INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
KONSENTRASI TEKNIK ELEKTRONIKA

SKRIPSI

PERANCANGAN DAN PEMBUATAN KRS ONLINE MENGGUNAKAN SMS

Disusun Oleh :

AAN NOERCAHYADI
Nim. 99.17.165

APRIL 2005

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN
SISTEM KRS ON LINE MENGGUNAKAN SMS

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat Guna Mencapai Gelar Sarjana Teknik

Disusun Oleh :

AAN NOERCAHYADI
NIM : 9917165

Diperiksa dan Disetujui Oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Poerwanto.MT	Ir. Mimien Mustikawati
NIP. 131 574 847	NIP. 103 0000 352

Mengetahui
Ketua Jurusan T. Elektro S-1

Ir. F Yudi Limpraptono, MT
NIP. P.103 9500 274

KONSENTRASI TEKNIK ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	: Aan Noercahyadi
NIM	: 9917165
Jurusan	: Teknik Elektro S-1
Kosentrasi	: Teknik Elektronika
Judul Skripsi	: Perancangan Dan Pembuatan Sistem KRS Online Menggunakan SMS

Dipertahankan dihadapan Team Penguji Skripsi Jenjang Strata Satu (S-1)

Pada :

Hari	: Senin
Tanggal	: 28 Maret 2005
Dengan Nilai	: 83,7 (A)

Panitia Ujian Skripsi

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
DEKAN

(Ir. Mochtar Asroni , MSME)
Ketua

(Ir. F. Yudi Limpraptono, MT)
Sekretaris

Anggota Penguji

(Ir. Yusuf Ismail Nakoda, MT)
Penguji Pertama

(Ir. Eko Nurcahyo)
Penguji Kedua

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Segala puji hanya milik-Mu Gusti Allah yang telah memberikan rahmat, taufik dan hidayah-Mu sehingga segala ikhtiar manusia dapat terlaksana. Skripsi kami dengan judul :

"PERANCANGAN DAN PEMBUATAN SISTEM KRS ON LINE MENGGUNAKAN SMS "

Pembuatan skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang Strata-1 di Institut Teknologi Nasional Malang. Laporan Skripsi ini merupakan tanggung jawab tertulis atas ilmu pengetahuan yang didapat selama penyusunan mengikuti kuliah.

Atas selesainya Skripsi ini, penulis mengucapkan terima kasih kepada :

- Bapak Dr. Ir. Abraham Lomi, MT, selaku Rektor Institut Teknologi Nasional Malang.
- Bapak Ir. F. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro S-1.
- Ibu Ir, Mimien Mustikawati, selaku Seketaris Jurusan Teknik Elektro S-1.
- Bapak Ir. Poerwanto, MT, selaku Dosen Pembimbing I. yang telah memberikan, pengarahan, serta ilmu-ilmu yang sangat berharga sehingga Skripsi ini dapat diselesaikan.

- Ibu Ir. Mimien Mustikawati, selaku Dosen Pembimbing II. yang telah memberikan bimbingan, serta ilmu-ilmu yang sangat berharga sehingga Skripsi ini dapat diselesaikan.
- Bapak dan Ibuku tercinta yang selalu mendoakan dan penuh kasih saying memberikan dorongan selama penulisan skripsi ini.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat penulis harapkan.

Akhir kata, penulis mohon maaf kepada semua pihak bilamana salama penyusunan Skripsi ini membuat kasalahan. Dan semoga Skripsi ini dapat bermanfaat bagi kita semua.

Malang, Maret 2005

Penulis

ABSTRAK

Penulisan : Aan Noercahyadi, Nim: 9917165, Fakultas Teknologi Industri, Jurusan Teknik Elektro, Program Studi Teknik Elektronika, ITN Malang,
Judul Skripsi : PERANCANGAN DAN PEMBUATAN SISTEM KRS ONLINE MENGGUNAKAN SMS.
Dosen Pembimbing I : Ir. Poerwanto, MT.
Dosen Pembimbing II : Ir. Mimien Mustikawati.

Layanan Short Message Services (SMS) dapat menyampaikan informasi teks singkat yang diakses melalui handphone, fasislitas tersebut dapat dimanfaatkan sebagai jalur layanan administrasi berbasis teks. Dalam sistem ini, SMS akan digunakan sebagai jalur pelayanan informasi pendidikan didalam kampus berupa informasi teks Indeks Prestasi mahasiswa dan layanan Kartu Rencana Studi. Sistem ini terdiri dari: database yang tersimpan pada PC, HANDPHONE, RANGKAIAN INTERFACE, serta komponen pendukung lainnya.

Untuk dapat mengirim dan membaca data SMS yang terdapat dalam handphone, digunakan perintah AT-Command melewati port RS-232. RANGKAIAN INTERFACE berfungsi menjembatani level tegangan digital yang berbeda antara PC dan HANDPHONE. Apabila format SMS benar maka komputer akan mencari informasi yang sesuai pada database kemudian mengirimkannya melalui SMS kepada nomor handphone peminta informasi.

Dari hasil pengujian dan analisa didapatkan bahwa laju data komunikasi serial antara handphone dan pc sebesar 19200bps, menggunakan tipe handphone SIEMENS C-35. Kecepatan respon balasan SMS tergatung terhadap layanan provider GSM. AT command merupakan perintah standart yang dipergunakan mengendalikan handphone mulai dari fungsi mengangkat dan melakukan panggilan hingga membaca dan mengirimkan SMS serta fungsi fungsi lainnya.

Agar PC dapat membaca dan mengirim sms pada handphone, digunakan perintah AT command yang dikirim melalui port RS-232 menuju handphone. Namun, terlebih dahulu RANGKAIAN INTERFACE mengubah level tegangan RS-232 menjadi level tegangan yang sesuai dengan level tegangan handphone (TTL). Apabila format SMS sesuai maka komputer akan mencari pada database dan mengirimkan SMS u handphone, terlebih dahulu rangkaian interface mengubah level tegangan RS-232 menjadi level tegangan yang sesuai dengan handphone yakni level tegangan TTL Demikian pula sebaliknya bila handphone mengirimkan data ke PC.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pada setiap akhir semester mahasiswa akan menerima Kartu Hasil Studi (KHS) yang berisi nilai setiap mata kuliah yang ditempuh dalam semester tersebut. Penilaian ini diberikan dosen berdasarkan perhitungan nilai tugas, kuis, ujian tengah dan akhir semester. Indeks prestasi akademik dihitung dari nilai mata kuliah yang diperoleh mahasiswa selama satu semester, sedangkan indeks prestasi komulatif dihitung dari semua nilai mata kuliah yang telah di tempuh mahasiswa. Mahasiswa mengisi KRS dengan mata kuliah yang akan diambilnya dengan persetujuan dosen wali.

Beberapa hal yang perlu diperhatikan Mahasiswa dalam mengisi KRS yaitu jumlah sks yang boleh diambil berdasarkan KRS dan IP, mata kuliah yang ditawarkan dan mata kuliah prasyarat. Banyaknya sks yang boleh diambil, mata kuliah yang ditawarkan dan mata kuliah yang prasyarat. Banyaknya sks yang boleh diambil ditentukan berdasarkan indeks prestasi Mahasiswa, karena itu diperlukan KHS. Mata kuliah prasyarat adalah mata kuliah yang harus sudah pernah ditempuh atau lulus sebelum menempuh mata kuliah lainnya. Aturan mengenai mata kuliah prasyarat ini dimuat dalam kurikulum yang diambil oleh setiap mahasiswa/i.

Besarnya data akademik mahasiswa dan kurikulum yang berubah-ubah memerlukan media penyimpanan yang cepat dalam pencarian dan perubahan data.

Media penyimpanan berupa dokumen kertas tidak efisien karena apabila diperlukan pencarian dan perubahan satu nilai mahasiswa dari data yang besar tersebut akan memerlukan waktu dan tenaga yang besar. Media penyimpanan yang lebih baik dari media kertas adalah media digital seperti *harddisk* atau *tape backup* dengan sistem *database.*

Demikian pula dalam menghadapi masalah yang terjadi pada saat ini adalah lambannya proses pengisian KRS karena jumlah mahahsiswa yang banyak sedangkan petugas yang melayani hanya beberapa orang. Proses pemeriksaan KRS oleh petugas secara manual juga akan memperlama proses pengisian KRS. Salah satu alternatif untuk memecahkan masalah ini adalah menggunakan sistem jaringan komputer .

Karena keunggulan penggunaan media telepon tersebut maka dirancang suatu alat yang dapat memberikan pelayanan akademik untuk mengisi KRS dan mengetahui nilai IPK melalui Hendphone.

Untuk merealisasikan alat layanan akademik melalui ponsel diperlukan suatu perangkat yang dapat beropersai melalui jalur ponsel dengan spesifikasi sesuai ketentuan pengguna ponsel sehingga alat ini tidak membebani jaringan telepon. Disisi *server* aplikasi dibuat perangkat lunak yang dapat mengontrol suatu program yang akan dikirim tersebut dan mengakses *database* akademik mahasiswa. Antarmuka menerima data yang dikirim melalui ponsel dan mengirimkan data tersebut ke komputer yang terhubung dengan *database server.* Kemudian komputer tersebut akan mengolah data yang diterima dari antarmuka tersebut untuk mengakses data di *database server* sesuai dengan permintaan dari mahasiswa.

1.2.Rumusan Masalah

Dari beberapa permasalahan yang ada, maka pada proposal ini masalah dititik beratkan pada :

1. Bagaimana cara menyanbungkan handphone sebagai penerima sms dengan computer.
2. Bagaimana merencanakan dan memproses sms yang masuk ke handphone untuk dibaca oleh computer.

1.3. Batasan Masalah

Agar permasalahan yang dibahas tidak meluas maka perlu adanya pembatasan permasalahan. Adapun batasan masalah yang meliputi :

1. Antara handphone dan computer disambungkan dengan Level Converter.
2. Perangkat lunak dibuat menggunakan Borland Delphi 6 pada sistem operasi Windows Xp.
3. Menggunakan handphone siemens C35 dengan kecepatan data antara computer dan handphone sebesar 19200 bps.

1.4. Tujuan Masalah

Tujuan pokok dari penulisan skripsi ini adalah merancang dan merealisasikan alat layanan akademik melalui handphone, sehingga mahasiswa dapat mengetahui

KRS dan mengisi KPS mereka tanpa harus datang di kampus dan tidak bergantung kepada petugas administrasi akademik

1.5. Metodologi Perencanaan

Adapun langkah-langkah yang diambil untuk menyelesaikan perubahan teoritis pada pembuatan perangkat keras dalam pelayanan informasi akademik adalah sebagai berikut:

a. Pengumpulan literatur

   Studi literature tentang teori Delphi secara umum, kemudian mempelajari teori Delphi dan cara mempelajari pembuatan aturan database pada program Delphi.

b. Perencanaan dan Pembuatan Alat

   Setelah melakukan pengumpulan literatur, maka dilakukan perencanaan perangkat keras dan perangkat lunak pengujian untuk mengetahui hasil rancangan yang telah dibuat.

1.6. Sistematika Penulisan

Sistematika penulisan dan gambaran secukupnya yang terdapat dalam setiap bab sebagai berikut:

BAB I : PENDAHULUAN

Membahas tentang latar belakang, rumusan masalah, tujuan batasan masalah, metodologi perencanaan, sitematika pembahasan.

BAB II : DASAR TEORI

Memuat teori penunjang tentang Borland Delphi 6, SQL Server 7 antarmuka pada komputer dan handphone secara umum yang diperlukan dalam perancangan dan pembuatan alat

BAB III : PERENCANAAN DAN PEMBUATAN ALAT

Membahas perancangan dan pembuatan perangkat keras dan perangkat lunak alat pelayanan informasi akakdemik melalui handphone

BAB IV : PENGUJIAN ALAT

Membahas tentang pengujian-pengujian terhadap sistem yang telah dibuat

BAB V : PENUTUP

Merupakan bagian-bagian penutup yang berisi kesimpulan dan saran yang bermanfaat bagi pengembangan lebih lanjut

BAB II

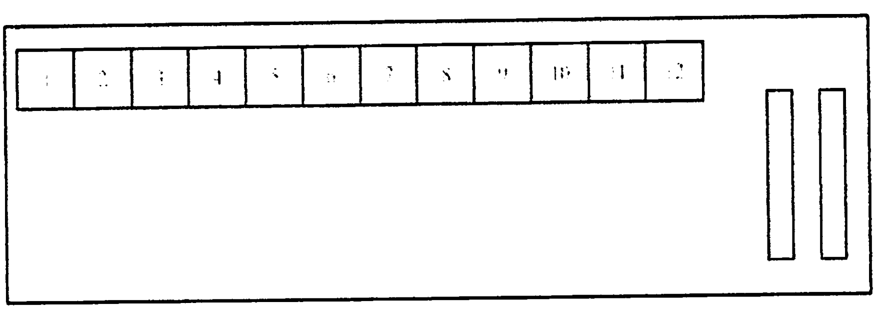
LANDASAN TEORI

2.1 Sistem Kerja SMS

2.1.1. Kabel Data

Kabel data adalah perangkat keras yang merupakan sarana penghubung antara ponsel dengan computer agar bisa berkomunikasi.

Macam data kabel yaitu : FBUS, MBUS, dan gabungan MBUS/FBUS. FBUS memerlukan 2 input/output untuk mengirimkan dan menerima data sedangkan MBUS hanya memerlukan 1 input/output saja. Hanya dengan kabel MBUS anda dapat mengakses data penting didalam ponsel untuk menservis ponsel, upgrade softwere, dan lain-lain. FBUS biasanya digunakan untuk memasukkan logo dan ringtone di ponsel serta untuk kebutuhan mobilitas seperti konek ke internet dan faxing. Kecepatan data ketiganya untuk kirim maupun terima biasanya 9,6 Kbps sampai 14,4 Kbps.



Gambar 2.1. Konektor Handphone C35

Koneksi handphone siemens pada dasarnya dapat menggunakan rangkaian yang sama dengan skematik rangkian pada pada gambar 2.1. Namun perbedaannya adalah pada perangkat lunak yang berkomunikasi dengan handphone siemens tersebut. Konektor siemens C35 dapat dilihat pada gambar diatas.

Untuk penggunaan masing-masing pin dari konektor diatas dapat dilihat pada table dibawah :

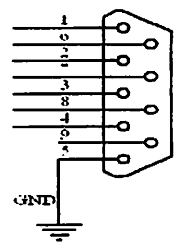
Tabel 2.1. Pinout Konektor C35

Pin	Nama	Fungsi	In/out
1	GND	Ground	
2	SELF SERVICE	Recognition/ control battery charger	In/Out
3	LOAD	Charging Voltage	In
4	BATTERY	Battery	Out
5	DATA OUT	Data sent	Out
6	DATA IN	Data recerved	In
7	Z_CLK	Recognition/ control accessories	
8	Z_DATA	Recognition/ control accessories	
9	MICG	Ground for microphone	In
10	MIC	Microphone input	
11	AUD	Loudspeaker	Out
12	AUDG	Ground for eksternal speaker	

2.1.2. Port Serial

Port komunikasi PC yang digunakan adlah Port Serial yang terdiri dari berbagai jenis sinyal yang dipakai mengatur pertukaran informasi antara DTE dan

DCE, semua terdapat 24 jenis sinyal tapi yang umum dipakai hanyalah 9 jenis sinyal.



GAMBAR 2.2. PORT SERIAL

Untuk sinyal yang lengkap adalah konektor DB25, sedangkan konektor DB9 hanya bisa dipakai untuk 9 sinyal yang umum dipakai.

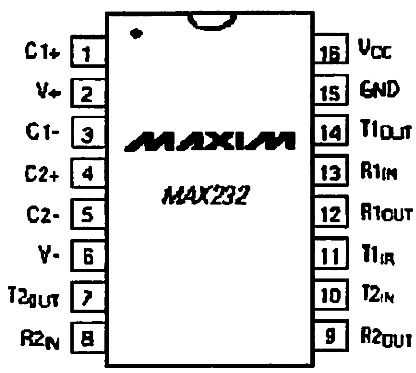
Port Serial dengan DB9 *tabel Connections* yang mempunyai 9 buah pin dengan fungsinya masing-masing ditunjukkan pada tabel 2.2.

TABEL 2.2. KONEKTOR DB 9 [1]

No	Konektor	
1.	Received Line Signal Detector (Carrier Detect/DCD)	Handshake from DCE
2.	Received Data (RD)	Data from DCE
3.	Transmit Data (TD)	Data from DTE
4.	Data Terminal Ready (DTR)	Handshake fromDTE
5.	Signal Ground	Reference point for signals
6.	Data Set Ready (DST)	Handshake from DCE
7.	Request To Send (RTS)	Handshake from DTE
8.	Clear To Send (CTD)	Handshake from DCE
9.	Ring Indicator	Handshake from DCE

Sinyal-sinyal tersebut ada yang menuju ke DCE ada pula yang berasal dari DCE. Sinyal yang menuju DCE artinya DTE berfungsi sebagai output data DCE berfungsi sebagai input, misalnya sinyal TD, pada sisi DTE kaki TD adalah output, dan kaki ini dihubungkan ke kaki TD pada DCE yang berfungsi sebagai input. Kebalikan sinyal TD adalah RD, sinyal ini berasal dari DCE dan dihubungkan ke kaki RD pada TDE yang berfungsi sebagai input.

2.2. RS MAX 232



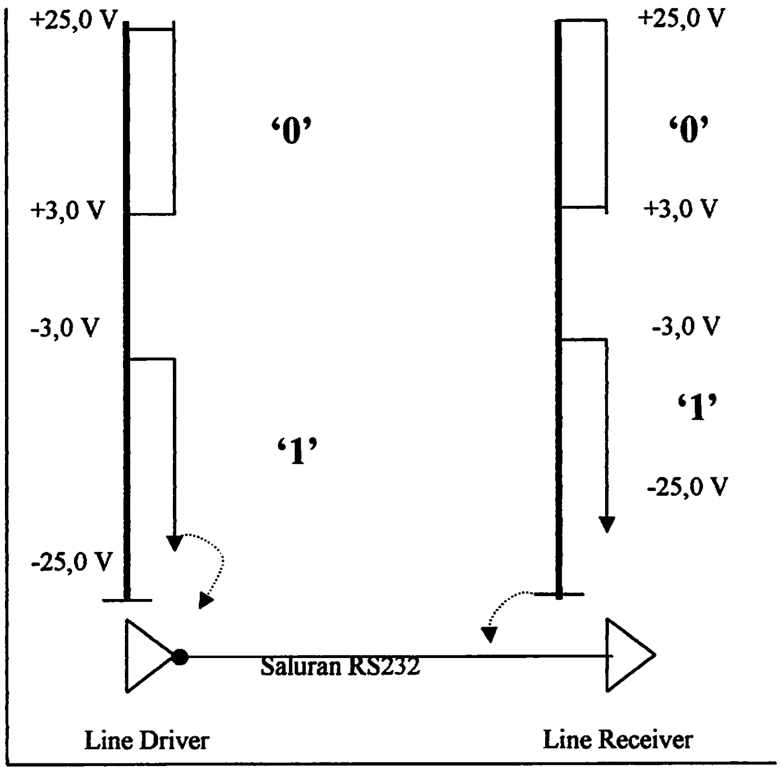
GAMBAR 2.3. RS MAX 232

RS MAX232 tersusun dari 2 bagian yaitu : RS232 Line Driver yang berfungsi mengubah level tegangan TTL ke level tegangan RS232 dan RS232 Line Receiver yang berfungsi mengubah tegangan RS232 ke level tegangan TTL.

Standard RS232 ditetapkan oleh *Electronic Industry Association* dan *Telecomunication Industry Association* pada tahun 1962. Standard ini hanya menyangkut komunikasi data antara komputer (Data Terminal Equipment – DTE) dengan alat-alat pelengkap komputer maupun alat-alat digital lainnya (Data Circuit-Terminating Equipment – DCE).

Ada 3 hal pokok yang diatur standard RS232, antara lain adalah :

1. Bentuk sinyal dan level tegangan yang dipakai



GAMBAR 2.4. LEVEL TEGANGAN RS232

Level tegangan sinyal RS232 adalah seperti terlihat pada gambar 2-3. Dalam standard RS232, tegangan antara +3 sampai +25 Volt baik pada input Line Receiver maupun pada output Line Driver dianggap sebagai level tegangan '0', dan tegangan antara -3 nsampai -25 Volt dianggap sebagai level tegangan ' 1'.

2. Penentuan jenis sinyal dan konektor yang dipakai, serta susunan sinyal pada kaki-kaki di konektor. Jenis-jenis sinyal yang dipakai mengatur

pertukaran informasi antara DTE dan DCE, semuanya terdapat 24 jenis sinyal tapi yang umum dipakai hanyalah 9 jenis sinyal.

3. Penentuan tata cara pertukaran informasi antara komputer dan alat-alat perlengkapnya maupun alat-alat digital.

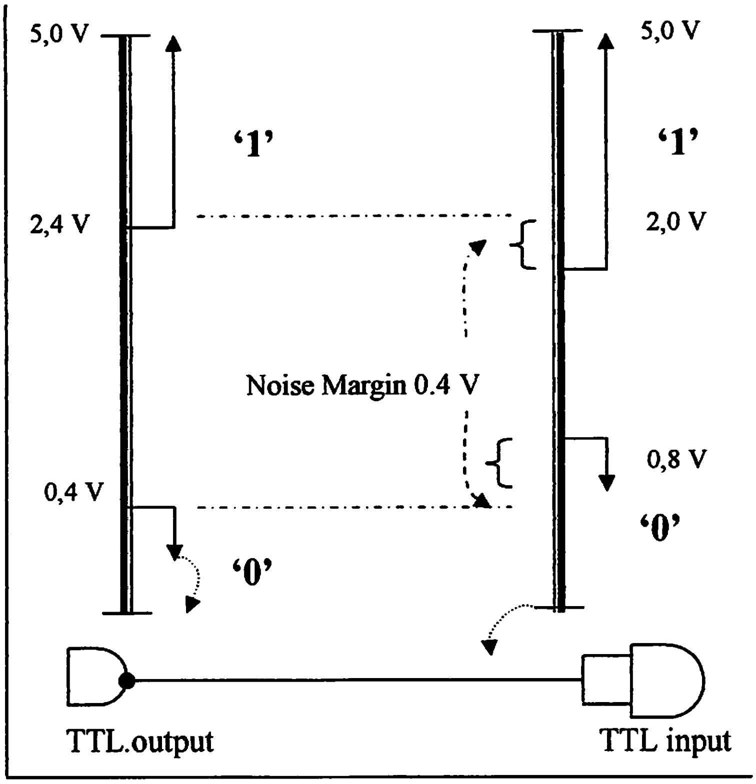
Sedangkan arus masukan yang dibutuhkan rata-rata sebesar 5mA.

2.3 Handphone

Handphone merupakan salah satu alat digital yang mempunyai kemampuan komunikasi yang tinggi dibanding dengan yang lain terutama untuk komunikasi data. Handphone tidak mempunyai standard komunikasi seperti yang ada pada PC, selain itu ada perbedaan bentuk sinyal yang dihasilkan keduanya sehingga agar bisa berkomunikasi perlu diketahui level tegangan sinyal handphone.

Alat digital pada umumnya bekerja pada level tegangan TTL, yang dibuat atas dasar catu daya +5 Volt. Hal tersebut juga berlaku untuk handphone. Level tegangan sinyal handphone ditunjukkan pada gambar 2.4.

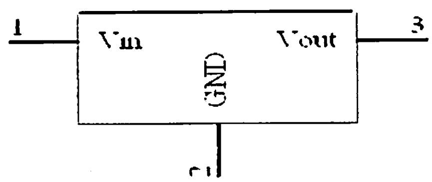
Rangkaian input TTL menganggap tegangan kurang dari 0,8 Volt sebagai level tegangan '0' dan tegangan lebih dari 2.0 Volt dianggap sebagai level tegangan '1', level tegangan ini sering dikatakan sebagai level tegangan TTL. Untuk menjamin output bisa diumpankan ke input dengan baik, tegangan output TTL saat level '0' dijamin lebih rendah dari 0,4 Volt, atau 0,4 lebih rendah dari tagangan yang dituntut oleh input TTL.



GAMBAR 2.5. LEVEL TEGANGAN TTL

Sedangkan tegangan output TTL pada saat level '1' dijamin lebih tinggi dari 2,4 Volt, atau 0,4 Volt lebih tinggi dari tegangan yang dituntut oleh unput TTL.

2.4 IC LM 7805



GAMBAR 2.6.IC LM 7805

IC LM7805 adalah regulator dengan 3 terminal yaitu terminal input, output dan ground. Fungsinya disini adalah sebagai catu daya untuk IC RS MAX232. IC ini dapat diperoleh dalam kemasan TO-3 aluminium. Regulator seri ini sudah cukup mudah digunakan tanpa komponen ekstern tambahan. Tegangan output yang dihasilkan sebesar 5 Volt pada kondisi arus keluar antara 0,5 mA sampai dengan 1 A dengan tegangan input antara 7,5 Volt – 20 Volt.

2.5 Pengantar Pemrograman Visual dengan Delphi

2.5.1. Windows dan Delphi

Program aplokasi berbasis Microsoft Windows mempunyai antar muka grafis yang dikenal sebagai GUI (Graphical User Interface). GUI menjadi tampilan program lebih mudah. Jika dibandingkan dengan tampilan berbasis DOS yang dipenuhi perintah-perintah dal bentuk teks, program dengan GUI lebih mudah digunakan. Hal ini disebabkan antar muka grafis menyajikan obyek dalam tampilan program bagaikan obyek fisik yang sebenarnya. Delphi merupakan salah satu perangkat pemrograman visual berbasis MS Windows yang menggunakan pascal sebagai bahasa dasar.

2.5.2. Pengenalan Windows Delphi 6.0

Sebagai pengantar pengenalan terhadap Delphi, perlu diketahui bagian-bagian berikut :

a. *Window* Utama

   Window ini merupakan fasilitas utama perancangan, sebenarnya *window* hanya berisi MenuBar, tetapi apabila diinginkan dapat diaktifkan ToolBar tertentu supaya ditampilkan dalam *window* ini.

b. Menu Bar

   Merupakan menu utama yang menyajikan item-item menu dalam satu baris. Dari menu inilah, pemrogram dapat memilih proses-proses tertentu untuk mengembangkan suatu program aplikasi.

c. Palet Komponen

   Palet komponen adalah bagian dari *window* utama yang merupakan kumpulan ToolBar dengan nama Standard, Additional, Win32, System dan sebagainya.

d. Window Form

   Window ini merupakan rancangan tampilan antarmuka atau form program aplikasi. Pada saat perancangan, dalam *window* ini dapat di isi dengan berbagai komponen penyusun obyek program.

e. Window Kode

   Window ini merupakan area pengeditan kode-kode dalam bahasa pascal yang terkait dengan satuan rancangan *form.*

f. *Window Objek Tree View*

Window ini terisi semua komponen atau objek yang digunakan dalam suatu *form* aplikasi beserta properti lengkap untuk masing-masing komponen dan *event* yang digunakan.

g. *Window* Project Manager

Window ini berisi semua file souce code program aplikasi. Dari *window* ini pemrogram dapat melakukan peraturan terhadap *souce code* rancangan program.

2.5.3. Pemrograman Event-Driven & Property

Pemrogram visual dikenal sebagai pemrogram *Event-Driven.* Pengertiannya adalah melakukan pemrograman sebagai respon terhadap berbagai event yang muncul terhadap obyek-obyek penyusun sebuah program. Dalam pemrograman Delphi, yang direspon adalah event tertentu yang memang dirasa harus direspon lebih lanjut. Misalnya didesain suatu tampilan yang terdapat beberapa tombol didalamnya, yang direspon adalah apakah yang akan terjadi jika suatu tombol diklik dengan *mouse. Event* ditangani oleh bagian dari program yang disebut dengan prosedur *event-handler.*

Sebuah komponen penyusun program aplikasi pasti mempunyai sifat yang berbeda dengan komponen lainnya. Sifat yang menjelaskan detail dari suatu komponen atau obyek inilah yang disebut dengan properti. Misalnya suatu tombol yang terdapat dalam *form* pasti mempunyai warna, ukuran,posisi pada tampilan dan sebagainya yang menunjukkan detail tombol tersebut.

2.5.4. Ukuran Langkah Pemrograman Delphi

Pada dasarnya tujuan akhir pemrograman adalah menghasilkan sebuah program aplikasi berbentuk *file* yang dapat dieksekusi (*executable file* yang berekstensi .exe) dari MS-DOS prompt, Windows Expoler atau menu *run* pada MS Windows. Pemrograman harus melakukan serangkaian langkah untuk menyusun komponen-komponen penyusun program yang tersimpan dalam sejumlah *file souce code* yang disebut dengan kompilasi. Urutan langkah pemrograman visual adalah sebagai berikut :

a. Menyiapkan komponen penyusunan *form* atau membuat obyek dalam *form* secara visual. Pengaturan setiap obyek dilakukan secara visual atau dengan mengubah propertinya pada *windows* Object Inspector. Hasil langkah ini adalah sebuah file berekstensi dfm (Delphi form filr). Setiap form akan tersimpan sebagai sebuah file.

b. Mengindentifikasi *event* terhadap obyek-obyek tertentu yang dirasa perlu ditangani dan menyusunnya sebagai suatu kode program yang disebut prosedur *event handler*. Prosedur *event handler* untuk setiap *form*, tersimpan sebagai sebuah *file* unit dengan ekstensi pas (pascal). *File* unit ini dengan sendirinya tersimpan pada saat rancangan *form* disimpan sebagai file dfm.

c. Mengkompilasi program. Ada beberapa cara untuk melakukan proses ini yaitu bisa dilakukan dengan menekan ToolBar Run, bisa juga dilakukan dengan menggunakan menu Run \ Run. Hasil kompilasi ini adalah sebuah *file exe* dengan nama yang sama dengan nama *file* proyek yang dibuat pada saat *source code* pertama kali disimpan. Nama *file* proyek berekstensi dpf (Delphi proyek file).

Mengenai source code perlu ditekankan disini, bahwa *source code* sebuah program aplikasi terdiri dari *file* dfm, *file* pas dan *file* dpr. *File* dpr pasti hanya sebuah, sedangkan *file* lainnya dapat lebih dari sebuah.

BAB III

PERANCANGAN DAN PEMBUATAN SISTEM KRS ONLINE MENGGUNAKAN SMS

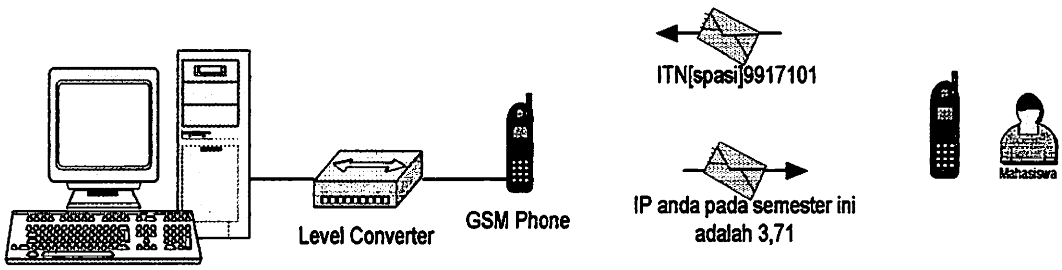
3.1. Pendahuluan

Dalam bab ini akan dibahas mengenai perencangan dan pembuatan handware Dan softwere pada alat yang dibuat yaitu : "Perancangan Dan Pembuatan Sistem KRS Online menggunakan SMS ". Sedangkan hardware yang diperlukan antara lain meliputi Rangkaian *Interface*, HP SIEMENS C35, dan PC.

3.2. Perancangan Hardware

3.2.1. Blok Diagram

Blok diagram sistem perangkat keras (hardware) diperhatian pada gambar 3.1. Sistem pada gambar tersebut digunakan pada HP. Ketika mengirim SMS, SMS akan masuk ke HP. Kemudian SMS akan diinterface ke komputer dan selanjutnya isi SMS tersebut akan membaca kedalam karakter-karakter yang ada pada program dan dibaca. Kumpulan karakter-karakter tersebut dicari untuk dikelompokkan sebagai SMS untuk mengirim data yang diminta. Penyambungan antar handphone dengan komputer dihubungkan ke COM serial komputer menggunakan rangkian interface.



GAMBAR 3.1. DIAGRAM SISTEM PERANCANGAN

Level Conventer adalah perangkat keras yang merupakan sarana penghubung antara ponsel dan komputer dimana Fungsinya sebagai penerjemah antara ponsel dan komputer. Keuntungan adalah fungsinya yang tidak batas karena setiap batas digerakan oleh perangkat lunak (*software*).

GAMBAR 3.2. BLOK PERANCANGAN DAN PEMBUATAN

Pada diagram blok diatas terlihat bahwa secara garis besar system ini terdiri dari beberapa sub sistem, yaitu :

- Komputer

  Berfungsi untuk menjalankan program dengan menggunakan database pada hasil prestasi akademik, dimana pemograman tersebut di jalankan sesuai yang diinginkan atau permintaan.

- Level Converter

  Berfungsi sebagai pengubah tegangan dari handphone ke CPU dihubungkan ke COM1 atau COM2 pada komputer dan akan dibaca oleh Data base, dimana didalam pengiriman data tersebut akan menghasilkan program yang diinginkan pada setiap pengguna *handphone.*

- Handphone

  Berfungsi sebagai penerima sinyal program dari Level Converter yang dikirim oleh komputer agar dapat dihasilkan data yang diinginkan dari sipengirim.

3.2.2. Handphone

Handphone saat ini bukanlah alat yang mewah, hampir semua orang dapat memilikinya. Semakin banyak merk atapun jenis handphone saat ini sering menimbukan salah pada *interface* dengan PC.

Oleh karana itu agar sebuah PC dapat berkomunikasi dengan *Handphone* maka perlu diketahui penggunaan masing-masing pin pada tiap jenis *handphone,* adapun *handphone* yang digunakan pada perangkat ini adalah merek SIEMENS tipe C35 yang mempunyai 8 pin konektor sebagai berikut :

TABEL 3.1. PINOUT KONEKTOR SIEMENS C35

No	Pin	Fungsi
1.	GND	Ground
2.	SELF SERVIVE	Recognition/control battery charger
3.	LOAD	Charging Voltage
4.	BATTERY	Battery
5.	DATA OUT	Data set
6.	DAT IN	Data received
7.	MIC	Micropohone input
8.	AUGND	Ground for eksternal speaker

3.2.3. Perancangan Level Converter

PC mengirimkan dan menerima data melalui port RS-232 dengan level tegangan yang berbeda dengan tegangan logic TTL. Untuk itu dibutuhkan pengubah level tegangan RS-232 ke level TTL, disaat kita mengirimkan dari port serial. Sedangkan yang menghasilkan output TTL, harus dirubah dahulu menjadi level tegangan yang sesuai untuk port RS-232. Hal ini dapat dilakukan oleh IC Maxim 232.

Perangkat level conventer menggunakan IC Maxim 232 dapat dibangun dengan menambahkan lima komponen eksternalnya. Komponen tersebut berupa capasitor C_1,C_2,C_3,C_4,C_5 yang nilainya telah ditentukan pada datasheet sebesar 1µF. Pada IC dersebut terdapat dua buah jalur mengubah level tegangan TTL menjadi level tegangan RS-232, jalur pertama adalah pin 11 (T1 in) yang outputnya pada pin 14 (T1 out) serta jalur kedua adalah pin 10 (T2 in) yang outputnya pada pin 7 (T2 out). Terdapat juga dua buah jalur mengubah level tegangan RS-232 menjadi level TTL,Jalur pertama adalah pin 13 (R1 in) yang outputnya pada pin 12 (R1 out),serta jalur kedua adalah pin 8 (R2in) yang outputnya pada pin 9 (R2in).

3.2.4. Perancangan Perangkat Lunak

Perancangan perangkat lunak didahului dengan perancangan protokol sistem, kemudian dilanjutkan dengan pembuatan rutin-rutin Server yang bertugas sebagai penyedia data dan pengirim paket data sesuai dengan protokol yang direncanakan. pada sisi client dibuat rutin penerima data yang menerima data sesuai protokol yang telah direncanakan.

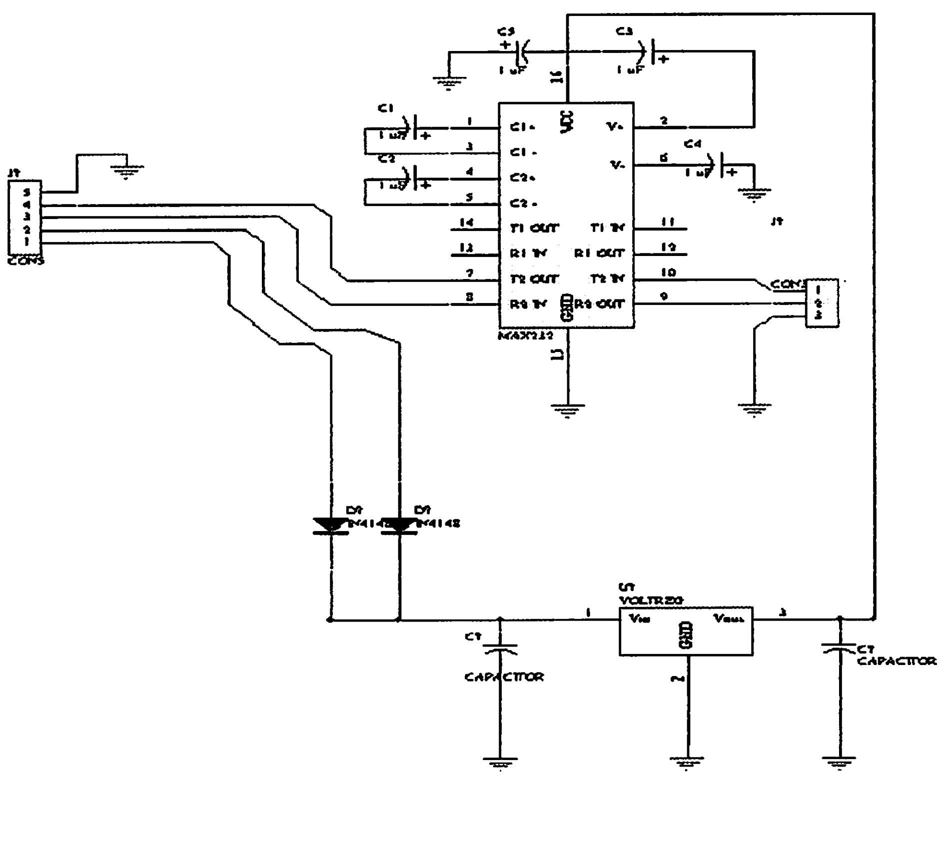
3.2.5. Perancangan Protokol

Untuk mendukung proses pertukaran data diperlukan kesepakatan pihak menyangkut urutan proses, format data, pengamatan, teknik pencegahan error dan sebagainya. Semua hal tersebut dirangkum menjadi sebuah tata aturan yang disebut protokol dan harus disepakati oleh dua belah pihak.

3.2.6. Interface Handphone Dengan PC

Rangkaian interface adalah perangkat keras yang merupakan sarana penghubung antara *handphone* dan komputer agar bisa berkomunikasi lewat COM PORT.

Untuk menginterfacekan *handphone* SIEMEN C35 dengan PC dapat digunakan skematik rangkaian pada gambar berikut:



GAMBAR 3.3. RANGKAIAN INTERFACE SIEMENS C35 SEBAGAI LEVEL CONVENTER DENGAN PC

Data dari handphone yang masuk dan keluar pada pin T2 Out dan R1 In mempunyai tegangan 0 V untuk level tegangan '0' dan 5 Volt untuk level tegangan '1'. Sedangkan dari port serial pada PC yang keluar dan masuk dari pin Standar Battery mempunyai tegangan 11 Volt untuk level tegangan '0' dan -11 Volt untuk level tegangan '1'. Agar data dari handphone dapat dibaca oleh port serial maka level tegangan *handphone* harus diubah ke level tegangan port serial menggunakan IC RS MAX232. RSMAX232 sendiri membutuhkan tegangan input sebesar 5 Volt untuk menjalankannya. Pada alat ini catu daya untuk RS MAX232 disediakan oleh IC 7805 dimana tegangan inputnya diambil dari tegangan pada data-data yang keluar dari port serial sebesar 11 Volt.

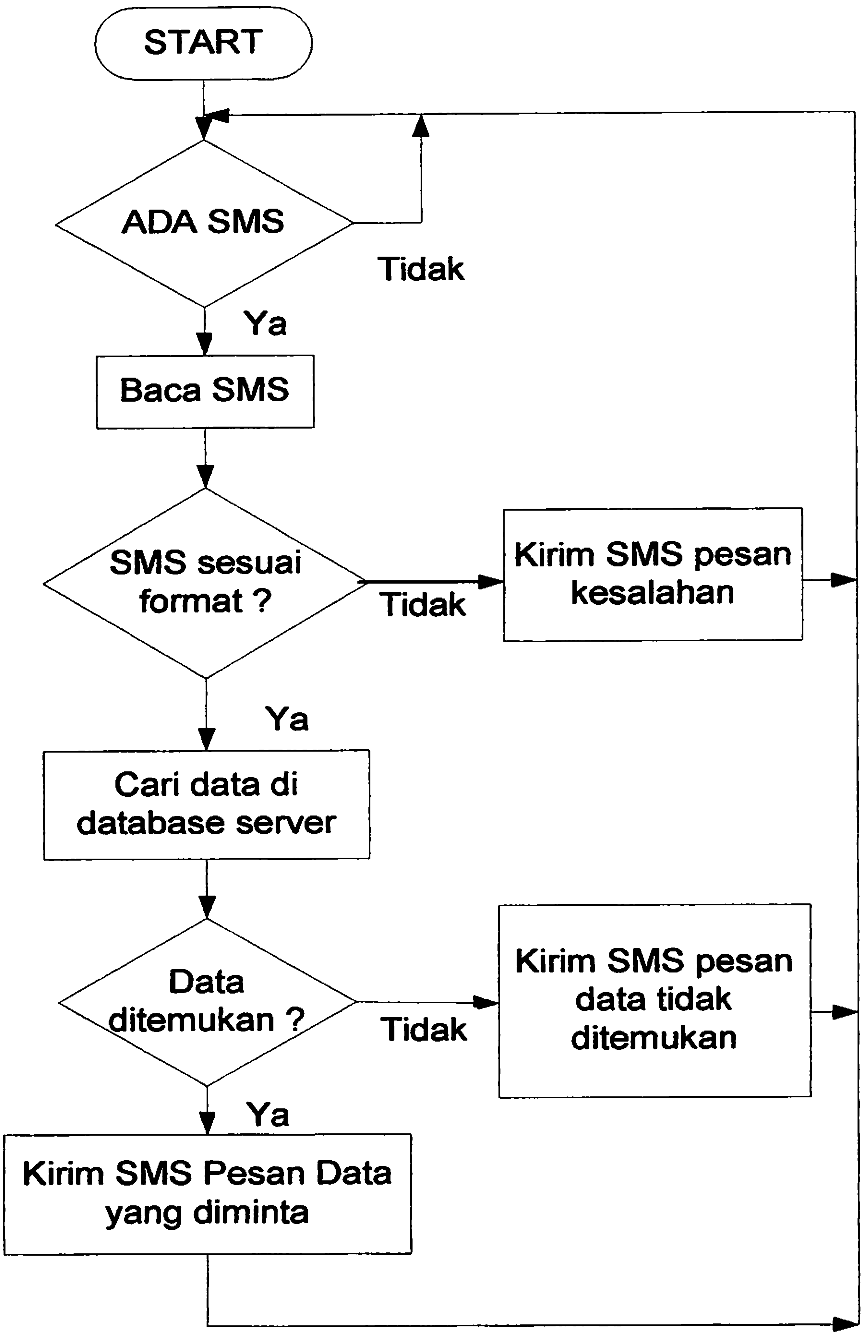
3.2.7. Kabel Data.

Kabel data mempunyai bentuk yang berbeda-beda sesuai dengan jenis *handphone*nya. Tiap *handphone* biasanya mempunyai pasangan kabel data masing-masing. Kabel data yang digunakn alat ini adalah kabel data milik Siemens C35 yang termasuk jenis MBUS yang memiliki 1 buah input/output.

3.3. Perancangan Perangkat Lunak (Softwere)

Program adalah kumpulan dari instruksi untuk mengendalikan atau mengoperasikan sistem perangkat keras (hardware). Adapun langkah-langkah pembuatan program ini adalah sebagai berikut :

1. Membuat diagram alir (flowchart) program yang akan dibuat.
2. Mengubah diagram aliran tersebut ke dalam bahasa pemprogram.
3. Menyusun komponen-komponen yang diinginkan pada form beserta event dan property yang dikenai pada masing-masing komponen sampai didapat tampilan yang sesuai.
4. Mengkompilasikan program yang telah dibuat kedalam memori, sampai menghasilkan program yang paling sesuai.
5. Memasukan program yang telah selesai, dan sistem akan bekerja dengan baik jika perancangan perangkat lunak (software) sesuai dengan perangkat keras (hardware) yang mendukung.



GAMBAR 3.4. FLOWCHART PROGRAM UTAMA

BAB IV

PENGUJIAN ALAT

4.1 Pendahuluan

Pengujian alat ini dimaksudkan untuk mengetahui apakah alat yang dirancang dapat berfungsi dengan baik serta bermanfaat bagi pengguna.

4.2 Peralatan

Peralatan yang digunakan antara lain :

1. Rangkaian Interface dengan level Converter
2. Handphone Siemens C35
3. PC

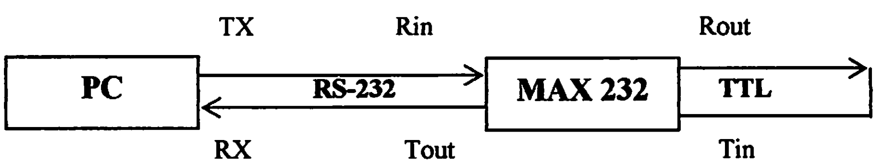
4.3 Pengujian dan Pengukuran

Proses pengujian dan pengambilan data terbagi menjadi 2 tahap pengujian, yaitu pengujian terhadap perangkat keras dan pengujian pada piranti lunak. Pengujian perangkat keras meliputi pengujian terhadap rangkaian level converter . Sedangkan pada bagian piranti lunak diujikan tentang protocol dan jalur yang mengatur keseluruhan kerja system.

4.3.1 Pengujian Perangkat Keras

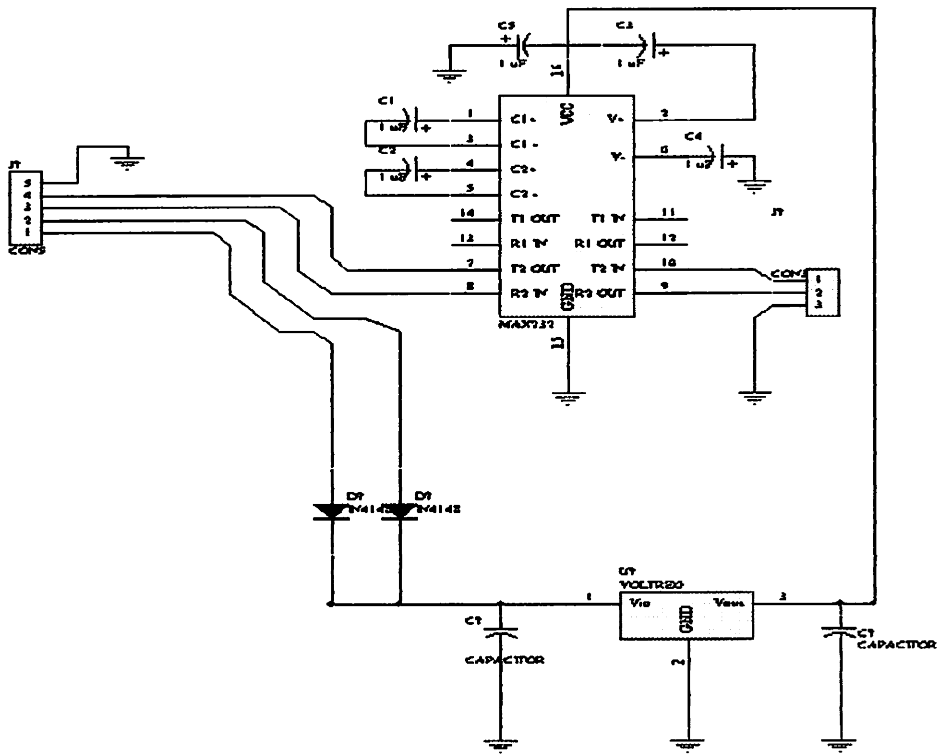
4.3.1.1 Level Converter

Pengujian rangkaian level converter dilakukan dengan mengumpan baliknya data yang dikirim melalui port RS-232. Data serial yang dikirimkan lewat port serial setelah melewati IC Maxim 232 akan berubah menjadi data dengan level tegangan TTL, data dengan level tegangan TTL ini kemudian dibalikkan kembali melewati IC Maxim 232 sehingga berubah kembali menjadi level tegangan RS-232 untuk kemudian diterima oleh PC.



Gambar 4.1. Metode Pengujian Level Converter

Implementasi pengujian rangkaian Level converter dilakukan dengan menghubungkan pin 3 (Tx) dari port RS-232 dengan pin 8 (R2in) IC Max232. Logika dengan level tegangan RS-232 yang masuk pada pin 8 (R2in) akanberubah menjadi level TTL pada pin 9 (T2out), Kemudian logika TTL yang masuk pada pin 18 (T2in) IC Max232 akan berubah menjadi level tegangan RS-232 pada pin 7 (T2out) IC Max232. Schema dari rangkaian pengujian diperhatikan pada gambar 4.2 berikut :



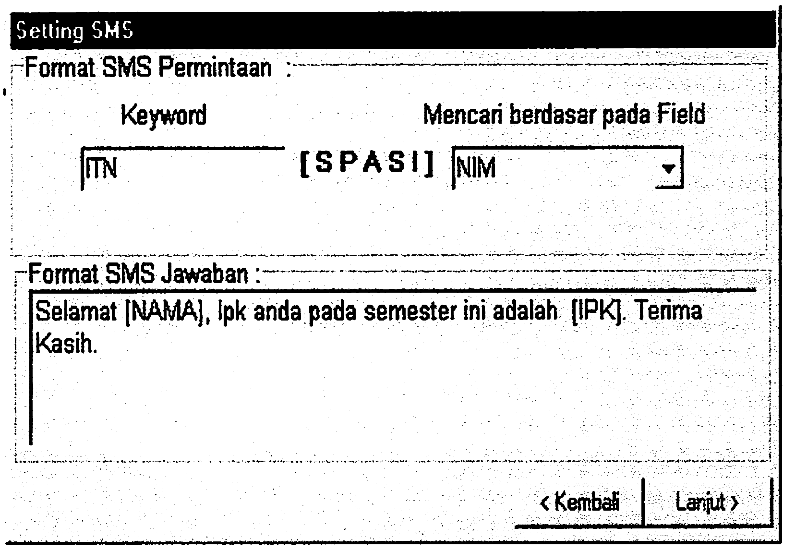
Gambar 4.2.Rangkaian Pengujian Level Converter

Untuk keperluan pengujian rangkaian ini, digunakan perangkat lunak sederhana yang dapat bertugas mengirim data berupa teks dan menampilkan data yang diterimanya. Data dikirim dari salah satu port RS-232 PC dan diterima sendiri oleh port tersebut, setelah diumpan balikkan oleh rangkaian Level converter Max 232.

4.3.2 Pengujian Perangkat Lunak

4.3.2.1 Pengujian Event

Sebelum perangkat lunak dibangun menjadi tampilan akhir, maka sebelumnya diujikan terlebih dahulu prosedur, maupun komponen penting lain yang menyusun dari keseluruhan software. Hal tersebut dilakukan untuk memastikan software telah sesuai dengan protocol dari system yang dibuat.

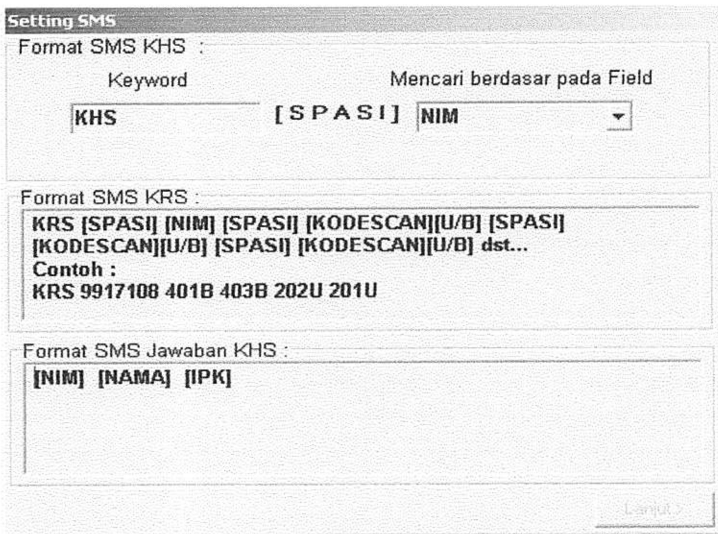


Gambar 4.3. Piranti Lunak Server dan Client Sebelum Permintaan

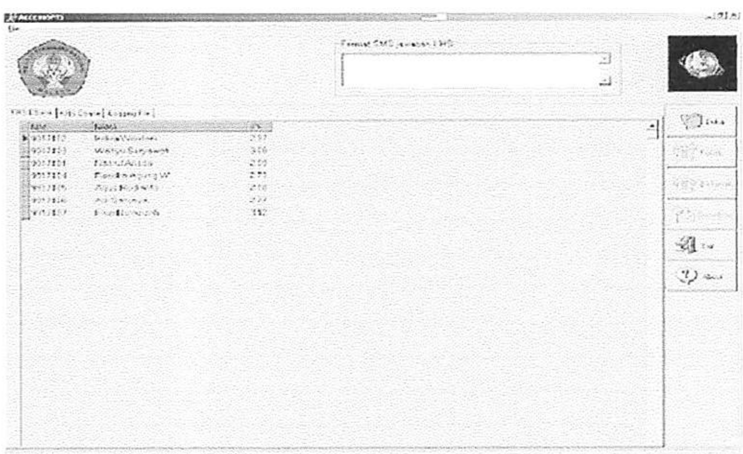
4.3.2.2 Paket Data

Paket data tersusun atas header dan isi data yang dikirim. Header memuat informasi tentang data yang dikirim, meliputi tujuan, nama, IPK, KRS, yang diinginkan dari setiap mahasiswa/i.

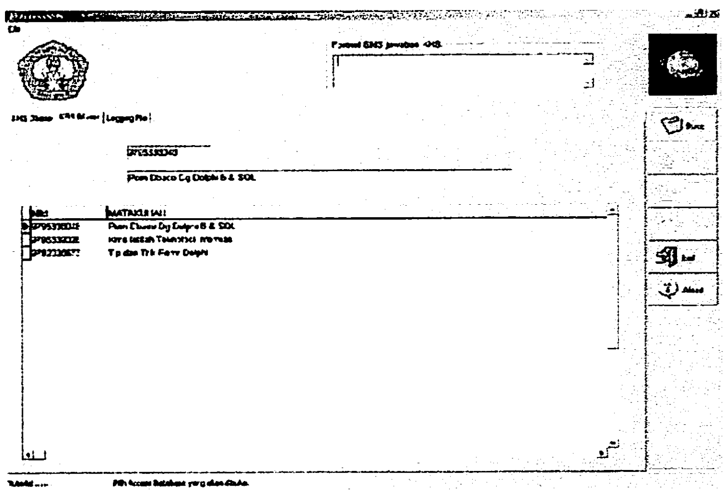
Untuk menguji prosedur penyusun paket data yang telah dibuat, akan dimuat file **"KRS [SPASI] [NIM] [SPASI] [KODE SCAN] [U/B] [SPASI] [KODE SCAN] dst...."** Yang akan dikirim ke nomor tujuan. Setelah paket data disusun, akan ditampilkan kedalam komponen memo untuk diperiksa kesesuaiannya. Dari hasil pengujian yang didapat hasil seperti berikut :



Gambar 4.4. Hasil Pengujian Format Paket Data



Gambar 4.5. Hasil Pengujian Paket Data IPK



Gambar 4.6. Hasil Pengujian Paket Data KRS

Dari hasil pengujian, paket data berupa header dan isi data telah sesuai dengan file yang dibuka. Dengan demikian paket data telah benar dan selanjutnya siap dikirim melalui port RS-232.

4.3.2.3 Pengalamatan

Pengalamatan digunakan untuk mengatur hak penerima data. Halnya Nim yang sesuai dengan informasi yang tercantum pada Header saja yang boleh menerima isi data yang mengikuti header data tersebut. Pengujian pengalamatan dilakukan pengiriman paket data untuk melihat hak penerima software client.

Dari hasil pengujian didapat bahwa, hanya Nim yang dituju yang dapat membaca file yang dikirim. Pengalamatan tujuan berdasar grup angkatan atau jurusan juga mampu hanya mengirimkan paket data kegrup angkatan dan jurusan yang dituju.

4.3.2.4 Progress Bar

Dalam proses menerima sebuah file, dibutuhkan suatu program yang menandakan sampai sejauh mana/seberapa besar sebuah file telah diterima.

Gambar 4.7. Progress Bar Dari Hasil IPK Dan KRS

Nilai maksimum progressbar adalah ukuran file dalam byte. Informasi ukuran file diambil dari header paket data yang dikirim. Berdasarkan hasil pengujian, nilai progressbar akan terus bertambah seiring byte yang diterima. Progressbar akan terkirim setelah menerima byte data jumlah nilai ukuran file yang tercantum pada informasi header.

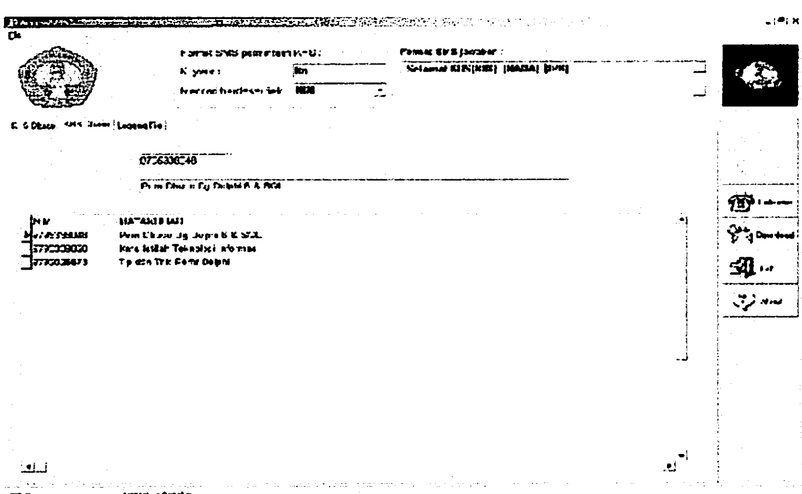
4.3.2.5 Segmentasi

Segmentasi membagi file yang diterima menjadi beberapa segment, sehingga tidak perlu menunggu file terkirim secara kekeluruhan untuk dapat melihat, pengujian dilakukan dengan memperbesar atau memperkecil hasil segment yang dapat diatur melalui pengiriman data.

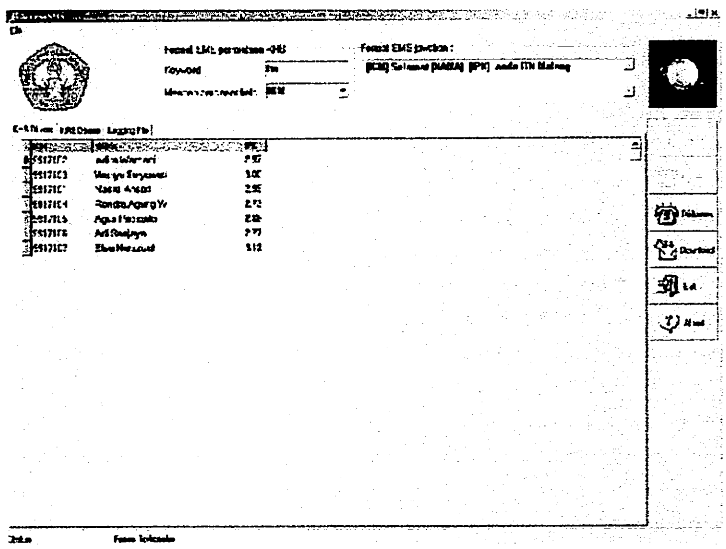
Dari hasil pengujian didapat bahwa semakin kecil data segment, semakin cepat atau semakin sering ditampilkan pada data. Sebaliknya semakin besar permintaan data segment, semakin lambat pengiriman yang ditampilkan kedalam hasil data tersebut. Dari hasil tersebut disimpulkan bahwa segmentesi file telah bekerja sesuai perencanaan atau permintaan.

4.3.2.6 Pengujian Server

Pengujian server dilakukan dengan melakukan rutin-rutin tanggung jawab server, meliputi pengiriman file, pengiriman teks. Tampilan pengujian tersebut seperti dilakukan dalam gambar berikut :



Gambar 4.8. Tampilan Pengujian Pengiriman KRS



Gambar 4.9. Tampilan Pengujian Pengiriman IPK

Dari hasil pengujian, server mampu membangun paket data dengan benar, baik data yang untuk pengiriman file maupun pengiriman pesan text dan pengiriman header data dan isi paket data yang diminta. Ketika tidak sedang mengirimkan data, karakter pandu juga berhasil dikirimkan melalui port RS-232. Dengan demikian disimpulkan piranti lunak server telah bekerja sesuai perencanaan.

BAB V

PENUTUP

5.1. Kesimpulan

Dari pembahasan perencanaan dan pembahasan pemproses SMS dapat diambil beberapa kesimpilan yaitu :

1. Untuk jalur komunikasi data antara handphone dengan PC karena adanya perbedaan level tegangan maka digunakan RS MAX232 sebagai penghubung.
2. Catu daya RS MAX232 sebesar 5 Volt disediakan oleh IC 7805 yang sumber tegangannya diambilkan dari tegangan yang dihasilkan data-data pada port serial, sedangkan kebutuhan arusnya sebesar 5 mA juga terpenuhi oleh IC 7805.
3. Untuk mengirim dan menerima data dari dan ke handphone melalui service center menggunakan perintah-perintah AT Command sebagai bahasa komunikasi.
4. Isi SMS dari handphone yang berupa kode Protocol Data Unit (PDU) harus dikonversi terdahulu menjadi karakter yang dapat dibaca agar dapat ditampilkan ke monitor.
5. Alat ini dapat bekerja dengan baik sesuai fungsinya dapat memproses SMS dari handphone untuk ditampilkan ke monitor serta dapat digunakan untuk menerima dari pengirim SMS.

5.2 Saran

1. Agar system dapat berfungsi harus diperhatikan data-data tentang handphone yang akan digunakan antara lain tipe handphone.

DAFTAR PUSTAKA

1. Khang, Ir. Bustam 2002.Trik *Pemrograman Aplikasi Berbasis SMS.* Jakarta:PT Elex Media Komputindo.
2. Manual Reference AT Command Set SIEMENS Mobile Phone.
3. Pramono, Djoko. *Contoh Penggunaan Rutin-rutin Borland Delphi.*Jakarta:PT Elex Media Komputindo.
4. Wahana Computer. *Panduan Praktis Pemrograman Borland Delphi 6.0.* Yogyakarta:ANDI Offset.
5. Majalah Komputek Tahun 2002

LAMPIRAN

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

LEMBAR BIMBINGAN SKRIPSI

1. Nama	: Aan Noercahyadi
2. NIM	: 9917165
3. Jurusan	: Teknik Elektro S-1
4. Kosentrasi	: Teknik Elektronika
5. Judul Skripsi	: Perancangan Dan Pembuatan Sistem KRS OnLine Menggunakan SMS
6. Tanggal Pengajuan Skripsi	: 8 Januari 2005
7. Selesai Menulis Skripsi	: 14 Maret 2005
8. Dosen Pembimbing I	: Ir. Poerwanto, MT
9. Dosen Pembimbing II	: Ir. Mimien Mustikawati
10. Telah Dievaluasi Dengan Nilai	: 87 (Delapan Puluh Tujuh)

Diperiksa dan Disetujui:

Dosen Pembimbing I

Ir. Poerwanto, MT
NIP. 131 574 847

Dosen Pembimbing II

Ir.Mimien Mustikawati
NIP. 103 0000 352

Mengetahui:
Ketua Jurusan T. Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP. P. 1039500 274

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

FORMULIR PERBAIKAN SKRIPSI

Dari hasil ujian Skripsi Jenjang Strata Satu (S-1) Jurusan Teknik Elektro, Kosentrasi Elektronika yang diselenggarakan pada :

Hari : Senin
Tanggal : 28 Maret 2005

Telah dilakukan Perbaikan Skripsi Oleh :

1. Nama : Aan Noercahyadi
2. NIM : 9917165
3. Jurusan : Teknik Elektro S-1
4. Kosentrasi : Teknik Elektronika
5. Judul Skripsi : Perancangan dan Pembuatan Sistem KRS OnLine Menggunakan SMS

Perbaikan Meliputi :

No	Materi Perbaikan	Keterangan
1	Perancangan Dan Pembuatan KRS On Line Menggunakan SMS	

Mengetahui :

Dosen Pembimbing I

(Ir. Poerwanto, MT)

Dosen Pembimbing II

(Ir.Mimien Mustikawati)

Penguji Pertama

(Ir. Yusuf Ismail Nakoda, MT)

Penguji Kedua

(Ir. Eko Nurcahyo)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : AAH. N
N I M : 9917165.
Perbaikan meliputi :

Utk judul agar dirubah: PERANCANGAN dan PEMBUATAN SISTEM KRS ON LINE MENGGUNAKAN SMS.

Malang, 28 - 03 - 2005

(EKO N)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA :
N I M :
Perbaikan meliputi :

Malang, 200

(______________________)

INSTITUT TEKNOLOGI NASIONAL
JL BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Aan Noercahyadi
Nim : 9917165
Masa Bimbingan : 8-Jan-2005 s/d 11-Jul-2005
Judul Skripsi : Perancangan dan pembuatan sistem informasi akademik dan KRS on line menggunkan SMS

NO	Tanggal	Uraian	Paraf Pembimbing
1.	23-02-04.	Bab I :	
2.		Batasan masalah →	[signature]
3.		teknis / operasional	
4.	Bab II.	Ref. dicantumkan.	[signature]
5.	Bab III	Rancangan hasil	
6.		[illegible] → diperbaiki.	[signature]
7.		Lanjutkan bab IV dst.	[signature]
8.	Bab IV.	Acc.	[signature]
9.	Bab V	Acc.	
10.		Acc Seminar [illegible].	

Malang, 2004
Dosen Pembmbing

[signature]

Ir. Poerwanto, MS

Form. S-4a

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

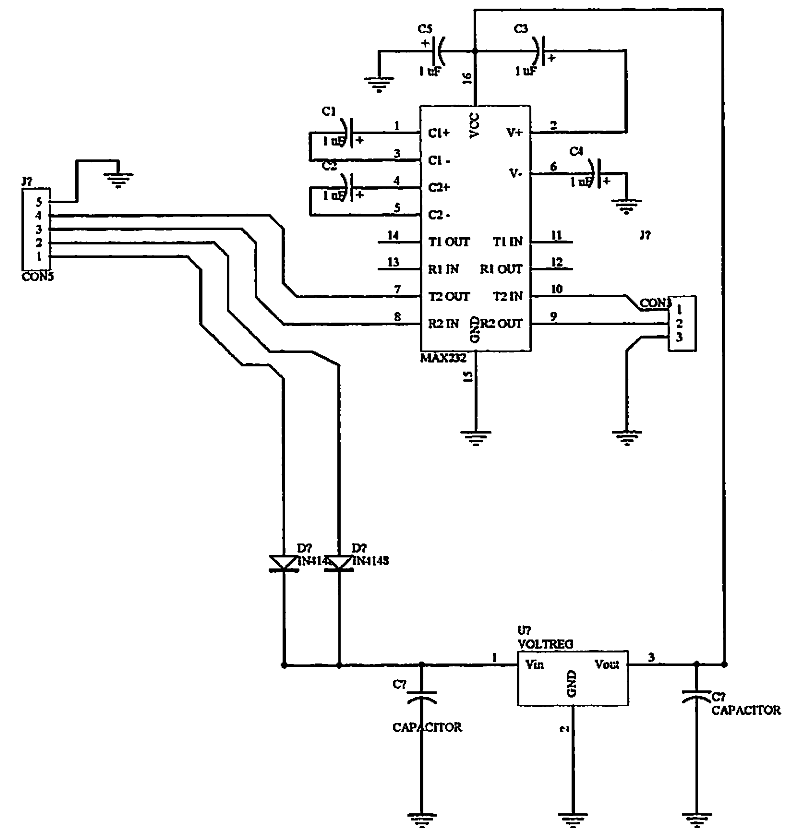
FORMULIR BIMBINGAN SKRIPSI

Nama : Aan Noercahyadi
Nim : 9917165
Masa Bimbingan : 8-Jan-2005 s\d 11-jul-2005
Judul Skripsi : Perancangan dan pembuatan Sistem Informasi Akademik dan KRS OnLine Menggunakan SMS

No	Tanggal	Uraian	Paraf Pembimbing
1.	9/3 '05	- Bab IV, III, II, I direvisi - Gambar rangk. lengkap sertakan	[signature]
2.		Acc seminar	[signature]
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Malang, 2005
Dosen Pembimbing

Ir. Mimien Mustikawati



Title	SIEMENS INTERFACE	
Size B	Number	Revision
Date: 14-Mar-2005	Sheet of	
File: D:\TTp Ade\My Disc (G)\skripsi jadi\TA\Pbrdv\rteyle Addi\RCAHYADI\9917165		

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, ImgList, ToolWin, ComCtrls, Menus, ExtCtrls, StdCtrls, TabNotBk,
  Grids, DBGrids, DB, DBTables, Mask, DBCtrls,Inifiles, Gauges, Spin,
  Buttons, ADODB, VaClasses, VaComm, Animate, GIFCtrl;

type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    Timer1: TTimer;
    Timer2: TTimer;
    Label3: TLabel;
    StatusBar1: TStatusBar;
    ADOConnection1: TADOConnection;
    DataSource1: TDataSource;
    ADOTable1: TADOTable;
    Comport1: TVaComm;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Mbuka: TMenuItem;
    Mkonek: TMenuItem;
    Mdiskonek: TMenuItem;
    Mabout: TMenuItem;
    Mtutup: TMenuItem;
    Panel1: TPanel;
    GroupBox2: TGroupBox;
    erespon: TMemo;
    Panel2: TPanel;
    Panel3: TPanel;
    bopen: TBitBtn;
    bconnect: TBitBtn;
    bdisconnect: TBitBtn;
    bdownload: TBitBtn;
    BAbout: TBitBtn;
    Bexit: TBitBtn;
    Panel4: TPanel;
    TabbedNotebook1: TTabbedNotebook;
    Label2: TLabel;
    Memo1: TMemo;
    estatus: TEdit;
    DBGrid1: TDBGrid;
    Memo2: TMemo;
```

```
    Panel5: TPanel;
    animate1: TRxGIFAnimator;
    Image1: TImage;
    ADOTable2: TADOTable;
    DBGrid2: TDBGrid;
    DataSource2: TDataSource;
    Dbnimkrs: TDBEdit;
    DBmatakuliah: TDBEdit;
    Label1: TLabel;
    procedure Timer1Timer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure bconnectClick(Sender: TObject);
    procedure bopenClick(Sender: TObject);
    procedure Comport1RxChar(Sender: TObject; Count: Integer);
    procedure bdownloadClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure bdisconnectClick(Sender: TObject);
    procedure BexitClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BAboutClick(Sender: TObject);
//   procedure BitBtn8Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  loadsms:boolean=false;
  hit:integer=1;

function pdu_kirim(tujuan,isi:shortstring):shortstring;external 'pdusms.dll';
function unpdu_isi(pdu:shortstring):shortstring;external 'pdusms.dll';
function unpdu_waktu(pdu:shortstring):shortstring;external 'pdusms.dll';
function unpdu_nomor(pdu:shortstring):shortstring;external 'pdusms.dll';


implementation

uses Unit2,Ukorek, Unit3, Unit5;
{$R *.dfm}

procedure TForm1.Timer1Timer(Sender: TObject);
Begin
loadsms:=false;
```

```
 hit:=hit+1;
 if hit>=6 then
  begin
  timer1.enabled:=false;
  hit:=1;
  animate1.animate:=false;
  statusbar1.Panels[1].Text:='Koneksi Terputus';
  estatus.clear;
  MessageDlg('Ponsel Tidak Terhubung dg Komputer, hubungkan ' + chr(13) +
 'ponsel dengan komputer kemudian tekan OK',mtError,[mbOk], 0);
  comport1.writetext('ATE1'+chr(13));
  statusbar1.Panels[1].Text:='Menunggu Koneksi';
  timer1.enabled:=true;
  end;

 if pos('OK',estatus.text)<>0 then
 begin
 timer1.enabled:=false;
 hit:=1;
 animate1.animate:=True;
 statusbar1.Panels[1].Text:='Ponsel Terkoneksi';
 memo1.clear;
 loadsms:=true;
 comport1.writetext('at+CMGL=0'+ chr(13));
 end;

 if pos('ERROR',estatus.text)<>0 then
 begin
  timer1.enabled:=false;
  hit:=1;
  animate1.animate:=false;
  statusbar1.Panels[1].Text:='Koneksi Terputus';
  estatus.clear;
  comport1.writetext('ATE1'+chr(13));
  statusbar1.Panels[1].Text:='Menunggu Koneksi';
  timer1.enabled:=true;
 end;

if loadsms=false then comport1.writetext('ATE1'+chr(13));
estatus.clear;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
timer2.enabled:=false;
timer1.enabled:=true;
```

```
end;

procedure TForm1.bconnectClick(Sender: TObject);
begin
 mkonek.enabled:=true;
 bconnect.enabled:=false;
 statusbar1.Panels[0].Text:='Status.....';
 statusbar1.Panels[1].Text:='Menunggu koneksi';
 bdisconnect.Enabled:=true;
 estatus.clear;
 loadsms:=false;
 comport1.open;
 comport1.SetDTR(true);
 bdownload.Enabled:=true;
 timer1.enabled:=true;
end;

procedure TForm1.bopenClick(Sender: TObject);
begin
form3.show;
end;


procedure TForm1.Comport1RxChar(Sender: TObject; Count: Integer);
var x:integer;
   mkorek:tkorek;
begin
 if loadsms=false then
 begin
 estatus.Text := estatus.text + ComPort1.ReadText;
 end;

 if loadsms=true then
 begin
 Memo1.Text := Memo1.Text + ComPort1.ReadText;
 x:=memo1.lines.capacity;
  if pos('OK',memo1.lines[x-1])<>0 then
  begin         //awal koreksi
  loadsms:=false;
  mkorek:=tkorek.create(false);
  end;
end;
end;

procedure TForm1.bdownloadClick(Sender: TObject);
begin
```

```
timer2.Enabled:=false;
timer1.enabled:=true;
end;


procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var konfirm:word;
begin
 konfirm:=MessageDlg('Apakah aplikasi diakhiri
?',mtconfirmation,[mbYes,mbNo], 0);
 if konfirm=mryes then
 begin
 action:=cafree;
 memo2.Lines.SaveToFile('smslog.txt');
 end else
 action:=canone;
end;


procedure TForm1.bdisconnectClick(Sender: TObject);
begin
mdiskonek.enabled:=true;
bdisconnect.Enabled:=false;
bdownload.enabled:=false;
timer2.Enabled:=false;
timer1.enabled:=false;
comport1.close;
animate1.animate:=false;
bconnect.enabled:=true;
statusbar1.Panels[0].Text:='Status.....';
statusbar1.Panels[1].Text:='Koneksi ponsel dihentikan. Tekan "Konek" untuk
terkoneksi';
end;

procedure TForm1.BexitClick(Sender: TObject);
begin
close;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
opendialog1.initialdir:=extractfilepath(Application.exename);
if fileexists('smslog.txt')then memo2.Lines.LoadFromFile('smslog.txt');
end;

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

```
begin
form5.show;
end;

end.
```

4323; Rev 10; 8/01

MAXIM

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

General Description

ıe MAX220–MAX249 family of line drivers/receivers is tended for all EIA/TIA-232E and V.28/V.24 communica-ıns interfaces, particularly applications where ±12V is ɔt available.

ıese parts are especially useful in battery-powered sys-ms, since their low-power shutdown mode reduces ɔwer dissipation to less than 5μW. The MAX225, AX233, MAX235, and MAX245/MAX246/MAX247 use) external components and are recommended for appli-ıtions where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop 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

art Iumber	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
IAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
IAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
IAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
IAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
IAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
IAX231 (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
IAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
IAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
IAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
IAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
IAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
IAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
IAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
AX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
AX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
AX239 (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
IAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
IAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
IAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
IAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
IAX244	+5	8/10	4	1.0	No	—	120	High slew rate
IAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
IAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
IAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
IAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
IAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

·5V-Powered, Multichannel RS-232 ɔrivers/Receivers

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

pply Voltage (V_{CC})..-0.3V to +6V
ɔut Voltages
IN..-0.3V to (V_{CC} - 0.3V)
IN (Except MAX220) ....±30V
IN (MAX220)..±25V
OUT (Except MAX220) (Note 1) ....±15V
OUT (MAX220)..±13.2V
ıtput Voltages
OUT...±15V
OUT ....-0.3V to (V_{CC} + 0.3V)
iver/Receiver Output Short Circuited to GND.........Continuous
ɔntinuous Power Dissipation (T_A = +70°C)
6-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
8-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

ɔte 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
ɔte 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

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

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

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
ꞮS-232 TRANSMITTERS						
Ɔutput Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±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				200	116	kb/s
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 $R2_{IN}$	0.8	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
	Shrinking V_{OUT} = V_{CC}		10	30		

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

CC = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

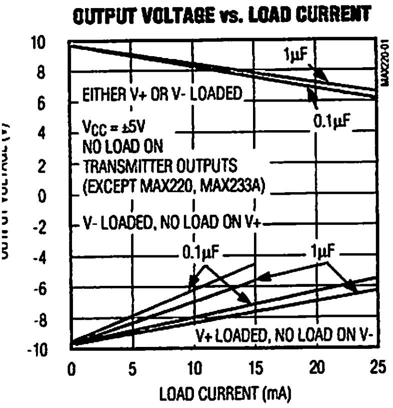
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{N}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{N}$ Input Threshold High	MAX242		2.0	1.4		V
)perating Supply Voltage			4.5		5.5	V
'CC Supply Current ($\overline{SHDN}$ = V_{CC}), igures 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		
;hutdown 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
'ransition 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	
ransmitter Propagation Delay LL to RS-232 (normal operation), igure 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	
leceiver Propagation Delay S-232 to TLL (normal operation), igure 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	
leceiver Propagation Delay S-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
leceiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
leceiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
ransmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
ransmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
ransmitter + 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		

lote 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

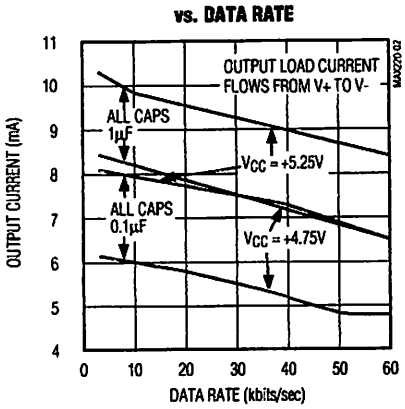
Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243

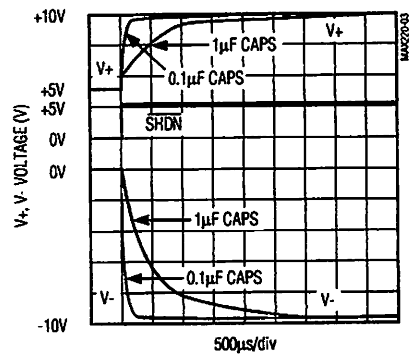
OUTPUT VOLTAGE vs. LOAD CURRENT



AVAILABLE OUTPUT CURRENT vs. DATA RATE



MAX222/MAX242 ON-TIME EXITING SHUTDOWN



BSOLUTE MAXIMUM RATINGS—MAX223/MAX230–MAX241

'CC .... -0.3V to +6V
'+ .... (V_{CC} - 0.3V) to +14V
'- .... +0.3V to -14V
nput Voltages
T_{IN} .... -0.3V to (V_{CC} + 0.3V)
R_{IN} .... ±30V
)utput Voltages
T_{OUT} .... (V+ + 0.3V) to (V- - 0.3V)
R_{OUT} .... -0.3V to (V_{CC} + 0.3V)
hort-Circuit Duration, T_{OUT} .... Continuous
ontinuous Power Dissipation (T_A = +70°C)
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C).... 800mW
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C).... 842mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C).... 889mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Plastic DIP (derate 9.09mW/°C above +70°C)...... 500mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C)......... 762mW
20-Pin Wide SO (derate 10 00mW/°C above +70°C)....... 800mW
24-Pin Wide SO (derate 11.76mW/°C above +70°C)....... 941mW
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
44-Pin Plastic FP (derate 11.11mW/°C above +70°C)..... 889mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C).......... 727mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C)........ 800mW
20-Pin CERDIP (derate 11.11mW/°C above +70°C)........ 889mW
24-Pin Narrow CERDIP
(derate 12.50mW/°C above +70°C) 1W
24-Pin Sidebraze (derate 20.0mW/°C above +70°C).......... 1.6W
28-Pin SSOP (derate 9.52mW/°C above +70°C)............. 762mW
Operating Temperature Ranges
MAX2 _ _ C _ _ .... 0°C to +70°C
MAX2 _ _ E _ _ .... -40°C to +85°C
MAX2 _ _ M _ _ .... -55°C to +125°C
Storage Temperature Range .... -65°C to +160°C
Lead Temperature (soldering, 10sec) .... +300°C

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

LECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241

AX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, CC = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
)utput Voltage Swing	All transmitter outputs loaded with 3kΩ to ground		±5.0	±7.3		V
/CC Power-Supply Current	No load, T_A = +25°C	MAX232/233		5	10	mA
		MAX223/230/234–238/240/241		7	15	
		MAX231/239		0.4	1	
/+ Power-Supply Current		MAX231		1.8	5	mA
		MAX239		5	15	
ihutdown Supply Current	T_A = +25°C	MAX223		15	50	µA
		MAX230/235/236/240/241		1	10	
nput Logic Threshold Low	T_{IN}; EN, $\overline{SHDN}$ (MAX233); $\overline{EN}$, SHDN (MAX230/235–241)				0.8	V
nput Logic Threshold High	T_{IN}		2.0			V
	EN, $\overline{SHDN}$ (MAX223); $\overline{EN}$, SHDN (MAX230/235/236/240/241)		2.4			
ogic Pull-Up Current	T_{IN} = 0V			1.5	200	µA
eceiver Input Voltage perating Range			-30		30	V

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241 (continued)

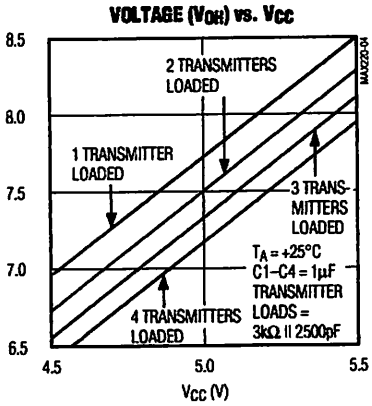
(MAX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, V_{CC} = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS			MIN	TYP	MAX	UNITS
RS-232 Input Threshold Low	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)		0.8	1.2		V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)		0.6	1.5		
RS-232 Input Threshold High	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)			1.7	2.4	V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)			1.5	2.4	
RS-232 Input Hysteresis	V_{CC} = 5V, no hysteresis in shutdown			0.2	0.5	1.0	V
RS-232 Input Resistance	T_A = +25°C, V_{CC} = 5V			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 1.6mA (MAX231/232/233, I_{OUT} = 3.2mA)					0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1mA			3.5	V_{CC} - 0.4		V
TTL/CMOS Output Leakage Current	0V ≤ R_{OUT} ≤ V_{CC}; EN = 0V (MAX223); $\overline{EN}$ = V_{CC} (MAX235–241)				0.05	±10	µA
Receiver Output Enable Time	Normal operation	MAX223			600		ns
		MAX235/236/239/240/241			400		
Receiver Output Disable Time	Normal operation	MAX223			900		ns
		MAX235/236/239/240/241			250		
Propagation Delay	RS-232 IN to TTL/CMOS OUT, C_L = 150pF	Normal operation			0.5	10	µs
		$\overline{SHDN}$ = 0V (MAX223)	t_{PHLS}		4	40	
			t_{PLHS}		6	40	
Transition Region Slew Rate	MAX223/MAX230/MAX234–241, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V			3	5.1	30	V/µs
	MAX231/MAX232/MAX233, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V				4	30	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V			300			Ω
Transmitter Output Short-Circuit Current					±10		mA

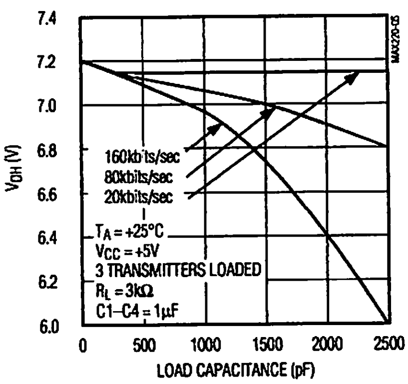
Typical Operating Characteristics

MAX223/MAX230–MAX241

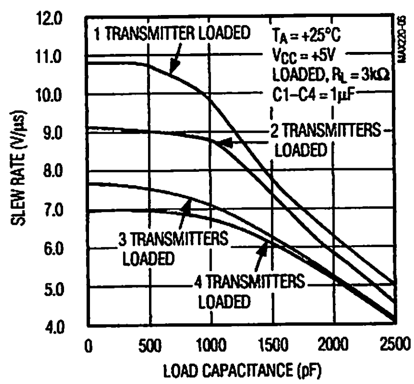
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. V_{CC}



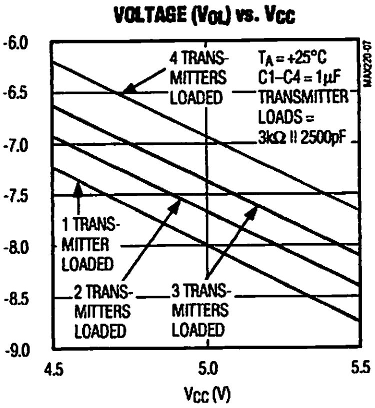
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



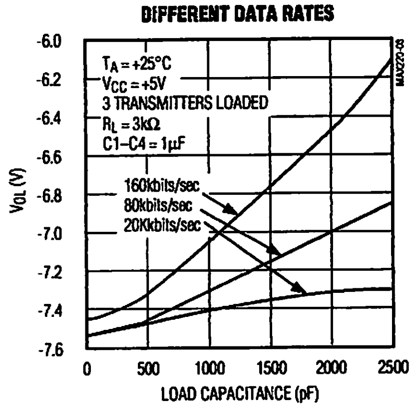
TRANSMITTER SLEW RATE vs. LOAD CAPACITANCE



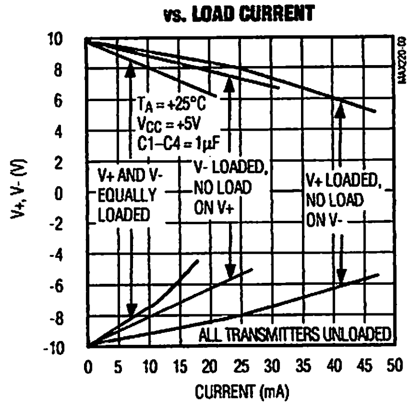
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. V_{CC}



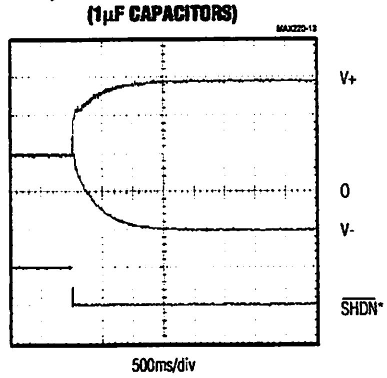
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



TRANSMITTER OUTPUT VOLTAGE (V+, V-) vs. LOAD CURRENT



V+, V- WHEN EXITING SHUTDOWN (1µF CAPACITORS)



*SHUTDOWN POLARITY IS REVERSED FOR NON MAX241 PARTS

BSOLUTE MAXIMUM RATINGS—MAX225/MAX244–MAX249

pply Voltage (V_{CC}) ....-0.3V to +6V
ɔut Voltages
IN, $\overline{ENA}$, $\overline{ENB}$, $\overline{ENR}$, $\overline{ENT}$, $\overline{ENRA}$,
$\overline{NRB}$, $\overline{ENTA}$, $\overline{ENTB}$.................................-0.3V to (V_{CC} + 0.3V)
IN ....±25V
OUT (Note 3)..±15V
IOUT ....-0.3V to (V_{CC} + 0.3V)
ort Circuit (one output at a time)
OUT to GND...Continuous
IOUT to GND...Continuous

Continuous Power Dissipation (T_A = +70°C)
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
40-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ...611mW
44-Pin PLCC (derate 13.33mW/°C above +70°C) 1.07W
Operating Temperature Ranges
MAX225C_ _, MAX24_C_ _ ....0°C to +70°C
MAX225E_ _, MAX24_E_ _ ....-40°C to +85°C
Storage Temperature Range ....-65°C to +160°C
Lead Temperature (soldering,10sec) ....+300°C

ɔte 4: Input voltage measured with transmitter output in a high-impedance state, shutdown, or V_{CC} = 0V.

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

LECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249

1AX225, V_{CC} = 5.0V ±5%; MAX244–MAX249, V_{CC} = +5.0V ±10%, external capacitors C1–C4 = 1μF; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
ꞮS-232 TRANSMITTERS						
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Tables 1a–1d	Normal operation		10	50	μA
		Shutdown		±0.01	±1	
Data Rate	Tables 1a–1d, normal operation			120	64	kbits/sec
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±7.5		V
Output Leakage Current (shutdown)	Tables 1a–1d	$\overline{ENA}$, $\overline{ENB}$, $\overline{ENT}$, $\overline{ENTA}$, $\overline{ENTB}$ = V_{CC}, V_{OUT} = ±15V		±0.01	±25	μA
		V_{CC} = 0V, V_{OUT} = ±15V		±0.01	±25	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V (Note 4)		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±30		mA
ꞮS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±25	V
RS-232 Input Threshold Low	V_{CC} = 5V		0.8	1.3		V
RS-232 Input Threshold High	V_{CC} = 5V			1.8	2.4	V
RS-232 Input Hysteresis	V_{CC} = 5V		0.2	0.5	1.0	V
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
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	Normal operation, outputs disabled, Tables 1a–1d, 0V ≤ V_{OUT} ≤ V_{CC}, $\overline{ENR\_}$ = V_{CC}			±0.05	±0.10	μA

LECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249 (continued)

AX225, V_{CC} = 5.0V ±5%; MAX244–MAX249, V_{CC} = +5.0V ±10%, external capacitors C1–C4 = 1µF; T_A = T_{MIN} to T_{MAX}; unless oth-
vise noted.)

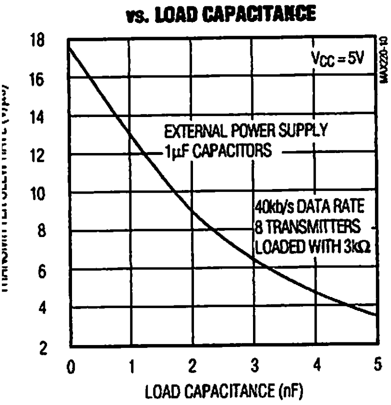
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
OWER SUPPLY AND CONTROL LOGIC						
)perating Supply Voltage		MAX225	4.75		5.25	V
		MAX244–MAX249	4.5		5.5	
'CC Supply Current normal operation)	No load	MAX225		10	20	mA
		MAX244–MAX249		11	30	
	3kΩ loads on all outputs	MAX225		40		
		MAX244–MAX249		57		
;hutdown Supply Current	T_A = +25°C			8	25	µA
	T_A = T_{MIN} to T_{MAX}				50	
;ontrol Input	Leakage current				±1	µA
	Threshold low			1.4	0.8	V
	Threshold high		2.4	1.4		
\C CHARACTERISTICS						
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		5	10	30	V/µs
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}			1.3	3.5	µs
	t_{PLHT}			1.5	3.5	
Receiver Propagation Delay TLL to RS-232 (normal operation), Figure 2	t_{PHLR}			0.6	1.5	µs
	t_{PLHR}			0.6	1.5	
Receiver Propagation Delay TLL to RS-232 (low-power mode), Figure 2	t_{PHLS}			0.6	10	µs
	t_{PLHS}			3.0	10	
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}			350		ns
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}			350		ns
Receiver-Output Enable Time, Figure 3	t_{ER}			100	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}			100	500	ns
Transmitter Enable Time	t_{ET}	MAX246–MAX249 (excludes charge-pump start-up)		5		µs
		MAX225/MAX245–MAX249 (includes charge-pump start-up)		10		ms
Transmitter Disable Time, Figure 4	t_{DT}			100		ns

Note 5: The 300Ω minimum specification complies with EIA/TIA-232E, but the actual resistance when in shutdown mode or V_{CC} = 0V is 10MΩ as is implied by the leakage specification.

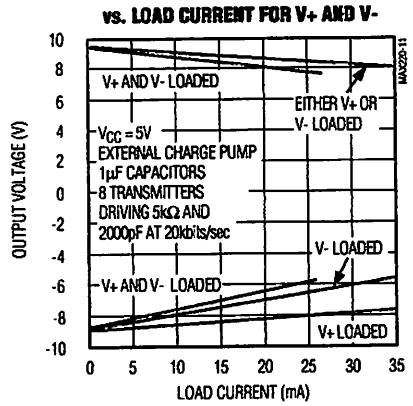
Typical Operating Characteristics

MAX225/MAX244–MAX249

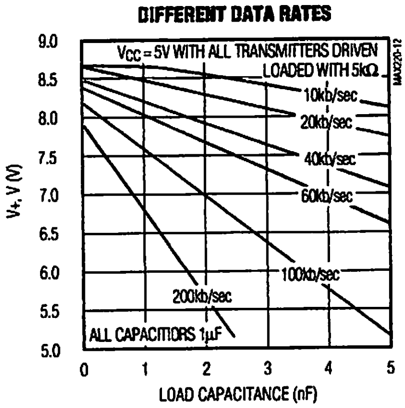
TRANSMITTER SLEW RATE vs. LOAD CAPACITANCE



OUTPUT VOLTAGE vs. LOAD CURRENT FOR V+ AND V-



TRANSMITTER OUTPUT VOLTAGE (V+, V-) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



Annex A (Normative): AT Command Set for Mobile Message Services

A.1. Introduction

This document specifies a common AT command set for messaging services, which meet the following overall setup (see figure 1): a terminal adaptor (TA) is controlling the message services of a mobile equipment (ME) on the basis of AT commands sent from a terminal equipment (TE). ME handles all actual receiving and sending of the messages over communications network. Messages can be stored in memories with a connection to the TA or ME block. Messages are characterised as point to point messages (normal user to user operation) and broadcast messages (teletext type messages provided by the network). The physical connection and protocol between TE and TA and between TA and ME are beyond the scope of this specification. TA may also be an integrated part of the TE or the ME.

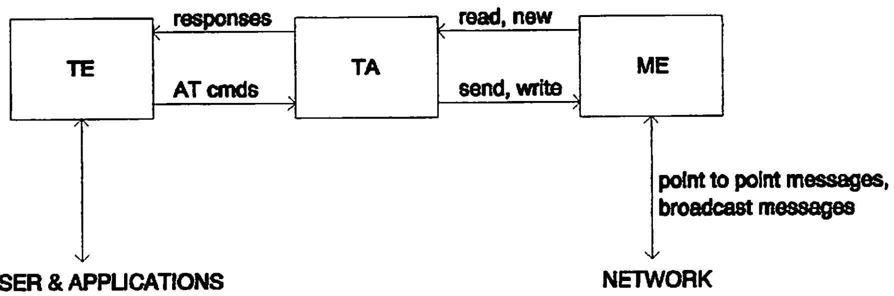


Figure 1: Setup

Due to differences in messaging services all commands cannot be precisely specified at the parameter level in this document. This means that additional specifications are needed for different services. At minimum these service specific documents must specify the exact formats of command parameters stated here to be service specific.

Command set specified here is planned to be used in the following manner. Mandatory command +CSMS is used to list all messaging services supported by the TA and then select one of them. This returns the message types supported by the ME (mobile terminated point to point messages, mobile originated point to point messages and broadcast messages). Now the TE is aware of the service available.

Second step is to choose the preferred memory storage to be used with all point to point message operation. This is done with mandatory command +CPMS. Special memory for broadcast messages handles their storing.

This command set supports two data formats to represent the messages: PDU (protocol data unit) and text format. PDU format has all data associated with the message in one 8 bit binary format parameter and text mode has separate parameters for different message parameters. Service specific documentation must specify the exact format of these. Hence the third step is to use mandatory command +CMGF to list available formats and select the one to be used with send, list, read and write commands.

Different kinds of messages received from network can be not only stored to a memory, but also forwarded to the TE. Fourth optional step is to use command +CNMI to select which messages to forward to the TE using unsolicited result codes.

Now the system should be ready for message handling commands +CMGS (sending from TE), +CMSS (sending directly from storage), +CMGL (listing of several messages in storage), +CMGR (reading one message in storage), +CMGW (writing to storage) and +CMGD (deleting from storage). The syntax or response of some of these commands is dependent on the message format (PDU or text). In text mode the PDU is built up (in the TA or in the ME) before sending to the network or writing to a storage.

Last three commands are probably common for many services and hence included here. Services may have a service centre (SC) through which mobile originated messages are routed and command +CSCA can be used to select the address for this centre. Many services must include a set of additional parameters to be used when messages are sent or stored in the text mode. These and any other settings given in service specific documentations can be saved and restored from a specific memory using commands +CSAS and +CRES.

The AT command syntax in this document relies on standards:

- TIA/EIA-602 Data Transmission Systems and Equipment - Serial Asynchronous Automatic Dialing and Control

- TIA/EIA-615	Data Transmission Systems and Equipment - Serial Asynchronous Automatic Dialing and Control - Extended Command Syntax

Some elements used in the following sections:

<CR>	carriage return character, which value is specified in register S3
<DLE>	IA5 (international alphabet 5; ITU-T T.50) character 16
<ETX>	IA5 character 3
<LF>	linefeed character, which value is specified in register S4
ERROR	if verbose result codes are enabled with command V1 and command's parameter values are not accepted by the TA, result <CR><LF>ERROR<CR><LF> is sent to the TE; if numeric result codes are enabled with command V0, result 4<CR> is sent instead
OK	if verbose result codes are enabled with command V1 and all commands in command line has been performed successfully, result <CR><LF>OK<CR><LF> is sent to the TE; if numeric result codes are enabled with command V0, result 0<CR> is sent instead
<...>	name enclosed in angle brackets is a syntactical element; bracket themselves don't appear in the command line
[...]	optional part of command or result code is enclosed in square brackets; brackets themselves don't appear in the command line; if command parameter is left out, it equals to its previous value
underline	underlined defined parameter value is the recommended default setting of this parameter (this value should be used when parameter isn't given in command line)

NOTE 1: The procedure to give PDUs (in PDU mode) and message bodies (in text mode) after command line isn't standardized syntax.

NOTE 2: Including double quote (IA5 34) and backslash (IA5 92) characters inside a string shall be done with the procedure specified in TIA-615.

NOTE 3: <DLE><ETX> method used to end PDU and RAW type message bodies is a common method for facsimile transmission using AT commands. Single <DLE> characters inside body are doubled and double <DLE> characters are replaced with <DLE><SUB> (IA5 26).

A.2. Select Message Service +CSMS

Parameter Command Syntax

Command	Possible response(s)
+CSMS=<service>	+CSMS: <mt>,<mo>,<bm> +CMS ERROR: <err> ERROR
+CSMS?	+CSMS: <service>,<mt>,<mo>,<bm>
+CSMS=?	+CSMS: (list of supported <service>'s)

Description

Set command selects messaging service <service>. It returns the types of messages supported by the ME: <mt> for mobile terminated messages, <mo> for mobile originated messages and <bm> for broadcast type messages. If chosen service isn't supported by the ME (but is supported by the TA), +CMS ERROR: <err> shall be returned. See chapter A.15 for a list of <err> values.

Also read command returns supported message types along the current service setting.

Test command returns a list of all services supported by the TA.

Defined Values

<service>:

1 GSM (ETSI Global System for Mobile communications)
2 CDMA (TIA IS-95 digital cellular)
3 TDMA (TIA IS-54 digital cellular)
...127 other values between 0 and 127 are reserved
128...255 manufacturer specific values

<mt>, <mo>, <bm>:

0 type not supported
1 type supported

Implementation

Mandatory.

A.3. Preferred Message Storage +CPMS

Parameter Command Syntax

Command	Possible response(s)
+CPMS=<storage>	+CPMS: <used>,<total> +CMS ERROR: <err> ERROR
+CPMS?	+CPMS: <storage>,<used>,<total>
+CPMS=?	+CPMS: (list of supported <storage>'s)

Description

Set command selects memory storage <storage>, which is used by 'send from storage', list, read, write and delete commands. Also all incoming messages are stored to the selected storage, unless they are specified to be forwarded directly to the TE (see command New Message Indications +CNMI). Broadcast type messages are stored in special storage regardless of this setting. If storing of incoming message to the preferred storage is not possible, the ME tries to store the message to another storage. If no storage accepts the message, the ME shall reject the message and inform the SC about it or discard the message depending on the service in use. Selecting storage for broadcast messages affects only the list, read and delete commands (received new point to point messages are stored to the point to point message storage which was set before selecting broadcast storage; also 'send from storage' and write commands use that storage). Successful command returns number of messages currently in the storage <used> and total number of messages <total> that may be stored there. If chosen storage isn't appropriate for the ME (but is supported by the TA), +CMS ERROR: <err> shall be returned. See chapter A.15 for a list of possible <err> values.

Test command returns a list of all memory storages supported by the TA.

Defined Values

<storage> values reserved by this document:

"BM"	broadcast message storage; setting affects only the list, read and delete commands
"ME"	ME's message storage
"MT"	any of the storages associated with ME
"SM"	SIM's (subscriber identity module) message storage
"TA"	TA's message storage

<used>, <total>:
integer type value

Implementation

Mandatory (even when only one memory available).

A.4. Message Format +CMGF

Parameter Command Syntax

Command	Possible response(s)
+CMGF=[<mode>,<chset>]	OK ERROR
+CMGF?	+CMGF: <mode>,<chset>
+CMGF=?	+CMGF: (list of supported <mode>'s), (list of supported <chset>'s)

Description

Set command tells the TA, which input and output format of messages to use. <mode> indicates the format of messages used with send, list, read and write commands and unsolicited result codes resulting from received messages. Mode can be either PDU mode (entire message data units used) or text mode (only address and body of the messages used). Text mode also requires parameter <chset>, which informs the character set used in the message body. The characters of the body are then converted to alphabet used in the currently selected message service.

Read command gives current setting. If text mode isn't supported, <chset> field should be empty.

Test command shall return supported modes, data bit counts and character sets in the TA as compound values. If text mode isn't supported, list of supported <chset>'s should be empty parentheses.

Defined Values

<mode>:

0 PDU mode
1 text mode

<chset> values reserved by this document (service specific documents should offer the conversion tables):

"GSM"	GSM character set (GSM 03.40 Annex 2)
"IA5"	international alphabet number 5 (ITU-T T.50)
"IBMPC"	IBM PC character set
"IBMPCDN"	IBM PC D/N character set
"8859-1"	ISO 8859 Latin 1 character set
"8859-2"	ISO 8859 Latin 2 character set
"8859-3"	ISO 8859 Latin 3 character set
"8859-4"	ISO 8859 Latin 4 character set
"8859-5"	ISO 8859 Latin/Cyrillic character set
"8859-6"	ISO 8859 Latin/Arabic character set
"8859-7"	ISO 8859 Latin/Greek character set
"8859-8"	ISO 8859 Latin/Hebrew character set
"8859-9"	ISO 8859 Latin 5 character set
"8859-10"	ISO 8859 Latin 6 character set
"RAW"	no conversion

Implementation

Mandatory (even when only one mode implemented).

A.5. New Message Indications to TE +CNMI

Parameter Command Syntax

Command	Possible response(s)
+CNMI=[<mode>,<mt>,<bm>,<ds>,<bfr>]	OK +CMS ERROR: <err> ERROR
+CNMI?	+CNMI: <mode>,<mt>,<bm>,<ds>,<bfr>
+CNMI=?	+CNMI: (list of supported <mode>'s), (list of supported <mt>'s), (list of supported <bm>'s), (list of supported <ds>'s), (list of supported <bfr>'s)

Description

Set command selects the procedure, how receiving of new messages is indicated to the TE. <mode> controls the processing of unsolicited result codes specified within this command, <mt> sets the result code indication routing for mobile terminated point to point messages, <bm> for broadcast messages and <ds> for delivery status reports offered to mobile originated messages. <bfr> defines the handling method for buffered result codes when <mode> 1, 2 or 3 is enabled. If ME doesn't support requested item (although TA does), +CMS ERROR: <err> is returned. See chapter A.15 for a list of <err> values.

Test command gives the settings supported by the TA as compound values.

NOTE: Command Select Message Service +CSMS should be used to detect ME's support of mobile terminated and broadcast messages.

Defined Values

<mode> (NOTE: setting affects only to unsolicited result codes specified within this command):

0 buffer unsolicited result codes in the TA; if TA's result code buffer is full, codes can be buffered in some other place or be discarded
1 discard unsolicited result codes when TA-TE link is reserved (e.g. in on-line data mode); otherwise forward them directly to the TE
2 buffer unsolicited result codes in the TA when TA-TE link is reserved (e.g. in on-line data mode) and flush them to the TE after reservation; otherwise forward them directly to the TE
3 forward unsolicited result codes directly to the TE; TA-TE link specific inband technique used to embed result codes and data when TA is in on-line data mode

<mt>:

0 no new point to point message indications are routed to the TE

1 indication of received new point to point message is routed to the TE using unsolicited result code +CMT IND: <storage>,<index> (see Result Code Parameter Values below)

2 received new point to point messages are routed directly to the TE using result code +CMT PDU: <pdu> (PDU mode enabled) or +CMT TEXT: <addr>,<length><CR><LF><text>[<CR><LF><prm>,<prm>,...] (text mode enabled) (see Result Code Parameter Values below)

<bm>:

0 no new broadcast message indications are routed to the TE

1 indication of received new broadcast message is routed to the TE using unsolicited result code +CBM IND: <storage>,<index> (see Result Code Parameter Values below)

2 received new broadcast messages are routed directly to the TE using result code +CBM PDU: <pdu> (PDU mode enabled) or +CBM TEXT: <addr>,<length><CR><LF><text>[<CR><LF><prm>,<prm>,...] (text mode enabled) (see Result Code Parameter Values below)

<ds>:

0 no delivery status reports are routed to the TE

1 received delivery status reports are routed to the TE using unsolicited result code +CDS PDU: <pdu> (PDU mode enabled) or +CDS TEXT: <ref>[,<prm>,<prm>,...] (text mode enabled) (see Result Code Parameter Values below)

<bfr>:

0 TA's buffer of unsolicited result codes defined within this command is flushed to the TE when <mode> 1-3 is entered

1 TA's buffer of unsolicited result codes defined within this command is cleared when <mode> 1-3 is entered

Result Code Parameter Values

<storage> where new message is placed; values reserved by this document:

"BM" broadcast message storage

"ME" ME's message storage

"SM" SIM's (subscriber identity module) message storage

"TA" TA's message storage

<index> number where new message is placed in the storage:
integer type; value in the range of location numbers supported by <storage>

<pdu>:
service specific; <DLE><ETX> method is used inside <pdu> to indicate the tail of the PDU

<addr>:
string type; length and format are service specific

<length>:
integer type value indicating the length of the message body <text> (in bytes)

<text>:
service specific message body string (without double quotes); it can contain characters which are in the currently selected character set

<prm>:
type and values are service specific

<ref>:
integer type value identifies the message which is reported; value is service specific

Implementation

Mandatory when any of the new message indications implemented.

A.6. Send Message +CMGS

Action Command Syntax

Command	Possible response(s)
if PDU mode (+CMGF=0): +CMGS<CR><pdu> **if text mode (+CMGF=1):** +CMGS=<addr><CR><text><exitmethod>	**if delivery status reports not supported and sending successful:** OK **if delivery status reports supported and sending successful:** +CMGS: <ref> **otherwise:** +CMS ERROR: <err> ERROR
+CMGS=?	OK

Description

Execution command sends message from a TE to the network. Both PDU and text mode command formats are specified. In text mode message body <text> is sent to given address <addr> and all current service specific settings are used when transmitting the message to the network. <pdu> and <text> are given after the command line execution. In text mode this can be a sort of interactive mode, where backspace can be used to delete last character and carriage returns can be used. A linefeed character may be added after every carriage return. The implementation of this mode is manufacturer specific. Service (and character set) specific <exitmethod> is used to indicate the ending of the message body. If a delivery status report using a message specific reference value is supported by the service, reference value <ref> is returned to the TE on successful message delivery. Value can be used to identify message upon unsolicited delivery status report result code. If sending fails in a network or an ME error, +CMS ERROR: <err> is returned. See chapter A.15 for a list of <err> values. This command should be abortable when network can 'lose' messages.

Defined Values

<pdu>:
service specific; <DLE><ETX> method is used inside <pdu> to indicate the tail of the PDU

<addr>:
string type; length and format are service specific

<text>:
this is the message body string (without double quotes); it can contain characters which are in the currently selected character set

<exitmethod>:
service specific character(s) to indicate the tail of the message body. **NOTE:** If current character set is "RAW", same <DLE><ETX> method as specified for PDU mode shall be applied

<ref>:
integer type; values are service specific

Implementation

Optional.

A.7. Send Message from Storage +CMSS

Action Command Syntax

Command	Possible response(s)
+CMSS=<index>	**if delivery status reports not supported and sending successful:** OK **if delivery status reports supported and sending successful:** +CMSS: <ref> **otherwise:** +CMS ERROR: <err> ERROR
+CMSS=?	OK

Description

Execution command sends message with location value <index> from preferred message storage to the network. If a delivery status report using a message specific reference value is supported by the service, reference value <ref> is returned to the TE on successful message delivery. Value can be used to identify message upon unsolicited delivery status report result code. If sending fails in a network or an ME error, +CMS ERROR: <err> is returned. See chapter A.15 for a list of <err> values. This command should be abortable when network can 'lose' messages.

Defined Values

<index>:
integer type; value in the range of location numbers supported by the preferred memory storage

<ref>:
integer type; values are service specific

Implementation

Optional.

A.8. List Messages +CMGL

Action Command Syntax

Command	Possible response(s)
+CMGL[=<stat>]	**if PDU mode (+CMGF=0) and command successful:** +CMGL: <index>,<stat>,<pdu> [+CMGL: <index>,<stat>,<pdu> ...] **if text mode (+CMGF=1) and command successful:** +CMGL: <index>,<stat>,<addr>,<length><CR><LF> <text><CR><LF>[<prm>,<prm>,...] [<CR><LF>+CMGL: <index>,<stat>,<addr>,<length><CR><LF> <text><CR><LF>[<prm>,<prm>,...] ...] **otherwise:** +CMS ERROR: <err> ERROR
+CMGL=?	+CMGL: (list of supported <stat>'s)

Description

Execution command returns messages with status value <stat> from preferred message storage to the TE. If PDU mode is enabled, service specific message data units <pdu> are returned. If text mode is enabled, three mandatory parameters to be returned are: message's status <stat>, sender's or recipient's address <addr> and message's text part <text>. Also additional service specific parameters <prm> can be returned. If status of the message is 'received unread', status in the storage changes to 'received read'. If listing fails, +CMS ERROR: <err> is returned. See chapter A.15 for <err> values.

Test command shall give a list of all status values supported by the TA.

Defined Values

<index>:
integer type; value in the range of location numbers supported by the preferred memory storage

<stat>:
0 received unread (i.e. new messages)
1 received read
2 stored unsent
3 stored sent
4 all messages

<pdu>:
service specific; <DLE><ETX> method is used inside <pdu> to indicate the tail of the PDU

<addr>:
string type; length and format are service specific

<length>:
integer type value indicating the length of the message body <text> (in bytes)

<text>:
service specific message body string (without double quotes); it can contain characters which are in the currently selected character set

<prm>:
type and values are service specific

Implementation

Optional.

A.9. Read Message +CMGR

Action Command Syntax

Command	Possible response(s)
+CMGR=<index>	**if PDU mode (+CMGF=0) and command successful:** +CMGR: <stat>,<pdu> **if text mode (+CMGF=1) and command successful:** +CMGR: <stat>,<addr>,<length><CR><LF> <text><CR><LF>[<prm>,<prm>,...] **otherwise:** +CMS ERROR: <err> ERROR
+CMGR=?	OK

Description

Execution command returns message with location value <index> from preferred message storage to the TE. If PDU mode is enabled, status of the message and service specific message data unit <pdu> is returned. If text mode is enabled, three mandatory parameters to be returned are: message's status <stat>, string type sender's address <addr> and string type text part of the message <text>. Also additional service specific parameters <prm> can be returned. If status of the message is 'received unread', status in the storage changes to 'received read'. If reading fails, +CMS ERROR: <err> is returned. See chapter A.15 for <err> values.

Defined Values

<index>:
integer type; value in the range of location numbers supported by the preferred memory storage

<stat>:
0 received unread
1 received read
2 stored unsent
3 stored sent

<pdu>:
service specific; <DLE><ETX> method is used inside <pdu> to indicate the tail of the PDU

<addr>:
string type; length and format are service specific

<length>:
integer type value indicating the length of the message body <text> (in bytes)

<text>:
service specific message body string (without double quotes); it can contain characters which are in the currently selected character set

<prm>:
type and values are service specific

Implementation

Optional.

A.10. Write Message to Memory +CMGW

Action Command Syntax

Command	Possible response(s)
if PDU mode (+CMGF=0): +CMGW[=<stat>]<CR><pdu> **if text mode (+CMGF=1):** +CMGW=<addr>[,<stat>]<CR><text><exitmethod>	+CMGW: <index> +CMS ERROR: <err> ERROR
+CMGW=?	OK

Description

Execution command stores message to preferred memory storage. Memory location <index> of the stored message is returned. The contents of message data unit <pdu> is service specific. In text mode, message body <text> is stored with address <addr> and all current service specific settings needed in the network path. <pdu> and <text> are given after the command line execution. In text mode this can be a sort of interactive mode, where backspace can be used to delete last character and carriage returns can be used. A linefeed character may be added after every carriage return. The implementation of this mode is manufacturer specific. Service (and character set) specific <exitmethod> is used to indicate the ending of this interactive mode. By default message's status will be set to 'stored unsent', but parameter <stat> allows also other status values to be given. If writing fails, +CMS ERROR: <err> is returned. See chapter A.15 for <err> values.

Defined Values

<stat>:
0 received unread
1 received read
<u>2</u> stored unsent
3 stored sent

<pdu>:
service specific; <DLE><ETX> method is used inside <pdu> to indicate the tail of the PDU

<addr>:
string type; length and format are service specific

<text>:
service specific message body string (without double quotes); it can contain characters which are in the currently selected character set

<exitmethod>:
service specific character(s) to indicate the tail of the message body. NOTE: If current character set is "RAW", same <DLE><ETX> method as specified for PDU mode shall be applied

<index>:
integer type; value in the range of location numbers supported by the preferred memory storage

Implementation

Optional.

A.11. Delete Message +CMGD

Action Command Syntax

Command	Possible response(s)
+CMGD=<index>	OK +CMS ERROR: <err> ERROR
+CMGD=?	OK

Description

Execution command deletes message from preferred message storage's memory location <index>. If deleting fails, +CMS ERROR: <err> is returned. See chapter A.15 for <err> values.

Defined Values

<index>:
integer type; value in the range of location numbers supported by the preferred memory storage

Implementation

Optional.

A.12. Service Centre Address +CSCA

Parameter Command Syntax

Command	Possible response(s)
+CSCA=<addr>	OK ERROR
+CSCA?	+CSCA: <addr>
+CSCA=?	OK

Description

Set command updates the string type service centre address <addr>, through which mobile originated messages are transmitted (if supported by service). Setting is stored in a place where it can be used when send, send from storage or write commands are used.

Defined Values

<addr>:
service specific

Implementation

Optional.

A.13. Save Settings +CSAS

Action Command Syntax

Command	Possible response(s)
+CSAS[=<profile>]	OK +CMS ERROR: <err> ERROR
+CSAS=?	+CSAS: (list of supported <profile>'s)

Description

Execution command saves active message service settings to a non-volatile memory. A TA can contain several profiles of settings. The location of memories is TA specific and which settings are saved is dependent on the service used. See chapter A.15 for <err> values.

Test command shall display the supported storages for reading and writing of settings.

NOTE: Settings to be saved should be listed in service specific extensions specifications.

Defined Values

<profile>:
0...255 manufacturer specific profile number where settings are to be stored

Implementation

Optional.

A.14. Restore Settings +CRES

Action Command Syntax

Command	Possible response(s)
+CRES[=<profile>]	OK +CMS ERROR: <err> ERROR
+CRES=?	+CRES: (list of supported <profile>'s)

Description

Execution command restores message service settings from non-volatile memory to active memory. A TA can contain several profiles of settings. The location of memories is TA specific and which settings are restored is dependent on the service used. See chapter A.15 for <err> values.

Test command shall display the supported storages for reading and writing of settings.

NOTE: Settings to be restored should be listed in service specific extensions specifications.

Defined Values

<profile>:

<u>0</u>...255	manufacturer specific profile number from where settings are to be restored

Implementation

Optional.

A.15. Message Service Failure Result Code +CMS ERROR

Result +CMS ERROR: <err> indicates an error related to mobile equipment or network. The operation is similar to ERROR result code. None of the following commands in the same command line is executed. Neither ERROR nor OK result code shall be returned.

Defined Values

<err> values used by common messaging commands:

0-127	values reserved for service specific extensions
128	no proper connection to the ME
129	invalid index
130	service reserved
131	operation not allowed
132	operation not supported
133	memory full
134	service centre unknown
135	invalid PDU mode parameter
136	invalid text mode parameter
137	no SIM (or equivalent) inserted
200	unknown error
...255	other values between 128 and 255 are reserved
256...	values over 255 are manufacturer specific

Interfacing the Serial / RS232 Port V5.0

aimer : While every effort has been made to make sure the information in this document is correct, the author can not be liable for amages whatsoever for loss relating to this document. Use this information at your own risk.

Table of Contents

·oduction

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

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

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

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

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

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

rt One : Hardware (PC's)

Hardware Properties

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

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

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

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

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

Serial Pinouts (D25 and D9 Connectors)

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

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

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

Pin Functions

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

Null Modems

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

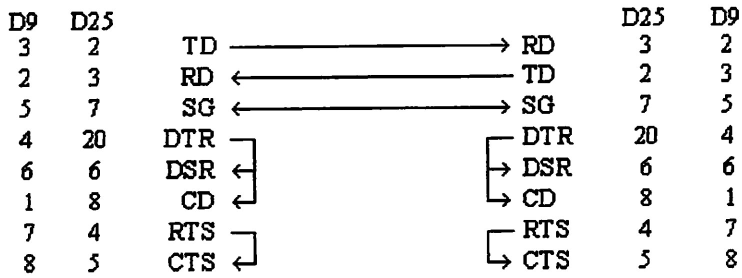


Figure 1 : Null Modem Wiring Diagram

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

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

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

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

LoopBack Plug

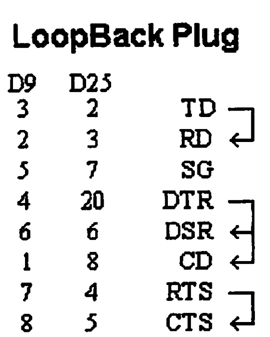


Figure 2 : Loopback Plug Wiring Diagram

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

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

)TE / DCE Speeds

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

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

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

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

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

Flow Control

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

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

Hardware flow control is also known as RTS/CTS flow control. It uses two wires in your serial cable rather than extra characters transmitted in your data lines. Thus hardware flow control will not slow down transmission times like Xon-Xoff does. When the computer wishes to send data it takes active the Request to Send line. If the modem has room for this data, then the modem will reply by taking active the Clear to Send line and the computer starts sending data. If the modem does not have the room then it will not send a Clear to Send.

The UART (8250 and Compatibles)

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

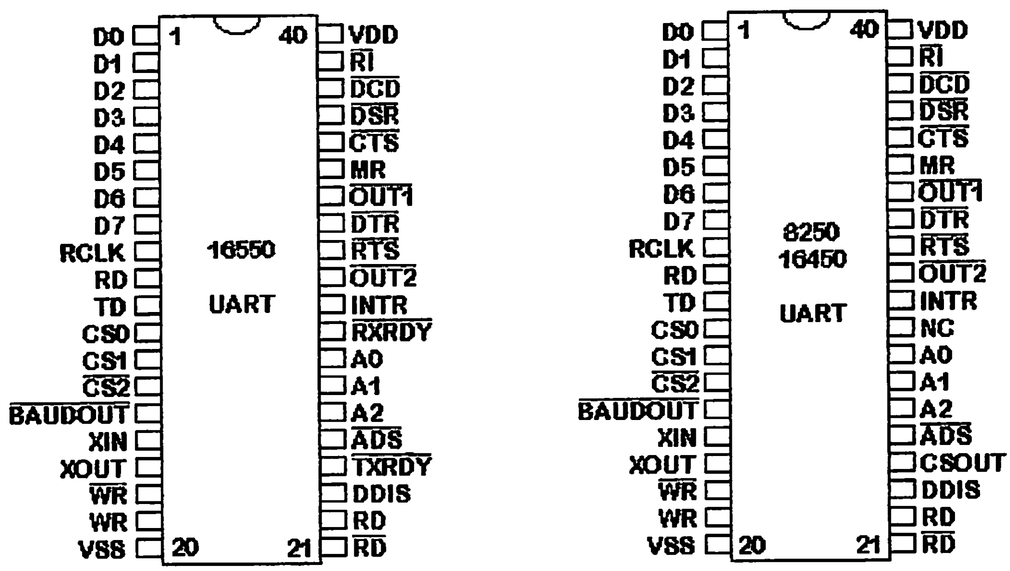


Figure 3 : Pin Diagrams for 16550, 16450 & 8250 UARTs

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

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

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

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

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

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

Table 2 : Pin Assignments for 16550A UART

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

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

Types of UARTS (For PC's)

8250	First UART in this series. It contains no scratch register. The 8250A was an improved version of the 8250 which operates faster on the bus side.
8250A	This UART is faster than the 8250 on the bus side. Looks exactly the same to software than 16450.
8250B	Very similar to that of the 8250 UART.
16450	Used in AT's (Improved bus speed over 8250's). Operates comfortably at 38.4KBPS. Still quite common today.

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

rt Two : Serial Port's Registers (PC's)

Port Addresses & IRQ's

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

Table 3 : Standard Port Addresses

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

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

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

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

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

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

 ptraddr=(unsigned int far *)0x00000400;

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

Table of Registers

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

Table 5 : Table of Registers

)LAB ?

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

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

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

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

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

Table 6 : Table of Commonly Used Baudrate Divisors

nterrupt Enable Register (IER)

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

Table 7 : Interrupt Enable Register

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

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

Interrupt Identification Register (IIR)

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

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

Table 8 : Interrupt Identification Register

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

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

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

First In / First Out Control Register (FCR)

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

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

Table 9 : FIFO Control Register

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

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

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

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

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

Line Control Register (LCR)

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

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

Table 10 : Line Control Register

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

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

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

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

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

Modem Control Register (MCR)

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

Table 11 : Modem Control Register

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

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

:ine Status Register (LSR)

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

Table 12 : Line Status Register

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

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

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

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

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

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

Modem Status Register (MSR)

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

Table 13 : Modem Status Register

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

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

Scratch Register

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

·t 3 : Programming (PC's)

?olling or Interrupt Driven?

When writing a communications program you have two methods available to you. You can poll the UART, to see if any new data is available or you can set up an interrupt handler to remove the data from the UART when it generates a interrupt. Polling the UART is a lot slower method, which is very CPU intensive thus can only have a maximum speed of around 34.8 KBPS before you start losing data. Some newer Pentium Pro's may be able to achieve better rates that this. The other option is using a Interrupt handler, and that's what we have used here. It will very easily support 115.2K BPS, even on low end computers.

Termpoll.c - A sample Comms Program using Polling

```
/* Name        : Sample Comm's Program - Polled Version - termpoll.c     */
/* Written By : Craig Peacock <cpeacock@senet.com.au>                    */
/* Date        : Saturday 22nd February 1997                             */
/*          Copyright 1997 CRAIG PEACOCK <cpeacock@senet.com.au>         */
/*           See http://www.senet.com.au/~cpeacock/serial.htm            */
/*                          For More Information                         */

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

#define PORT1 0x3F8
/* Defines Serial Ports Base Address */
/* COM1 0x3F8                         */
/* COM2 0x2F8                          */
/* COM3 0x3E8                          */
/* COM4 0x2E8                          */

void main(void)
{
 int c;
 int ch;
 outportb(PORT1 + 1 , 0);    /* Turn off interrupts - Port1 */
```

```
/*         PORT 1 - Communication Settings         */

outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
outportb(PORT1 + 0 , 0x03);  /* Set Baud rate - Divisor Latch Low Byte */
                             /* Default 0x03 =  38,400 BPS */
                             /*         0x01 = 115,200 BPS */
                             /*         0x02 =  56,700 BPS */
                             /*         0x06 =  19,200 BPS */
                             /*         0x0C =   9,600 BPS */
                             /*         0x18 =   4,800 BPS */
                             /*         0x30 =   2,400 BPS */
outportb(PORT1 + 1 , 0x00);  /* Set Baud rate - Divisor Latch High Byte */
outportb(PORT1 + 3 , 0x03);  /* 8 Bits, No Parity, 1 Stop Bit */
outportb(PORT1 + 2 , 0xC7);  /* FIFO Control Register */
outportb(PORT1 + 4 , 0x0B);  /* Turn on DTR, RTS, and OUT2 */

printf("\nSample Comm's Program. Press ESC to quit \n");

do { c = inportb(PORT1 + 5);          /* Check to see if char has been */
                                      /* received.                     */
    if (c & 1) {ch = inportb(PORT1); /* If so, then get Char           */
                printf("%c",ch);}     /* Print Char to Screen          */

    if (kbhit()){ch = getch();          /* If key pressed, get Char */
                 outportb(PORT1, ch);} /* Send Char to Serial Port */

   } while (ch !=27); /* Quit when ESC (ASC 27) is pressed */
}
```

Polling the UART should not be dismissed totally. It's a good method for diagnostics. If you have no idea of what address your card is at or what IRQ you are using you can poll the UART at several different addresses to firstly find which port your card is at and which one your modem is attached to. Once you know this information, then you can set up the Interrupt routines for the common IRQs and by enabling one IRQ at a time using the Programmable Interrupt Controller you can find out your IRQ, You don't even need a screw driver!

Buff1024.c - An Interrupt Driven Sample Comms Program

```
/* Name        : Sample Comm's Program - 1024 Byte Buffer - buff1024.c   */
/* Written By : Craig Peacock <cpeacock@senet.com.au>                     */
/*      Copyright 1997 CRAIG PEACOCK <cpeacock@senet.com.au>              */
/*        See http://www.senet.com.au/~cpeacock/serial.htm                */
/*                        For More Information                            */

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

#define PORT1 0x2E8  /* Port Address Goes Here */
/* Defines Serial Ports Base Address */
/* COM1 0x3F8                        */
/* COM2 0x2F8                        */
/* COM3 0x3E8                        */
/* COM4 0x2E8                        */
#define INTVECT 0x0B /* Com Port's IRQ here */
                     /* (Must also change PIC setting) */

int bufferin = 0;
int bufferout = 0;
char ch;
char buffer[1025];

void interrupt (*oldport1isr)();

void interrupt PORT1INT()  /* Interrupt Service Routine (ISR) for PORT1 */
{
 int c;
 do { c = inportb(PORT1 + 5);
      if (c & 1) {buffer[bufferin] = inportb(PORT1);
```

```
                    bufferin++;
                    if (bufferin == 1024) bufferin = 0;}
      }while (c & 1);
  outportb(0x20,0x20);
}

void main(void)
{
 int c;
 outportb(PORT1 + 1 , 0);         /* Turn off interrupts - Port1 */

 oldport1isr = getvect(INTVECT); /* Save old Interrupt Vector for */
                                 /* later recovery */

 setvect(INTVECT, PORT1INT);     /* Set Interrupt Vector Entry */
                                 /* COM1 - 0x0C */
                                 /* COM2 - 0x0B */
                                 /* COM3 - 0x0C */
                                 /* COM4 - 0x0B */

 /*          PORT 1 - Communication Settings          */

 outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
 outportb(PORT1 + 0 , 0x03);  /* Set Baud rate - Divisor Latch Low Byte */
                              /* Default 0x03 =  38,400 BPS */
                              /*         0x01 = 115,200 BPS */
                              /*         0x02 =  56,700 BPS */
                              /*         0x06 =  19,200 BPS */
                              /*         0x0C =   9,600 BPS */
                              /*         0x18 =   4,800 BPS */
                              /*         0x30 =   2,400 BPS */
 outportb(PORT1 + 1 , 0x00);  /* Set Baud rate - Divisor Latch High Byte */
 outportb(PORT1 + 3 , 0x03);  /* 8 Bits, No Parity, 1 Stop Bit */
 outportb(PORT1 + 2 , 0xC7);  /* FIFO Control Register */
```

```
outportb(PORT1 + 4 , 0x0B);  /* Turn on DTR, RTS, and OUT2 */

outportb(0x21,(inportb(0x21) & 0xF7));  /* Set Programmable Interrupt */
                                        /* Controller */
                                        /* COM1 (IRQ4) - 0xEF  */
                                        /* COM2 (IRQ3) - 0xF7  */
                                        /* COM3 (IRQ4) - 0xEF  */
                                        /* COM4 (IRQ3) - 0xF7  */

outportb(PORT1 + 1 , 0x01);  /* Interrupt when data received */

printf("\nSample Comm's Program. Press ESC to quit \n");

do {

    if (bufferin != bufferout){ch = buffer[bufferout];
                               bufferout++;
                               if (bufferout == 1024) bufferout = 0;
                               printf("%c",ch);}

    if (kbhit()){c = getch();
                 outportb(PORT1, c);}

   } while (c !=27);

outportb(PORT1 + 1 , 0);      /* Turn off interrupts - Port1 */
outportb(0x21,(inportb(0x21) | 0x08));  /* MASK IRQ using PIC */
                                        /* COM1 (IRQ4) - 0x10  */
                                        /* COM2 (IRQ3) - 0x08  */
                                        /* COM3 (IRQ4) - 0x10  */
                                        /* COM4 (IRQ3) - 0x08  */
setvect(INTVECT, oldport1isr); /* Restore old interrupt vector */
}
```

Note: *The source code on the earier pages is not a really good example on how to program but is rather cut down to size giving quick results, and making it easier to understand. Upon executing your communications program, it would be wise to store the status of the UART registers, so that they all can be restored before you quit the program. This is to cause the least upset to other programs which may also be trying to use the communications ports.*

The first step to using interrupts is to work out which interrupt services your serial card. Table 13 shows the base addresses and IRQ's of some standard ports. IRQ's 3 and 4 are the two most commonly used. IRQ 5 and 7 are sometimes used.

Interrupt Vectors

Once we know the IRQ the next step is to find it's interrupt vector or software interrupt as some people may call it. Basically any 8086 processor has a set of 256 interrupt vectors numbered 0 to 255. Each of these vectors contains a 4 byte code which is an address of the Interrupt Service Routine (ISR). Fortunately C being a high level language, takes care of the addresses for us. All we have to know is the actual interrupt vector.

INT (Hex)	IRQ	Common Uses
08	0	System Timer
09	1	Keyboard
0A	2	Redirected
0B	3	Serial Comms. COM2/COM4
0C	4	Serial Comms. COM1/COM3
0D	5	Reserved/Sound Card
0E	6	Floppy Disk Controller
0F	7	Parallel Comms.
70	8	Real Time Clock
71	9	Reserved
72	10	Reserved
73	11	Reserved
74	12	PS/2 Mouse

75	13	Maths Co-Processor
76	14	Hard Disk Drive
77	15	Reserved

Table 14 : Interrupt Vectors (Hardware Only)

The above table shows only the interrupts which are associated with IRQ's. The other 240 are of no interest to us when programming RS-232 type communications.

For example if we were using COM3 which has a IRQ of 4, then the interrupt vector would be 0C in hex. Using C we would set up the vector using the instruction `setvect(0x0C, PORT1INT);` where PORT1INT would lead us to a set of instructions which would service the interrupt.

However before we proceed with that I should say that it is wise to record the old vectors address and then restore that address once the program is finished. This is done using `oldport1isr = getvect(INTVECT);` where oldport1isr is defined using `void interrupt (*oldport1isr)();`

Not only should you store the old vector addresses, but also the configuration the UART was in. Why you Ask? Well it's simple, I wrote a communications program which was fully featured in the chat side of things. It had line buffering, so no body could see my spelling mistakes or how slowly I typed. It included anti-bombing routines and the list goes on. However I couldn't be bothered to program any file transfer protocols such as Zmodem etc into my communications program. Therefore I either had to run my communications program in the background of Telemate using my communications program for chat and everything else it was designed for and using Telemate to download files. Another method was to run, say Smodem as a external protocol to my communications program.

Doing this however would mean that my communications program would override the original speed, parity etc and then when I returned to the original communications program, everything stopped. Therefore by saving the old configuration, you can revert back to it before you hand the UART back over to the other program. Makes sense? However if you don't have any of these programs you can save yourself a few lines of code. This is what we have done here.

Interrupt Service Routine (ISR)

Now, could we be off track just a little? Yes that's right, PORT1INT is the label to our interrupt handler called a Interrupt Service Routine (ISR). You can put just about anything in here you want. However calling some DOS routines can be a problem.

```
void interrupt PORT1INT()
{
 int c;
 do { c = inportb(PORT1 + 5);
      if (c & 1) {
                  buffer[bufferin] = inportb(PORT1);
                  bufferin++;
                  if (bufferin == 1024) bufferin = 0;
                 }
    } while (c & 1);
outportb(0x20,0x20);
}
```

From the example above we check to see if there is a character to receive and if their is we remove it from the UART and place it in a buffer contained in memory. We keep on checking the UART, in case FIFO's are enabled, so we can get all data available at the time of interrupt.

The last line contains the instruction `outportb(0x20,0x20);` which tells the Programmable Interrupt Controller that the interrupt has finished. The Programmable Interrupt Controller (PIC) is what we must go into now. All of the routines above, we have assumed that everything is set up ready to go. That is all the UART's registers are set correctly and that the Programmable Interrupt Controller is set.

The Programmable Interrupt Controller handles hardware interrupts. Most PC's will have two of them located at different addresses. One handles IRQ's 0 to 7 and the other IRQ's 8 to 15. Mainly Serial communications interrupts reside on IRQ's under 7, thus PIC1 is used, which is located at 0020 Hex.

Bit	Disable IRQ	Function
7	IRQ7	Parallel Port
6	IRQ6	Floppy Disk Controller
5	IRQ5	Reserved/Sound Card
4	IRQ4	Serial Port
3	IRQ3	Serial Port
2	IRQ2	PIC2
1	IRQ1	Keyboard
0	IRQ0	System Timer

Table 15 : PIC1 Control Word (0x21)

Multi-Comm ports are getting quite common, thus table 16 includes data for PIC2 which is located at 0xA0. PIC2 is responsible for IRQ's 8 to 15. It operates in exactly the same way than PIC1 except that EOI's (End of Interrupt) goes to port 0xA0 while the disabling (Masking) of IRQ's are done using port 0xA1.

Bit	Disable IRQ	Function
7	IRQ15	Reserved
6	IRQ14	Hard Disk Drive
5	IRQ13	Maths Co-Processor
4	IRQ12	PS/2 Mouse
3	IRQ11	Reserved
2	IRQ10	Reserved

1	IRQ9	IRQ2
0	IRQ8	Real Time Clock

Table 16 : PIC2 Control Word (0xA1)

Most of the PIC's initiation is done by BIOS. All we have to worry about is two instructions. The first one is `outportb(0x21,(inportb(0x21) & 0xEF);` which selects which interrupts we want to Disable (Mask). So if we want to enable IRQ4 we would have to take 0x10 (16) from 0xFF (255) to come up with 0xEF (239). That means we want to disable IRQ's 7,6,5,3,2,1 and 0, thus enabling IRQ 4.

But what happens if one of these IRQs are already enabled and then we come along and disable it? Therefore we input the value of the register and using the & function output the byte back to the register with our changes using the instruction `outportb(0x21,(inportb(0x21) & 0xEF);`. For example if IRQ5 is already enabled before we come along, it will enable both IRQ4 and IRQ5 so we don't make any changes which may affect other programs or TSR's.

The other instruction is `outportb(0x20,0x20);` which signals an end of interrupt to the PIC. You use this command at the end of your interrupt service routine, so that interrupts of a lower priority will be accepted.

UART Configuration

Now we get to the UART settings (Finally)

It's a good idea to turn off the interrupt generation on the UART as the first instruction. Therefore your initialization can't get interrupted by the UART. I've then chosen to set up our interrupt vectors at this point. The next step is to set the speed at which you wish to communicate at. If you remember the process, we have to set bit 7 (The DLAB) of the LCR so we can access the Divisor Latch High and Low Bytes. We have decided to set the speed to 38,400 Bits per second which should be find for 16450's and 16550's. This requires a divisor of 3, thus our divisor latch high byte will be 0x00 and a divisor latch low byte, 0x03.

In today's standards the divisor low latch byte is rarely used but it still pays us to write 0x00 to the register just in case the program before us just happened to set the UART at a very very low speed. BIOS will normally set UARTs at 2400 BPS when the computer is first booted up which still doesn't require the Divisor Latch Low byte.

The next step would be to turn off the Divisor latch access bit so we can get to the Interrupt Enable Register and the receiver/transmitter buffers. What we could do is just write a 0x00 to the register clearing it all, but considering we have to set up our word length, parity as so forth in the line control register we can do this at the same time. We have decided to set up 8 bits, no parity and 1 stop bit which is normally used today. Therefore we write 0x03 to the line control register which will also turn off the DLAB for us saving one more I/O instruction.

The next line of code turns on the FIFO buffers. We have made the trigger level at 14 bytes, thus bits 6 and 7 are on. We have also enabled the FIFO's (bit 0). It's also good practice to clear out the FIFO buffers on initialization. This will remove any rubbish which the last program may of left in the FIFO buffers. Due to the fact that these two bits are self resetting, we don't have to go any further and turn off these bits. If my arithmetic is correct all these bits add up to 0xC7 or 199 for those people which still work in decimal.

Then DTR, RTS and OUT 2 is taken active by the instruction `outportb(PORT1 + 4,0x0B);`. Some cards (Both of Mine) require OUT2 active for interrupt requests thus I'm normally always take it high. All that is left now is to set up our interrupts which has be deliberately left to last as to not interrupt our initialization. Our interrupt handler is only interested in new data being available so we have only set the UART to interrupt when data is received.

Main Routine (Loop)

Now we are left with,

```
do {
    if (bufferin != bufferout){
                                 ch = buffer[bufferout];
                                 bufferout++;
                                 if (bufferout == 1024) bufferout = 0;
                                 printf("%c",ch);
                                }
    if (kbhit()){
                  c = getch();
                  outportb(PORT1, c);
                 }
   } while (c !=27);
```

which keeps repeating until c = 27. This occurs when the ESC key is hit.

The next *if* statement checks to see if a key has been hit. (`kbhit()`) If so, it gets the character using the `getch()` statement and outputs it to the receiver buffer. The UART then transmits the character to the modem. What we have assumed here, is that the person using the Communications Program can't type as fast as the UART can send. However if the program wishes to send something, then a check should be made to see if BIT 5 of the Line Status Register is set before attempting to send a byte to the transmitter register.

*For more information on Interrupts, try **Using Interrupts**,*
http://www.geocities.com/SiliconValley/Bay/8302/interupt.pdf (62k)

Determining the type of UART via software

The type of UART you have installed in your system can be determined without even needing a screwdriver in most cases. As you can see from the Types of UARTs, each UART has minor differences, all we have to do it test these.

The first procedure we do is to set bit 0 to '1' in the FIFO control register. This tries to enable the FIFO buffers. Then we read bits 6 and 7 from the interrupt identification register. If both bits are '1' then the FIFO buffers are enabled. This would mean the UART is a 16550a. If the FIFO's were enabled but not usable then it would be a 16550. If there is no FIFO buffer enabled it is most likely to be a 16450 UART, but could be a 8250, 8250A or 8250B on very old systems.

AT's have a fast bus speed which the 8250 series of UART can't handle to well thus it is very unlikely to be found in any AT. However if you wish to test for them as well you can follow the same test as above to distinguish 16550's or 16550A's from the rest. If no FIFOs are enabled then a possible UART is the 16450, 8250, 8250A or 8250B. Once it is established the it could be one of these four chips, try writing a byte to the scratch register and then read it back and compare the results. If the results match then you must have a scratch register, if they don't you either don't have a scratch register, or it doesn't work to well.

From the descriptions of the UART above if you read back your byte from the scratch register then the UART must be a 16450 or 8250A. (Both have scratch registers) If you don't read back your byte then it's either a 8250 or 8250B.

The 16750 has 64 byte FIFO's, thus the easiest way to test for it's presence is to enable the 64 byte buffer using the FIFO Control Register and then read back the status of the Interrupt Identification Register. However I have never tested this.

t 4 : *Interfacing Devices to RS-232 Ports*

RS-232 Waveforms

So far we have introduced RS-232 Communications in relation to the PC. RS-232 communication is asynchronous. That is a clock signal is not sent with the data. Each word is synchronized using it's start bit, and an internal clock on each side, keeps tabs on the timing.

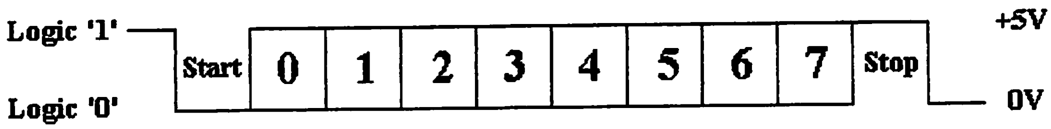


Figure 4 : TTL/CMOS Serial Logic Waveform

The diagram above, shows the expected waveform from the UART when using the common 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1). A transmission starts with a start bit which is (Logic 0). Then each bit is sent down the line, one at a time. The LSB (Least Significant Bit) is sent first. A Stop Bit (Logic 1) is then appended to the signal to make up the transmission.

The diagram, shows the next bit after the Stop Bit to be Logic 0. This must mean another word is following, and this is it's Start Bit. If there is no more data coming then the receive line will stay in it's idle state(logic 1). We have encountered something called a "Break" Signal. This is when the data line is held in a Logic 0 state for a time long enough to send an entire word. Therefore if you don't put the line back into an idle state, then the receiving end will interpret this as a break signal.

The data sent using this method, is said to be *framed*. That is the data is *framed* between a Start and Stop Bit. Should the Stop Bit be received as a Logic 0, then a framing error will occur. This is common, when both sides are communicating at different speeds.

The above diagram is only relevant for the signal immediately at the UART. RS-232 logic levels uses +3 to +25 volts to signify a "Space" (Logic 0) and -3 to -25 volts for a "Mark" (logic 1). Any voltage in between these regions (ie between +3 and -3 Volts) is undefined. Therefore this signal is put through a "RS-232 Level Converter". This is the signal present on the RS-232 Port of your computer, shown below.

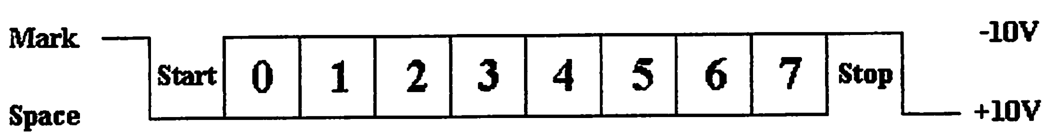


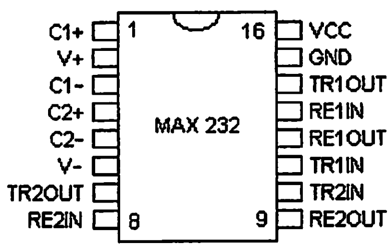
Figure 5 : RS-232 Logic Waveform

The above waveform applies to the Transmit and Receive lines on the RS-232 port. These lines carry serial data, hence the name Serial Port. There are other lines on the RS-232 port which, in essence are *Parallel* lines. These lines (RTS, CTS, DCD, DSR, DTR, RTS and RI) are also at RS-232 Logic Levels.

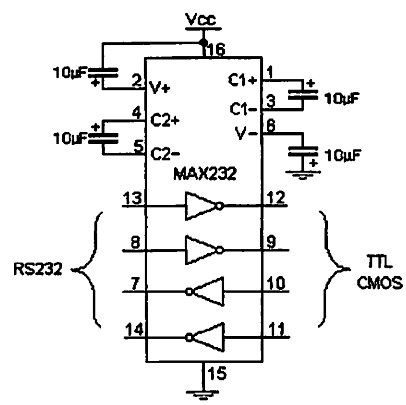
RS-232 Level Converters

Almost all digital devices which we use require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts. As we have already covered, this is done by RS-232 Level Converters.

Two common RS-232 Level Converters are the 1488 RS-232 Driver and the 1489 RS-232 Receiver. Each package contains 4 inverters of the one type, either Drivers or Receivers. The driver requires two supply rails, +7.5 to +15v and -7.5 to -15v. As you could imagine this may pose a problem in many instances where only a single supply of +5V is present. However the advantages of these I.C's are they are cheap.



Above: (Figure 6) Pinouts for the MAX-232, RS-232 Driver/Receiver.



Right: (Figure 7) Typical MAX-232 Circuit.

Another device is the MAX-232. It includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This I.C. also includes two receivers and two transmitters in the same package. This is handy in many cases when you only want to use the Transmit and Receive data Lines. You don't need to use two chips, one for the receive line and one for the transmit. However all this convenience comes at a price, but compared with the price of designing a new power supply it is very cheap.

There are also many variations of these devices. The large value of capacitors are not only bulky, but also expensive. Therefore other devices are available which use smaller capacitors and even some with inbuilt capacitors. *(Note : Some MAX-232's can use 1 micro farad Capacitors).* However the MAX-232 is the most common, and thus we will use this RS-232 Level Converter in our examples.

Making use of the Serial Format

In order to do anything useful with our Serially transmitted data, we must convert it back to Parallel. *(You could connect an LED to the serial port and watch it flash if you really want too, but it's not extremely useful).* This in the past has been done with the use of UART's. However with the popularity of cheap Microcontroller's, these can be more suited to many applications. We will look into the advantages and disadvantages of each method.

8250 and Compatible UARTs

We have already looked at one type of UART, the 8250 and compatibles found in your PC. These devices have configuration registers accessible via the data and address buses which have to be initialized before use. This is not a problem if your device which you are building uses a Microprocessor. However if you are making a stand alone device, how are you going to initialize it?

Most Microprocessors / Microcontrollers these days can be brought with build-in Serial Communication Interfaces (SCI). Therefore there is little need to connect a 40 pin 16550 to, for example a 68HC11 when you can buy one built in. If you are still in love with the Z-80 or 8086 then an 16550 may be option! *(or if you are like myself, the higher chip count the better. After all it looks more complicated and impressive! - But a headache to debug!)*

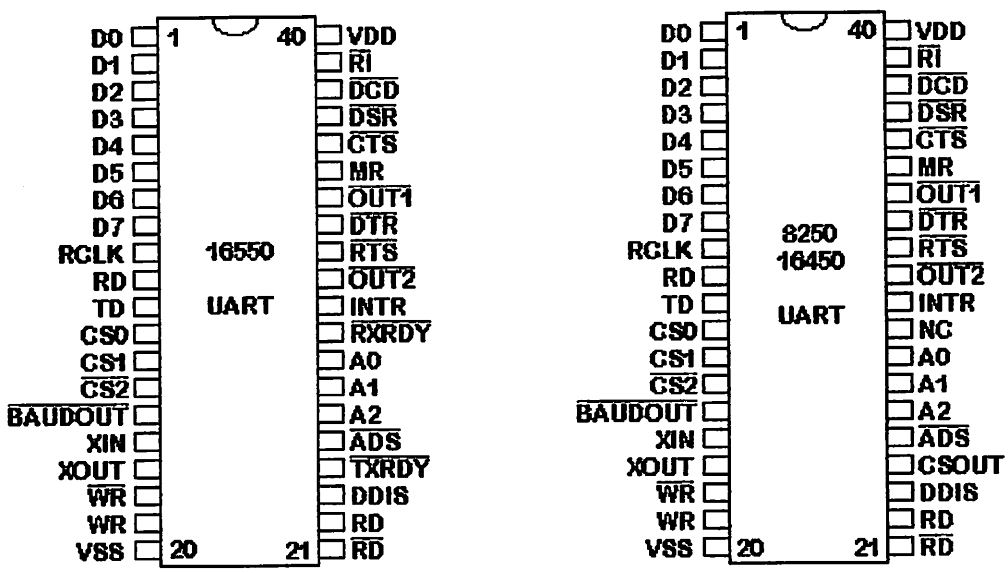


Figure 8 : Pin Diagrams for 16550, 16450 & 8250 UARTs

For more information on the 16550 and compatible UART's see ***The UART (8250 and Compatibles)*** *in Part One of this tutorial or consult the PC16550D data sheet from National Semiconductor (http://www.natsemi.com)*

CDP6402, AY-5-1015 / D36402R-9 etc UARTs

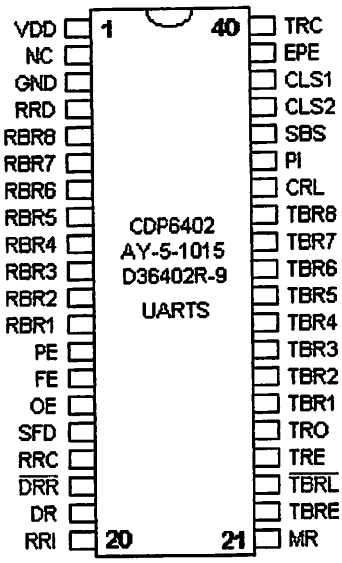


Figure 9 : Pinout of CDP6402

There are UARTs such as the CDP6402, AY-5-1015 / D36402R-9 and compatibles. These differ from the 8250 and compatibles, by the fact that they have separate Receive and Transmit data buses and can be configured by connecting certain pins to various logic levels. These are ideal for applications where you don't have a Microprocessor available. Such an example is if you want to connect a ADC0804 (Analog to Digital Converter) to the UART, or want to connect a LCD Display to the Serial Line. These common devices use a 8 bit parallel data bus.

The CDP6402's *Control Register* is made up of Parity Inhibit (PI), Stop Bit Select (SBS), Character Length Select (CLS1 and 2) and Even Parity Enable (EPE). These inputs can be latched using the Control Register Load (CRL) or if you tie this pin high, changes made to these pins will immediately take effect.

Pin Number	Abbr.	Full Name	Notes
Pin 1	VDD	+ 5v Supply Rail	Connect to Supply (VCC +5V)
Pin 2	NC	Not Connected	Not Connected.
Pin 3	GND	Ground	Ground.
Pin 4	RRD	Receiver Register Disable	When driven high, outputs RBR8:RBR1 are High Impedance.
Pin 5:12	RBR8: RBR1	Receiver Buffer Register	Receiver's Data Bus
Pin 13	PE	Parity Error	When High, A Parity Error Has Occurred.
Pin 14	FE	Framing Error	When High, A Framing Error Has Occurred. i.e. The Stop Bit was not a Logic 1.
Pin 15	OE	Overrun Error	When High, Data has been received but the nData Received Reset had not yet been activated.

Pin 16	SFD	Status Flag Disable	When High, Status Flag Outputs (PE, FE, OE, DR and TBRE) are High Impedance
Pin 17	RRC	Receiver Register Clock	x16 Clock input for the Receiver Register.
Pin 18	nDRR	Data Received Reset	Active Low. When low, sets Data received Output Low (i.e. Clears DR)
Pin 19	DR	Data Received	When High, Data has been received and placed on outputs RBR8:RBR1.
Pin 20	RRI	Receiver Register In	RXD - Serial Input. Connect to Serial Port, Via RS-232 receiver.
Pin 21	MR	Master Reset	Resets the UART. *UART should be reset after applying power.*
Pin 22	TBRE	Transmitter Buffer Register Empty	When High, indicates that Transmitter Buffer Register is Empty, thus all bits including the stop bit have been sent.
Pin 23	nTBRL	Transmitter Buffer Load / Strobe	Active Low. When low, data present on TBR8:TBR1 is placed in Transmitter Buffer Register. A Low to High Transition on this pin, then sends the data.
Pin 24	TRE	Transmitter Register Empty	When High, Transmitter Register is Empty, thus can accept another byte of data to be sent.
Pin 25	TRO	Transmitter Register Out (TXD)	TXD - Serial Output. Connect to Serial Port, Via RS-232 Transmitter.
Pin 26:33	TBR8: TBR1	Transmitter Buffer Register	Data Bus, for Transmitter. Place Data here which you want to send.
Pin 34	CRL	Control Register Load	When High, Control Register (PI, SBS, CLS2, CLS1, EPE) is Loaded. *Can be tied high, so changes on these pins occur instantaneously.*
Pin 35	PI	Parity Inhibit	When High, No Parity is Used for Both Transmit and Receive. When Low, Parity is Used.
Pin 36	SBS	Stop Bit Select	A High selects 2 stop bits. (1.5 for 5 Character Word Lengths) A Low selects one stop bit.

Pin 37:38	CLS2: CLS1	Character Length Select	Selects Word Length. 00 = 5 Bits, 01 = 6 Bits, 10 = 7 Bits and 11 = 8 Bits.
Pin 39	EPE	Even Parity Enable	When High, Even Parity is Used, When Low, Odd Parity is Used.
Pin 40	TRC	Transmitter Register Clock	16x Clock input for Transmitter.

Table 18 : Pin Description for CDP6402, AY-5-1015 / D36402R-9 and compatible UART's

However one disadvantage of these chips over the 8250's is that these UART's have no inbuilt Programmable Baud Rate Generator, and no facility to connect a crystal directly to it. While there are Baud Rate Generator Chips such as the AY-5-8116, a more cheaper *(and common)* alternative is the 74HC4060 14-bit Binary Counter and Oscillator.

The 74HC4060, being a 14 bit binary counter/divider only has outputs for some of it's stages. Only Q4 to Q14 is available for use as they have external connections. This means higher Baud Rates are not obtainable from common crystals, such as the 1.8432 Mhz and 2.4576 Mhz. The UART requires a clock rate 16 times higher than the Baud Rate you will be using. eg A baud rate of 9600 BPS requires a input clock frequency of 153.6 Khz.

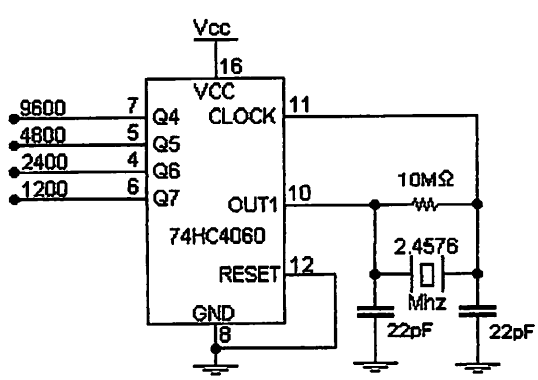


Figure 10 : Baud Rate Generator using a 74HC4060

Output	1.8432Mhz	2.4546Mhz
Out 2	115.2 KBPS	153.6 KBPS
Q4	7200 BPS	9600 BPS
Q5	3600 BPS	4800 BPS
Q6	1800 BPS	2400 BPS
Q7	900 BPS	1200 BPS
Q8	450 BPS	600 BPS
Q9	225 BPS	300 BPS

Table 19 : Possible Baud Rates using a 74HC4060

The 1.8432 Mhz crystal gives some unfamiliar Baud Rates. While many of these won't be accepted by terminal programs or some hardware, they are still acceptable if you write your own serial programs. For example the PC's baud rate divisor for 7200 BPS is 16, 3600 BPS is 32, 1800 BPS is 64 etc. If you require higher speeds, then it is possible to connect the UART to the OUT2 pin. This connection utilizes the oscillator, but has no frequency division applied. Using OUT2 with a 1.8432 Mhz crystal connected gives a baud rate of 115,200 BPS. The CMOS CDP6402 UART can handle up to 200 KBPS at 5 volts, however your MAX-232 may be limited to 120 KBPS, but is still within range.

Microcontrollers

It is also possible to use microcontrollers to transmit and receive Serial data. As we have already covered, some of these MCU's (Micro Controller Units) have built in UART's among other things. Take the application we have used above. We want to monitor analog voltages using a ADC and then send them serially to the PC. If the Microcontroller also has a ADC built in along with the UART or SCI, then we could simply program the device and connect a RS-232 Line Driver. This would minimize your chip count and make your PCB much smaller.

Take the second example, displaying the serial data to a common 16 character x 2 line LCD display. A common problem with the LCD modules, is they don't accept cartridge returns, line-feeds, form-feeds etc. By using a microcontroller, not only can you emulate the UART, but you can also program it to clear the screen, should a form-feed be sent or advance to the next line should a Line-feed be sent.

The LCD example also required some additional logic (An Inverter) to reset the data receive line on the UART, and provide a -ve edge on the enable of the LCD to display the data present on the pins. This can all be done using the Microcontroller and thus reducing the chip count and the cost of the project.

Talking of chip count, most Microcontrollers have internal oscillators thus you don't require the 74HC4060 14 Bit Binary Counter and Oscillator. Many Microcontrollers such as the 68HC05J1A and PIC16C84 have a smaller pin count, than the 40 Pin UART. This not only makes the project smaller in size, it reduces complexity of the PCB.

But there are also many disadvantages of the Microcontroller. The major one, is that you have to program it. For the hobbyist, you may not have a development system for a Microcontroller or a method of programming it. Then you have to learn the micro's code and work out how to tackle the problem. At least with the UART, all you did was plug it in, wire it up and it worked. You can't get much simpler that that.

So far we have only discussed Full Duplex Transmission, that is that we can transmit and receive at the same time. If our Microcontroller doesn't have a SCI then we can *Emulate* a RS-232 port using a Parallel line under software control. However Emulation has it's dis-advantages. It only supports slow transmission speeds, commonly 2400, 9600 or maybe even 19,200 BPS if you are lucky. The other disadvantage is that it's really only effective in half duplex mode. That is, it can only communicate in one direction at any one given time. However in many applications this is not a problem.

As there are many different types of Micro-Controllers all with their different instruction sets, it is very hard to give examples here which will suit everyone. Just be aware that you can use them for serial communications and hopefully at a later date, I can give a limited number of examples with one micro.

Craig Peacock's Interfacing the PC

http://www.senet.com.au/~cpeacock

http://www.geocities.com/SiliconValley/Bay/8302/

Any errors, ideas, criticisms or problems, please contact the author at cpeacock@senet.com.au

3:10pm Friday 30th January 1998