timer1: TTimer;
    Edit4: TEdit;
    BSTART: TButton;
    JADWAL1: TMenuItem;
    TbPresensinotDetil: TTable;
    BitBtn1: TBitBtn;
    procedure IDDosen1Click(Sender: TObject);
    procedure IDMahasiswa1Click(Sender: TObject);
//  procedure Penunjuk(kelas:Integer);
    procedure MataKuliah1Click(Sender: TObject);
    procedure Mahasiswa1Click(Sender: TObject);
    procedure Dosen1Click(Sender: TObject);
    procedure ComPort1RxChar(Sender: TObject; Count: Integer);
    procedure kirimData(data : String);
    procedure MK_Mahasiswa(tempat : string);
    procedure NIM_Mahasiswa(tempat : string);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure KRS1Click(Sender: TObject);
```

```
    procedure BKIRIMClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure BSTARTClick(Sender: TObject);
    procedure JADWAL1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  data : String;
  baris_MK,baris_Presensi:array[0..10]of integer;
  NIP:array[0..10]of String;

implementation

uses Unit2, Unit3, Unit4, Unit5, Unit6, Unit7, Unit8;

{$R *.dfm}

procedure TForm1.kirimData(data : String);
begin
  ComPort1.FlowControl.ControlRTS:=rtsDisable;
  sleep(5);
  ComPort1.WriteStr(data);
  sleep(100);
  ComPort1.ClearBuffer(true,true);
  ComPort1.FlowControl.ControlRTS:=rtsEnable;
end;

procedure TForm1.MK_Mahasiswa(tempat : string);
var    i : Integer;
       kirim:string;
begin
i:=1;
kirimData('$'+tempat);
 repeat
  begin
  kirim:=TbMK.fieldbyname('NamaMK').AsString[i];
```

```
   kirimData(kirim);
   i:=i+1;
   end;
  until i > length(TbMK.fieldbyname('NamaMK').AsString);
kirimData('Z');
end;

procedure TForm1.NIM_Mahasiswa(tempat : string);
var i : Integer;
      kirim:string;
begin
i:=1;
kirimData('$'+tempat);
  repeat
  begin
  kirim:=TbPresensiMhs.fieldbyname('Nama').AsString[i];
   kirimData(kirim);
   i:=i+1;
   end;
  until i > length(TbPresensiMhs.fieldbyname('Nama').AsString);
kirimData('Z');
end;

procedure Penunjuk(Kelas:integer);
var i,j :Integer ;
   nip1:String;
begin
i:=1;
j:=1;
Form1.TbMK.First;
form1.TbPresensiMhs.First;
nip1:=NIP[kelas];
if Form1.TbIDDosen.FindKey([nip1]) then
  begin
   while baris_MK[Kelas] <> i do
   begin
   form1.TbMK.Next;
   i:=i+1;
   end;

   while baris_Presensi[Kelas] <> j do
   begin
   form1.TbPresensiMhs.Next;
```

```
    j:=j+1;
    end;
  end;
end;

function waktu (WAKTU1:String;WAKTU2:String):Integer;
var jamsl1,jamsl2:TStringList;
      jam1,jam2,menit1,menit2,TOTAL:Integer;
begin
  jamsl1:=TStringList.Create;
  jamsl1.Clear;
  jamsl1.Delimiter:=':';
  jamsl1.DelimitedText:=WAKTU1;
  jam1:=StrToInt(jamsl1[0]);
  menit1:=StrToInt(jamsl1[1]);


  jamsl2:=TStringList.Create;
  jamsl2.Clear;
  jamsl2.Delimiter:=':';
  jamsl2.DelimitedText:=WAKTU2;
  jam2:=StrToInt(jamsl2[0]);
  menit2:=StrToInt(jamsl2[1]);

  form1.E1.Text:=IntToStr(jam1);
  form1.E2.Text:=IntToStr(jam2);
  form1.E3.Text:=IntToStr(menit1);
  form1.E4.Text:=IntToStr(menit2);

  TOTAL:=(jam2-jam1)*60;
  Form1.Etot1.Text:=IntToStr(TOTAL);
  if menit2 <= menit1 then TOTAL:=TOTAL-(menit1-menit2)
  else TOTAL:=TOTAL+(menit2-menit1);
waktu:=TOTAL;
end;

procedure TForm1.IDDosen1Click(Sender: TObject);
begin
Form1.Visible:=false;
FIDDosen.Show;
end;

procedure TForm1.IDMahasiswa1Click(Sender: TObject);
```

```
begin
FMahasiswa.Show;
Form1.Visible:=false;
end;

procedure TForm1.MataKuliah1Click(Sender: TObject);
begin
Form1.Visible:=false;
FMK.Show;
end;

procedure TForm1.Mahasiswa1Click(Sender: TObject);
begin
FPresensiKuliah.Show;
Form1.Visible:=false;
end;

procedure TForm1.Dosen1Click(Sender: TObject);
begin
FPresensiDosen.Show;
Form1.Visible:=false;
end;

procedure TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
var    data1,WAKTU1,WAKTU2 : String;
       datasl : TStringList;
       kondisi,kondisi1 : Boolean;
       no : Integer;
begin
ComPort1.ReadStr(data1,count);
Timer1.Enabled:=false;
Memo2.Text:=Memo2.Text+data1;
data:=data+data1;
edit1.Text:=data;

  datasl:=TStringList.Create;
  datasl.Clear;
  datasl.Delimiter:=',';
  datasl.DelimitedText:=data;

 if length(data)=13 then
  begin
```

```
 TbIDDosen.First;
  repeat
  kondisi:= (TbIDDosen.FieldByName('NIP').AsString=datasl[1]) and
(TbIDDosen.FieldByName('Password').AsString=datasl[2]);
  begin


  if kondisi then
   begin
   StatusBar1.SimpleText:='Dosen';
   TbDosen.First;
    repeat
    begin
    kondisi1:=(TbDosen.FieldByName('NIP').AsString=datasl[1]) and
(TbDosen.FieldByName('waktu').AsInteger = 0);

          if kondisi1 then
          begin
          WAKTU1:=TimeToStr(TbDosen.FieldByName('Mulai').AsDateTime);
          WAKTU2:=TimeToStr(Time);
          E5.Text:=WAKTU1;
          E6.Text:=WAKTU2;
          TbDosen.Edit;
          TbDosen.FieldByName('selesai').AsDateTime:=Time;
          TbDosen.FieldByName('Waktu').AsInteger:=waktu(WAKTU1,WAKTU2);
          TbDosen.Post;
          kirimData('$'+datasl[0]+'8');
          end
          else TbDosen.Next;
     end;
     until TbDosen.Eof or kondisi1;

          if not kondisi1 then
          begin
          TbDosenNotDetil.Last;
          no:=TbDosenNotDetil.FieldByName('No').AsInteger+1;
          NIP[StrToInt(datasl[0])]:=datasl[1];
          TbDosen.Append;

TbDosen.FieldByName('NIP').AsString:=TbIDDosen.FieldByName('NIP').AsString;

TbDosen.FieldByName('Nama').AsString:=TbIDDosen.FieldByName('Nama').AsStri
ng;
```

```
            TbDosen.FieldByName('Tanggal').AsDateTime:=Date;
            TbDosen.FieldByName('Mulai').AsDateTime:=Time;
            TbDosen.FieldByName('Waktu').AsInteger:=0;
            TbDosen.FieldByName('Tempat').AsString:=datasl[0];
            TbDosen.FieldByName('no').AsInteger:=no;
            TbDosen.Post;
            TbDosenNotDetil.Close;
            TbDosenNotDetil.Open;
            kirimData('$'+datasl[0]+'1');
            TbMK.First;
            MK_Mahasiswa(datasl[0]);
            baris_MK[StrToInt(datasl[0])]:=1;
            end;

    end
   else TbIDDosen.Next;

   end;
   until TbIDDosen.Eof or kondisi;

   if not kondisi then
      begin
    StatusBar1.SimpleText:='Bukan Dosen';
    kirimData('$'+datasl[0]+'9');
    end;
  data:='';
  Timer1.Enabled:=true;
  end;
if length(data)=3 then
      begin
           if datasl[1]='G' then
           begin
            if baris_MK[StrToInt(datasl[0])] <= 1 then
              begin
              baris_MK[StrToInt(datasl[0])]:=1;
              Penunjuk(StrToInt(datasl[0]));
              end
            else
              begin
              baris_MK[StrToInt(datasl[0])]:=baris_MK[StrToInt(datasl[0])]-1;
              Penunjuk(StrToInt(datasl[0]));
              end;
       //     TbMK.Prior;
```

```
        MK_Mahasiswa(datasl[0]);
        end;

        if datasl[1]='H' then
        begin
         Penunjuk(StrToInt(datasl[0]));
         if not TbMK.Eof then
           begin
           baris_MK[StrToInt(datasl[0])]:=baris_MK[StrToInt(datasl[0])]+1;
           Penunjuk(StrToInt(datasl[0]));
           end;
    //     TbMK.Next;
        MK_Mahasiswa(datasl[0]);
        end;

   //     if datasl[1]='D' then
    //    begin
     //   Timer1.Enabled:=true;
      //  end;

        if datasl[1]='E' then
        begin
        TbDosen.Edit;

TbDosen.FieldByName('Kode').AsString:=TbMk.FieldByName('Kode').AsString;
        TbDosen.Post;
        TbPresensiMhs.Filtered:=false;
        TbPresensiMhs.Last;
        no:=TbPresensiMhs.FieldByName('No').AsInteger+1;

            TbMahasiswa.First;
            repeat
            TbPresensinotDetil.Append;

TbPresensinotDetil.FieldByName('NIM').AsString:=TbMahasiswa.FieldByName('NI
M').AsString;

TbPresensinotDetil.FieldByName('Nama').AsString:=TbMahasiswa.FieldByName('N
ama').AsString;

TbPresensinotDetil.FieldByName('Kode').AsString:=TbMK.FieldByName('Kode').A
sString;
                TbPresensinotDetil.FieldByName('Tanggal').AsDateTime:=Date;
```

```
            TbPresensinotDetil.FieldByName('No').AsInteger:=no;
            TbPresensinotDetil.Post;
            no:=no+1;
            TbMahasiswa.Next;
            until TbMahasiswa.Eof;

          TbPresensiMhs.Close;
          TbPresensiMhs.Open;

          TbPresensiMhs.Filter:='tanggal = '+QuotedStr(DateToStr(Date));
          TbPresensiMhs.Filtered:=true;
          TbPresensiMhs.First;
          baris_Presensi[StrToInt(datasl[0])]:=1;
          NIM_Mahasiswa(datasl[0]);
          end;

          if datasl[1]='C' then
          begin
          if baris_Presensi[StrToInt(datasl[0])] <= 1 then
            begin
            baris_Presensi[StrToInt(datasl[0])]:=1;
            Penunjuk(StrToInt(datasl[0]));
            end
           else
            begin
            baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]-
1;
            Penunjuk(StrToInt(datasl[0]));
            end;
          //TbPresensiMhs.Prior;
          NIM_Mahasiswa(datasl[0]);
          end;

          if datasl[1]='F' then
          begin
          Penunjuk(StrToInt(datasl[0]));
           if not TbPresensiMhs.Eof then
            begin

baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]+1;
            Penunjuk(StrToInt(datasl[0]));
            end;
          //TbPresensiMhs.Next;
```

```
            NIM_Mahasiswa(datasl[0]);
            end;

            if datasl[1]='I' then
            begin
            Penunjuk(StrToInt(datasl[0]));

            TbPresensiMhs.Edit;
            TbPresensiMhs.FieldByName('keterangan').AsString:='SAKIT';
            TbPresensiMhs.Post;
            if not TbPresensiMhs.Eof then
               begin

baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]+1;
               Penunjuk(StrToInt(datasl[0]));
               end;
            // TbPresensiMhs.Next;
            NIM_Mahasiswa(datasl[0]);
            end;

            if datasl[1]='2' then
            begin
            Penunjuk(StrToInt(datasl[0]));

            TbPresensiMhs.Edit;
            TbPresensiMhs.FieldByName('keterangan').AsString:='IJIN';
            TbPresensiMhs.Post;
            if not TbPresensiMhs.Eof then
               begin

baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]+1;
               Penunjuk(StrToInt(datasl[0]));
               end;
            //TbPresensiMhs.Next;
            NIM_Mahasiswa(datasl[0]);
            end;

            if datasl[1]='3' then
            begin
            Penunjuk(StrToInt(datasl[0]));

            TbPresensiMhs.Edit;
            TbPresensiMhs.FieldByName('keterangan').AsString:='ALPHA';
```

```
            TbPresensiMhs.Post;
            if not TbPresensiMhs.Eof then
              begin

baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]+1;
              Penunjuk(StrToInt(datasl[0]));
              end;
            //TbPresensiMhs.Next;
            NIM_Mahasiswa(datasl[0]);
            end;

            if datasl[1]='4' then
            begin
            Penunjuk(StrToInt(datasl[0]));

            TbPresensiMhs.Edit;
            TbPresensiMhs.FieldByName('keterangan').AsString:='';
            TbPresensiMhs.Post;
            if not TbPresensiMhs.Eof then
              begin

baris_Presensi[StrToInt(datasl[0])]:=baris_Presensi[StrToInt(datasl[0])]+1;
              Penunjuk(StrToInt(datasl[0]));
              end;
            //TbPresensiMhs.Next;
            NIM_Mahasiswa(datasl[0]);
            end;

      data:='';
      Timer1.Enabled:=true;
      end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
ComPort1.FlowControl.ControlRTS:=rtsEnable;
            baris_MK[1]:=1;
            baris_MK[2]:=1;
            baris_Presensi[1]:=1;
            baris_Presensi[2]:=1;
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
//MK_Mahasiswa(Edit4.Text);
end;

procedure TForm1.KRS1Click(Sender: TObject);
begin
Form1.Visible:=false;
FKRS.Show;
end;

procedure TForm1.BKIRIMClick(Sender: TObject);
begin
kirimData(Edit4.Text);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
          IF LKELAS.Caption='KELAS 1' then
            begin
            LKELAS.Caption:='KELAS 2';
            kirimData('$25');
            end
          else
            begin
            LKELAS.Caption:='KELAS 1';
            kirimData('$15');
            end;
end;

procedure TForm1.BSTARTClick(Sender: TObject);
begin
IF BSTART.Caption='START' THEN
     BEGIN
     BSTART.Caption:='STOP';
     Timer1.Enabled:=TRUE;

     END
ELSE   BEGIN
     BSTART.Caption:='START';
     Timer1.Enabled:=FALSE;
     END;
```

```
end;

procedure TForm1.JADWAL1Click(Sender: TObject);
begin
Form1.Visible:=false;
FJadwal.Show;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
data:='';
end;

end.
```

Features

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

Description

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

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

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

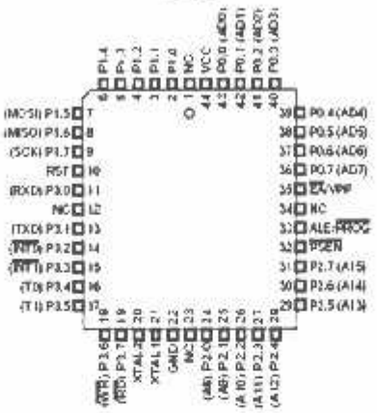
AT89S51

Rev. 2487A–10/01

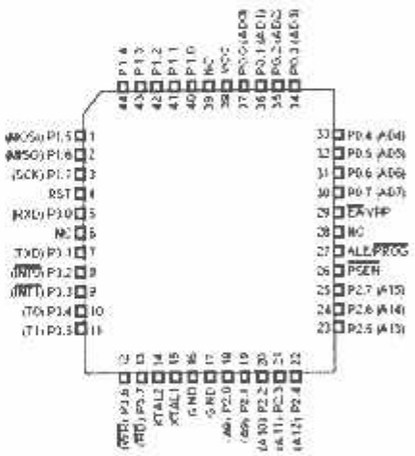
Pin Configurations

PDIP

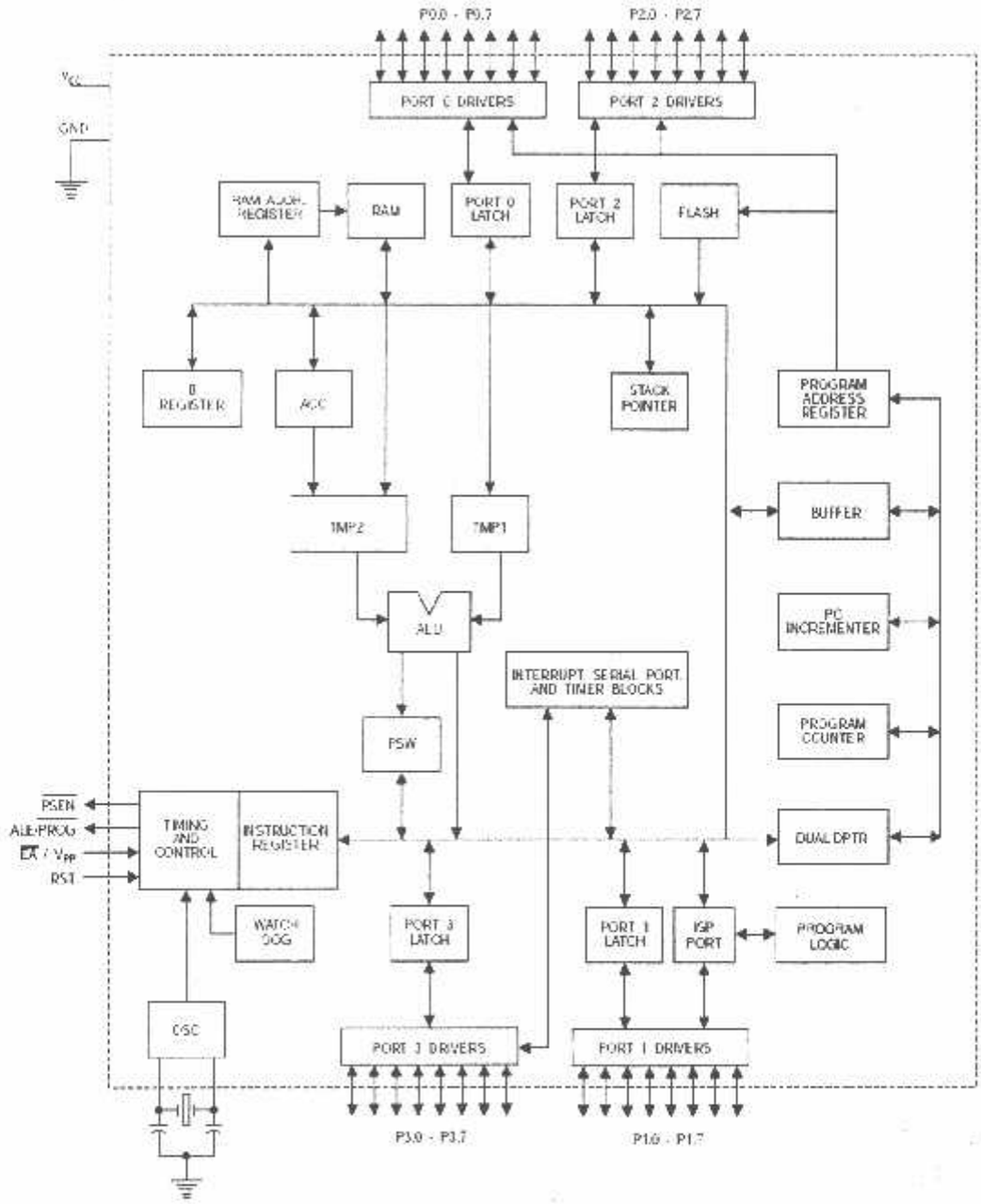
PLCC



TQFP



Block Diagram



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.

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

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

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 AT89S51, 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.

$\overline{PSEN}$

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

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

$\overline{EA}$/VPP

External Access Enable. $\overline{EA}$ 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, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ 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 1.

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

Table 1. AT89S51 SFR Map and Reset Values

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

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.

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

Table 2. AUXR: Auxiliary Register

AUXR — Address = 8EH — Reset Value = XXX00XX0B

Not Bit Addressable

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

– Reserved for future expansion

DISALE: 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

DISRTO: Disable/Enable Reset out

DISRTO

0 — Reset pin is driven High after WDT times out

1 — Reset pin is input only

WDIDLE: Disable/Enable WDT in IDLE mode

WDIDLE

0 — WDT continues to count in IDLE mode

1 — WDT halts counting in IDLE mode

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

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

Table 3. AUXR1: Auxiliary Register 1

AUXR1

Address = A2H

Reset Value = XXXXXXX0B

Not Bit Addressable

Bit	7	6	5	4	3	2	1	0
			–	–	–	–	–	DPS

– Reserved for future expansion

DPS Data Pointer Register Select

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

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

Data Memory

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

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 AT89S51 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

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

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S51 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and 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 AT89S51 operates the same way as the UART in the AT89C51. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

Timer 0 and 1

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

Interrupts

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

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

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

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

Table 4. Interrupt Enable (IE) Register

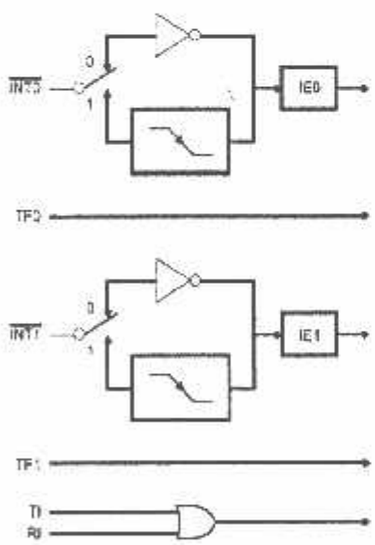
(MSB)							(LSB)
EA	–	–	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

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

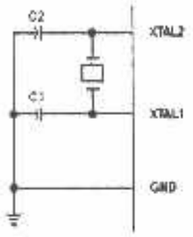
Figure 1. Interrupt Sources



Oscillator Characteristics

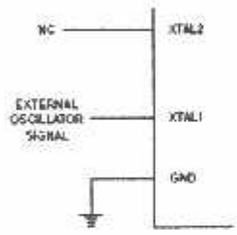
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected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

Figure 2. Oscillator Connections



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

Figure 3. External Clock Drive Configuration



Idle Mode

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

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

Power-down Mode

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

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

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

Program Memory Lock Bits

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

Table 6. Lock Bit Protection Modes

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

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

Programming the Flash – Parallel Mode

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

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

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

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

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

Ready/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) = 51H indicates 89S51
(200H) = 06H

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

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

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

Programming the Flash – Serial Mode

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

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

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

Serial Programming Algorithm

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

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

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

Power-off sequence (if needed):

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

Set RST to "L".

Turn V_{CC} power off.

Data Polling: The 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 8 on page 18.

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.

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

Table 7. Flash Programming Modes

Mode	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	P2.3-0	P1.7-0
												Address	
Write Code Data	5V	H	L	(1)	12V	L	H	H	H	H	D_{IN}	A11-8	A7-0
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A11-8	A7-0
Write Lock Bit 1	5V	H	L	(2)	12V	H	H	H	H	H	X	X	X
Write Lock Bit 2	5V	H	L	(2)	12V	H	H	H	L	L	X	X	X
Write Lock Bit 3	5V	H	L	(2)	12V	H	L	H	H	L	X	X	X
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L	(3)	12V	H	L	H	L	L	X	X	X
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	51H	0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	08H	0010	00H

Notes:
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 4. Programming the Flash Memory (Parallel Mode)

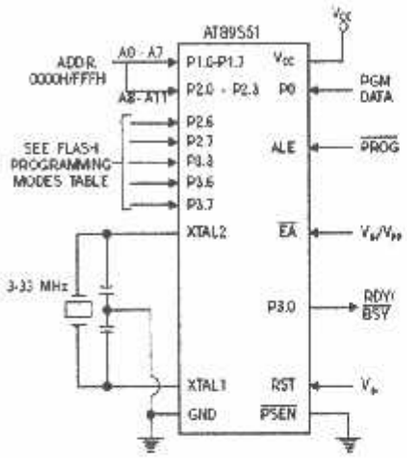
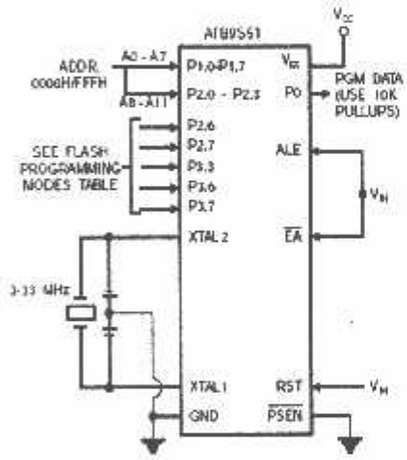


Figure 5. Verifying the Flash Memory (Parallel Mode)



19-0122; Rev 5; 2/96

MAXIM

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

General Description

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120µA and 500µA of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1µA. All parts operate from a single 5V supply.

Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488–MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

Applications

Low-Power RS-485 Transceivers

Low-Power RS-422 Transceivers

Level Translators

Transceivers for EMI-Sensitive Applications

Industrial-Control Local Area Networks

Features

- **In µMAX Package: Smallest 8-Pin SO**
- **Slew-Rate Limited for Error-Free Data Transmission (MAX483/487/488/489)**
- **0.1µA Low-Current Shutdown Mode (MAX481/483/487)**
- **Low Quiescent Current:**
  120µA (MAX483/487/488/489)
  230µA (MAX1487)
  300µA (MAX481/485/490/491)
- **-7V to +12V Common-Mode Input Voltage Range**
- **Three-State Outputs**
- **30ns Propagation Delays, 5ns Skew (MAX481/485/490/491/1487)**
- **Full-Duplex and Half-Duplex Versions Available**
- **Operate from a Single 5V Supply**
- **Allows up to 128 Transceivers on the Bus (MAX487/MAX1487)**
- **Current-Limiting and Thermal Shutdown for Driver Overload Protection**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX481CPA	0°C to +70°C	8 Plastic DIP
MAX481CSA	0°C to +70°C	8 SO
MAX481CUA	0°C to +70°C	8 µMAX
MAX481C/D	0°C to +70°C	Dice*

Ordering Information continued at end of data sheet.

* *Contact factory for dice specifications.*

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF TRANSMITTERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC}) 12V
Control Input Voltage ($\overline{RE}$, DE) ..... -0.5V to (V_{CC} + 0.5V)
Driver Input Voltage (DI) ..... -0.5V to (V_{CC} + 0.5V)
Driver Output Voltage (A, B) ..... -8V to +12.5V
Receiver Input Voltage (A, B) ..... -8V to +12.5V
Receiver Output Voltage (RO) ..... -0.5V to (V_{CC} +0.5V)
Continuous Power Dissipation (T_A = +70°C)
8-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 727mW
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) 800mW
8-Pin SO (derate 5.88mW/°C above +70°C) 471mW
14-Pin SO (derate 8.33mW/°C above +70°C) 667mW
8-Pin µMAX (derate 4.1mW/°C above +70°C) 830mW
8-Pin CERDIP (derate 8.00mW/°C above +70°C) 640mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
Operating Temperature Ranges
MAX4_ _C_ _/MAX1487C_A ..... 0°C to +70°C
MAX4_ _E_ _/MAX1487E_A ..... -40°C to +85°C
MAX4_ _MJ_/MAX1487MJA ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Differential Driver Output (no load)	V_{OD1}					5	V
Differential Driver Output (with load)	V_{OD2}	R = 50Ω (RS-422)		2			V
		R = 27Ω (RS-485), Figure 4		1.5		5	
Change in Magnitude of Driver Differential Output Voltage for Complementary Output States	ΔV_{OD}	R = 27Ω or 50Ω, Figure 4				0.2	V
Driver Common-Mode Output Voltage	V_{OC}	R = 27Ω or 50Ω, Figure 4				3	V
Change in Magnitude of Driver Common-Mode Output Voltage for Complementary Output States	ΔV_{OD}	R = 27Ω or 50Ω, Figure 4				0.2	V
Input High Voltage	V_{IH}	DE, DI, $\overline{RE}$		2.0			V
Input Low Voltage	V_{IL}	DE, DI, $\overline{RE}$				0.8	V
Input Current	I_{IN1}	DE, DI, $\overline{RE}$				±2	µA
Input Current (A, B)	I_{IN2}	DE = 0V; V_{CC} = 0V or 5.25V, all devices except MAX487/MAX1487	V_{IN} = 12V			1.0	mA
			V_{IN} = -7V			-0.8	
		MAX487/MAX1487, DE = 0V, V_{CC} = 0V or 5.25V	V_{IN} = 12V			0.25	mA
			V_{IN} = -7V			-0.2	
Receiver Differential Threshold Voltage	V_{TH}	-7V ≤ V_{CM} ≤ 12V		-0.2		0.2	V
Receiver Input Hysteresis	ΔV_{TH}	V_{CM} = 0V			70		mV
Receiver Output High Voltage	V_{OH}	I_O = -4mA, V_{ID} = 200mV		3.5			V
Receiver Output Low Voltage	V_{OL}	I_O = 4mA, V_{ID} = -200mV				0.4	V
Three-State (high impedance) Output Current at Receiver	I_{OZR}	0.4V ≤ V_O ≤ 2.4V				±1	µA
Receiver Input Resistance	R_{IN}	-7V ≤ V_{CM} ≤ 12V, all devices except MAX487/MAX1487		12			kΩ
		-7V ≤ V_{CM} ≤ 12V, MAX487/MAX1487		48			kΩ

DC ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS			MIN	TYP	MAX	UNITS
No-Load Supply Current (Note 3)	I_{CC}	MAX488/MAX489, DE, DI, $\overline{RE}$ = 0V or V_{CC}				120	250	µA
		MAX490/MAX491, DE, DI, $\overline{RE}$ = 0V or V_{CC}				300	500	
		MAX481/MAX485, $\overline{RE}$ = 0V or V_{CC}	DE = V_{CC}			500	900	
			DE = 0V			300	500	
		MAX1487, $\overline{RE}$ = 0V or V_{CC}	DE = V_{CC}			300	500	
			DE = 0V			230	400	
		MAX483/MAX487, $\overline{RE}$ = 0V or V_{CC}	DE = 5V	MAX483		350	650	
				MAX487		250	400	
			DE = 0V			120	250	
Supply Current in Shutdown	I_{SHDN}	MAX481/483/487, DE = 0V, $\overline{RE}$ = V_{CC}				0.1	10	µA
Driver Short-Circuit Current, V_O = High	I_{OSD1}	-7V ≤ V_O ≤12V (Note 4)			35		250	mA
Driver Short-Circuit Current, V_O = Low	I_{OSD2}	-7V ≤ V_O ≤12V (Note 4)			35		250	mA
Receiver Short-Circuit Current	I_{OSR}	0V ≤ V_O ≤ V_{CC}			7		95	mA

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Driver Input to Output	t_{PLH}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		10	30	60	ns
	t_{PHL}			10	30	60	
Driver Output Skew to Output	t_{SKEW}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF			5	10	ns
Driver Rise or Fall Time	t_R, t_F	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	MAX481, MAX485, MAX1487	3	15	40	ns
			MAX490C/E, MAX491C/E	5	15	25	
			MAX490M, MAX491M	3	15	40	
Driver Enable to Output High	t_{ZH}	Figures 7 and 9, C_L = 100pF, S2 closed			40	70	ns
Driver Enable to Output Low	t_{ZL}	Figures 7 and 9, C_L = 100pF, S1 closed			40	70	ns
Driver Disable Time from Low	t_{LZ}	Figures 7 and 9, C_L = 15pF, S1 closed			40	70	ns
Driver Disable Time from High	t_{HZ}	Figures 7 and 9, C_L = 15pF, S2 closed			40	70	ns
Receiver Input to Output	t_{PLH}, t_{PHL}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	MAX481, MAX485, MAX1487	20	90	200	ns
			MAX490C/E, MAX491C/E	20	90	150	
			MAX490M, MAX491M	20	90	200	
$\lvert t_{PLH} - t_{PHL} \rvert$ Differential Receiver Skew	t_{SKD}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF			13		ns
Receiver Enable to Output Low	t_{ZL}	Figures 5 and 11, C_{RL} = 15pF, S1 closed			20	50	ns
Receiver Enable to Output High	t_{ZH}	Figures 5 and 11, C_{RL} = 15pF, S2 closed			20	50	ns
Receiver Disable Time from Low	t_{LZ}	Figures 5 and 11, C_{RL} = 15pF, S1 closed			20	50	ns
Receiver Disable Time from High	t_{HZ}	Figures 5 and 11, C_{RL} = 15pF, S2 closed			20	50	ns
Maximum Data Rate	f_{MAX}			2.5			Mbps
Time to Shutdown	t_{SHDN}	MAX481 (Note 5)		50	200	600	ns

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487 (continued)

(V_{CC} = 5V ±5% T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Enable from Shutdown to Output High (MAX481)	$t_{ZH(SHDN)}$	Figures 7 and 9, C_L = 100pF, S2 closed		40	100	ns
Driver Enable from Shutdown to Output Low (MAX481)	$t_{ZL(SHDN)}$	Figures 7 and 9, C_L = 100pF, S1 closed		40	100	ns
Receiver Enable from Shutdown to Output High (MAX481)	$t_{ZH(SHDN)}$	Figures 5 and 11, C_L = 15pF, S2 closed, A - B = 2V		300	1000	ns
Receiver Enable from Shutdown to Output Low (MAX481)	$t_{ZL(SHDN)}$	Figures 5 and 11, C_L = 15pF, S1 closed, B - A = 2V		300	1000	ns

SWITCHING CHARACTERISTICS—MAX483, MAX487/MAX488/MAX489

(V_{CC} = 5V ±5% T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Input to Output	t_{PLH}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250	900	2000	ns
	t_{PHL}		250	800	2000	
Driver Output Skew to Output	t_{SKEW}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		100	800	ns
Driver Rise or Fall Time	t_R, t_F	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250		2000	ns
Driver Enable to Output High	t_{ZH}	Figures 7 and 9, C_L = 100pF, S2 closed	250		2000	ns
Driver Enable to Output Low	t_{ZL}	Figures 7 and 9, C_L = 100pF, S1 closed	250		2000	ns
Driver Disable Time from Low	t_{LZ}	Figures 7 and 9, C_L = 15pF, S1 closed	300		3000	ns
Driver Disable Time from High	t_{HZ}	Figures 7 and 9, C_L = 15pF, S2 closed	300		3000	ns
Receiver Input to Output	t_{PLH}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250		2000	ns
	t_{PHL}		250		2000	
$\lvert t_{PLH} - t_{PHL} \rvert$ Differential Receiver Skew	t_{SKD}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		100		ns
Receiver Enable to Output Low	t_{ZL}	Figures 5 and 11, C_{RL} = 15pF, S1 closed		20	50	ns
Receiver Enable to Output High	t_{ZH}	Figures 5 and 11, C_{RL} = 15pF, S2 closed		20	50	ns
Receiver Disable Time from Low	t_{LZ}	Figures 5 and 11, C_{RL} = 15pF, S1 closed		20	50	ns
Receiver Disable Time from High	t_{HZ}	Figures 5 and 11, C_{RL} = 15pF, S2 closed		20	50	ns
Maximum Data Rate	f_{MAX}	t_{PLH}, t_{PHL} < 50% of data period	250			kbps
Time to Shutdown	t_{SHDN}	MAX483/MAX487 (Note 5)	50	200	600	ns
Driver Enable from Shutdown to Output High	$t_{ZH(SHDN)}$	MAX483/MAX487, Figures 7 and 9, C_L = 100pF, S2 closed			2000	ns
Driver Enable from Shutdown to Output Low	$t_{ZL(SHDN)}$	MAX483/MAX487, Figures 7 and 9, C_L = 100pF, S1 closed			2000	ns
Receiver Enable from Shutdown to Output High	$t_{ZH(SHDN)}$	MAX483/MAX487, Figures 5 and 11, C_L = 15pF, S2 closed			2500	ns
Receiver Enable from Shutdown to Output Low	$t_{ZL(SHDN)}$	MAX483/MAX487, Figures 5 and 11, C_L = 15pF, S1 closed			2500	ns

NOTES FOR ELECTRICAL/SWITCHING CHARACTERISTICS

Note 1: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 2: All typical specifications are given for V_{CC} = 5V and T_A = +25°C.

Note 3: Supply current specification is valid for loaded transmitters when DE = 0V.

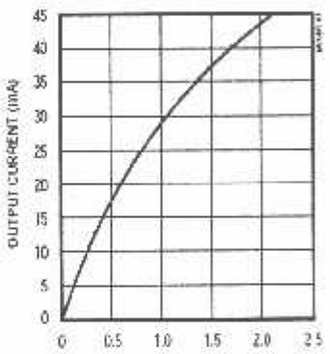
Note 4: Applies to peak current. See *Typical Operating Characteristics*.

Note 5: The MAX481/MAX483/MAX487 are put into shutdown by bringing RE high and DE low. If the inputs are in this state for less than 50ns, the parts are guaranteed not to enter shutdown. If the inputs are in this state for at least 600ns, the parts are guaranteed to have entered shutdown. See *Low-Power Shutdown Mode* section.

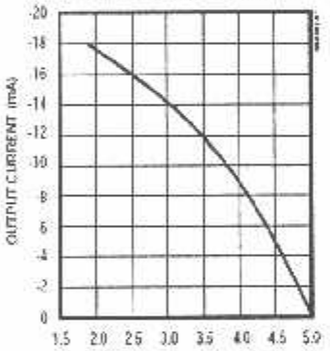
Typical Operating Characteristics

(V_{CC} = 5V, T_A = +25°C, unless otherwise noted.)

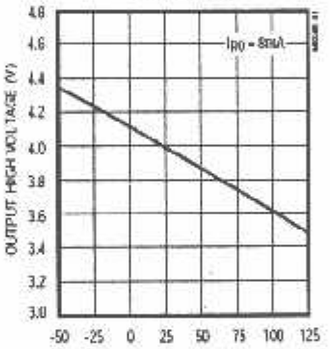
OUTPUT CURRENT vs. RECEIVER OUTPUT LOW VOLTAGE



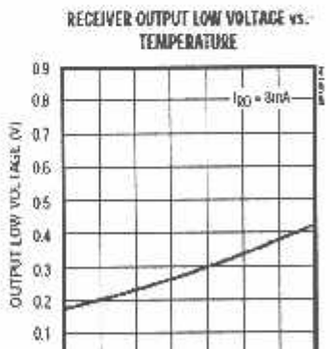
OUTPUT CURRENT vs. RECEIVER OUTPUT HIGH VOLTAGE



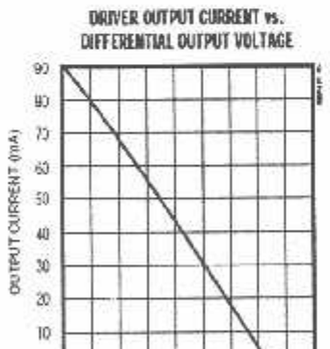
RECEIVER OUTPUT HIGH VOLTAGE vs. TEMPERATURE



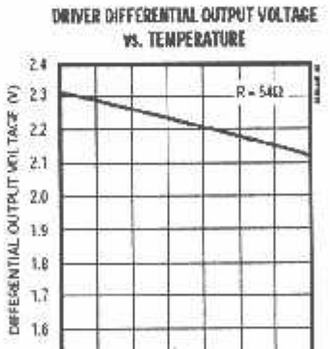
RECEIVER OUTPUT LOW VOLTAGE vs. TEMPERATURE



DRIVER OUTPUT CURRENT vs. DIFFERENTIAL OUTPUT VOLTAGE



DRIVER DIFFERENTIAL OUTPUT VOLTAGE vs. TEMPERATURE

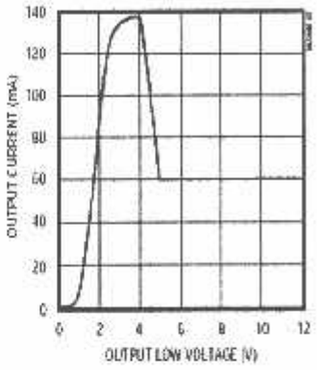


Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

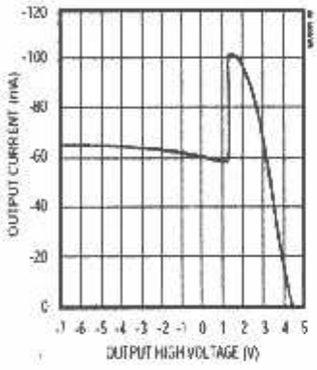
Typical Operating Characteristics (continued)

(VCC = 5V, TA = +25°C, unless otherwise noted.)

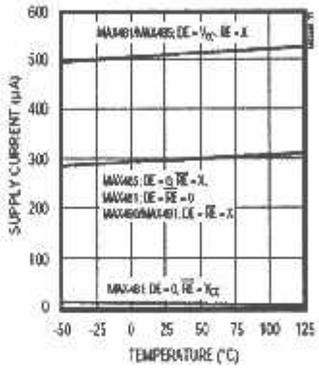
OUTPUT CURRENT vs. DRIVER OUTPUT LOW VOLTAGE



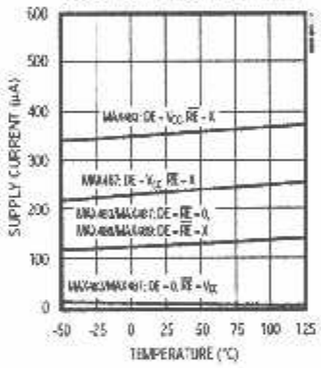
OUTPUT CURRENT vs. DRIVER OUTPUT HIGH VOLTAGE



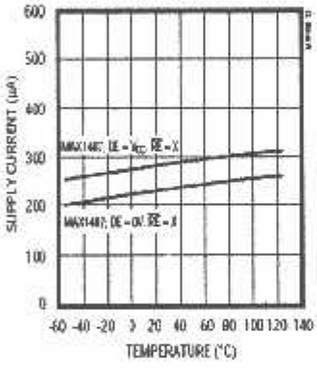
MAX481/MAX485/MAX490/MAX491 SUPPLY CURRENT vs. TEMPERATURE



MAX483/MAX487–MAX489 SUPPLY CURRENT vs. TEMPERATURE



MAX1487 SUPPLY CURRENT vs. TEMPERATURE



Pin Description

PIN					NAME	FUNCTION
MAX481/MAX483/ MAX485/MAX487/ MAX1487		MAX488/ MAX490		MAX489/ MAX491		
DIP/SO	µMAX	DIP/SO	µMAX	DIP/SO		
1	3	2	4	2	RO	Receiver Output. If A > B by 200mV, RO will be high; if A < B by 200mV, RO will be low.
2	4	—	—	3	$\overline{RE}$	Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when $\overline{RE}$ is high.
3	5	—	—	4	DE	Driver Output Enable. The driver outputs, Y and Z, are enabled by bringing DE high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low.
4	6	3	5	5	DI	Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low.
5	7	4	6	6, 7	GND	Ground
—	—	5	7	9	Y	Noninverting Driver Output
—	—	6	8	10	Z	Inverting Driver Output
6	8	—	—	—	A	Noninverting Receiver Input and Noninverting Driver Output
—	—	8	2	12	A	Noninverting Receiver Input
7	1	—	—	—	B	Inverting Receiver Input and Inverting Driver Output
—	—	7	1	11	B	Inverting Receiver Input
8	2	1	3	14	VCC	Positive Supply: 4.75V ≤ VCC ≤ 5.25V
—	—	—	—	1, 8, 13	N.C.	No Connect—not internally connected

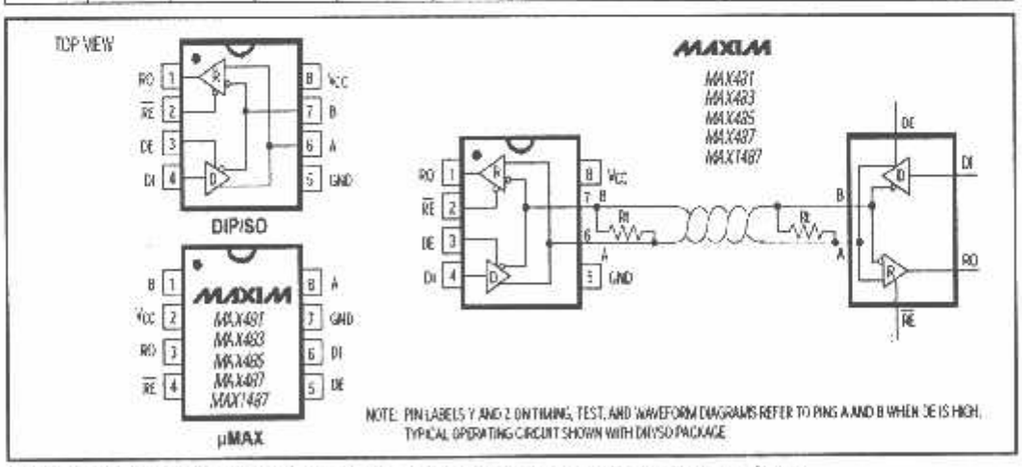


Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

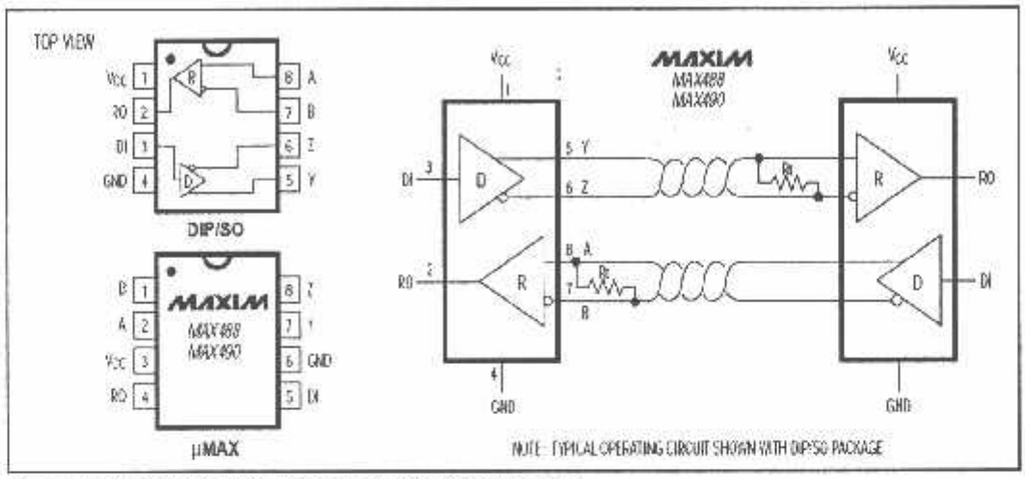


Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

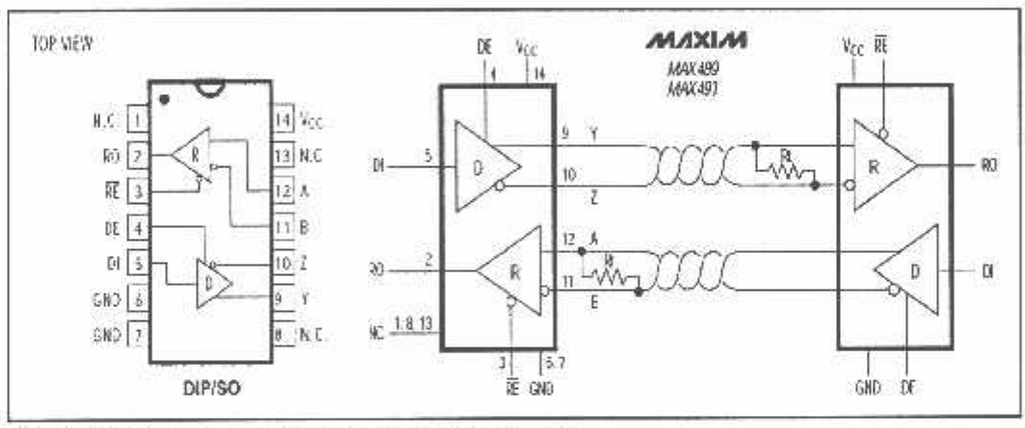


Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

Applications Information

The MAX481/MAX483/MAX485/MAX487–MAX491 and MAX1487 are low-power transceivers for RS-485 and RS-422 communications. The MAX481, MAX485, MAX490, MAX491, and MAX1487 can transmit and receive at data rates up to 2.5Mbps, while the MAX483, MAX487, MAX488, and MAX489 are specified for data rates up to 250kbps. The MAX488–MAX491 are full-duplex transceivers while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are half-duplex. In addition, Driver Enable (DE) and Receiver Enable ($\overline{RE}$) pins are included on the MAX481, MAX483, MAX485, MAX487, MAX489, MAX491, and MAX1487. When disabled, the driver and receiver outputs are high impedance.

MAX487/MAX1487: 128 Transceivers on the Bus

The 48kΩ, ¼-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12kΩ input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488–MAX491 have standard 12kΩ Receiver Input impedance.

Test Circuits

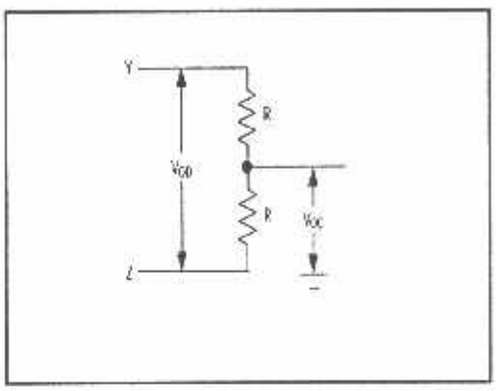


Figure 4. Driver DC Test Load

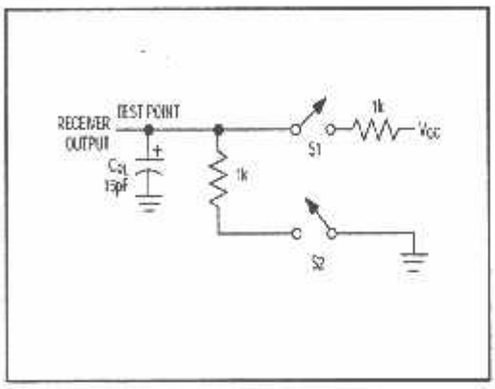


Figure 5. Receiver Timing Test Load

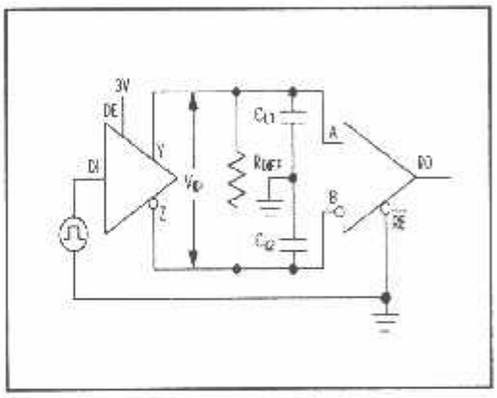


Figure 6. Driver/Receiver Timing Test Circuit

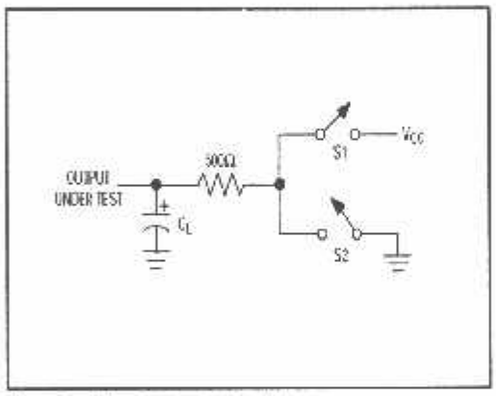


Figure 7. Driver Timing Test Load

MAX483/MAX487/MAX488/MAX489: Reduced EMI and Reflections

The MAX483 and MAX487–MAX489 are slew-rate limited, minimizing EMI and reducing reflections caused by improperly terminated cables. Figure 12 shows the driver output waveform and its Fourier analysis of a 150kHz signal transmitted by a MAX481, MAX485, MAX490, MAX491, or MAX1487. High-frequency harmonics with large amplitudes are evident. Figure 13 shows the same information displayed for a MAX483, MAX487, MAX488, or MAX489 transmitting under the same conditions. Figure 13's high-frequency harmonics have much lower amplitudes, and the potential for EMI is significantly reduced.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Switching Waveforms

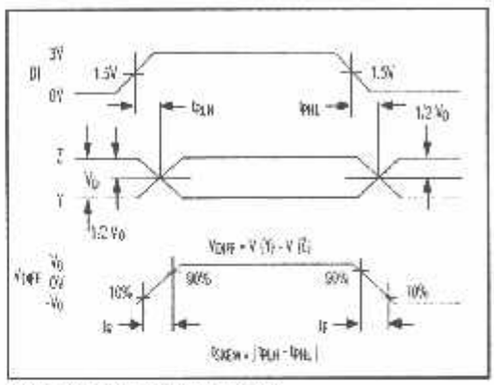


Figure 8. Driver Propagation Delays

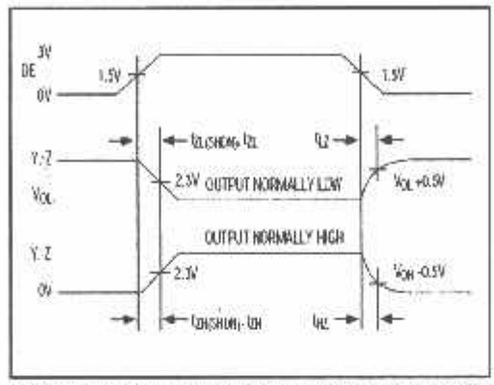


Figure 9. Driver Enable and Disable Times (except MAX488 and MAX490)

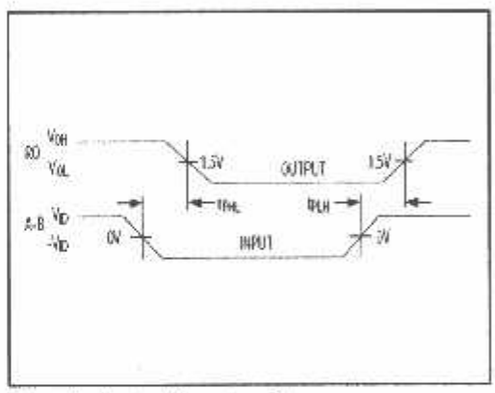


Figure 10. Receiver Propagation Delays

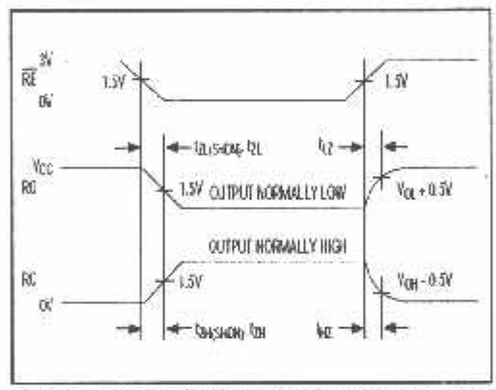


Figure 11. Receiver Enable and Disable Times (except MAX488 and MAX490)

Function Tables (MAX481/MAX483/MAX485/MAX487/MAX1487)

Table 1. Transmitting

INPUTS			OUTPUTS	
$\overline{RE}$	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	High-Z	High-Z
1	0	X	High-Z*	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

Table 2. Receiving

INPUTS			OUTPUT
$\overline{RE}$	DE	A-B	RO
0	0	≥ +0.2V	1
0	0	≤ -0.2V	0
0	0	Inputs open	1
1	0	X	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

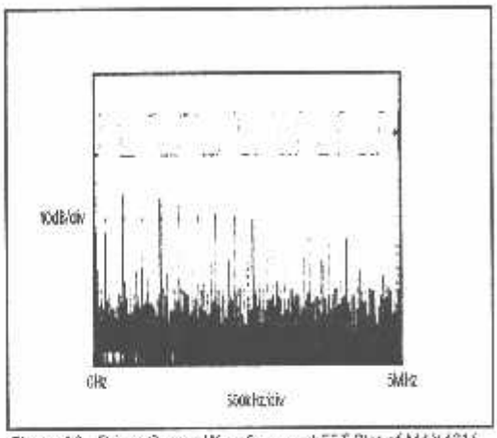


Figure 12. Driver Output Waveform and FFT Plot of MAX481/MAX485/MAX490/MAX491/MAX1487 Transmitting a 150kHz Signal

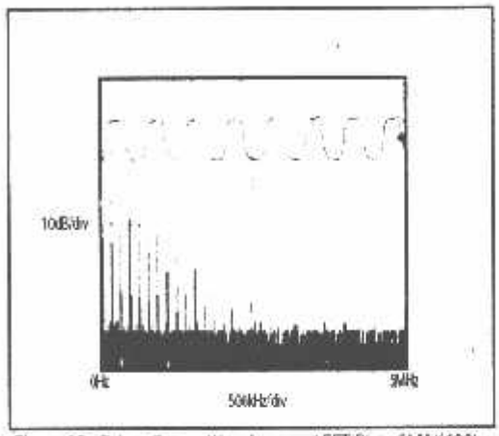


Figure 13. Driver Output Waveform and FFT Plot of MAX483/MAX487–MAX489 Transmitting a 150kHz Signal

Low-Power Shutdown Mode (MAX481/MAX483/MAX487)

A low-power shutdown mode is initiated by bringing both $\overline{RE}$ high and DE low. The devices will not shut down unless both the driver and receiver are disabled. In shutdown, the devices typically draw only 0.1µA of supply current.

$\overline{RE}$ and DE may be driven simultaneously; the parts are guaranteed not to enter shutdown if $\overline{RE}$ is high and DE is low for less than 50ns. If the inputs are in this state for at least 600ns, the parts are guaranteed to enter shutdown.

For the MAX481, MAX483, and MAX487, the t_{ZH} and t_{ZL} enable times assume the part was not in the low-power shutdown state (the MAX485/MAX488–MAX491 and MAX1487 can not be shut down). The $t_{ZH(SHDN)}$ and $t_{ZL(SHDN)}$ enable times assume the parts were shut down (see *Electrical Characteristics*).

It takes the drivers and receivers longer to become enabled from the low-power shutdown state ($t_{ZH(SHDN)}$, $t_{ZL(SHDN)}$) than from the operating mode (t_{ZH}, t_{ZL}). (The parts are in operating mode if the $\overline{RE}$, DE inputs equal a logical 0, 1 or 1, 1 or 0, 0.)

Driver Output Protection

Excessive output current and power dissipation caused by faults or by bus contention are prevented by two mechanisms. A foldback current limit on the output stage provides immediate protection against short circuits over the whole common-mode voltage range (see *Typical Operating Characteristics*). In addition, a thermal shutdown circuit forces the driver outputs into a high-impedance state if the die temperature rises excessively.

Propagation Delay

Many digital encoding schemes depend on the difference between the driver and receiver propagation delay times. Typical propagation delays are shown in Figures 15–18 using Figure 14's test circuit.

The difference in receiver delay times, $|t_{PLH} - t_{PHL}|$, is typically under 13ns for the MAX481, MAX485, MAX490, MAX491, and MAX1487 and is typically less than 100ns for the MAX483 and MAX487–MAX489.

The driver skew times are typically 5ns (10ns max) for the MAX481, MAX485, MAX490, MAX491, and MAX1487, and are typically 100ns (800ns max) for the MAX483 and MAX487–MAX489.

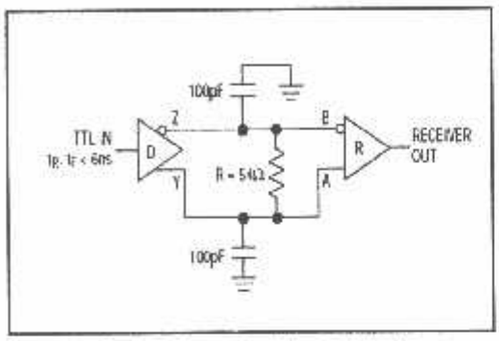


Figure 14. Receiver Propagation Delay Test Circuit

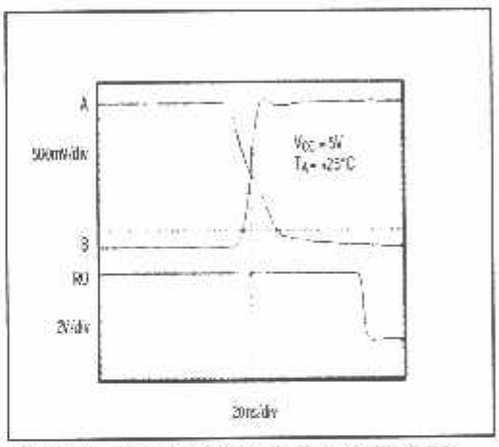


Figure 15. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PHL}

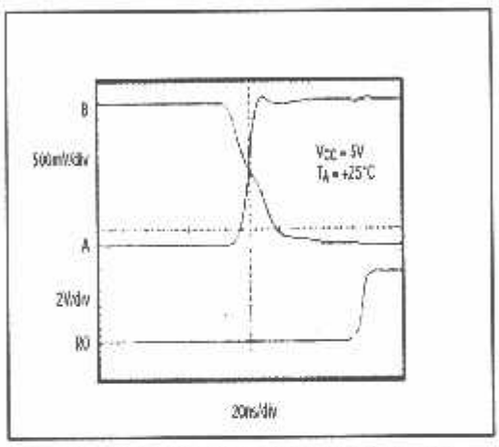


Figure 16. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PLH}

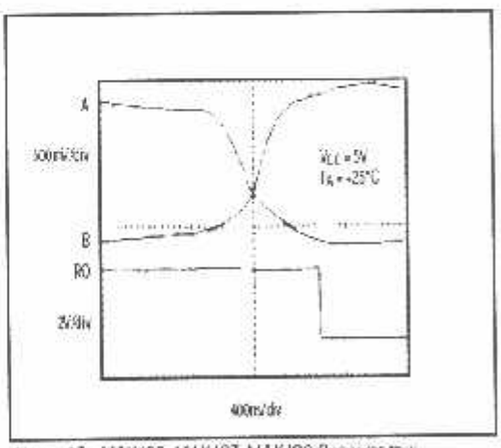


Figure 17. MAX483, MAX487–MAX489 Receiver t_{PHL}

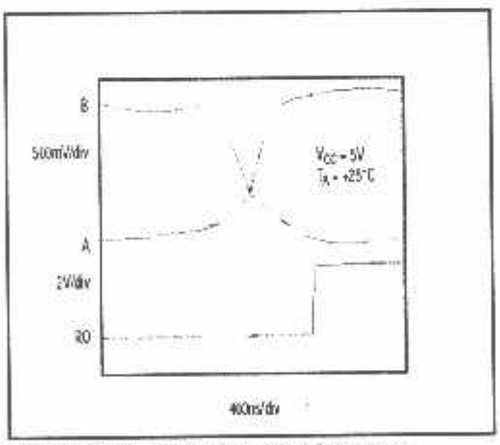


Figure 18. MAX483, MAX487–MAX489 Receiver t_{PLH}

Line Length vs. Data Rate

The RS-485/RS-422 standard covers line lengths up to 4000 feet. For line lengths greater than 4000 feet, see Figure 23.

Figures 19 and 20 show the system differential voltage for the parts driving 4000 feet of 26AWG twisted-pair wire at 110kHz into 120Ω loads.

Typical Applications

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 transceivers are designed for bidirectional data communications on multipoint bus transmission lines.

Figures 21 and 22 show typical network applications circuits. These parts can also be used as line repeaters, with cable lengths longer than 4000 feet, as shown in Figure 23.

To minimize reflections, the line should be terminated at both ends in its characteristic impedance, and stub lengths off the main line should be kept as short as possible. The slew-rate-limited MAX483 and MAX487–MAX489 are more tolerant of imperfect termination.

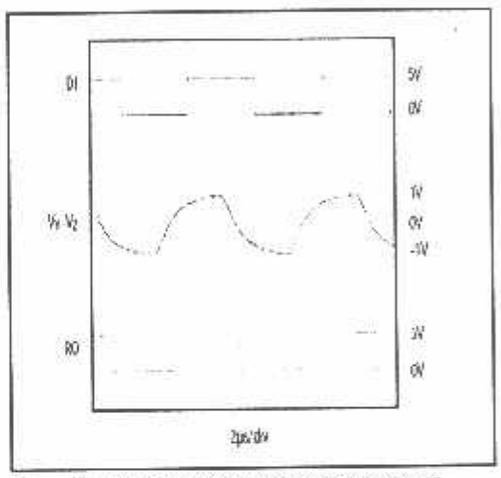


Figure 19. MAX481/MAX485/MAX490/MAX491/MAX1487 System Differential Voltage at 110kHz Driving 4000ft of Cable

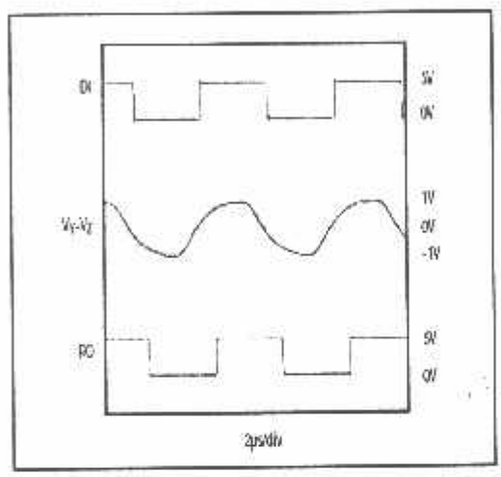


Figure 20. MAX483, MAX487–MAX489 System Differential Voltage at 110kHz Driving 4000ft of Cable

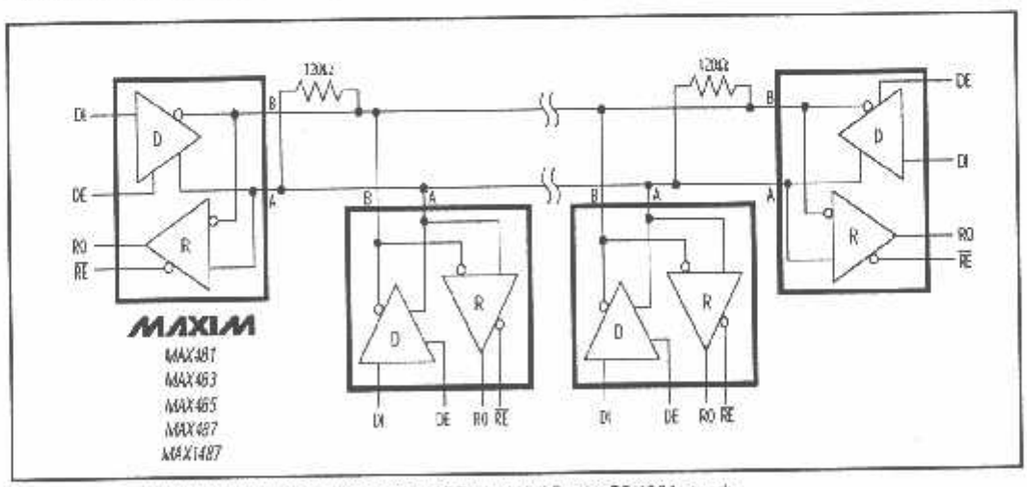


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

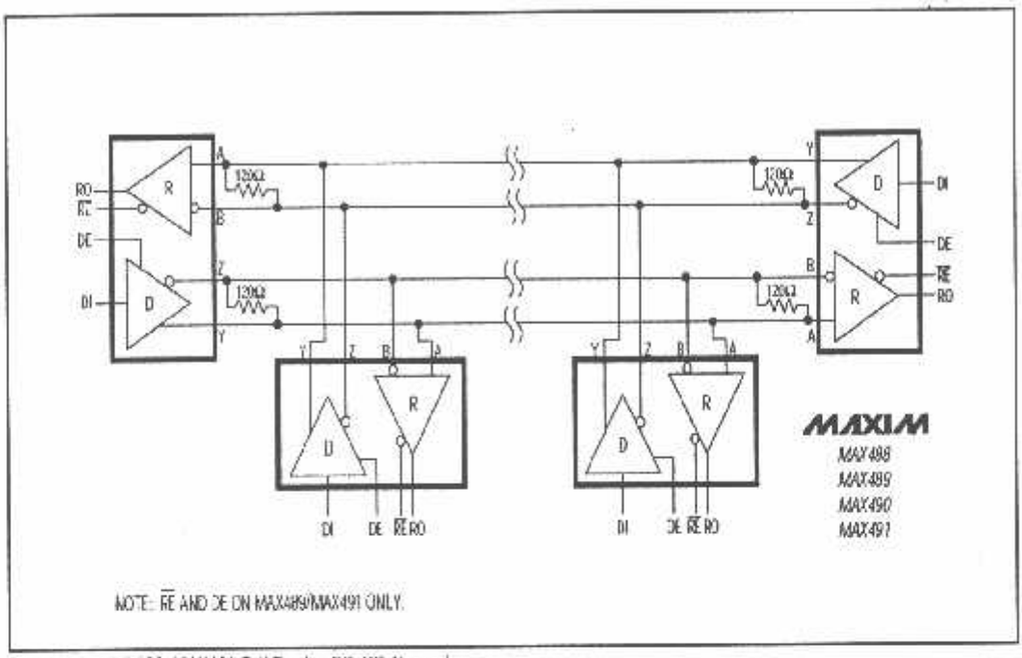


Figure 22. MAX488–MAX491 Full-Duplex RS-485 Network

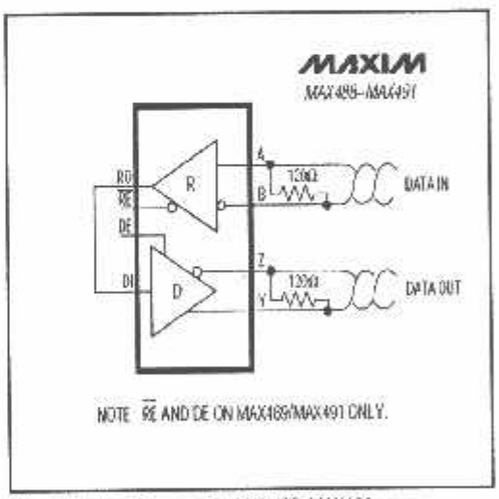


Figure 23. Line Repeater for MAX488–MAX491

Isolated RS-485

For isolated RS-485 applications, see the MAX253 and MAX1480 data sheets.

Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX481EPA	-40°C to +85°C	8 Plastic DIP
MAX481ESA	-40°C to +85°C	8 SO
MAX481MJA	-55°C to +125°C	8 CERDIP
MAX483CPA	0°C to +70°C	8 Plastic DIP
MAX483CSA	0°C to +70°C	8 SO
MAX483CUA	0°C to +70°C	8 µMAX
MAX483C/D	0°C to +70°C	Dice*
MAX483EPA	-40°C to +85°C	8 Plastic DIP
MAX483ESA	-40°C to +85°C	8 SO
MAX483MJA	-55°C to +125°C	8 CERDIP
MAX485CPA	0°C to +70°C	8 Plastic DIP
MAX485CSA	0°C to +70°C	8 SO
MAX485CUA	0°C to +70°C	8 µMAX
MAX485C/D	0°C to +70°C	Dice*
MAX485EPA	-40°C to +85°C	8 Plastic DIP
MAX485ESA	-40°C to +85°C	8 SO
MAX485MJA	-55°C to +125°C	8 CERDIP
MAX487CPA	0°C to +70°C	8 Plastic DIP
MAX487CSA	0°C to +70°C	8 SO
MAX487CUA	0°C to +70°C	8 µMAX
MAX487C/D	0°C to +70°C	Dice*
MAX487EPA	-40°C to +85°C	8 Plastic DIP
MAX487ESA	-40°C to +85°C	8 SO
MAX487MJA	-55°C to +125°C	8 CERDIP
MAX488CPA	0°C to +70°C	8 Plastic DIP
MAX488CSA	0°C to +70°C	8 SO
MAX488CUA	0°C to +70°C	8 µMAX
MAX488C/D	0°C to +70°C	Dice*
MAX488EPA	-40°C to +85°C	8 Plastic DIP
MAX488ESA	-40°C to +85°C	8 SO
MAX488MJA	-55°C to +125°C	8 CERDIP
MAX489CPD	0°C to +70°C	14 Plastic DIP
MAX489CSD	0°C to +70°C	14 SO
MAX489C/D	0°C to +70°C	Dice*
MAX489EPD	-40°C to +85°C	14 Plastic DIP
MAX489ESD	-40°C to +85°C	14 SO
MAX489MJD	-55°C to +125°C	14 CERDIP

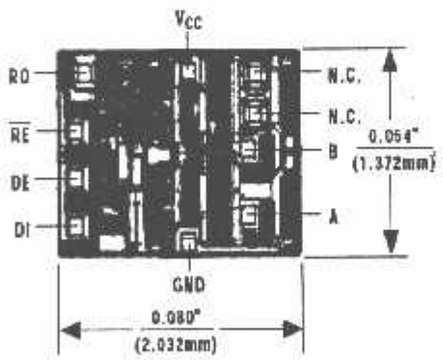
Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX490CPA	0°C to +70°C	8 Plastic DIP
MAX490CSA	0°C to +70°C	8 SO
MAX490CUA	0°C to +70°C	8 µMAX
MAX490C/D	0°C to +70°C	Dice*
MAX490EPA	-40°C to +85°C	8 Plastic DIP
MAX490ESA	-40°C to +85°C	8 SO
MAX490MJA	-55°C to +125°C	8 CERDIP
MAX491CPD	0°C to +70°C	14 Plastic DIP
MAX491CSD	0°C to +70°C	14 SO
MAX491C/D	0°C to +70°C	Dice*
MAX491EPD	-40°C to +85°C	14 Plastic DIP
MAX491ESD	-40°C to +85°C	14 SO
MAX491MJD	-55°C to +125°C	14 CERDIP
MAX1487CPA	0°C to +70°C	8 Plastic DIP
MAX1487CSA	0°C to +70°C	8 SO
MAX1487CUA	0°C to +70°C	8 µMAX
MAX1487C/D	0°C to +70°C	Dice*
MAX1487EPA	-40°C to +85°C	8 Plastic DIP
MAX1487ESA	-40°C to +85°C	8 SO
MAX1487MJA	-55°C to +125°C	8 CERDIP

* *Contact factory for dice specifications.*

Chip Topographies

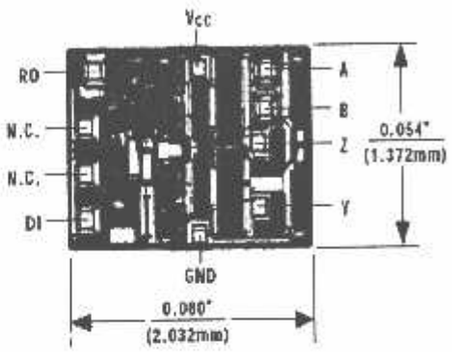
MAX481/MAX483/MAX485/MAX487/MAX1487



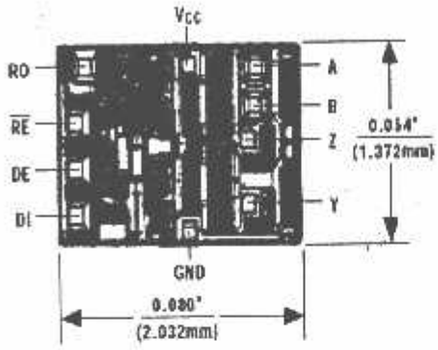
Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Chip Topographies (continued)

MAX488/MAX490



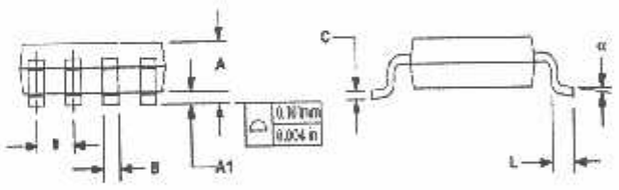
MAX489/MAX491

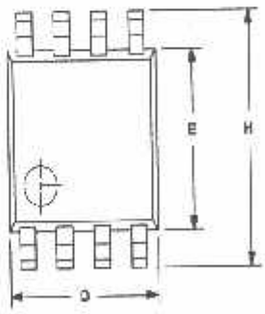


TRANSISTOR COUNT: 248

SUBSTRATE CONNECTED TO GND

Package Information





DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.036	0.044	0.91	1.11
A1	0.004	0.008	0.10	0.20
B	0.010	0.014	0.25	0.36
C	0.005	0.007	0.13	0.18
D	0.116	0.120	2.95	3.05
E	0.116	0.120	2.95	3.05
e	0.0256		0.65	
H	0.188	0.198	4.78	5.03
L	0.016	0.026	0.41	0.66
α	0	6	0	6

21-0036D

8-PIN µMAX
MICROMAX SMALL-OUTLINE
PACKAGE

19-4323; Rev 7b; 11/97

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multi-Drop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	4.7/10	No	—	120	Ultra low power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/-12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/0	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to -13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/-12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±30V
T_{OUT} (Note 1) ..... ±15V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) ..842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ..889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ..696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) ..762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) ..762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) ..800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) ..640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) ..800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) ..842mW
Operating Temperature Ranges
MAX2__AC__ MAX2__C__ ..... 0°C to +70°C
MAX2__AE__ MAX2__E__ ..... -40°C to +85°C
MAX2__AM__ MAX2__M__ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±9		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate	All except MAX220, normal operation			200	116	kbits/sec
	MAX220			22	20	
Transmitter Output Resistance	V_{CC} = V_+ = V_- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V			±7	±22	mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.6	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Sinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX221), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TTL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TTL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TTL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 2: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

HARRIS SEMICONDUCTOR

ICL232

+5V Powered Dual RS-232 Transmitter/Receiver

December 1993

Features

- Meets All RS-232C Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/µs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges

Applications

- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
  - Dataloggers

Description

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

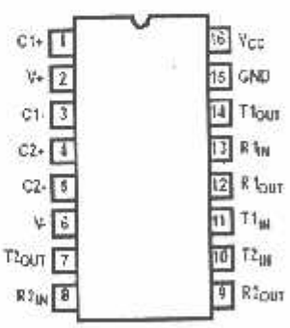
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to ±30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

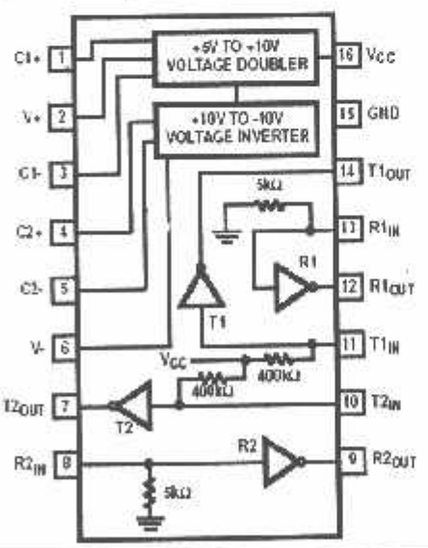
PART NUMBER	TEMPERATURE RANGE	PACKAGE
ICL232CPE	0°C to +70°C	16 Lead Plastic DIP
ICL232CJE	0°C to +70°C	16 Lead Ceramic DIP
ICL232CBE	0°C to +70°C	16 Lead SOIC (W)
ICL232IPE	-40°C to +85°C	16 Lead Plastic DIP
ICL232IJE	-40°C to +85°C	16 Lead Ceramic DIP
ICL232IBE	-40°C to +85°C	16 Lead SOIC (W)
ICL232MJE	-55°C to +125°C	16 Lead Ceramic DIP

Pinouts

ICL232
(PDIP, CDIP, SOIC)
TOP VIEW



Functional Diagram



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper I.C. Handling Procedures.

File Number 3020.2

Absolute Maximum Ratings

V_{CC} to Ground ..... (GND -0.3V) <V_{CC} < 6V
V+ to Ground ..... (V_{CC} -0.3V) <V+ < 12V
V- to Ground ..... -12V < V- < (GND +0.3V)
Input Voltages
$T1_{IN}$, $T2_{IN}$ ..... (V- -0.3V) < V_{TXIN} < (V+ +0.3V)
$R1_{IN}$, $R2_{IN}$ ..... ±30V
Output Voltages
$T1_{OUT}$, $T2_{OUT}$ ..... (V- -0.3V) < V_{TXOUT} < (V+ +0.3V)
$R1_{OUT}$, $R2_{OUT}$ ..... (GND -0.3V) < V_{RXOUT} < (V_{CC} +0.3V)
Short Circuit Duration
$T1_{OUT}$, $T2_{OUT}$ ..... Continuous
$R1_{OUT}$, $R2_{OUT}$ ..... Continuous
Storage Temperature Range ..... -65°C to +150°C
Lead Temperature (Soldering 10s) ..... +300°C

Thermal Information

Thermal Resistance	θ_{JA}	θ_{JC}
Ceramic DIP Package	80°C/W	24°C/W
Plastic DIP Package	100°C/W	-
SOIC Package	100°C/W	-

Maximum Power Dissipation 250mW
Operating Temperature Range
ICL232C ..... 0°C to +70°C
ICL232I ..... -40°C to +85°C
ICL232M ..... -55°C to +125°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Specifications

Test Conditions: V_{CC} = +5V ±10%, T_A = Operating Temperature Range, Test Circuit as in Figure 8 Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS
Transmitter Output Voltage Swing, T_{OUT}	$T1_{OUT}$ and $T2_{OUT}$ loaded with 3kΩ to Ground	±5	±9	±10	V
Power Supply Current, I_{CC}	Outputs Unloaded, T_A = +25°C	-	5	10	mA
T_{IN} Input Logic Low, V_{IL}		-	-	0.8	V
T_{IN} Input Logic High, V_{IH}		2.0	-	-	V
Logic Pullup Current, I_P	$T1_{IN}$, $T2_{IN}$ = 0V	-	15	200	µA
RS-232 Input Voltage Range, V_{IN}		-30	-	+30	V
Receiver Input Impedance, R_{IN}	V_{IN} = ±3V	3.0	5.0	7.0	kΩ
Receiver Input Low Threshold, V_{IN} (H-L)	V_{CC} = 5.0V, T_A = +25°C	0.8	1.2	-	V
Receiver Input High Threshold, V_{IN} (L-H)	V_{CC} = 5.0V, T_A = +25°C	-	1.7	2.4	V
Receiver Input Hysteresis, V_{HYST}		0.2	0.5	1.0	V
TTL/CMOS Receiver Output Voltage Low, V_{OL}	I_{OUT} = 3.2mA	-	0.1	0.4	V
TTL/CMOS Receiver Output Voltage High, V_{OH}	I_{OUT} = -1.0mA	3.5	4.6	-	V
Propagation Delay, t_{PD}	RS-232 to TTL	-	0.5	-	µs
Instantaneous Slew Rate, SR	C_L = 10pF, R_L = 3kΩ, T_A = +25°C (Notes 1, 2)	-	-	30	V/µs
Transition Region Slew Rate, SR_T	R_L = 3kΩ, C_L = 2500pF Measured from +3V to -3V or -3V to +3V	-	3	-	V/µs
Output Resistance, R_{OUT}	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V	300	-	-	Ω
RS-232 Output Short Circuit Current, I_{SC}	$T1_{OUT}$ or $T2_{OUT}$ shorted to GND	-	±10	-	mA

NOTES:

1. Guaranteed by design.
2. See Figure 4 for definition.

Typical Performance Curves

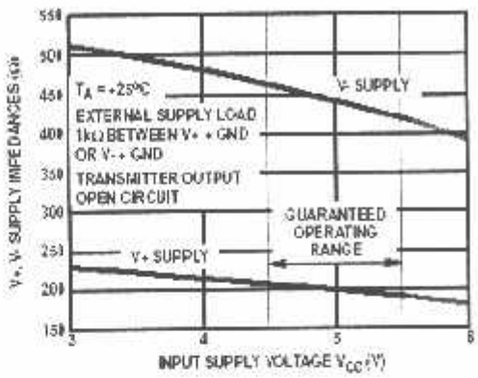


FIGURE 1. V+, V- OUTPUT IMPEDANCES vs V_{CC}

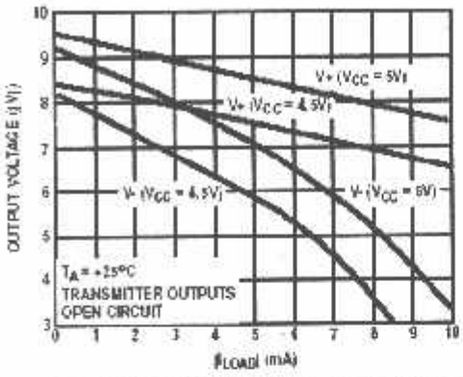


FIGURE 2. V+, V- OUTPUT VOLTAGES vs LOAD CURRENT

Pin Descriptions

PLASTIC DIP, CERAMIC DIP	SOIC	PIN NAME	DESCRIPTION
1	1	C1+	External capacitor "+" for internal voltage doubler.
2	2	V+	Internally generated +10V (typical) supply.
3	3	C1-	External capacitor "-" for internal voltage doubler.
4	4	C2+	External capacitor "+" internal voltage inverter.
5	5	C2-	External capacitor "-" internal voltage inverter.
6	6	V-	Internally generated -10V (typical) supply.
7	7	$T2_{OUT}$	RS-232 Transmitter 2 output ±10V (typical).
8	8	$R2_{IN}$	RS-232 Receiver 2 input, with internal 5K pulldown resistor to GND.
9	9	R2out	Receiver 2 TTL/CMOS output.
10	10	$T2_{IN}$	Transmitter 2 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
11	11	$T1_{IN}$	Transmitter 1 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
12	12	$R1_{OUT}$	Receiver 1 TTL/CMOS output.
13	13	$R1_{IN}$	RS-232 Receiver 1 input, with internal 5K pulldown resistor to GND.
14	14	$T1_{OUT}$	RS-232 Transmitter 1 output ±10V (typical).
15	15	GND	Supply Ground.
16	16	VCC	Positive Power Supply +5V ±10%

Detailed Description

The ICL232 is a dual RS-232 transmitter/receiver powered by a single +5V power supply which meets all EIA RS232C specifications and features low power consumption. The functional diagram illustrates the major elements of the ICL232. The circuit is divided into three sections: a voltage doubler/inverter, dual transmitters, and dual receivers.

Voltage Converter

An equivalent circuit of the dual charge pump is illustrated in Figure 3.

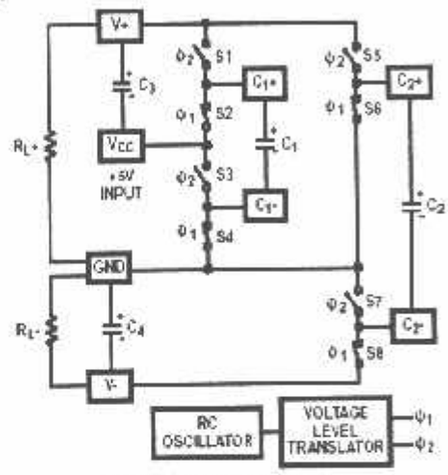


FIGURE 3. DUAL CHARGE PUMP

The voltage quadrupler contains two charge pumps which use two phases of an internally generated clock to generate +10V and -10V. The nominal clock frequency is 16kHz. During phase one of the clock, capacitor C1 is charged to V_{CC}. During phase two, the voltage on C1 is added to V_{CC}, producing a signal across C2 equal to twice V_{CC}. At the same time, C3 is also charged to $2V_{CC}$, and then during phase one, it is inverted with respect to ground to produce a signal across C4 equal to $-2V_{CC}$. The voltage converter accepts input voltages up to 5.5V. The output impedance of the doubler (V+) is approximately 200Ω, and the output impedance of the inverter (V-) is approximately 450Ω. Typical graphs are presented which show the voltage converters output vs input voltage and output voltages vs load characteristics. The test circuit (Figure 8) uses 1µF capacitors for C1-C4, however, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, and increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.

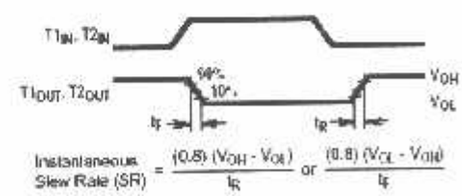


FIGURE 4. SLEW RATE DEFINITION

Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of V_{CC}, or 1.3V for V_{CC} = 5V. A logic 1 at the input results in a voltage of between -5V and V- at the output, and a logic 0 results in a voltage between +5V and (V+ -0.6V). Each transmitter input has an internal 400kΩ pullup resistor so any unused input can be left unconnected and its output remains in its low state. The output voltage swing meets the RS-232C specification of ±5V minimum with the worst case conditions of: both transmitters driving 3kΩ minimum load impedance, V_{CC} = 4.5V, and maximum allowable operating temperature. The transmitters have an internally limited output slew rate which is less than 30V/µs. The outputs are short circuit protected and can be shorted to ground indefinitely. The powered down output impedance is a minimum of 300Ω with ±2V applied to the outputs and V_{CC} = 0V.

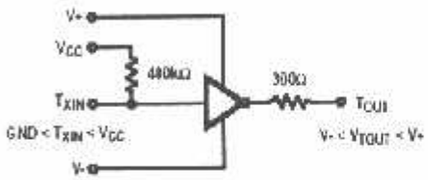


FIGURE 5. TRANSMITTER

Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even if the power is off (V_{CC} = 0V). The receivers have a typical input threshold of 1.3V which is within the +3V limits, known as the transition region, of the RS-232 specification. The receiver output is 0V to V_{CC}. The output will be low whenever the input is greater than 2.4V and high whenever the input is floating or driven between +0.8V and -30V. The receivers feature 0.5V hysteresis to improve noise rejection.

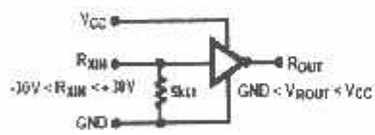


FIGURE 6. RECEIVER

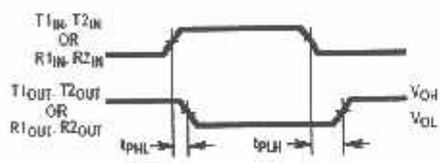


FIGURE 7. PROPAGATION DELAY DEFINITION

Test Circuits

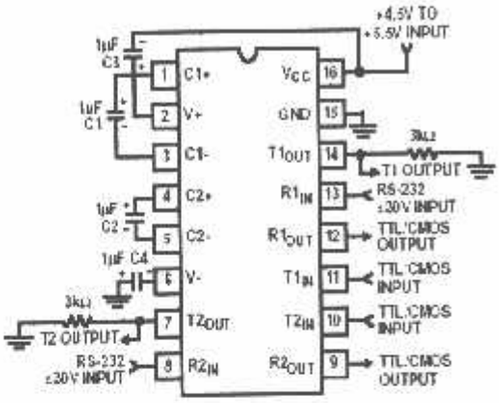


FIGURE 9. GENERAL TEST CIRCUIT

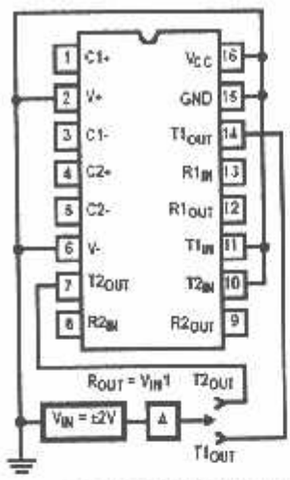


FIGURE 9. POWER-OFF SOURCE RESISTANCE CONFIGURATION

Applications

The ICL232 may be used for all RS-232 data terminal and communication links. It is particularly useful in applications where ±12V power supplies are not available for conventional RS-232 interface circuits. The applications presented represent typical interface configurations.

A simple duplex RS-232 port with CTS/RTS handshaking is illustrated in Figure 10. Fixed output signals such as DTR (data terminal ready) and DSRS (data signaling rate select) is generated by driving them through a 5kΩ resistor connected to V+.

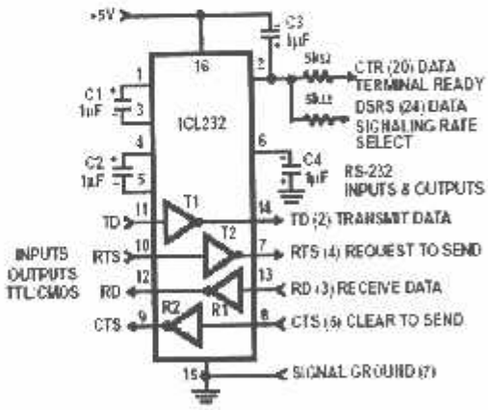


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

In applications requiring four RS-232 inputs and outputs (Figure 11), note that each circuit requires two charge pump capacitors (C1 and C2) but can share common reservoir capacitors (C3 and C4). The benefit of sharing common reservoir capacitors is the elimination of two capacitors and the reduction of the charge pump source impedance which effectively increases the output swing of the transmitters.

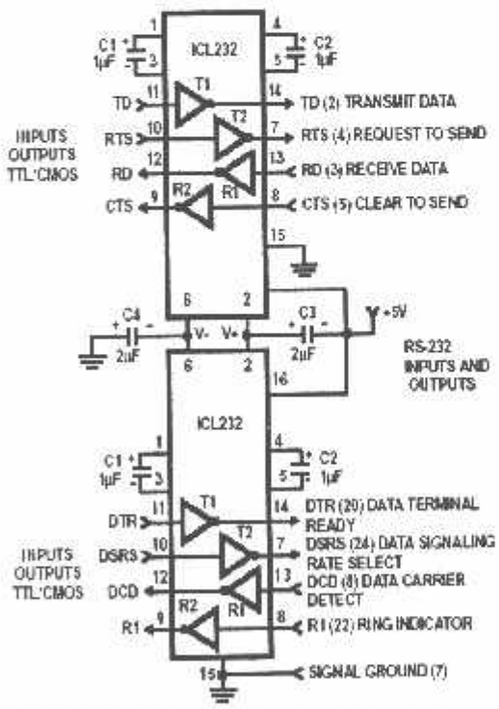


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS

DALLAS SEMICONDUCTOR

Application Note 83
Fundamentals of RS–232 Serial Communications

Due to it's relative simplicity and low hardware overhead (as compared to parallel interfacing), serial communications is used extensively within the electronics industry. Today, the most popular serial communications standard in use is certainly the EIA/TIA–232–E specification. This standard, which has been developed by the Electronic Industry Association and the Telecommunications Industry Association (EIA/TIA), is more popularly referred to simply as "RS–232" where "RS" stands for "recommended standard". In recent years, this suffix has been replaced with "EIA/TIA" to help identify the source of the standard. This paper will use the common notation of "RS–232" in its discussion of the topic.

The official name of the EIA/TIA–232–E standard is "Interface Between Data Terminal Equipment and Data Circuit–Termination Equipment Employing Serial Binary Data Interchange". Although the name may sound intimidating, the standard is simply concerned with serial data communication between a host system (Data Terminal Equipment, or "DTE") and a peripheral system (Data Circuit–Terminating Equipment, or "DCE").

The EIA/TIA–232–E standard which was introduced in 1962 has been updated four times since its introduction in order to better meet the needs of serial communication applications. The letter "E" in the standard's name indicates that this is the fifth revision of the standard.

RS–232 SPECIFICATIONS

RS–232 is a "complete" standard. This means that the standard sets out to ensure compatibility between the host and peripheral systems by specifying 1) common voltage and signal levels, 2)common pin wiring configurations, and 3) a minimal amount of control information between the host and peripheral systems. Unlike many standards which simply specify the electrical characteristics of a given interface, RS–232 specifies electrical, functional, and mechanical characteristics in order to meet the above three criteria. Each of these aspects of the RS–232 standard is discussed below.

ELECTRICAL CHARACTERISTICS

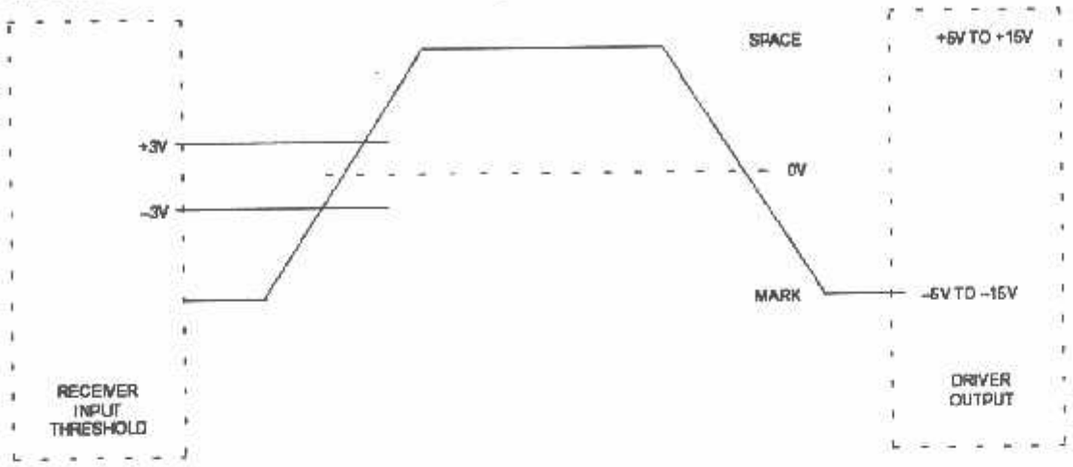
The electrical characteristics section of the RS–232 standard includes specifications on voltage levels, rate of change of signal levels, and line impedance.

The original RS–232 standard was defined in 1962. As this was before the days of TTL logic, it should not be surprising that the standard does not use 5 volt and ground logic levels. Instead, a high level for the driver output is defined as being +5 to +15 volts and a low level for the driver output is defined as being between –5 and –15 volts. The receiver logic levels were defined to provide a 2 volt noise margin. As such, a high level for the receiver is defined as +3 to +15 volts and a low level is –3 to –15 volts. Figure 1 illustrates the logic levels defined by the RS–232 standard. It is necessary to note that, for RS–232 communication, a low level (–3 to –15 volts) is defined as a logic 1 and is historically referred to as "marking". Likewise a high level (+3 to +15 volts) is defined as a logic 0 and is referred to as "spacing".

The RS–232 standard also limits the maximum slew rate at the driver output. This limitation was included to help reduce the likelihood of cross–talk between adjacent signals. The slower the rise and fall time, the smaller the chance of cross talk. With this in mind, the maximum slew rate allowed is 30 V/µs. Additionally, a maximum data rate of 20k bits/second has been defined by the standard. Again with the purpose of reducing the chance of cross talk.

The impedance of the interface between the driver and receiver has also been defined. The load seen by the driver is specified to be 3kΩ to 7kΩ. For the original RS–232 standard, the cable between the driver and the receiver was also specified to be a maximum of 15 meters in length. This part of the standard was changed in revision "D" (EIA/TIA–232–D). Instead of specifying the maximum length of cable, a maximum capacitive load of 2500 pF was specified which is clearly a more adequate specification. The maximum cable length is determined by the capacitance per unit length of the cable which is provided in the cable specifications.

RS–232 LOGIC LEVEL SPECIFICATIONS Figure 1



FUNCTIONAL CHARACTERISTICS

Since RS–232 is a "complete" standard, it includes more than just specifications on electrical characteristics. The second aspect of operation that is covered by the standard concerns the functional characteristics of the interface. This essentially means that RS–232 has defined the function of the different signals that are used in the interface. These signals are divided into four different categories: common, data, control, and timing. Table 1 illustrates the signals that are defined by the RS–232 standard. As can be seen from the table there is an overwhelming number of signals defined by the standard. The standard provides an abundance of control signals and supports a primary and secondary communications channel. Fortunately few applications, if any, require all of these defined signals. For example, only eight signals are used for a typical modem. Some simple applications may require only four signals (two for data and two for handshaking) while others may require only data signals with no handshaking. Examples of how the RS–232 standard is used in some "real world" applications are discussed later in this paper. The complete list of defined signals is included here as a reference, but it is beyond the scope of this paper to review the functionality of all of these signals.

MECHANICAL INTERFACE CHARACTERISTICS

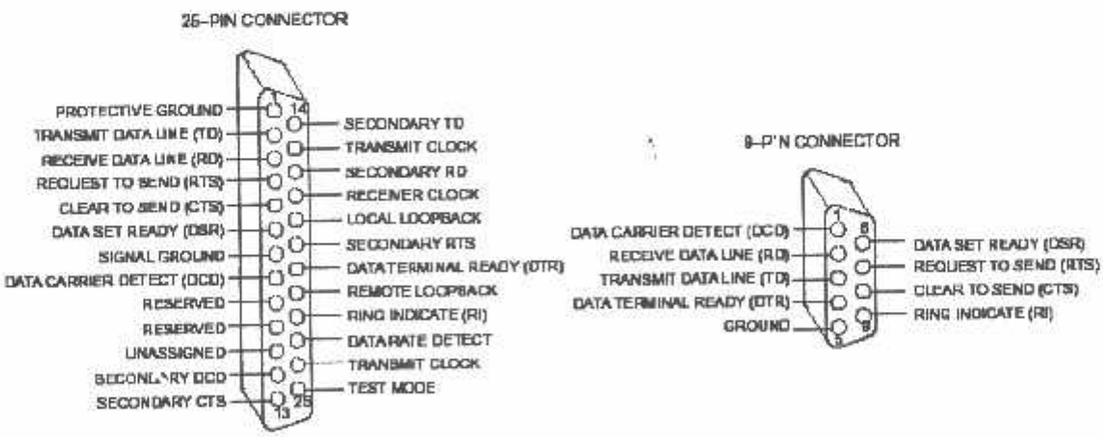
The third area covered by RS–232 concerns the mechanical interface. In particular, RS–232 specifies a 25–pin connector. This is the minimum connector size that can accommodate all of the signals defined in the functional portion of the standard. The pin assignment for this connector is shown in Figure 2. The connector for DCE equipment is male for the connector housing and female for the connection pins. Likewise, the DTE connector is a female housing with male connection pins. Although RS–232 specifies a 25–position connector, it should be noted that often this connector is not used. This is due to the fact that most applications do not require all of the defined signals and therefore a 25–pin connector is larger than necessary. This being the case, it is very common for other connector types to be used. Perhaps the most popular is the 9–position DB9S connector which is also illustrated in Figure 2. This connector provides the means to transmit and receive the necessary signals for modem applications, for example. This will be discussed in more detail later.

RS–232 DEFINED SIGNALS Table 1

CIRCUIT MNEMONIC	CIRCUIT NAME*	CIRCUIT DIRECTION	CIRCUIT TYPE
AB	Signal Common	–	Common
BA	Transmitted Data (TD)	To DCE	Data
BB	Received Data (RD)	From DCE	
CA	Request to Send (RTS)	To DCE	Control
CB	Clear to Send (CTS)	From DCE	
CC	DCE Ready (DSR)	From DCE	
CD	DTE Ready (DTR)	To DCE	
CE	Ring Indicator (RI)	From DCE	
CF	Received Line Signal Detector** (DCD)	From DCE	
CG	Signal Quality Detector	From DCE	
CH	Data Signal Rate Detector from DTE	To DCE	
CI	Data Signal Rate Detector from DCE	From DCE	
CJ	Ready for Receiving	To DCE	
RL	Remote Loopback	To DCE	
LL	Local Loopback	To DCE	
TM	Test Mode	From DCE	
DA	Transmitter Signal Element Timing from DTE	To DCE	
DB	Transmitter Signal Element Timing from DCE	From DCE	Timing
DD	Receiver Signal Element Timing From DCE	From DCE	
SBA	Secondary Transmitted Data	To DCE	Data
SBB	Secondary Received Data	From DCE	
SCA	Secondary Request to Send	To DCE	Control
SCB	Secondary Clear to Send	From DCE	
SCF	Secondary Received Line Signal Detector	From DCE	

*Signals with abbreviations in parentheses are the eight most commonly used signals.
**This signal is more commonly referred to as Data Carrier Detect (DCD).

RS–232 CONNECTOR PIN ASSIGNMENTS Figure 2



PRACTICAL RS–232 IMPLEMENTATION

Most systems designed today do not operate using RS–232 voltage levels. Since this is the case, level conversion is necessary to implement RS–232 communication. Level conversion is performed by special RS–232 IC's. These IC's typically have line drivers that generate the voltage levels required by RS–232 and line receivers that can receive RS–232 voltage levels without being damaged. These line drivers and receivers typically invert the signal as well since a logic 1 is represented by a low voltage level for RS–232 communication and likewise a logic 0 is represented by a high logic level. Figure 3 illustrates the function of an RS–232 line driver/receiver in a typical modem application. In this particular example, the signals necessary for serial communication are generated and received by the Universal Asynchronous Receiver/Transmitter (UART). The RS–232 line driver/receiver IC performs the level translation necessary between the CMOS/TTL and RS–232 interface.

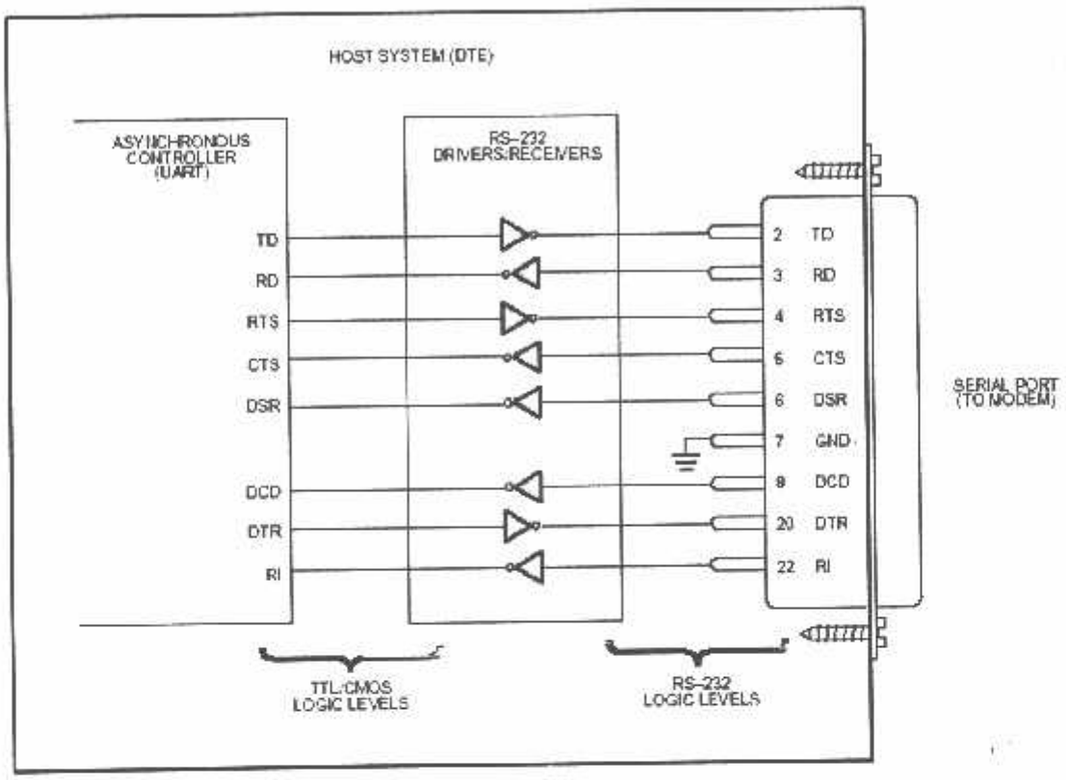
The UART just mentioned performs the "overhead" tasks necessary for asynchronous serial communication. For example, the asynchronous nature of this type of communication usually requires that start and stop bits be initiated by the host system to indicate to the peripheral system when communication will start and stop. Parity bits are also often employed to ensure that the data sent has not been corrupted. The UART usually generates the start, stop, and parity bits when transmitting data and can detect communication errors upon receiving data. The UART also functions as the intermediary between byte–wide (parallel) and bit–wide (serial) communication; it converts a byte of data into a serial bit stream for transmitting and converts a serial bit stream into a byte of data when receiving.

Now that an elementary explanation of the TTL/CMOS to RS–232 interface has been provided we can consider some "real world" RS–232 applications. It has already been noted that RS–232 applications rarely follow the RS–232 standard precisely. Perhaps the most significant reason this is true is due to the fact that many of the defined signals are not necessary for most applications. As such, the unnecessary signals are omitted. Many applications , such as a modem, require only nine signals (two data signals, six control signals, and ground). Other applications may require only five signals (two for data, two for handshaking, and ground), while others may require only data signals with no handshake control. We will begin our investigation of "real world" implementations by first considering the typical modem application.

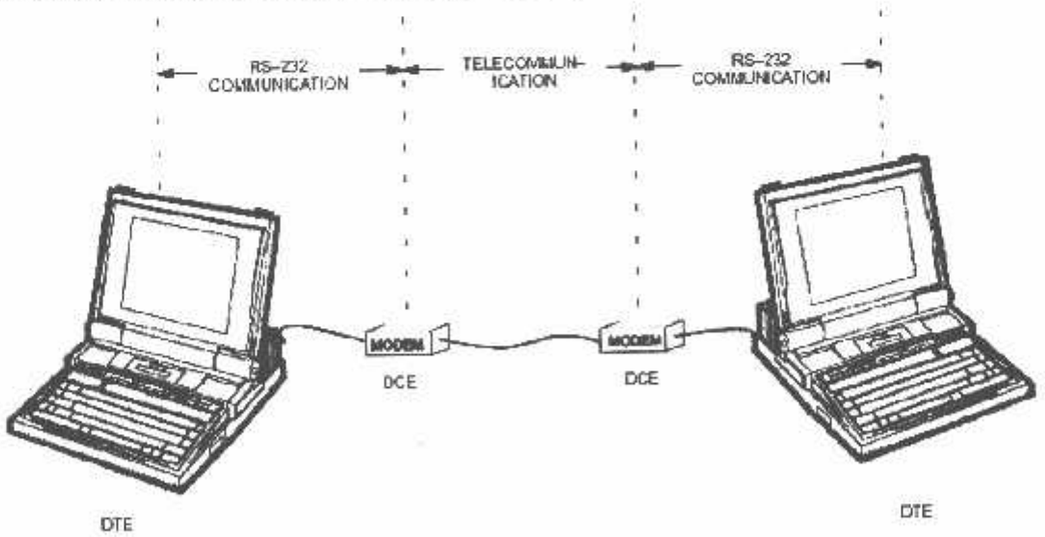
RS–232 IN MODEM APPLICATIONS

Modem applications are one of the most popular uses for the RS–232 standard. Figure 4 illustrates a typical modem application utilizing the RS–232 interface standard. As can be seen in the diagram, the PC is the DTE and the modem is the DCE. Communication between each PC and its associated modem is accomplished using the RS–232 standard. Communication between the two modems is accomplished via telecommunication. It should be noted that although a microcomputer is usually the DTE in RS–232 applications, this is not mandatory according to a strict interpretation of the standard.

TYPICAL RS-232 MODEM APPLICATION Figure 3



MODEM COMMUNICATION BETWEEN TWO PC'S Figure 4



Many modem applications require only nine signals (including ground). Although some designers choose to use a 25–pin connector, it is not necessary since there are only nine interface signals between the DTE and DCE. With this in mind, many have chosen to use to use 9– or 15–pin connectors (see Figure 2 for 9–pin connector pin assignment). The "basic nine" signals used in modem communication are illustrated in Figure 3. Note that with respect to the DTE, three RS–232 drivers and five receivers are necessary. The functionality of these signals is described below. Note that for the following signal descriptions, "ON" refers to a high RS–232 voltage level (+5 t o +15 volts) and "OFF" refers to a low RS–232 voltage level (–5 to –15 volts). Keep in mind that a high RS–232 voltage level actually represents a logic 0 and a low RS–232 voltage level refers to a logic 1.

Transmitted Data (TD): One of two separate data signals. This signal is generated by the DTE and received by the DCE.

Received Data (RD): The second of two separate data signals. This signals is generated by the DCE and received by the DTE.

Request to Send (RTS): When the host system (DTE) is ready to transmit data to the peripheral system (DCE), RTS is turned ON. In simplex and duplex systems, this condition maintains the DCE in receive mode. In half–duplex systems, this condition maintains the DCE in receive mode and disables transmit mode. The OFF condition maintains the DCE in transmit mode. After RTS is asserted, the DCE must assert CTS before communication can commence.

Clear to Send (CTS): CTS is used along with RTS to provide handshaking between the DTE and the DCE. After the DCE sees an asserted RTS, it turns CTS ON when it is ready to begin communication.

Data Set Ready (DSR): This signal is turned on by the DCE to indicate that it is connected to the telecommunications line.

Data Carrier Detect (DCD): This signal is turned ON when the DCE is receiving a signal from a remote DCE which meets its suitable signal criteria. This signal remains ON as long as the a suitable carrier signal can be detected.

Data Terminal Ready (DTR): DTR indicates the readiness of the DTE. This signal is turned ON by the DTE when it is ready to transmit or receive data from the DCE. DTR must be ON before the DCE can assert DSR.

Ring Indicator (RI): RI, when asserted, indicates that a ringing signal is being received on the communications channel.

The signals described above form the basis for modem communication. Perhaps the best way to understand how these signals interact is to give a brief step by step example of a modem interfacing with a PC. The following step s describe a transaction in which a remote modem calls a local modem.

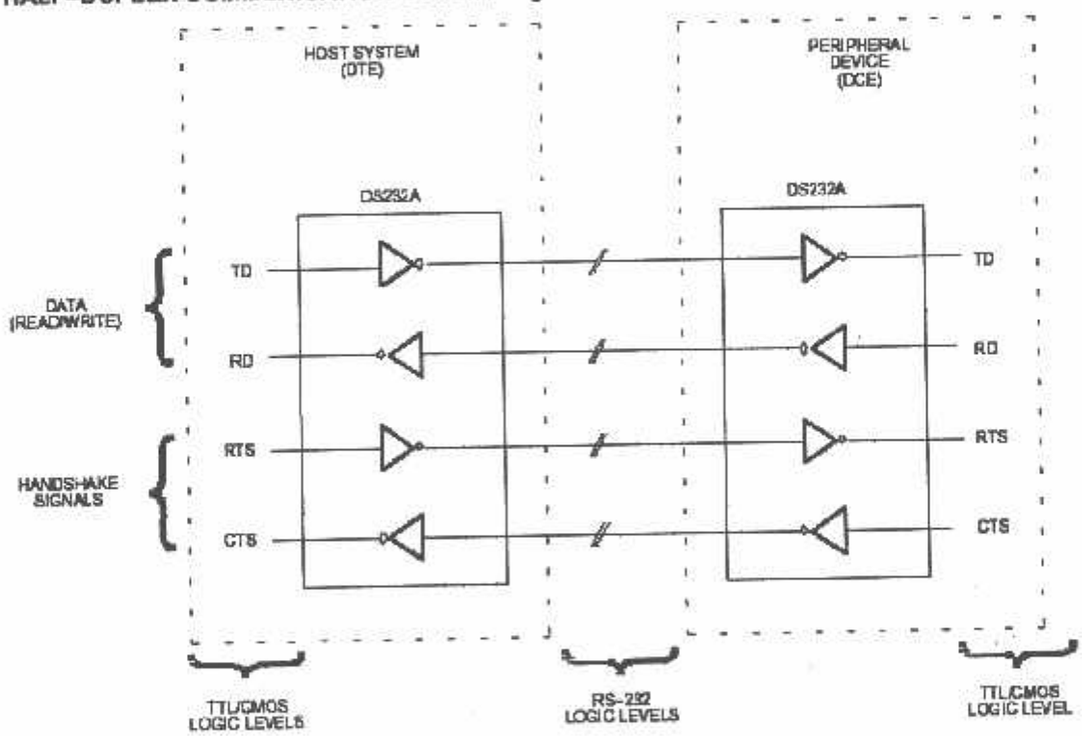
1. The local PC monitors the RI (Ring Indicate) signal via software.
2. When the remote modem wants to communicate with the local modem, it generates an RI signal. This signal is transferred by the local modem to the local PC.
3. The local PC responds to the RI signal by asserting the DTR (Data Terminal Ready) signal when it is ready to communicate.
4. After recognizing the asserted DTR signal, the modem responds by asserting DSR (Data Set Ready) after it is connected to the communications line. DSR indicates to the PC that the modem is ready to exchange further control signals with the DTE to commence communication. When DSR is asserted, the PC begins monitoring DCD for indication that data is being sent over the communication line.
5. The modem asserts DCD (Data Carrier Detect) after it has received a carrier signal from the remote modem that meets the suitable signal criteria.
6. At this point data transfer can began. If the local modem has full–duplex capability, the CTS (Clear to Send) and RTS (Request to Send) signals are held in the asserted state. If the modem has only half–duplex capability, CTS and RTS provide the handshaking necessary for controlling the direction of the data flow. Data is transferred over the RD and TD signals.
7. When the transfer of data has been completed, the PC disables the DTR signal. The modem follows by inhibiting the DSR and DCD signals. At this point the PC and modem are in the original state described in step number 1.

RS–232 IN MINIMAL HANDSHAKE APPLICATIONS

Even though the modem application discussed above is simplified from the RS–232 standard in terms of the number of signals needed, it is still more complex than the requirements of many systems. For many applications, two data lines and two handshake control lines are all that is necessary to establish and control communication between a host system and a peripheral system. For example, an environmental control system may need to interface with a thermostat using a half–duplex communication scheme. At times the control systems may desire to read the temperature from the thermostat and at other times may need to load temperature trip points to the thermostat. In this type of simple application, five signals may be all that is necessary (two for data, two for handshake control, and ground).

Figure 5 illustrates a simple half–duplex communication interface. As can be seen in this diagram, data is transferred over the TD (Transmit Data) and RD (Receive Data) pins and handshake control is provided by the RTS (Ready to Send) and CTS (Clear to Send) pins. RTS is driven by the DTE to control the direction of data. When it is asserted, the DTE is placed in transmit mode. When RTS is inhibited, the DTE is placed in receive mode. CTS, which is generated by the DCE, controls the flow of data. When asserted, data can flow. However, when CTS is inhibited, the transfer of data is interrupted. The transmission of data is halted until CTS is reasserted.

HALF–DUPLEX COMMUNICATION SCHEME Figure 5



RS–232 APPLICATION LIMITATIONS

As mentioned earlier in this paper, the RS–232 standard was first introduced in 1962. In the more than three decades since, the electronics industry has changed immensely and therefore there are some limitations in the RS–232 standard. One limitation, the fact that over twenty signals have been defined by the standard, has already been addressed – simply do not use all of the signals or the 25–pin connector if they are not necessary. Other limitations in the standard are not necessarily as easy to correct, however.

GENERATION OF RS–232 VOLTAGE LEVELS

As we saw in the section on RS–232 electrical characteristics, RS–232 does not use the conventional 0 and 5 volt levels implemented in TTL and CMOS designs. Drivers have to supply +5 to +15 volts for a logic 0 and –5 to –15 volts for a logic 1. This means that extra power supplies are needed to drive the RS–232 voltage levels. Typically, a +12 volt and a –12 volt power supply are used to drive the RS–232 outputs. This is a great inconvenience for systems that have no other requirements for these power supplies. With this in mind, RS–232 products manufactured by Dallas Semiconductor have on–chip charge–pump circuits that generate the necessary voltage levels for RS–232 communication. The first charge pump essentially doubles the standard +5 volt power supply to provide the voltage level necessary for driving a logic 0. A second charge pump, inverts this voltage and provides the voltage level necessary for driving a logic 1. These two charge pumps allow the RS–232 interface products to operate from a single +5 volt supply.

MAXIMUM DATA RATE

Another limitation in the RS–232 standard is the maximum data rate. The standard defines a maximum data rate of 20k bits/second. This is unnecessarily slow for many of today's applications. RS–232 products manufactured by Dallas Semiconductor guarantee up to 250k bits/second and typically can communicate up to 350k bits/second. While providing a communication rate at this frequency, the devices still maintain a maximum 30V/µs maximum slew rate to reduce the likelihood of cross–talk between adjacent signals.

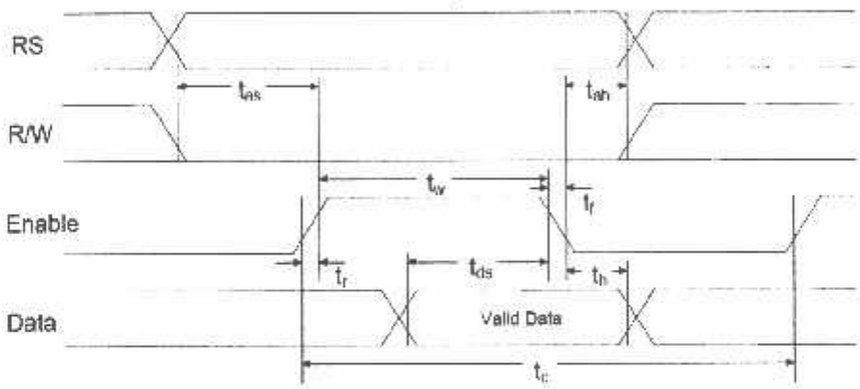
MAXIMUM CABLE LENGTH

A final limitation to discuss concerning RS–232 communication is cable length. As we have already seen, the cable length specification that was once included in the RS–232 standard has been replaced by a maximum load capacitance specification of 2500 pF. To determine the total length of cable allowed, one must determine the total line capacitance. Figure 6 shows a simple approximation for the total line capacitance of a conductor. As can be seen in the diagram, the total capacitance is approximated by the sum of the mutual capacitance between the signal conductors and the conductor to shield capacitance (or stray capacitance in the case of unshielded cable).

As an example, let's assume that the user has decided to use non–shielded cable when interconnecting the equipment. The cable mutual capacitance (Cm) of the cable is found in the cable's specifications to be 20 pF per foot. If we assume that the input capacitance of the receiver is 20 pF, this leaves the user with 2480 pF for the interconnecting cable. From the equation in Figure 6, the total capacitance per foot is found to be 30 pF. Dividing 2480 pF by 30 pF reveals that the maximum cable length is approximately 80 feet. If a longer cable length is required, the user would need to find a cable with a smaller mutual capacitance.

Instruction	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Description	Clocks
NOP	0	0	0	0	0	0	0	0	0	0	No Operation	0
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears display & sets address counter to zero.	165
Cursor Home	0	0	0	0	0	0	0	0	1	0	Sets address counter to zero, returns shifted display to original position DDRAM contents remains unchanged.	3
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction, and specifies automatic shift.	3
Display Control	0	0	0	0	0	0	1	D	C	B	Turns display (D), cursor on/off (C) or cursor blinking(B).	3
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	0	0	Moves cursor and shift display. DDRAM contents remains unchanged.	3
Function Set	0	0	0	0	1	DL	N	M	G	0	Sets interface data width(DL), number of display lines (N,M) and voltage generator control (G).	3
Set CGRAM Addr	0	0	0	1	Character Generator RAM						Sets CGRAM Address	3
Set DDRAM Addr	0	0	1	Display Data RAM Address							Sets DDRAM Address	3
Busy Flag & Addr	0	1	BF	Address Counter							Reads Busy Flag & Address Counter	0
Read Data	1	0	Read Data								Reads data from CGRAM or DDRAM	3
Write Data	1	1	Write Data								Writes data from CGRAM or DDRAM	3

Write Cycle

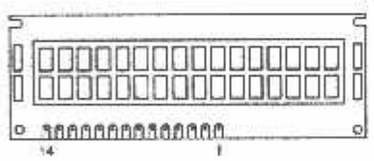


Parameter	Symbol	Min [1]	Typ [1]	Max [1]	Unit
Enable Cycle Time	t_c	500	-	-	ns
Enable Pulse Width (High)	t_w	230	-	-	ns
Enable Rise/Fall Time	t_r,t_f	-	-	20	ns
Address Setup Time	t_{as}	40	-	-	ns
Address Hold Time	t_{ah}	10	-	-	ns
Data Setup Time	t_{ds}	80	-		ns
Data Hold Time	t_h	10	-	-	ns

Note [1] The above specifications are a indication only. Timing will vary from manufacturer to manufacturer.

Note [2] A 2 line by 16 Character LCD Module is Pictured. Data will work on most 1 line x 16 character, 1 line x 20 character, 2 line x 16 character, 2 line x 20 character, 4 lines x 20 character, 2 lines x 40 character etc. modules compatible with the HD44780 LCD Module.

[2]



Pin No	Name	I/O	Description
1	Vss	Power	GND
2	Vdd	Power	+5v
3	Vo	Analog	Contrast Control
4	RS	Input	Register Select
5	R/W	Input	Read/Write
6	E	Input	Enable *(Strobe)*
7	D0	I/O	Data *LSB*
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data *MSB*