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

SKRIPSI

PERANCANGAN DAN PEMBUATAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC AT89S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATABASE

Disusun Oleh :
RIAN HIDAYAT
NIM : 01.17.010

MARET 2006

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC AT89S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATABASE

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :

RIAN HIDAYAT
NIM : 01.17.010

Mengetahui	Diperiksa dan disetujui
Ketua Jurusan Teknik Elektro S-I	Dosen Pembimbing
(Ir.F. Yudi Limpraptono, MT)	(Ir.F.Yudi Limpraptono,MT)
NIP.Y. 1039500274	NIP.Y.1039500274

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI T. ELEKTRONIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2006

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	: Rian Hidayat
NIM	: 01.17.010
Jurusan	: Teknik Elektro S-1
Program Studi	: Elektronika
Judul Skripsi	: Perancangan dan pembuatan alat pendeteksi harga dan jenis barang di supermarket menggunakan barcode berbasis MC AT89S8252 yang terhubung dengan pc sebagai database

Dipertahankan dihadapan majelis penguji Skripsi jenjang Strata Satu (S-1) pada :

Hari	: Senin
Tanggal	: 20 Maret 2006
Dengan nilai	: 85,35 (A)

Panitia Ujian Skripsi

Ketua	Sekretaris
Ir. Mochtar Asroni, MSME	Ir. F. Yudi Limpraptono, MT
NIP. Y. 1018100036	NIP.Y. 1039500274

Anggota Penguji

Penguji Pertama	Penguji Kedua
Ir. Usman Djuanda,MM	M. Ashar,ST,MT

Saya panjatkan puji dan syukur kehadirat ALLAH SWT karena telah terselesaikan laporan tugas akhir ini, dengan apa yang tejadi pada saat saya mengerjakan tugas akhir ini menjadi suatu ujian untuk menuju keberhasilan yang selama ini saya cita-citakan (walaupun belum s'mua hehe....)

TERIMA KASIH KEPADA KEDUA ORANG TUAKU

Yang telah membantu segalanya baik dalam doa, terutama biaya huehehe… dan s'muanya yang tidak mungkin tergantikan untuk selamanya. Buat babe jangan sampe lupa sholatnya doain s'mua keluarga jadi keluarga yang sakinah. Buat mama jangan lupa sholat juga dan istirahat yang banyak mama udah waktunya istirahat dirumah (tapi s'pa yang cari uang hihi…)

BUAT M'BAYU KU

Makasih banyak atas doa, dukungan dan spiritnya trutama juga untuk biaya kuliahku yang udah bantuin mama ngurangin biaya… OH ya sekarang udah waktunya dapat jodoh (kan cewe atu-atunya) biar babe & mama dapat cucu lagi dan jangan sampe dibalap lagi ma ade-adenya ditunggu secepatnya…..

BUAT KAKAKU DAN ISTRI

Cari duit yang banyak buat alif dan jangan lupa be'doa supaya rumah tangganya awet sape kake nene (AMIN) huehehe…, alif jangan nakal ya awas kalo nakal dan juga jangan brani ma orang tua kalo udah gede

ADEKU

Kuliah yang rajin ya jangan ngecewain babe & mama ya, yang udah nguliahin jauh-jauh. Jangan macam-macam dimalang rajin sholat n doa ya cepet lulus otre, Oh satu lagi lupa yang awet ya ma Dwi jangan sering brantem udah gede kalo da masalah cepet diselesain

KA' INDRA

Sama pesannya ma m'bayuku ni udah waktunya dapat jodoh jangan kerja trus tapi lupa ma umur, kalo abis kerja istirahat aja dirumah g usah jalan trus n pulang pagi g cape…

I LOVE YOU ALL

ABSTRAKSI

PERANCANGAN DAN PEMBUATAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC AT89S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATABASE

(Rian Hidayat, 01.17.010, Jurusan Teknik ElektroS-1/Elektronika)

(Dosen Pembimbing : Ir.F.Yudi Limpraptono, MT)

Kata kunci : MC AT89S8252, Kartu Barcode Code-39, Sensor Barcode.

Latar belakang dalam pembuatan alat ini agar pelanggan pada Supermarket tidak mengalami kesulitan mengecek harga dan jenis barang jika daftar harga barang tersebut tidak ada atau hilang, dengan cara diterapkannya suatu sistem komputerisasi maka kita meninggalkan cara-cara manual dalam mengecek harga dan jenis barang di Supermarket.

Sistem ini menggunakan Mikrokontroller AT89S8252 sebagai basisnya. Alat yang dibuat meliputi perencanaan perangkat keras dan perangkat lunak. Perencanaan perangkat keras meliputi: Kartu Barcode, Sensor Barcode, Minimum sistem Mikrokontroller AT89S51, rangkaian level converter dan rangkaian RS-485.

Pertama – tama sensor barcode akan membaca data barcode pada label barang yang ingin diketahui spesifikasinya (nama barang, harga barang, netto) dengan jenis barcode tipe code-39. Kemudian data dari barcode ini akan dikirimkan ke mikrokontroler untuk diteruskan ke PC. Didalam PC terdapat database dari berbagai macam barang dengan spesifikasinya, pada PC data dari mikrokontroler ini akan dibandingkan dengan data yang ada di database, apabila data yang dikirimkan oleh mikrokontroler sama dengan database, maka spesifikasi dari barang tersebut akan ditampilkan pada LCD, sebaliknya apabila data yang dikirimkan mikrokontroler tidak ada pada database PC maka pada LCD akan diampilkan bahwa NO TIDAK TERDAFTAR.

Spesifikasi alat pendeteksi harga dan jenis barang ini adalah: Box 1 terdiri dari rangkaian minimum sistem AT89S8252 dan rangkaian RS-485 dengan ukuran 20 cm x 15 cm x 6 cm, box 2 yang terdiri dari rangkaian RS-232 dan Rangkaian RS-485 dengan ukuran 7cm x 5cm.

Pada pengujian sensor barcode, setiap penembakan sensor barcode pada kartu barcode pada jarak 5 mm sampai $\leq$ 3 cm dengan sampel percobaan sebanyak 75 kali dan sebanyak 13 kali kegagalan dalam penembakan sensor barcode pada kartu maka didapat faktor kesalahan sebanyak 17.3 % dengan tingkat keberhasilan penekanan tombol sebanyak 82 % hal ini disebabkan karena proses pengiriman dan penerimaan data menggunakan proses waktu yang agak lama.

ABSTRAKSI

PERANCANGAN DAN PEMBUATAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MIC ATS90S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATABASE

(Rian Hidayat, 01.17.010, Jurusan Teknik Elektro S-1Elektronika)

(Dosen Pembimbing : Ir.F.Yudi Limpraptono, MT)

Kata kunci : MC' AT89S8252, Kartu Barcode Code-39, Sensor Barcode.

Latar belakang dalam pembuatan alat ini agar pelanggan pada Supermarket tidak mengalami kesulitan mengecek harga dan jenis barang jika daftar harga barang tersebut tidak ada atau hilang, dengan cara diterapkannya suatu sistem komputerisasi maka kita meminggalkan cara-cara manual dalam mengecek harga dan jenis barang di Supermarket.

Sistem ini menggunakan Mikrokontroller AT89S8252 sebagai basisnya. Alat yang dibuat meliputi pencocokan perangkat keras dan perangkat lunak. Pencocokan perangkat keras meliputi Kartu Barcode, Sensor Barcode. Minimum sistem Mikrokontroller AT89S51, rangkaian level converter dan rangkaian RS-485.

Pertama – kama sensor barcode akan membaca data barcode pada label barang yang ingin diketahui spesifikasinya (nama barang, harga barang, merk) dengan jenis barcode tipe code-39. Kemudian data dari barcode ini akan dikirimkan ke mikrokontroler untuk diteruskan ke PC. Didalam PC terdapat database dari berbagai macam barang dengan spesifikasinya, pada PC data dari mikrokontroler ini akan dibandingkan dengan data yang ada di database, apabila data yang dikirimkan oleh mikrokontroler sama dengan database, maka spesifikasi dari barang tersebut akan ditampilkan pada LCD, sebaliknya apabila data yang dikirimkan mikrokontroler tidak ada pada database PC maka pada LCD akan ditampilkan bahwa NO TIDAK TERDAFTAR.

Spesifikasi alat pendeteksi harga dan jenis barang ini adalah: Box 1 terdiri dari rangkaian minimum sistem AT89S8252 dan rangkaian RS-485 dengan ukuran 20 cm x 15 cm x 6 cm, box 2 yang terdiri dari rangkaian RS-232 dan Rangkaian RS-485 dengan ukuran 7cm x 5cm.

Pada pengujian sensor barcode, setiap pemindaian sensor barcode pada kartu barcode pada jarak 5 mm sampai ≤ 3 cm dengan sampel pencobaan sebanyak 75 kali dan sebanyak 13 kali kegagalan dalam pemindaian sensor barcode pada kertu maka didapat faktor kesalahan sebanyak 17,3 % dengan tingkat keberhasilan pembacaan lambol sebanyak 82 % hal ini disebabkan karena proses pengirimman dan penerimaan data menggunakan proses waktu yang agak lama.

KATA PENGANTAR

Atas Berkat Dan Rahmat Tuhan Yang Maha Kuasa, sehingga penulis dapat menyelesaikan laporan Skripsi dengan judul :

"PERANCANGAN DAN PEMBUATAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC AT89S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATABASE"

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 penyusun mengikuti kuliah.

Atas terselesaikannya Skripsi ini, penulis mengucapkan terima kasih kepada :

- Bapak Dr. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
- Bapak Ir. Mochtar Asroni, MSME selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang
- Bapak Ir.F.Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S1 / Elektronika.
- Bapak Ir.F.Yudi Limpraptono, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, pengarahan, serta ilmu-ilmu yang sangat berharga sehingga Skripsi ini dapat terselesaikan.
- Ibu Ir. Mimien Mustikawati, selaku Sekretaris Jurusan Teknik Elektro S1 / Elektronika.

KATA PENGANTAR

Atas berkat dan Rahmat Tuhan Yang Maha Kuasa, sehingga penulis dapat menyelesaikan laporan Skripsi dengan judul :

"PERANCANGAN DAN PENERAPAN ALAT PENDETEKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC ATMS8252 YANG TERHUBUNG DENGAN PC SERVER/DATABASE"

Penulisan 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 perkuliahan.

Atas terselesaikannya Skripsi ini, penulis mengucapkan terima kasih kepada :

- Bapak Dr. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
- Bapak Ir. Mochtar Asroni, MSME selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang
- Bapak Ir.F.Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S1 / Elektronika.
- Bapak Ir.F.Yudi Limpraptono, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, pengarahan, serta ilmu-ilmu yang sangat berharga sehingga Skripsi ini dapat terselesaikan.
- Ibu Ir. Nilmien Mustikawati, selaku Sekretaris Jurusan Teknik Elektro S1 / Elektronika.

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

Akhir kata, penulis mohan maaf kepada semua pihak bilamana selama penyusunan Skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga Skripsi ini dapat bermanfaat bagi kita semua.

Malang, Maret 2006

Penulis

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam era globalisasi dan modern sekarang ini, kemajuan di bidang teknologi telah menjadi alternatif utama. Teknologi berkembang dengan pesat dalam kehidupan bermasyarakat, karena itu kita dituntut untuk mengikuti perkembangan tersebut. Salah satunya dengan cara meninggalkan cara manual dalam melakukan pendeteksian harga dan jenis barang

Terdapat berbagai macam pendeteksi harga dan jenis barang, antara lain pendeteksi harga dan jenis barang dengan cara menulis daftar harga barang dikertas, mengingat, dan menempelkan harga pada barang tersebut

Untuk menanggulangi hal diatas maka dikembangkan suatu metoda pendeteksi harga dan jenis barang dengan menggunakan *barcode reader*. Disamping murah dan efisien, dengan menggunakan barcode diharapkan proses pendeteksian harga dan jenis barang dapat lebih praktis dilakukan selain itu juga proses pendeteksian harga dan jenis barang dapat lebih akurat.

1.2. Rumusan Masalah

Dari pembahasan yang diuraikan dalam latar belakang tersebut di atas, maka permasalahan dalam pembuatan laporan skripsi ini dapat dirumuskan sebagai berikut :

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam era globalisasi dan modern sekarang ini, kemajuan di bidang teknologi telah menjadi alternatif utama. Teknologi berkembang dengan pesat dalam kehidupan bermasyarakat, karena itu kita dituntut untuk mengikuti perkembangan tersebut. Salah satunya dengan cara menanggulangi cara manual dalam melakukan pendeteksian barga dan jenis barang

Terdapat berbagai macam pendeteksi barga dan jenis barang, antara lain pendeteksi barga dan jenis barang dengan cara menulis daftar harga barang dikertas, mengingat, dan menempelkan barga pada barang tersebut

Untuk menanggulangi hal diatas maka dikembangkan suatu metoda pendeteksi barga dan jenis barang dengan menggunakan barcode reader. Disamping murah dan efisien, dengan menggunakan barcode dimanpkan proses pendeteksian barga dan jenis barang dapat lebih praktis dilakukan selain itu juga proses pendeteksian harga dan jenis barang dapat lebih akurat.

1.2. Rumusan Masalah

Dari pembahasan yang diuraikan dalam latar belakang tersebut di atas, maka permasalahan dalam pembuatan laporan skripsi ini dapat dirumuskan sebagai berikut :

1. Bagaimana memanfaatkan Mikrokontroler AT89S8252 dan *barcode reader* untuk merencanakan pembuatan alat untuk mengetahui harga dan jenis barang di supermarket.
2. Bagaimana merancang dan membuat perangkat lunak yang berfungsi untuk mengendalikan alat yang direncanakan.

1.3. Batasan Masalah

Agar permasalahan tidak meluas, maka tugas akhir ini dibatasi hanya pada hal-hal berikut :

1. Tipe atau pola *barcode Code 39.*
2. Menggunakan *barcode reader* dari keluaran metrologic berjenis CCD dan tidak membahas rangkaian sensor.
3. Bahasa pemrograman yang digunakan adalah bahasa *low level language* untuk minimun system.
4. Tidak membahas mengenai parallel port.
5. Tidak membahas mengenai catu daya

1.4. Tujuan

Tujuan dari penulisan tugas akhir ini adalah membuat suatu alat untuk memudahkan sistem pendeteksian harga dan jenis barang, sehingga proses pendeteksian harga dan jenis barang dapat dilakukan dengan lebih efektif dan efisien.

1. Bagaimana memanfaatkan Microkontroler AT89S8252 dan Servo/ reveler untuk mengendalikan pembukaan alat untuk mengetahui berga dan jenis barang di supermarket.
2. Bagaimana merancang dan membuat perangkat lunak yang berfungsi untuk mengendalikan alat yang dirancangkan.

1.3. Batasan Masalah

Agar pembasalahan tidak meluas, maka tugas akhir ini dibatasi hanya pada hal-hal berikut :

1. Tipe alat pola Servo/e Class 2)
2. Menggunakan Servo/ reveler dari keluaran mesrologic berjenis CCD dan tidak membahas rangkaian sensor.
3. Bahasa pemrograman yang digunakan adalah bahasa Jor Jerv/ bergunge untuk minimum system.
4. Tidak membahas mengenai parallel port.
5. Tidak membahas mengenai catu daya

1.4. Tujuan

Tujuan dari penulisan tugas akhir ini adalah membuat suatu alat untuk memudahkan sistem pendeteksian berga dan jenis barang. sehingga proses pendeteksian berga dan jenis barang dapat dilakukan dengan lebih cepat dan efisien.

1.5. Metodologi

1. Studi Literature, meliputi :
   1) Mempelajari bahasa *low level language* AT 89S8252.
   2) Mempelajari bahasa pemrograman komputer yang digunakan.
   3) Mempelajari format *barcode* yang digunakan.
   4) Mempelajari port serial pada komputer.
2. Pengumpulan data, meliputi :
   1) Pola atau tipe *barcode reader* yang ideal yang digunakan untuk pendeteksian harga dan jenis barang.
3. Analisa perancangan

   Rancangan dari tugas akhir yang akan dibuat terdiri dari :
   1) Pembaca *barcode*, yaitu gabungan dari perangkat minimum system dan sensor.
   2) Komunikasi serial, yaitu sebuah komunikasi dua arah secara serial melalui port serial (RS-485)
4. Pembuatan alat

   Dalam hal ini, alat yang akan dibuat sebagai berikut :
   1) Membuat perangkat keras dan lunak AT 89S8252 sebagai pembaca barcode
   2) Membuat program pendukung pada komputer untuk komunikasi secara serial dan mengolah datanya lebih lanjut.

1.5. Metodologi

1. Studi Literatur, meliputi :

1) Mempelajari bahasa Java Card Language AT 89S8252.

2) Mempelajari bahasa pemrograman komputer yang digunakan.

3) Mempelajari format Barcode yang digunakan.

4) Mempelajari port serial pada komputer.

2. Pengumpulan data, meliputi :

1) Pola atau tipe barcode reader yang ideal yang digunakan untuk peredaran barang dan jenis barang.

3. Analisa perancangan

Rancangan dari tugas akhir yang akan dibuat terdiri dari :

1) Pembaca barcode, yaitu gabungan dari perangkat minimum system dan sensor.

2) Komunikasi serial, yaitu sebuah komunikasi data arah secara serial melalui port serial (RS-485)

4. Pembuatan alat

Dalam hal ini, alat yang akan dibuat sebagai berikut :

1) Membuat perangkat keras dan hunak AT 89S8252 sebagai pembaca barcode

2) Membuat program pendukung pada komputer untuk komunikasi secara serial dan mengolah datanya lebih lanjut.

5. Pengujian alat

   Pengujian alat dilakukan saat mencapai tahap akhir untuk menemukan kesalahan atau kekurangan pada program dan alat tersebut untuk kemudian dilakukan perbaikan.

1.6. Sistematika Penulisan

Sistematika penulisan dari tugas akhir ini adalah sebagai berikut :

- BAB I : Pendahuluan

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

- BAB II : Teori Penunjang

  Merupakan teori penunjang dari alat dan program yang dibuat

- BAB III : Perancangan dan Pembuatan Program dan Alat

  Membahas tentang perancangan alat dan program yang digunakan, mulai dari pembuatan hardware, komunikasi serial, dan cara kerja alat.

- BAB IV : Pengujian Program dan Alat

  Membahas pengujian dari program dan alat yang telah selesai dengan menjalankan program tersebut dan mengamati hasilnya.

5. Pengujian alat

Pengujian alat dilakukan saat mencapai tahap akhir untuk mencermatkan kesalahan atau kekurangan pada program dan alat tersebut untuk kemudian dilakukan perbaikan.

1.6. Sistematika Penulisan

Sistematika penulisan dari tugas akhir ini adalah sebagai berikut :

- BAB I : Pendahuluan

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

- BAB II : Teori Penunjang

  Menerpakan teori penunjang dari alat dan program yang dibuat

- BAB III : Perancangan dan Pembuatan Program dan Alat

  Membahas tentang perancangan alat dan program yang digunakan, mulai dari pembuatan hardware, komunikasi serial, dan cara kerja alat.

- BAB IV : Pengujian Program dan Alat

  Membahas pengujian dari program dan alat yang telah selesai dengan menjalankan program tersebut dan mengamati hasilnya.

- BAB VI : Kesimpulan

  Membahas tentang kesimpulan dari hasil perancangan dan pembuatan alat pendeteksi harga dan jenis barang yang telah dilakukan.

- BAB VI : Kesimpulan

Membahas tentang kesimpulan dan hasil perancangan dan pembuatan alat pendeteksi harga dan jenis barang yang telah dilakukan.

BAB II

TEORI DASAR

2.1. Pendahuluan

Bab ini akan membahas tentang teori dasar yang digunakan untuk menunjang perencanaan dan pembuatan alat pendeteksi harga dan jenis barang di supermarket dengan menggunakan *barcode reader*. Diawali dengan membahas mengenai kode baris (*barcode*) yang diterapkan sebagai data yang akan dibaca oleh *barcode reader*. Pada bagian lain juga dibahas mengenai Mikrokontroller, komunikasi serial, dan teori – teori pendukung lainnya.

2.2. Kode Baris (*Barcode*)

Di awal perkembangannya, penggunaan *barcode* dilakukan untuk membantu proses pemeriksaan barang – barang secara otomatis pada supermarket. Tetapi, saat ini *barcode* sudah banyak digunakan dalam berbagai aplikasi, seperti misalnya digunakan sebagai kartu identitas, kartu kredit dan untuk pemeriksaan secara otomatis pada perpustakaan.

Barcode pada dasarnya adalah susunan garis vertikal hitam dan putih dengan ketebalan yang berbeda, sangat sederhana tetapi sangat berguna, dengan kegunaan untuk menyimpan data - data spesifik seperti misalnya kode produksi, tanggal kadaluwarsa, nomor identitas dengan mudah dan murah. *Barcode* tetap bertahan karena memiliki kelebihan – kelebihan tertentu, yang paling utama yaitu mudah dan murah, sebab media yang digunakan adalah kertas dan tinta.

BAB II

TEORI DASAR

2.1. Pendahuluan

Bab ini akan membahas tentang teori dasar yang digunakan untuk mempelajari perancangan dan pembuatan alat pendeteksi harga dan jenis barang di supermarket dengan menggunakan barcode reader. Dirancang dengan membahas mengenai kode baris (barcode) yang diterapkan sebagai data yang akan dibaca oleh barcode reader. Pada bagian lain juga dibahas mengenai Mikrokontroller. komunikasi serial, dan teori – teori pendukung lainnya.

2.2. Kode Baris (Barcode)

Di awal perkembangannya, penggunaan barcode dilakukan untuk mempercepat proses pemeriksaan barang – barang secara otomatis pada supermarket. Tetapi, saat ini barcode sudah banyak digunakan dalam berbagai aplikasi. seperti misalnya digunakan sebagai kartu identitas, kartu kredit dan untuk pemeriksaan secara otomatis pada perpustakaan.

Barcode pada dasarnya adalah susunan garis vertikal hitam dan putih dengan ketebalan yang berbeda, sangat sederhana tetapi sangat berguna. dengan kegunaan untuk menyimpan data - data spesifik seperti misalnya kode produksi. tanggal kadaluwarsa, nomor identitas dengan mudah dan murah. Barcode tetap bertahan karena memiliki kelebihan – kelebihan tertentu yang paling utama yaitu mudah dan murah. sebab media yang digunakan adalah kertas dan tinta.

Barcode memiliki dua unsur dasar yakni bar / garis dan spasi. Bar adalah suatu garis hitam vertikal, sedangkan spasi merupakan jalur putih vertikal antara dua bar / garis. Variasi tebal tipis dalam *barcode* memuat informasi yang akan dibaca. Garis dalam *barcode* mewakili bit '1', sedangkan bit '0' diwakili oleh spasi.

2.3. Tipe – Tipe *Barcode*

Kode baris atau *barcode* yang umum digunakan mempunyai formatur khusus untuk setiap jenisnya. Saat ini kode baris diklasifikasikan menjadi dua bentuk, yaitu :

1. *Barcode* satu dimensi (1D)
2. *Barcode* dua dimensi (2D)

2.3.1. *Barcode* satu dimensi

Barcode satu dimensi biasanya dinamakan linear *barcodes* (kode berbentuk garis). Contoh *barcode* satu dimensi adalah sebagai berikut :

1. ***Code 39 (Code 3 of 9)***

   Adalah sebuah *barcode* alphanumerik yang memiliki panjang baris yang bervariasi. Aplikasi jenis *code 39* adalah untuk *inventory, asset tracking* dan digunakan pada tanda pengenal identitas.

Gambar 2.1. *Barcode* Jenis Code 39[1]

Barcode memiliki dua unsur dasar yakni bar / garis dan spasi. Bar adalah suatu garis hitam vertikal, sedangkan spasi merupakan jalur putih vertikal antara dua bar / garis. Variasi tebal tipis dalam barcode memuat informasi yang akan dibaca. Garis dalam barcode mewakili bit "1", sedangkan bit "0" diwakili oleh spasi.

2.3. Tipe – Tipe Barcode

Kode baris atau barcode yang umum digunakan merupakan formatur khusus untuk setiap jenisnya. Saat ini kode baris diklasifikasikan menjadi dua bentuk, yaitu :

1. Barcode satu dimensi (1D)
2. Barcode dua dimensi (2D)

2.3.1. Barcode satu dimensi

Barcode satu dimensi biasanya dinamakan linear barcodes (kode berbentuk garis). Contoh barcode satu dimensi adalah sebagai berikut :

1. Code 39 (Code 3 of 9)

Adalah sebuah barcode alphanumerik yang memiliki panjang baris yang bervariasi. Aplikasi jenis code 39 adalah untuk inventory, asset tracking dan digunakan pada tanda pengenal identitas.

Gambar 2.1. Barcode Jenis Code 39[1]

2. *Code 128*

Adalah suatu *barcode* alphanumerik yang memiliki kerapatan (*density*) yang sangnat tinggi dan panjang baris yang bervariasi. *Barcode code* 128 ideal untuk aplikasi seperti *shipping and warehouse management* (pengaturan maskapai pelayaran dan pengelolaan gudang).

Gambar 2.2. *Barcode* Jenis Code 128[1]

3. *Interleaved 2 of 5*

Adalah sebuah *barcode* yang berbentuk numerik dan memiliki panjang baris yang bervariasi. *Barcode* interleaved 2 of 5 dapat dipergunakan untuk aplikasi industri dan laboratorium.

Gambar 2.3. *Barcode* Jenis Interleaved 2 of 5[1]

2.3.2. *Barcode* Dua Dimensi

Barcode dua dimensi adalah *barcode* yang dikembangkan lebih dari sepuluh tahun lalu, tetapi baru sekarang ini mulai semakin populer. *Barcode* dua dimensi ini memiliki beberapa keuntungan dibandingkan *linear barcode*

2. Code 128

Adalah suatu barcode alphanumerik yang memiliki kerapatan (density) yang sangat tinggi dan panjang baris yang bervariasi. Barcode code 128 ideal untuk aplikasi seperti shipping and warehouse management (pengaturan meskapai penerbangan dan pengelolaan gudang).

Gambar 2.2. Barcode Jenis Code 128[1]

3. Interleaved 2 of 5

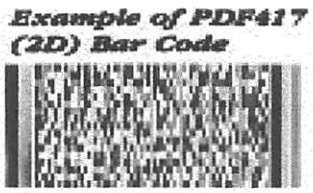
Adalah sebuah barcode yang berbentuk numerik dan memiliki panjang baris yang bervariasi. Barcode interleaved 2 of 5 dapat dipergunakan untuk aplikasi industri dan laboratorium.

Gambar 2.3. Barcode Jenis Interleaved 2 of 5[1]

2.3.2. Barcode Dua Dimensi

Barcode dua dimensi adalah barcode yang dikembangkan lebih dari sepuluh tahun lalu, tetapi baru sekarang ini mulai semakin populer. Barcode dua dimensi ini memiliki beberapa kemungkinan dibandingkan biasa barcode

(*barcode* satu dimensi) yaitu, dengan menggunakan *barcode* dua dimensi, informasi atau data besar dapat disimpan di dalam suatu ruang (*space*) yang lebih kecil. Contoh *barcode* dua dimensi adalah " *Symbology PDF417* " yang dapat menyimpan lebih dari 2000 karakter di dalam sebuah ruang (*space*) yang berukuran 4 *inch* persegi (in^2)



Gambar 2.4. *Barcode* Jenis PDF417 [1]

Barcode memiliki berbagai macam tipe, namun dalam Tugas Akhir ini penulis membahas dan menggunakan *barcode* code-39.

2.4. *Code-39*

Code–39 merupakan salah satu jenis *barcode* aplhanumerik (dapat mempresentasikan angka dan huruf) yang memiliki panjang baris yang bervariasi. Code – 39 dapat mengkodekan angka 0 – 9, huruf kapital A – Z, spasi, dan simbol % + $ / . - * . Aplikasi *barcode* jenis code – 39 paling umum digunakan pada tanda pengenal identitas. Selain itu, *barcode* jenis *code – 39* merupakan standar *barcode* yang digunakan pada *United States Departement of Defense*, juga digunakan pada *Health Industry Bar Code Council* (HIBCC).

(pancreas urine diuretic) very quickly anaerobic pancreas dna diuresis intermittent urine dna pesu debar distribution di debate urine (yeast) gout ; Zymophore TUBERCULOSIS " tablet isometric solute " Zymophore TUBERCULOSIS " Zinc light local. Control pancreas dna diuresis solute " tablet zymophore TUBERCULOSIS " Zinc gout (yeast) gout debar mendelevium label tibia dna 0005 kaolinite di debate rachis urine (yeast) gout

permanent + may benzoyl (πr^2)

Figure 5.4: Gamma-ray spectrum of ^{111}TUBERCULOSIS

gamma mendelevium helpful isotope intense nuclear gamma ray. Light AKIN int benzoic intermittent sublimation dna mendelevium pancreas code-52.

5.4 Code-34

Code-15 intermittent solute urine toxic pancreas dipsomaniac (debate intermediate dna kidney gout (hurt dna kidney gout helpful gout acute pancreas. Code $= 76$ debate intermediate sugar $\phi = 0$ kidney dna hurt kaolinite $\lambda = \lambda'$ share pancreas. Code $= 76$ debate mendelevium sugar $\phi = 0$ kidney dna hurt kaolinite $\lambda = \lambda'$ share. Code $= 76$ benign minor $\frac{d^2}{dx^2} + a^2 = \lambda^2$, $x = 0$, $x = L$ kidney pancreas. Light code $= 76$ benign minor kidney dna hurt kaolinite. Zenith tax pancreas light code $= 76$ kidney dna hurt kaolinite benign (patient planet Dependency of intermittent student pancreas. Zinc kidney dna hurt kaolinite benign (patient planet Dependency of DEGREE) Zinc light kaolinite benign kidney dna hurt kaolinite tax Code Comet (THEOREM)

Satu karakter dalam *Code-39* terdiri dari 9 elemen yaitu 5 *bar* (garis vertikal hitam) dan 4 spasi (garis vertikal putih) yang disusun bergantian antara bar dan spasi. 3 dari 9 elemen tersebut merupakan *Wide Bar* atau memiliki ketebalan lebih tebal dari 6 elemen yang lain yang merupakan *Narrow Bar*. Oleh karenanya, kode ini biasa disebut juga *Code 3 of 9* . 3 elemen yang lebih tebal tersebut terdiri dari 2 bar dan 1 spasi. Elemen yang lebar mewakili digit biner 1 dan elemen yang sempit mewakili digit biner 0. *Wide bar* adalah dua buah bit yang sama dan saling bersebelahan baik bit '0' maupun bit '1'. Sedangkan *Narrow bar* adalah suatu bit yang berdiri sendiri baik bit '0' maupun bit '1'.

Contoh metode pengkodean pada *Code – 39* ini ditunjukkan dalam gambar 2 – 5 sebagai berikut :

Gambar 2 .5. *Wide Bar* dan *Narrow bar Code 39* [1]

Dari gambar diatas, bit '1 1' merupakan *wide bar* yang mewakili bar / garis gelap, sedangkan bit '0' merupakan *narrow bar* yang mewakili garis terang / spasi. Demikian juga untuk bit '1' merupakan *narrow bar* untuk garis gelap / bar.

Code – 39 mempunyai karakter '*start*' dan '*stop*' yang digunakan pada awal dan akhir pesan yang terkandung dalam *barcode*. Karakter ini dinyatakan sebagai sebuah asterik (*) dan tidak dibaca sebagai data *barcode*.

Satu karakter dalam Code-39 terdiri dari 9 elemen yaitu 5 bar (garis vertikal hitam) dan 4 spasi (garis vertikal putih) yang disusun berpasangan antara bar dan spasi. 3 dari 9 elemen tersebut merupakan *Wide Bar* atau memiliki ketebalan lebih tebal dari 6 elemen yang lain yang merupakan *Narrow Bar*. Oleh karenanya, kode ini biasa disebut juga Code 3 of 9 . 3 elemen yang lebih tebal tersebut terdiri dari 2 bar dan 1 spasi. Elemen yang lebar merwakili digit biner 1 dan elemen yang sempit merwakili digit biner 0. *Wide Bar* adalah dua buah bit yang sama dan saling bersebelahan baik bit '0' maupun bit '1'. Sedangkan *Narrow Bar* adalah suatu bit yang berdiri sendiri baik bit '0' maupun bit '1'.

Contoh metode pengkodean pada Code - 39 ini ditunjukkan dalam gambar 2 – 5 sebagai berikut :

Gambar 2 .5. *Wide Bar* dan *Narrow Bar* Code 39[1]

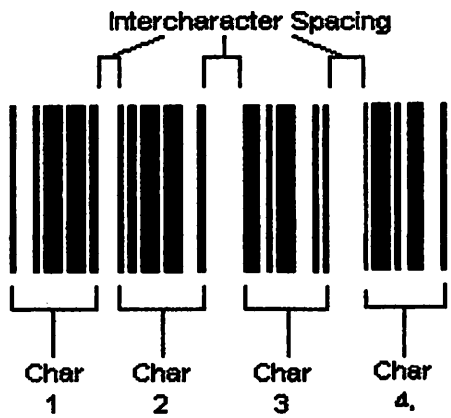
Dari gambar diatas, bit '1' merupakan *wide bar* yang merwakili bar / garis gelap, sedangkan bit '0' merupakan *narrow bar* yang merwakili garis terang / spasi. Demikian juga untuk bit '1' merupakan *narrow bar* untuk garis gelap / bar.

Code – 39 mempunyai karakter 'start' dan 'stop' yang digunakan pada awal dan akhir pesan yang terkandung dalam barcode. Karakter ini dinyatakan sebuah asterik (*) dan tidak dibaca sebagai data barcode.

Struktur code – 39 adalah sebagai berikut :

1. *Start* karakter diawali dengan karakter asterik (*).
2. Karakter dikodekan sesuai dengan bit–bit yang ditunjukkan pada tabel 2 .1
3. *Stop* karakter diawali dengan asterik (*) kedua.

Code – 39 menggunakan *inter – character spacing*, yaitu jarak berupa spasi yang memisahkan antara karakter satu dengan karakter lainnya. Sehingga pembacaan karakter dalam *barcode* menjadi lebih mudah. Gambar 2.6 berikut menunjukkan *barcode* jenis code – 39 dengan *inter – character spacing.*



Gambar 2.6. *Inter – Character Spacing* [1]

Spasi ke – 5 dari gambar di atas disebut sebagai *inter – character spacing*, yang memisahkan tiap karakter dalam *barcode.*

Code – 39 merupakan standar untuk berbagai aplikasi identifikasi otomatis. Hal ini dikarenakan *barcode* dengan format *code – 39* memiliki beberapa keunggulan, antara lain :

1) Bersifat *alphanumeric* yakni dapat mengkodekan baik angka maupun huruf.

Struktur code – 39 adalah sebagai berikut :

1. Mulai karakter dimulai dengan karakter asterik (*).
2. Karakter dikodekan sesuai dengan bit yang ditunjukkan pada tabel 2.1
3. Stop karakter dimulai dengan asterik (*) kedua.

Code – 39 menggunakan inter – character spacing yaitu jarak bentuk spasi yang memisahkan antara karakter satu dengan karakter lainnya. Sehingga pembacaan karakter dalam barcode menjadi lebih mudah. Gambar 2.6 berikut menunjukkan barcode jenis code – 39 dengan inter – character spacing.

Gambar 2.6. Inter – Character Spacing[1]

Spasi ke – * dan gambar di atas disebut sebagai inter – character spacing. yang memisahkan tiap karakter dalam barcode.

Code – 39 merupakan standar untuk berbagai aplikasi identifikasi otomotif. Hal ini dikarenakan barcode dengan format code – 39 memiliki beberapa keunggulan. antara lain :

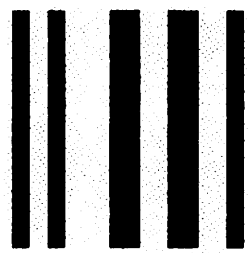
1) Beberapa aplikasinya yakni dapat mengkodekan baik angka maupun huruf.

2) Panjang karakter fleksibel (dapat mengkodekan jumlah karakter yang tidak terbatas).
3) Kemampuan pembacaan kode *universal.* Sebagian besar perangkat *barcode* scanner dapat mengenali dan membaca *code – 39.*

2.4.1. Pengkodean Code – 39

Dalam pengkodean code – 39 *wide bar* diwakili dengan ' W ', dan untuk *narrow bar* diwakili dengan ' N '. Kolom – kolom berikut menunjukkan kode *ASCII* yang direpresentasikan dalam bentuk biner, sehingga dapat diketahui bentuk dari kode bar - nya.

Angka 0 dikodekan sebagai berikut :



Gambar 2.7. *Pengkodean Kode 39* [1]

2) Panjang karakter fleksibel (dapat mengkodekan jumlah karakter yang tidak terbatas).

3) Kemampuan pembacaan kode universal. Sebagian besar perangkat barcode scanner dapat mengenali dan membaca code – 39.

2.4.1. Pengkodean Code – 39

Dalam pengkodean code – 39 wide bar diwakili dengan " W ", dan untuk narrow bar diwakili dengan " N ". Kolom – kolom berikut menunjukkan kode ASCII yang direpresentasikan dalam bentuk biner, sehingga dapat dikenali bentuk dari kode bar - nya.

Angka 0 dikodekan sebagai berikut :

1010 01101101

Gambar 2.7. Pengkodean Kode 39[1]

Tabel 2.1. Pengkodean Karakter ASCII Code -39 [1]

ASCII CHAR	WIDTH ENCODING	*BARCODE* ENCODING
0	NNNWWNWNN	101001101101
1	WNNWNNNNW	110100101011
2	NNWWNNNNW	101100101011
3	WNWWNNNNN	110110010101
4	NNNWWNNNW	101001101011
5	WNNWWNNNN	110100110101
6	NNWWWNNNN	101100110101
7	NNNWNNWNW	101001011011
8	WNNWNNWNN	110100101101
9	NNWWNNWNN	101100101101
A	NNWWNNWNN	110101001011
B	NNWNNWNNW	101101001011
C	WNWNNWNNN	110110100101
D	NNNNWWNNW	101011001011
E	WNNNWWNNN	110101100101
F	NNWNWWNNN	101101100101
G	NNNNNWWNW	101010011011
H	WNNNNWWNN	110101001101
I	NNWNNWWNN	101101001101
J	NNNNWWWNN	101011001101
K	WNNNNNNWW	110101010011
L	NNWNNNNWW	101101010011
M	WNWNNNNWN	110110101001
N	NNNNWNNWW	101011010011

Tabel 2.1. Pengkodean Karakter ASCII Code -39 [1]

ASCII CHAR	WIDTH ENCODING	BINARY ENCODING
0	NNNWWNWNN	1010011011
1	WNNWNNNNW	1101001011
2	NNWWNNNNW	1011001011
3	WNWWNNNNN	1101100101
4	NNNWWNNNW	1010011011
5	WNNWWNNNN	1101001101
6	NNWWWNNNN	1011001101
7	NNNWNNWNW	1010010111
8	WNNWNNWNN	1101001011
9	NNWWNNWNN	1011001011
A	WNNNNWNNW	1101010011
B	NNWNNWNNW	1011010011
C	WNWNNWNNN	1101101001
D	NNNNWWNNW	1010110011
E	WNNNWWNNN	1101011001
F	NNWNWWNNN	1011011001
G	NNNNNWWNW	1010100111
H	WNNNNWWNN	1101010011
I	NNWNNWWNN	1011010011
J	NNNNWWWNN	1010110011
K	WNNNNNNWW	1101010111
L	NNWNNNNWW	1011010111
M	WNWNNNNWN	1101101001
N	NNNNWNNWW	1010110111

O	WNNNWNNNWN	110101101001
P	NNWNWNNNWN	101101101001
Q	NNNNNNWWW	101010110011
R	WNNNNNWWN	110101011001
S	NNWNNNWWN	101101011001
T	NNNNWNWWN	101011011001
U	WWNNNNNNW	110010101011
V	NWWNNNNNW	100110101011
W	WWWNNNNNN	110011010101
X	NWNNWNNNW	100101101011
Y	WWNNWNNNN	110010110101
Z	NWWNWNNNN	100110110101
-	NWNNNNWNW	100101011011
.	WWNNNNWNN	110010101101
SPACE	NWWNNNWNN	100110101101
$	NWNWNWNNN	100100100101
/	NWNWNNNWN	100100101001
+	NWNNNWNWN	100101001001
%	NNNWNWNWN	101001001001
*	NWNNWNWNN	100101101101

2.5. CCD (Charge Couuple Device) Barcode Scanner

Kode – kode dalam bercode tidak dapat dibaca oleh komputer secara langsung, sehingga data *barcode* tersebut harus dikodekan terlebih dahulu menjadi format data yang dapat dibaca oleh komputer. Perangkat yang dapat

O	ИИИИИИИИИ	1001011010011
Q	ИИИИИИИИИ	1001011010011
R	ИИИИИИИИИ	1001011010011
S	ИИИИИИИИИ	1001011010011
T	ИИИИИИИИИ	1001011010011
U	ИИИИИИИИИ	1001010101011
V	ИИИИИИИИИ	1001101010011
W	ИИИИИИИИИ	1010101001011
X	ИИИИИИИИИ	1001011010011
Y	ИИИИИИИИИ	1001010110011
Z	ИИИИИИИИИ	1001101101001
[	ИИИИИИИИИ	1001010110101
\	ИИИИИИИИИ	1101010101101
]	ИИИИИИИИИ	1001101010110
^	ИИИИИИИИИ	1001001001001
_	ИИИИИИИИИ	1001001001001
`	ИИИИИИИИИ	1001001001001
a	ИИИИИИИИИ	1001001001001
b	ИИИИИИИИИ	1001001001001
c	ИИИИИИИИИ	1001001001001
d	ИИИИИИИИИ	1001001001001
e	ИИИИИИИИИ	1001001001001

3.2 CCD (Charge Coupla Device) Вигляд зсунки.

Код -- код даних розробо лідар одної дірки або контакт зсува

пишуть зарядів для умовах тягнуті плати фотодіодах зарядів одного

тягнуть фотон або умі одно одну дірки або контакт. Тягнуть умі одно

mengkodekan *barcode* menjadi format data tersebut dikenal sebagai *Barcode* Scanner atau *Barcode* Reader.

Sebuah *barcode* reader terdiri dari scanner, *decoder* dan kabel interface ke komputer. *Barcode* scanner men–*scan* kode – kode dalam *barcode* yang berupa susunan garis dan spasi, kemudian mengirimkannya ke *decoder*. *Decoder* akan menerjemahkan garis dan spasi pada output elektrikal dan data tersebut ditransmisikan ke komputer dengan format data yang dapat dibaca oleh komputer. Saat *barcode* di–*scan*, data dikirim ke komputer seperti penekanan *keyboard*.

Barcode scanner mempunyai berbagai tipe berdasarkan pada komponen yang digunakan dalam *transmit* dan *receive* pada pengkodean *barcode*. Dalam Tugas Akhir ini digunakan tipe *CCD Barcode* Scanner keluaran Metrologic.

CCD adalah salah satu jenis *barcode* scanner yang banyak digunakan secara luas. *CCD* merupakan satu deret *photocell* dalam satu *chip* semikonduktor. Tidak seperti pada satu *photocell* yang hanya dapat melihat satu bagian *barcode*, *CCD barcode* scanner dapat mengenali dan membaca keseluruhan bagian *barcode* dalam satu kali *scanning*. Sumber cahaya *barcode* dihasilkan oleh *LED* (*Light Emitting Diode*) yang terdapat pada bagian kepala *barcode scanner*. Lebar sensor CCD menentukan lebar maksimum dari *barcode* yang dapat dibaca. Jika lebar *barcode* melebihi lebar sensor, maka data dalam *barcode* tidak dapat di–*scan* dan dikodekan oleh *barcode scanner*.

Kualitas suatu *barcode scanner* dapat diketahui melalui reaksi kedipan *LED* saat pembacaan kode. Jika pembacaan dari sebuah *scanner* bagus

menghasilkan barcode menjadi format data tersebut dikenal sebagai Barcode Scanner atau Barcode Reader.

Sebuah barcode reader terdiri dari scanner, decoder dan kabel interface ke komputer. Barcode scanner non-scan code - kode dalam barcode yang berupa susunan garis dan spasi. kemudian menginterkannya ke decoder. Decoder akan menerjemahkan garis dan spasi pada output elektrikal dan data tersebut diterjemahkan ke komputer dengan format data yang dapat dibaca oleh komputer. Saat barcode di-scan, data dikirim ke komputer seperti pemakaian keyboard.

Barcode scanner mempunyai berbagai tipe berdasarkan pada komponen yang digunakan dalam prosesnyi dan receive pada pengkodean barcode. Dalam Tugas Akhir ini digunakan tipe CCD Barcode Scanner keluaran Metrologic.

CCD adalah salah satu jenis barcode scanner yang banyak digunakan secara luas. CCD menggunakan satu deret photocell dalam satu chip semikonduktor. Tidak seperti pada satu photocell yang hanya dapat membaca satu bagian barcode. CCD barcode scanner dapat mengenali dan membaca keseluruhan bagian barcode dalam satu kali scanning. Sumber cahaya barcode dihasilkan oleh LED (Light Emitting Diode) yang terdapat pada bagian kepala barcode scanner. Lebar sensor CCD menentukan lebar maksimum dari barcode yang dapat dibaca. Jika lebar barcode melebihi lebar sensor maka data dalam barcode tidak dapat di-scan dan dikodekan oleh barcode scanner.

Kualitas suatu barcode scanner dapat dikelahui melalui rasio kodipan LED saat pembacaan kode. Jika pembacaan dari sebuah scanner bagus

dan *valid*, maka *scanner* tersebut bereaksi dengan kedipan pendek pada *Led* – nya, dan sebaliknya bila bereaksi dengan kedipan panjang maka data yang dibaca kurang valid.

CCD barcode Scanner mempunyai berbagai macam tipe dengan spesifikasi yang berbeda pada tiap jenisnya, diantaranya : Tysso, Birch, Zebex, Metrologic, Datalogic, dll. Namun pada tugas akhir ini digunakan sensor *barcode* Metrologic karena pembacaan *barcode* tidak membutuhkan ketelitian yang tinggi, mudah dijumpai dipasaran serta harga yang terjangkau.

Model CCD *barcode scanner* ditunjukkan pada gambar 2.8. :

Gambar 2.8. Model *CCD Barcode* Scanner Metrologic [6]

Prinsip kerja *barcode* scanner adalah *LED* yang ada pada *barcode* scanner memancarkan sinar dan frekuensi, selanjutnya *photodiode* akan men – *scan* kode *barcode* yang terdiri dari susunan bar / garis dan spasi sehingga dapat diubah menjadi sinyal listrik yang berupa tegangan, komponen *interface* analog

dan nilai. maka scanner tersebut berarsi dengan kodipan pendek pada Acd – nya, dan sebaliknya bila berarsi dengan kodipan panjang maka data yang dibaca kurang valid.

CCD barcode Scanner merupakan berbagai macam tipe dengan spesifikasi yang berbeda pada tiap jenisnya, diantaranya : Tyxso, Birch, Xebex, Metrologic Danalogic, dll. Namun pada tugas akhir ini digunakan scanner barcode Metrologic karena pembacaan barcode tidak membutuhkan ketelitian yang tinggi, mudah dijumpai dipasaran serta harga yang terjangkau.

Model CCD barcode scanner ditunjukkan pada gambar 2.8. :

Gambar 2.8. Model CCD Barcode Scanner Metrologic [7]

Prinsip kerja barcode scanner adalah LED yang ada pada barcode scanner memancarkan sinar dan frekuensi, selanjutnya photodiode akan menerima kode barcode yang terdiri dari susunan bar / garis dan spasi sehingga dapat diubah menjadi sinyal listrik yang berupa tegangan. Komponen interface analog

akan menguatkan sinyal dan mengatur level sinyal yang dapat dikenali pada *digital controller* dan digital proses. Keseluruhan proses ini berlangsung di dalam memori dan dilakukan oleh mikrokontroller berdasarkan software pendukung yang telah disediakan. Hasil dari keseluruhan proses ini akan dibaca oleh komputer melalui I / O yang dalam hal ini berupa *keyboard controller*. Jadi hasil pembacaan *barcode* ini dikenali oleh komputer sebagai deretan sinyal yang serupa dengan penekanan *keyboard.*

Dalam pengoperasiannya, *CCD barcode* scanner memerlukan daya yang cukup kecil, selain itu tipe *CCD* dapat menerima daya yang berasal dari *port keyboard* komputer, sehingga tidak memerlukan daya eksternal. Dibandingkan dengan tipe – tipe yang lain, *CCD barcode* scanner memiliki daya tahan tinggi dan memerlukan perawatan yang lebih sederhana.

2.6. Mikrokontroler AT89S8252

Mikrokontroler AT89S8252 merupakan mikrokontroler 8 bit kompatibel dengan Standar industri MCS-51™ baik atas segi pemrograman maupun kaki tiap pin. Mikrokontroler AT89S8252 menpunyai 8 *Kby (FlashProgrammable and Read Only Memori*) pada dasarnya mikrokontroler adalah terdiri atas mikroprosesor, *timer*, dan *counter*, perangkat I/O dan internal memori. Mikrokontroler termasuk perangkat yang sudah didesain dalam chip tunggal.

Pada dasarnya mikrokontroler mempunyai fungsi yang sama dengan mikroprosesor yaitu untuk mengontrol suatu kerja system. Selain itu mikrokontroler juga dikemas dalam satu *chip* (*single chip*). Didalam

dalam mengunakan sinyal dan mengator level sinyal yang dapat dikenali pada digital controller dan digital proses. Kesalurahan proses ini berlangsung di dalam memori dan dilakukan oleh mikrokontroller berdasarkan software perdukung yang telah disediakan. Hasil dari keseluruhan proses ini akan dibaca oleh komputer melalui I/O yang dalam hal ini berupa keyboard controller. Jadi hasil pembacaan barcode ini dikenali oleh komputer sebagai deretan sinyal yang serupa dengan penekanan keyboard.

Dalam pengoperasianya, CCD barcode scanner memerlukan daya yang cukup kecil, selain itu tipe CCD dapat menerima daya eksternal. Dibandingkan dengan tipe – tipe yang lain, CCD barcode scanner memiliki daya tahan tinggi dan memerlukan perawatan yang lebih sederhana.

2.6. Microkontroler AT89S8252

Mikrokontroler AT89S8252 merupakan mikrokontroler 8 bit kompatibel dengan Standar industri MCS-51™ baik atas segi pemrograman maupun kaki tiap pin. Mikrokontroler AT89S8252 mempunyai 8 kbp (Flash)Programmable and Read Only Memory) pada dasarnya mikrokontroler adalah terdiri atas mikroprosesor, timer, dan counter, perangkat I/O dan internal memori. Mikrokontroler termasuk perangkat yang sudah didesain dalam chip tunggal.

Pada dasarnya mikrokontroler merupakan fungsi yang sama dengan microprocessor yaitu untuk mengontrol suatu kerja system. Selain itu mikrokontroler juga dikemas dalam satu chip (single chip). Didalam

mikrokontroler juga terdapat CPU, ALU, PC, SP, dan register seperti dalam mikroprosesor, tetapi juga ditambah dengan perangkat-perangkat lain seperti ROM, RAM, PIO, SIO, *counter* dan sebuah rangkaian *clock*. Mikroprosesor didesain dengan intruksi-intruksi lebih luas dan 8 bit instruksi yang digunakan membaca data instruksi dari internal memori ke ALU. Sebagai suatu system control mikrokontroler bila dibandingkan dengan mikroprosesor memiliki kemampuan dan segi ekonomis yang bisa diandalkan karena dalam mikrokontroler sudah terdapat RAM dan ROM. Sedangkan mikroprosesor didalamnya tidak terdapat keduanya. Terlihat bahwa mikrokontroler Atmel AT89S8252 memilki banyak fitur yang menguntungkan. Dipakainya Downloadable flash memori memungkinkan mikrokontroler ini bekerja sendiri tanpa diperlukan tambahan chip lainya. Sementara Flash memorinya mampu diprogram hingga seribu kali. Hal lain yang menguntungkan adalah system pemrograman menjadi lebih sederhana dan tidak memerlukan rangkaian yang rumit seperti rangkaian untuk memprogram produk Atmel lainya. Secara umum konfigurasi yang dimiliki mikrokontroler AT89S8252 adalah sebagai berikut :

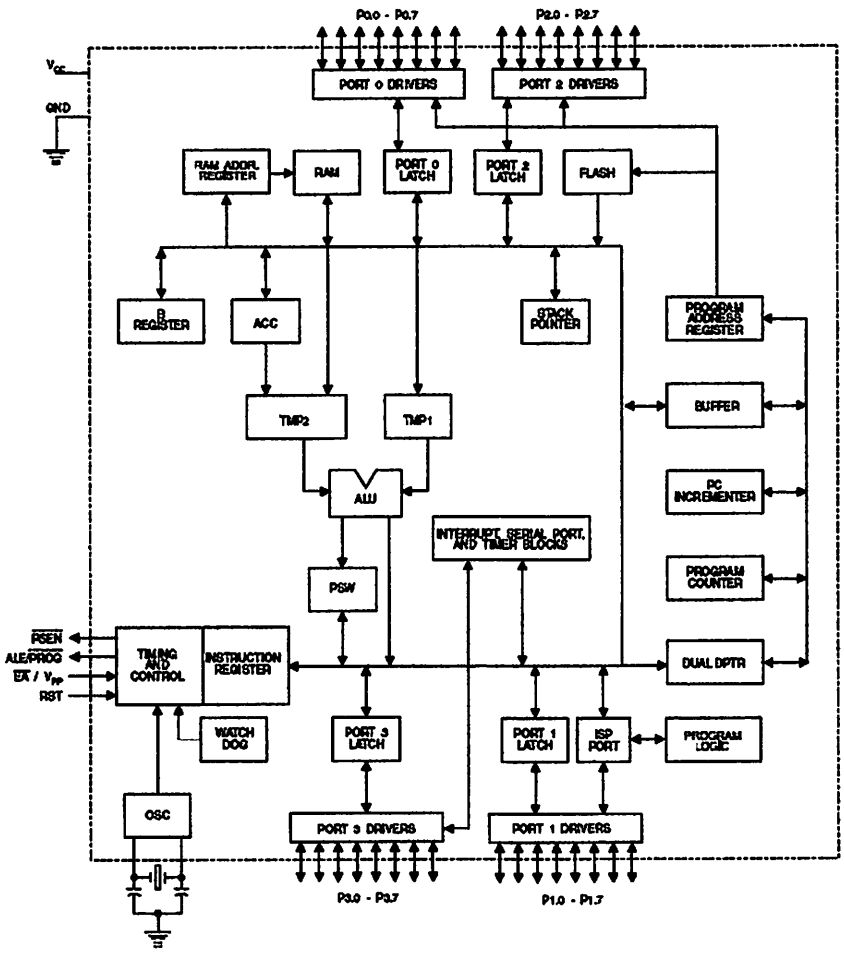
- Sebuah CPU 8 bit dengan menggunakan teknologi dari Atmel.
- 8K byte Downloadable Flash Memori.
- 2K byte EEPROM
- Sebuah *port* serial dengan control *full duplex* UART (Universal Asynchronous Receiver Transmiter.
- 256 byte RAM internal.
- 32 I/O yang dapat dipakai semua.
- 3 buah Timer/Counter 16 bit .

mikrokontroler juga terdapat CPU, ALU, PC, SP, dan register seperti dalam mikroprosesor, tetapi juga ditambah dengan perangkat-perangkat lain seperti ROM, RAM, PIO, SIO, counter dan sebuah rangkaian clock. Mikroprosesor didesain dengan instruksi-instruksi lebih luas dan 8 bit instruksi yang digunakan membaca data instruksi dari internal memori ke ALU. Sebagai suatu system control mikrokontroler bila dibandingkan dengan mikroprosesor memiliki kemampuan dan segi ekonomis yang bisa dimanfaatkan karena dalam mikrokontroler sudah terdapat RAM dan ROM. Sedangkan mikroprosesor didalamnya tidak terdapat keduanya. Terdiri dari bahwa mikrokontroler Atmel AT89S8252 memiliki banyak fitur yang menguntungkan. Diperlainya Downloadable flash memori memungkinkan mikrokontroler ini bekerja sendiri tanpa diperlukan tambahan chip lainya. Semestara Flash memorinya mampu diprogram hingga seribu kali. Hal lain yang menguntungkan adalah system pemrograman menjadi lebih sederhana dan tidak memerlukan rangkaian yang rumit seperti rangkaian untuk memprogram produk Atmel lainya. Secara umum konfigurasi yang dimiliki mikrokontroler AT89S8252 adalah sebagai berikut :

- Sebuah CPU 8 bit dengan menggunakan teknologi dari Atmel.
- 8K byte Downloadable Flash Memori.
- 2K byte EEPROM
- Sebuah port serial dengan control full duplex UART (Universal Asynchronous Receiver Transmitter.
- 256 byte RAM internal.
- 32 I/O yang dapat diprakan sendiri.
- 3 buah Timer/Counter 16 bit .

- SPI Serial Interface.
- Programmable Watchdog Timer .
- Dual Data Pointer.
- Frekuensi kerja 0 sampai 24 MHz
- Tegangan operasi 2,7 Volt sampai 6Volt.
- Kemampuan melaksanakan operasi perkalian, pembagian, dan operasi *Bolean* (bit)

Sedangkan untuk blok diagram AT89S8252 diperlihatkan dalam gambar 2-9 :



Gambar 2.9. Blok Diagram AT89S8252 [2]

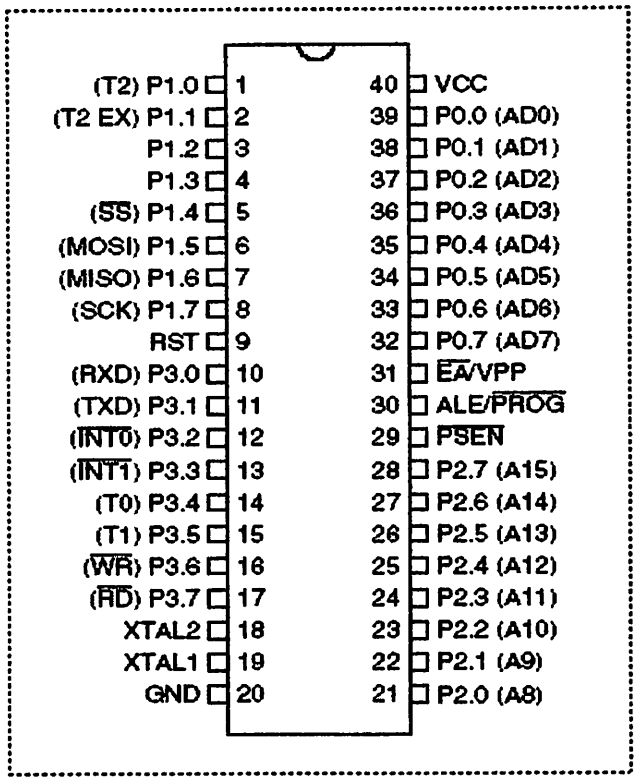
- SPI Serial Interface.
- Programmable Watchdog Timer.
- Dual Data Pointer.
- Frekuensi kerja 0 sampai 24 MHz
- Tegangan operasi 2,7 Volt sampai 6Volt.
- Kemampuan melaksanakan operasi perkalian, pembagian, dan operasi Boleen (bit)

Sedangkan untuk blok diagram AT89S8252 diperlihatkan dalam gambar 2-9 :

Gambar 2.9. Blok Diagram AT89S8252 [2]

2.6.1. Penjelasan Funsi Pin AT89S8252

Mikrktroler AT89S8252 mempunyai 40 pin seperti yang ditunjukkan dalam gambar 2-10 Fungsi-fungsi pin dijelaskan sebagai berikut :



Gambar 2.10. Susunan Pin AT89S8252 [2]

- Pin 1 sampai 8

  Port 1 yang terdiri atas pin 1 sampai 8 merupakan saluran masukan/keluaran dua arah.

- Pin 9

  RST merupakan saluran dua masukan untuk m*ereset* mikrokontroler dengan cara memberi masukan logika tinggi.

2.6.1. Penjelasan Fungsi Pin AT89S8252

Mikrokontroler AT89S8252 mempunyai 40 pin seperti yang ditunjukkan dalam gambar 2-10 Fungsi-fungsi pin dijelaskan sebagai berikut :

Gambar 2.10. Susunan Pin AT89S8252 [2]

- Pin 1 sampai 8

Port 1 yang terdiri atas pin 1 sampai 8 merupakan saluran masukan/keluaran dua arah.

- Pin 9

RST merupakan saluran dua masukan untuk menyet mikrokontroler dengan cara memberi masukan logika tinggi.

- Pin 10 sampai 17

  Port 3 yang terdiri atas pin 10 sampai pin 17 merupakan saluran masukan/keluaran dua arah dan mempunyai fungsi khusus seperti yang terlihat dalam tabel 2-2

- Pin 18 dan 19

  $XTAL_1$ dan $XTAL_2$ merupakan saluran untuk mengatur pewaktuan system. Untuk pewaktuan dapat menggunakan pewaktuan internal maupun eksternal .

- Pin 20

  Vss merupakan hubungan ke *ground* dari rangkaian.

- Pin 21 sampai 28

  Port 2 yang terdiri atas pin 21 sampai 28 merupakan saluran masukan/keluaran dua arah. *Port* ini mengeluarkan 8 bit bagian alamat tinggi (A8-A15) selama pengambilan instruksi dari memori program eksternal dan pengambilan data memori eksternal yang menggunakan mode pengalamatan 16 bit.

- Pin 29

  PSEN (*Program Store Enable*) merupakan sinyal baca untuk mengaktifkan memori program eksternal.

- Pin 30

  ALE/PROG (*Address Latch Enable*) merupakan pulsa yang berfungsi untuk menahan alamat rendah (A0-A7) dalam *port* 0, selama proses baca/tulis memori eksternal. Frekuensi ALE adalah 1/6 kali rekuensi osilator, dan digunakan sebagai pewaktu. Pin ini juga berfungsi sebagai

- Pin 10 sampai 17

  Port 3 yang terdiri atas pin 10 sampai pin 17 merupakan saluran masukan/keluaran dua arah dan mempunyai fungsi khusus seperti yang terlihat dalam tabel 2-2

- Pin 18 dan 19

  XTAL1 dan XTAL2 merupakan saluran untuk mengatur pewaktuan system. Untuk pewaktuan dapat menggunakan pewaktuan internal maupun eksternal .

- Pin 20

  Vss merupakan hubungan ke ground dari rangkaian.

- Pin 21 sampai 28

  Port 2 yang terdiri atas pin 21 sampai 28 merupakan saluran masukan/keluaran dua arah. Port ini mengeluarkan 8 bit bagian alamat tinggi (A8-A15) selama pengambilan instruksi dari memori program eksternal dan pengambilan data memori eksternal yang menggunakan mode pengalamatan 16 bit.

- Pin 29

  PSEN (Program Store Enable) merupakan singal baca untuk mengaktifkan memori program eksternal.

- Pin 30

  ALE/PROG (Address Latch Enable) merupakan pulsa yang berfungsi untuk memulan alamat rendah (A0-A7) dalam port 0, selama proses pengambilan memori eksternal. Frekuensi ALE adalah 1/6 kali frekuensi osilator dan digunakan sebagai pewaktu. Pin ini juga berfungsi sebagai

saluran program selama dilakukan pemrograman jika menggunakan memori program eksternal.

- Pin 31

  EA/VPP (External Access Enable) untuk mengatur penggunaan memori program eksternal dan internal. Pin ini harus dihubungkan dengan ground bila menggunakan memori program eksternal dan dihubungkan dengan VPP sebesar 12 Volt jika menggunakan memori program eksternal.

- Pin 32 sampai 39

  Port 0 yang terdiri atas pin 32 sampai 39 merupakan saluran masukan/keluaran. *Port* 0 merupakan saluran alamat rendah (A0-A7) yang dimultipleks dengan saluran *bus* data (D0-D7).

- Pin 40

  Vcc merupakan saluran masukan untuk catu daya positi sebesar 5 Volt DC dengan toleransi kurang lebih 10 %.

2.6.2. Masukan dan Keluaran

Untuk saluran dan keluaran terdapat 4 buah *port* yang masing-masing 8 bit. Saluran ini bersifat dua arah (*bidirectional*) yang berarti dapat difungsikan sebagai masukan atau keluaran, serta dapat dialamati per bit. *Port* 3 selain digunakan sebagai *port* masukan dan keluaran juga dapat digunakan sebagai fungsi pengganti sebagaimana yang terdapat dalam tabel 2-2. Sedangkan AT89S8252 memiliki fitur tambahan yang terdapat pada *port 1* seperti pada tabel 2-3.

saluran program selama dilakukan pemrograman jika menggunakan memori program eksternal.

- Pin 31

EA/VPP (External Access Enable) untuk mengatur penggunaan memori program eksternal dan internal. Pin ini harus dihubungkan dengan ground bila menggunakan memori program eksternal dan dihubungkan dengan VPP sebesar 12 Volt jika menggunakan memori program eksternal.

- Pin 32 sampai 39

Port 0 yang terdiri atas pin 32 sampai 39 merupakan saluran masukan/keluaran. Port 0 merupakan saluran alamat terdub (A0-A7) yang dimultipleks dengan saluran bus data (D0-D7).

- Pin 40

Vcc merupakan saluran masukan untuk catu daya positi sebesar 5 Volt DC dengan toleransi kurang lebih 10 %.

2.6.2. Masukan dan Keluaran

Untuk saluran dan keluaran terdapat 4 buah port yang masing-masing 8 bit. Saluran ini bersifat dua arah (bidirectional) yang berarti dapat difungsikan sebagai masukan atau keluaran. serta dapat dialamati per bit. Port 3 selain digunakan sebagai port masukan dan keluaran juga dapat digunakan sebagai fungsi pengganti sebagaimana yang terdapat dalam tabel 2-2. Sedangkan AT89S8252 memiliki fitur tambahan yang terdapat pada port 1 seperti pada tabel 2-3.

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

Tabel 2.2. Fungsi Pengganti *Port 3* [2]

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

Tabel 2.3. Fungsi khusus pada *port 1* AT89S8252 [2]

2.6.3. Osilator

Jantung dari AT89S8252 adalah rangkaian yang membangkitkan pulsa clock yang mesinkronkan semua operasi internal. Mikrokontroler AT89S8252 memiliki osilator internal (*on chip oscillator*) yang dapat digunakan sebagai sumber pewaktu (*clock*) bagi CPU. Untuk menggunakan internal diperlukan sebuah kristal atau resonator keramik antara pin XTAL1 dan pin

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

Tabel 2.2. Fungsi Pengganti Port 3 [2]

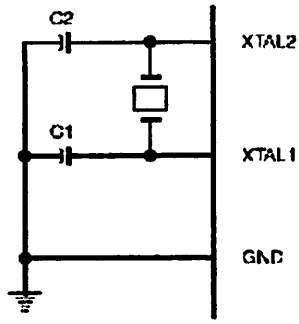
Port Pin	Alternate Function
P1.0	T2 (external count input) Timer/Counter 2, clock-out
P1.1	T2EX Timer/Counter 2 capture/reload trigger and direction control
P1.4	SS Slave port select input
P1.5	MOSI Data line used to send data in and receive data out when SPI is in master mode
P1.6	MISO Data line used to send data out and receive data in when SPI is in master mode
P1.7	SCK Clock output when SPI is in master mode or clock input when SPI is in slave mode

Tabel 2.3. Fungsi khusus pada port 1 AT89S8252 [2]

2.6.3. Osilator

Internal dari AT89S8252 adalah rangkaian yang membangkitkan pulsa clock yang mesinkronkan semua operasi internal. Mikrokontroler AT89S8252 memiliki osilator internal (or chip oscillator) yang dapat digunakan sebagai sumber pewaktu (clock) bagi CPU. Untuk menggunakan internal digerbangkan sebuah kristal atau resonator keramik antara pin XTAL1 dan pin

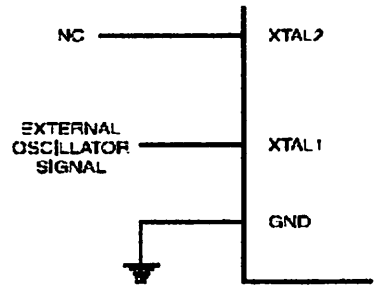
XTAL2 dan sebuah kapasitor ke ground. Konfigurasinya dapat dilihat pada gambar berikut.



Gambar 2.11. Konfigurasi Osilator Menggunakan Kristal [2]

Nilai C1 dan C2 adalah 10 pF – 30 pF bila menggunakan kristal, dan bernilai 10 pF – 40 pF bila menggunakan resonator keramik.

Untuk penggunaan dengan external clock, XTAL2 harus dibiarkan dalam kondisi tidak terhubung. Konfigurasinya dapat dilihat pada gambar berikut ini :



Gambar 2.12. Konfigurasi Osilator Dengan External Oscilator Signal [2]

2.6.4. Reset

Rangkaian *power on reset* diperlukan untuk m*ereset* mikrokontroler secara otomatis setiap catu daya *on*. Gambar 2-13 menunjukkan rangkaian *power*

XTAL2 dan sebuah kapasitor ke ground. Konfigurasinya dapat dilihat pada gambar berikut.

Gambar 2.11. Konfigurasi Osilator Menggunakan Kristal [2]

Nilai C1 dan C2 adalah 10 pF – 30 pF bila menggunakan kristal. dan bernilai 10 pF – 40 pF bila menggunakan resonator keramik.

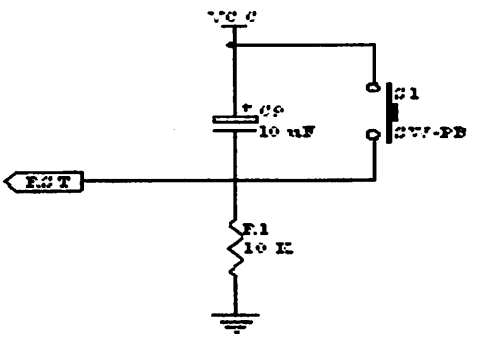
Untuk penggunaan dengan external clock, XTAL2 harus dibiarkan dalam kondisi tidak terhubung. Konfigurasinya dapat dilihat pada gambar berikut ini :

Gambar 2.12. Konfigurasi Osilator Dengan External Oscillator Signal [2]

2.6.4. Reset

Rangkaian power on reset dipergunakan untuk mereset mikrokontroler secara otomatis setiap catu daya on. Gambar 2-13 menunjukkan rangkaian power

on reset. Ketika catu daya diaktifkan, rangkaian *reset* menahan logika tinggi pin RST dengan jangka waktu yang ditentukan oleh besarnya pengisian muatan C.



Gambar 2.13. Rangkaian *Reset* [2]

2.6.5. Data Memori (EEPROM) Dan RAM

Berbeda dengan mikrokontroler standard MCS-51, mikrokontroler Atmel AT89S8252 juga dilengkapi dengan data memori yang berupa EEPROM (*Electrically Erasable Programmable Read Only Memory*). EEPROM yang dimaksud ini besarnya 2 *klo byte* (2K) dan dipakai untuk penyimpanan data.

EEPROM *on-chip* ini diakses dengan mengeset bit EEMEN pada register WMCON pada alamat 96H. Alamat EEPROM ini adalah 000H sampai 7FFH. Intruksi move digunakan untuk mengakses EEPROM internal ini. Bit EEMWE pada register WMCON harus diset ke-1 sebelum seberang lokasi pada EEPROM dapat ditulisi. Program pengguna harus *mereset* bit EEMWE ke '0'jika proses penulisan ke EEPROM tidak diperlukan lagi. Proses penulisan ke EEPROM dapat dilihat dengan membaca bit RDY/BSY pada SFR WMCON. Jika bit ini berlogika rendah maka berarti penulisan EEPROM sedang berlangsung,

ow reset. Kerika catu daya diaktifkan, rangkaian reset memakan logika tinggi pin RST dengan jangka waktu yang ditentukan oleh besarnya pengisian muatan C.

Gambar 2.13. Rangkaian Reset [2]

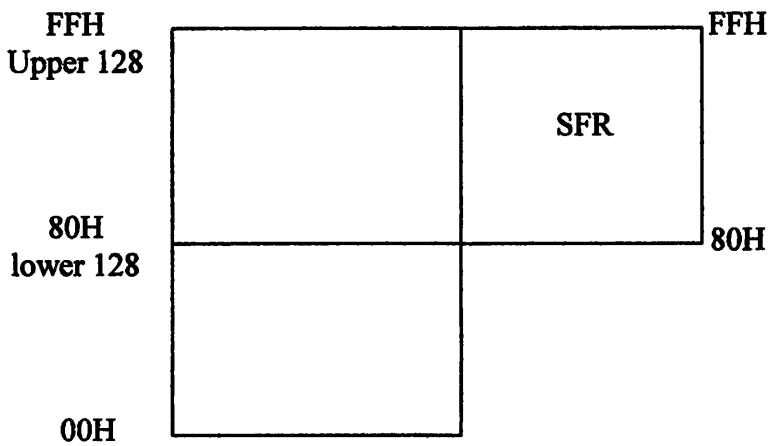
2.6.5. Data Memori (EEPROM) Dan RAM

Berbeda dengan mikrokontroler standard MCS-51. mikrokontroler Atmel AT89S8252 juga dilengkapi dengan data memori yang berupa EEPROM (Electrically Erasable Programmable Read Only Memory). EEPROM yang dimaksud ini besarnya 2 Kilo Byte (2K) dan dipakai untuk penyimpanan data.

EEPROM on-chip ini diakses dengan mengeset bit EEMEN pada register WMCON pada alamat 96H. Alamat EEPROM ini adalah 0000 sampai 7FFH. Instruksi move digunakan untuk mengakses EEPROM internal ini. Bit EEMWE pada register WMCON harus diset ke-1 sebelum sebarang lokasi pada EEPROM dapat ditulisi. Program penggunn harus mereset bit EEMWE ke '0' jika proses penulisan ke EEPROM tidak diperlukan lagi. Proses penulisan ke EEPROM dapat dilihat dengan membaca bit RDYBSY pada SFR WMCON. Jika bit ini berlogika rendah maka berarti penulisan EEPROM sedang berlangsung.

jika bit ini berlogika tinggi berarti penulisan sudah selesai dan penulisan lain dapat dimulai lagi.

Sedangkan RAM yang ada pada mikrokontroler AT89S8252 adalah berkapasitas 256 *byte* dan kompatibel dengan RAM yang ada pada mikrokontroler standard MCS-51.



Gambar 2.14. Memori Data Eksternal [2]

Pada *lower* 128 lokasi memori dibagi menjadi 3 bagian :

1. Register bank 0 – 3

   Lokasi bank register dimulai dari alamat 00H – 1 H yang terdiri dari 32 *bytes*. Register bank ini terdiri dari 4 buah register 8 bit yang dapat dipilih melalui pengaturan *program status word* register.

2. Bit *Addressing*

   Terdiri dari 16 *bytes* yang dimulai dari 20H – 2FH. Masing-masing dari 128 bit lokasi ini dapat dialamati secara langsung yaitu dari 00H sampai 7FH.

3. *Scratch Pad Area*

jika bit ini berlogika tinggi berarti pemilihan sudah selesai dan perulisan lain dapat dimulai lagi.

Sedangkan RAM yang ada pada mikrokontroler AT89S8252 adalah berkapasitas 256 byte dan kompatibel dengan RAM yang ada pada mikrokontroler standard MCS-51.

Gambar 2.14. Memori Data Eksternal [2]

Pada lower 128 lokasi memori dibagi menjadi 3 bagian :

1. Register bank 0 – 3

Lokasi bank register dimulai dari alamat 00H – 1 FH yang terdiri dari 32 byte. Register bank ini terdiri dari 4 buah register 8 bit yang dapat dipilih melalui pengaturan program status word register.

2. Bit Addressing

Terdiri dari 16 byte yang dimulai dari 20H – 2FH. Masing-masing dari 128 bit lokasi ini dapat dialamati secara langsung yaitu dari 00H sampai 7FH.

3. Scratch Pad Area

Lokasi dari alamat 30H – 7FH atau sebanyak 80 *bytes* yang dapat digunakan sebagai alamat bagi RAM.

2.6.6. Special Function Register (SFR)

Special Function Register merupakan register dengan tugas khusus. SFR pada mikrokontroler AT89S8252 kompatibel dengan mikrokontroler keluarga MCS-51 dan memiliki alamat 80H - FFH sehingga terdapat 128 lokasi alamat untuk SFR. Namun demikian pada mikrokontroler ini tidak berarti memiliki SFR sebanyak 128 buah. Berikut ini adalah gambar letak dari lokasi alamat SFR.

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 0000010X			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00000XXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DPOL 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0X000000	87H

Tabel 2.4. AT89S8252 SFR Map dan Reset Value [2]

Selain itu mikrokontroler AT89S8252 memiliki tambahan SFR . Hal ini tak lain adalah karena terdapatnya tambahan fitur pada mikrokontroler ini. SFR tambahan ini meliputi : T2CON (Timer 2 Register dengan alamat 0C8H),

Lokasi dari alamat 30H – 7FH atau sebanyak 80 byte yang dapat digunakan sebagai alamat bagi RAM.

2.6.6. Special Function Register (SFR)

Special Function Register merupakan register dengan tugas khusus. SFR pada mikrokontroler AT89S8252 kompatibel dengan mikrokontroler keluarga MCS-51 dan memiliki alamat 80H - FFH sehingga terdapat 128 lokasi alamat untuk SFR. Namun demikian pada mikrokontroler ini tidak berarti memiliki SFR sebanyak 128 buah. Berikut ini adalah gambar letak dari lokasi alamat SFR.

Tabel 2.4. AT89S8252 SFR Map dan Reset Value [1]

Selain itu mikrokontroler AT89S8252 memiliki tambahan SFR . Hal ini tak lain adalah karena terdapatnya tambahan fitur pada mikrokontroler ini. SFR tambahan ini meliputi : T2CON (Timer 2 Register dengan alamat 0C8H),

T2MOD (Timer 2 Mode dengan alamat 0C9H), WMCON (*Watcdog and Memory Control* Register dengan alamat 96H), SPCR (SPI *Control* Register dengan alamat D5H), SPSR (SPI Status Register dengan alamat AAH), SPDR (SPI Data Register dengan alamat 86H).

- SFR untuk Timer 2

  Mikrokontroler AT89S8252 terdapat tambahan sebuah Timer/Counter yang diberi nama timer 2 (sehingga AT89S8252 memiliki 3 Timer /Counter yaitu Timer/Counter 0, Timer/Counter 1, Timer/Counter 2). Pada Timer/Counter 2 ini dikendalikan oleh *Special Function* Register yang bernama T2CON (Timer 2 Control), T2MOD (Timer @ MODE) dan sepasang register RCAP2H, RCAP2L merupakan register *capture/reload* untuk Timer 2 dalam 16 bit *capture mode/auto reload mode.*

- SFR untuk *Watchdog* Memori.

  Untuk menggunakan *Watchdog* timer atau memori, maka dapat dilakukan dengan mengatur SFR yang bernama WMCON dengan alamat 96H.

- SFR pengontrol SPI

  Berbeda dengan mikrokontroler MCS-51, AT89S8252 memiliki fasilitas SPI (*Serial Perpheral Interface*). Fasilitas ini memungkinkan transfer data kecepatan tinggi secara sinkron antara mikrokontroler dengan peripheral atau antar mikrokontroler AT89S8252. Fitur ini meliputi :

  a. Full Duplek, 3 kawat dengan transfer data secara sinkron .
  b. Operasi Master atau Slave.
  c. Frekuensi maksimum 6 MHz.
  d. 4 bit rate terprogram.

T2MOD (Timer 2 Mode dengan alamat 0C9H). WMCON (Wavedog and Memory Control Register dengan alamat 96H). SPCR (SPI Control Register dengan alamat D5H). SPSR (SPI Status Register dengan alamat AAH). SPDR (SPI Data Register dengan alamat 86H).

- SFR untuk Timer 2

Mikrokontroler AT89S8252 terdapat tambahan sebuah Timer/Counter yang dikeri nama timer 2 (sehingga AT89S8252 memiliki 3 Timer /Counter yaitu Timer/Counter 0, Timer/Counter 1, Timer/Counter 2). Pada Timer/Counter 2 ini dikendalikan oleh Special Function Register yang bernama T2CON (Timer 2 Control). T2MOD (Timer / MODE) dan sepasang register RCAP2H, RCAP2L merupakan register cadangan reload untuk Timer 2 dalam 16 bit capture modecompare reload mode.

- SFR untuk Wavedog Memori.

Untuk menggunakan Wavedog timer atau memori, maka dapat dikendalikan dengan mengatur SFR yang bernama WMCON dengan alamat 96H.

- SFR pengontrol SPI

Berbeda dengan mikrokontroler MCS-51, AT89S8252 memiliki fasilitas SPI (Serial Peripheral Interface). Fasilitas ini memungkinkan transfer data kecepatan tinggi secara sinkron antara mikrokontroler dengan peripheral atau antar mikrokontroler AT89S8252. Fitur ini meliputi :

a. Full Duplex, 3 kawat dengan transfer data secara sinkron.

b. Operasi Master atau Slave.

c. Frekuensi maksimum 6 MHz.

d. 4 bit rate terprogram.

2.6.7. Timer dan Counter

Dalam mikrokontroler AT89S8252 terdapat tiga buah pewaktu/pencacah (timer/counter 16) 16 bit yang dapat diatur melalui perangkat lunak, yaitu pewaktu / pencacah 0 dan pewaktu / pencacah 1. Timer/counter ini diatur oleh *special function* register yaitu Timer/Counter *Control* (TCON alamat 88H), dan Timer/Counter Mode *Control* (TMOD alamat 89H). Selain itu nilai byte bawah dan byte atas dari Timer/Counter disimpan dalam register TL dan TH.

Jika difungsikan sebagai Timer, maka akan menggunakan system clock sebagai sumber masukan pulsanya. Jika sebagai Counter (pencacah), maka akan menggunakan pulsa dari luar (eksternal) sebagai masukan pulsanya. Pada Port 3 terdapat fungsi khusus yaitu TO (masukan luar untuk Timer/Counter 0) dan T1 (masukan luar untuk Timer/Counter 1). Pemilihan mode Timer/Counter dikontrol oleh register TMOD. Dengan memberikan nilai tertentu pada register TMOD dapat dipilih mode operasi untuk Timer/Counter 0 dan Timer/Counter 1 seperti terlihat dalam Tabel 2-5.

Mode	Timer/Counter 0	Timer/Counter 1
0	13 bit Timer	13 bit Timer
1	16 bit Timer	16 bit Timer
2	8 bit auto-reload	8 bit auto-reload
3	28 bit timer	Tidak bekerja

Tabel 2.5. Mode Operasi Timer/Counter 0 dan 1 [2]

2.6.7. Timer dan Counter

Dalam mikrokontroler AT89S8252 terdapat tiga buah pewaktu/pencacah (timer/counter 16) 16 bit yang dapat diatur melalui perangkat lunak. yaitu pewaktu / pencacah 0 dan pewaktu / pencacah 1. Timer/counter ini diatur oleh special function register yaitu TimerCounter Control (TCON alamat 88H), dan TimerCounter Mode Control (TMOD alamat 89H). Selain itu nilai byte bawah dan byte atas dari TimerCounter disimpan dalam register TL dan TH.

Jika difungsikan sebagai Timer, maka akan menggunakan system clock sebagai sumber masukan pulsanya. Jika sebagai Counter (pencacah), maka akan menggunakan pulsa dari luar (eksternal) sebagai masukan pulsanya. Pada Port 3 terdapat fungsi khusus yaitu T0 (masukan luar untuk Timer/Counter 0) dan T1 (masukan luar untuk Timer/Counter 1). Pemilihan mode Timer/Counter dikontrol oleh register TMOD. Dengan memberikan nilai tertentu pada register TMOD dapat dipilih mode operasi untuk Timer/Counter 0 dan Timer/Counter 1 seperti terlihat dalam Tabel 2-5.

Mode	Timer/Counter 0	Timer/Counter 1
0	13 bit Timer	13 bit Timer
1	16 bit Timer	16 bit Timer
2	8 bit auto-reload	8 bit auto-reload
3	2X bit timer	Tidak beroperasi

Tabel 2.5. Mode Operasi Timer/Counter 0 dan 1 [5]

Pada mikrokontroler AT89S8252 terdapat tambahan Timer 2. Timer yang lain adalah Timer 0 dan Timer 1. Timer 2 ini merupakan Timer/Counter 16 bit dan memiliki 3 mode operasi yaitu *capture, auto reload* (*up down counting*) dan baud rate generator. Untuk memilih mode ini dilakukan dengan mengatur bit pada SFR T2CON (Timer 2 Control Register). Timer 2 ini terdiri dari 2 buah timer 8 bit register yaitu TH2 dan TL2. pada fungsi Timer, register TL2 dinaikkan (increment) tiap siklus mesin. Karena siklus mesin terdiri dari 12 periode osilasi, maka count rate menjadi 1/12 dari frekuensi osilator. Sedangkan pada fungsi Counter, register dinaikkan berdasarkan tanggapan adanya transisi tinggi ke rendah pada pena yang bersesuaian (dalam hal ini pin T2 atau P1.0). Tabel berikut menunjukkan mode operasi yang dapat dijalankan pada timer 2.

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

Tabel 2.6. Mode Operasi Timer 2 [2]

RCLK = Receive clock enable. Jika diset menyebabkan serial port menggunakan pulsa overflow Timer 2 sebagai detak penerimaan pada serial port. Jika RCLK = 0 Timer 1 yang digunakan.

TCLK = Transmit clok enable. Jika diset menyebabkan serial port menggunakan pulsa overflow Timer 2 sebagai detak pengiriman. Jika TCLK = 0 pulsa overflow timer 1 yang digunakan.

Pada mikrokontroler AT89S8252 terdapat tambahan Timer 2. Timer yang lain adalah Timer 0 dan Timer 1. Timer 2 ini merupakan Timer/Counter 16 bit dan memiliki 3 mode operasi yaitu capture, auto reload (up down counting) dan baud rate generator. Untuk memilih mode ini dilakukan dengan mengatur bit pada SFR T2CON (Timer 2 Control Register). Timer 2 ini terdiri dari 2 buah timer 8 bit register yaitu TH2 dan TL2. pada fungsi Timer, register TL2 dimasukkan (increment) tiap siklus mesin. Karena siklus mesin terdiri dari 12 periode osilasi, maka count rate merupadi 1/12 dari frekuensi osilator. Sedangkan pada fungsi Counter, register dimasukkan berdasarkan tanggapan adanya transisi tinggi ke rendah pada pena yang bersesuaian (dalam hal ini pin T2 atau P1.0). Tabel berikut menunjukkan mode operasi yang dapat dijalankan pada timer 2.

RCLK + TCLK	CP/RL2	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	OFF

Tabel 2.6. Mode Operasi Timer 2 [7]

RCLK = Receive clock enable. Bila diset menyebabkan serial port menggunakan pulsa overflow Timer 2 sebagai detak penerimaan pada serial port. Jika RCLK = 0 Timer 1 yang digunakan.

TCLK = Transmit clock enable. Jika diset menyebabkan serial port menggunakan pulsa overflow Timer 2 sebagai detak pengiriman. Jika TCLK = 0 pulsa overflow timer 1 yang digunakan.

CP/RL2 = Pemilihan capture/Reload. Jika diset maka proses capture yang terjadi sedangkan jika bit ini diclear maka proses reload.

TR2 = Bit untuk mengatur start/stop untuk timer 2 jika TR2 = 1 Timer akan aktif.

2.6.8. Idle Mode

Saat *Idle Mode* mikrokontroler tidak melakukan apa-apa. Tetapi peralatan lain yang terhubung tetap aktif. Kondisi ini dapat dihentikan dengan sebuah *interrupt* atau dengan me-*reset* system.

2.6.9. Sistem Interupt

Mikrokontroler AT89S8252 mempunyai 6 buah sumber interrupt yang dapat membangkitkan permintaan interrupt, yaitu INTO, INT1, T0, T1, T2 dan port serial.

Saat terjadinya interrupt, mikrokontroler secara otomatis akan menuju ke *sub rutin* pada alamat tersebut. Setelah interrupt service selesai dikerjakan, mikrokontroler akan mengerjakan program semula. Dua sumber interrupt external adalah INTO dan INT1, dimana kedua interupsi eksternal akan aktif atau aktif transisi tergantung isi dari IT0 dan IT1 pada register TCON. Interupsi T0, T1, T2 aktif pada saat timer yang sesuai mengalami *roll over*, interupsi serial dibangkitkan dengan melakukan operasi OR pada R1 dan T1. Tiap-tiap sumber interupsi dapat *enable* atau *disable* secara otomatis.

Tingkat prioritas semua sumber interupsi dapat diprogram sendiri-sendiri dengan set atau *clear bit* pada SFRS IP (*interrupt Priority*).

CPRL2 = Pemilihan capture/Reload. Jika diset maka proses capture yang terjadi sedangkan jika bit ini diclear maka proses reload.

TR2 = Bit untuk mengatur start/stop untuk timer 2 jika TR2 = 1 Timer akan aktif.

2.6.8. Idle Mode

Saat Idle Mode mikrokontroler tidak melakukan apa-apa. Tetapi peralatan lain yang terhubung tetap aktif. Kondisi ini dapat dikontrikan dengan sebuah interrupt atau dengan me-reset system.

2.6.9. Sistem Interupt

Mikrokontroler AT89S8252 mempunyai 6 buah sumber interrupt yang dapat memberikan permintaan interrupt. yaitu INT0, INT1, T0, T1, T2 dan port serial.

Saat terjadinya interrupt, mikrokontroler secara otomatis akan menuju ke vektor pada alamat tersebut. Setelah interrupt service selesai dikerjakan, mikrokontroler akan mengerjakan program semula. Dua sumber interrupt external adalah INT0 dan INT1, dimana kedua interrupsi external akan aktif atau aktif transisi tergantung isi dari IT0 dan IT1 pada register TCON. Interrupsi T0, T1, T2 aktif pada saat timer yang sesuai mengalami roll over. interrupsi serial dibangkitkan dengan melakukan operasi OR pada RI dan TI. Tiap-tiap sumber interrupsi dapan enable atau disable secara otomatis.

Tingkat prioritas semua sumber interupsi dapat diprogram sendiri-sendiri dengan set atau clear bit pada SFR IP (interrupt Priority).

(MSB)(LSB)

EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	SPI and UART interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

Tabel 2.7. Alamat Sumber Interupsi [2]

2.7. LCD (*Liquid Crystal Display*)

Liquid Crystal Display (LCD) merupakan komponen elektronika yang digunakan untuk menampilkan suatu karakter baik itu berupa angka, huruf atau karakter tertentu sehingga tampilan tersebut dapat dilihat secara *visual.* Pemakaian LCD sebagai indicator tampilan banyak digunakan dikarenakan daya yang dibutuhkan LCD relatif kecil, disamping itu dapat juga menampilkan angka, huruf atau symbol dan karakter tertentu.

LCD terdiri atas tumpukan tipis atau sel dari dua lembar kaca dengan pinggiran tertutup rapat. Antara dua lembar kaca tersebut diberi bahan crystal cair (*liquid crystal*) yang tembus cahaya. Permukaan luar masing-masing keeping kaca mempunyai lapisan yang tembus cahaya seperti oksida timah (*tin oxide*) atau oksida indium (*indium oxide*). Sel mempunyai ketebalan 1×10^{-3} meter dan diisi dengan crystal cair.

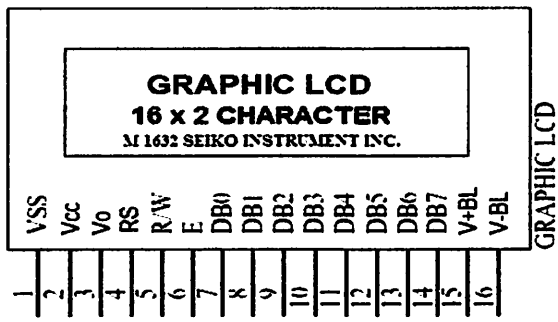
Tabel 2.7. Alamat Number Interupsi [2]

2.7. LCD (Liquid Crystal Display)

Liquid Crystal Display (LCD) merupakan komponen elektronika yang digunakan untuk menampilkan suatu karakter baik itu berupa angka, huruf atau karakter tertentu sehingga tampilan tersebut dapat dilihat secara visual. Pemakaian LCD sebagai indicator tampilan banyak digunakan dikarenakan daya yang dibutuhkan LCD relatif kecil, disamping itu dapat juga menampilkan angka, huruf atau symbol dan karakter tertentu.

LCD terdiri atas tumpukan tipis atau sel dari dua lembar kaca dengan pinggiran tertutup rapat. Antara dua lembar kaca tersebut diberi bahan crystal cair (liquid crystal) yang lembus cahaya. Permukaan luar masing-masing keeping kaca merupunyai lapisan yang lembus cahaya seperti oksida timah (tin oxide) atau oksida indium (indium oxide). Sel mempunyai ketebalan 1×10^{-5} meter dan diisi dengan crystal cair.

Crystal cair adalah suatu bahan yang akan mengalir seperti sebuah cairan tetapi struktur molekulnya seperti benda padat. Pada LCD terdapat suatu unit penghamburan cahaya, yang mana terdapat suatu proses *neumatic liquid crystal*. Pada proses tersebut permukaan penghantar indium oksida yang tembus pandang sehingga saat cahaya datang dan melewatinya struktur crystal cair akan kelihatan bersih. Jika diberikan tegangan pada permukaan penghantar, susunan molekul terganggu yang menyebabkan perbedaan penyebaran pada daerah yang terbentuk. Cahaya datang dipantulkan dalam arah yang berbeda pada titik temu antara penyebaran indeks yang berbeda pada daerah dengan hasil hamburan sinar yang menampakkan suatu lapisan kaca, hubungan antara permukaan berlawanan antara yang satu dengan yang lain. Berikut ini bentuk fisik dari LCD.



Gambar 2-15. Bentuk fisik dari LCD[3]

2.7.1. LCD Tipe M1632

LCD tipe M1632 merupakan suatu jenis tampilan yang menggunakan *liquid crystal* dalam menampilkan suatu karakter secara dot matrik. LCD ini memiliki cirri-ciri sebagai berikut :

- LCD ini terdiri dari 32 karakter dengan 2 baris masing-masing 16 karakter dengan display dot matrik 5x7

Crystal cair adalah suatu bahan yang akan mengalir seperti sebuah cairan tetapi struktur molekulnya seperti benda padat. Pada LCD terdapat suatu unit pengkamburan cahaya, yang mana terdapat suatu proses nematic liquid crystal. Pada proses tersebut permukaan pengkantur indium oksida yang tembus pandang sehingga saat cahaya datang dan melewatinya struktur crystal cair akan terlihat bersih. Jika dibenkan tegangan pada permukaan pengkantur, susunan molekul tergangga yang menyebabkan perbedaan penyebaran pada daerah yang terbentuk. Cahaya datang dipantulkan dalam arah yang berbeda pada titik tema antara penyebaran indeks yang berbeda pada daerah dengan hasil hamburan sinar yang menampakkan suatu lapisan kaca. hubungan antara permukaan berbayangan antara yang satu dengan yang lain. Bentuk ini bentuk fisik dari LCD.

Gambar 2-15. Bentuk fisik dari LCD[11]

2.7.1. LCD Tipe M1632

LCD tipe M1632 merupakan suatu jenis tampilan yang menggunakan liquid crystal dalam menampilkan suatu karakter secara dot matrik. LCD ini memiliki ciri-ciri sebagai berikut :

- LCD ini terdiri dari 32 karakter dengan 2 baris masing-masing 16 karakter dengan display dot matrik 5x7

- Karakter generator ROM dengan 192 tipe karakter
- Karakter generator RAM dengan 8 bit karakter
- 80x8 bit display data RAM
- Dapat diinterfacekan ke MCU 8 atau 4
- Dilengkapi dengan fungsi tambahan : *display clear, cursor home, display on-off, cursor on-off, display character blink, cursor shift* dan *display shift*
- Internal data
- Reset pada power on
- Tegangan +5 volt PSU tunggal

Liquid Crystal Display ini mempunyai konsumsi daya yang relatif rendah dan terdapat sebuah *controller* CMOS didalamnya. *Controller* tersebut sebagai pembangkit dari karakter RAM/ROM dan display data RAM. Semua fungsi tampilan dikontrol oleh suatu instruksi dan modul LCD dapat dengan mudah untuk diinterfacekan dengan mikrokontroller. Masukkan yang diperlukan untuk mengendalikan modul ini berupa bus data yang masih termultiflex dengan bus alamat serta 3 bit sinyal control. Sementara pengendalian dot matrik LCD dilakukan secara internal oleh *controller* yang sudah ada pada modul LCD.

Dasar-dasar pengoperasian LCD ini terdiri atas register, *busy flag, address counter, display* dan data RAM.

- Register

  Controller dari LCD mempunyai 2 buah register 8 bit yaitu register instruksi (IR) dan register data (DR). IR mempunyai instruksi seperti *display clear, cursor shift* dan *display data* (DD RAM)

- Karakter generator ROM dengan 192 tipe karakter
- Karakter generator RAM dengan 8 bit karakter
- 80x8 bit display data RAM
- Dapat diinterfacekan ke MCU 8 atau 4
- Diterpkapi dengan fungsi tambahan : display clear, cursor home, display on-off, cursor on-off, display character blink, cursor shift dan display shift
- Internal data
- Reset pada power on
- Tegangan +5 volt PSU tunggal

Liquid Crystal Display ini merupakan komponen data yang relatif rendah dan terdapat sebuah controller CMOS didalamnya. Controller tersebut sebagai pembangkit dari karakter RAM/ROM dan display data RAM. Semua fungsi tampilan dikontrol oleh suatu instruksi dan modul LCD dapat dengan mudah untuk diinterfacekan dengan microkontroller. Misalkan yang dipergunakan untuk mengendalikan modul ini berupa bus data yang masih termultiplex dengan bus alamat serta 3 bit signal control. Sementara pengendalian dot matrik LCD dilakukan secara internal oleh controller yang sudah ada pada modul LCD.

Dasar-dasar pengoperasian LCD ini terdiri atas register, busy flag, address counter, display dan data RAM.

- Register

Controller dari LCD mempunyai 2 buah register 8 bit yaitu register instruksi (IR) dan register data (DR). IR mempunyai instruksi seperti display clear, cursor shift dan display clear (DD RAM)

serta *character generator* (CG RAM). DR menyimpan data untuk ditulis di DD RAM atau CG RAM ataupun membaca data dari DD RAM atau CG RAM. Ketika data ditulis ke DD RAM atau CG RAM maka DR secara otomatis menulis data ke DD RAM atau CG RAM. Ketika data pada DD RAM atau CG RAM akan dibaca maka alamat data ditulis pada IR sedangkan data akan dimasukkan melalui DR dan mikrokontroller membaca data dari DR.

- *Busy Flag*

  Busy Flag menunjukkan bahwa modul siap untuk menerima instruksi selanjutnya. Sebagaimana yang terlihat pada table 1, register seleksi sinyal akan melalui DB, jika RS = 0 dan R/W = 1. Jika bernilai 1 maka modul LCD sedang melakukan kerja internal dan instruksi tidak akan diterima. Oleh karena itu status dari *flag* harus diperiksa sebelum melaksanakan instruksi selanjutnya.

RS	R/W	Operasi
0	0	Seleksi IR, IR *Write Display Clear*
0	1	*Busy Flag* (DB7) @ counter (DB0 – DB7) read
1	0	Seleksi DR, DR *Write*
1	1	Seleksi DR, DR *Write*

Tabel 2.8. Register Seleksi [3]

serta character generator (CG RAM). DR meng-input data untuk ditulis di DD RAM atau CG RAM ataupun membaca data dari DD RAM atau CG RAM. Ketika data ditulis ke DD RAM atau CG RAM maka DR secara otomatis meng-update data ke DD RAM atau CG RAM. Ketika data pada DD RAM atau CG RAM akan dibaca maka alamat data ditulis pada IR sedangkan data akan ditransfer melalui DR dan milikoncontroller membaca data dari DR.

- Busy Flag

Busy Flag menunjukkan bahwa modul siap untuk menerima instruksi selanjutnya. Sebagaimana yang terlihat pada table 1, register seleksi ninyal akan melalui DB. jika RS = 0 dan R/W = 1, maka bernilai 1 maka modul LCD sedang melaksanakan kerja internal dan instruksi tidak akan ditermia. Oleh karena itu status dari flag harus dipenksa sebelum melaksanakan instruksi selanjutnya.

RS	R/W	Operasi
0	0	Seleksi IR. IR Write /Display Clear
0	1	Busy Flag (DB7) & counter (DB0 - DB7) read
1	0	Seleksi DR. DR Write
1	1	Seleksi DR. DR Write

Tabel 2.8. Register Seleksi [7]

Pin	No. Term	I/O	Tujuan	Fungsi
DB0 – DB3	4	I/O	MPU	Sebagai lalu lintas data dan instruksi ked an dari MPU, lower byte
DB4 – DB7	4	I/O	MPU	Sebagai lalu lintas data dan instruksi ked an dari MPU, lower byte
E	1	1	MPU	Sinyal Start (read/ write)
R/W	1	1	MPU	Sinyal seleksi register, 0 = write : 1= read
RS	1	1	MPU	Sinyal seleksi register, 0 = Instruksi *register busy flag* & @ (read)
V0	1	-	PSU	Driver LCD
VCC	1	-	PSU	5 Volt
VSS	1	-	PSU	Ground ke terminal : 0 Volt

Tabel 2.9. Fungsi-fungsi Terminal pada LCD[3]

Instruksi-instruksi yang digunakan untuk pemrograman LCD tipe M1632 ini dapat dilihat dalam Tabel 2-10.

No.	Instruksi	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
1	Display Clear	0	0	0	0	0	0	0	0	0	1
2	Cursor Home	0	0	0	0	0	0	0	0	1	0

Pin	No. Term	I/O	Tujuan	Fungsi
DB0 – DB3	4	I/O	MPU	Sebagai jalur bintas data dan instruksi kod an dari MPU, lower byte
DB4 – DB7	4	I/O	MPU	Sebagai jalur bintas data dan instruksi kod an dari MPU, lower byte
E	1	I	MPU	Sinyal Start (read/ write)
R/W	1	I	MPU	Sinyal sedeksi register. 0 = write : 1= read
RS	1	I	MPU	Sinyal sedeksi register. 0 = Instruksi register bus/flag & 1 (read)
Vo	1	-	PSU	Driver LCD
VCC	1	-	PSU	5 Volt
VSS	1	-	PSU	Ground ke terminal : 0 Volt

Tabel 2.9. Fungsi-fungsi Terminal pada LCD[6]

Instruksi-instruksi yang digunakan untuk pemrograman LCD tipe M1632 ini dapat dilihat dalam Tabel 2-10.

No.	Instruksi	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	Display Clear	0	0	0	0	0	0	0	0	0	1
2	Cursor Home	0	0	0	0	0	0	0	0	1	0

3	Entry Set Mode	0	0	0	0	0	0	0	1	1/0	S
4	Display On – Off Control	0	0	0	0	0	0	1	D	C	B
5	Cursor Display Shift	0	0	0	0	0	1	S/C	R/L	0	0
6	Function Set	0	0	0	0	1	DL	N	F	0	0
7	CG RAM Address Set	0	0	0	1	Alamat Karakter					
8	DD RAM Address Set	0	0	1	Tampilan Alamat Data						
9	BF/Address Set	0	1	BF	Alamat Bus						
10	Penulisan ke CG RAM atau DD RAM	1	0	Byte Karakter							
11	Pembacaan CG RAM atau DD RAM	1	1	Byte Karakter							

Tabel 2.10. Instruksi pemrograman LCD [3]

2.7.2. DDRAM

DDRAM adalah merupakan memori tempat karakter yang ditampilkan berada. Contoh, untuk karakter 'A' atau 41H yang ditulis pada alamat 00, maka karakter tersebut akan tampil pada baris pertama dan kolom pertama dari LCD. Apabila karakter tersebut ditulis di alamat 40, maka karakter tersebut akan tampil pada baris kedua kolom pertama dari LCD.

3	Entry Set Mode	0	0	0	0	0	0	0	1	I/D	S
4	Display On – Off Control	0	0	0	0	0	0	1	D	C	B
5	Cursor Display Shift	0	0	0	0	0	1	S/C	R/L	0	0
6	Function Set	0	0	0	0	1	DL	N	F	0	0
7	CG RAM Address Set	0	0	0	1	Alamat Karakter					
8	DD RAM Address Set	0	0	1	Tampilan Alamat Data						
9	BF/Address Set	0	1	BF	Alamat Bus						
10	Pembacaan Ke CG RAM atau DD RAM	1	0	Byte Karakter							
11	Pembacaan CG RAM atau DD RAM	1	1	Byte Karakter							

Tabel 2.10. Instruksi pemrograman LCD [6]

2.7.2. DDRAM

DDRAM adalah memori/alat memori tempat karakter yang ditampilkan berada. Contoh, untuk karakter 'A' alat 41H yang ditulis pada alamat 00, maka karakter tersebut akan tampil pada baris pertama dan kolom pertama dari LCD. Apabila karakter tersebut ditulis di alamat 40, maka karakter tersebut akan tampil pada baris kedua kolom pertama dari LCD.

2.7.3. CGRAM

CGRAM adalah merupakan memori untuk menggambarkan pola sebuah karakter di mana bentuk dari karakter dapat diubah-ubah sesuai keinginan. Namun memori ini akan hilang saat power supply tidak aktif, sehingga pola karakter akan hilang.

2.7.4. CGROM

CGROM adalah merupakan memori untuk menggambarkan pola sebuah karakter di mana pola tersebut sudah ditentukan secara permanen dari HD44780 sehingga pengguna tidak dapat mengubah lagi. Namun karena ROM bersifat permanen, maka pola karakter tersebut tidak akan hilang walaupun power supply tidak aktif.

2.7.5. Register

HD44780, mempunyai dua buah Register yang aksesnya diatur dengan menggunakan kaki RS. Pada saat RS berlogika 0, maka register yang diakses adalah Register Perintah dan pada saat RS berlogika 1, maka register yang diakses adalah Register Data.

2.7.6. Register Perintah

Register ini adalah register di mana perintah-perintah dari mikrokontroler ke HD44780 pada saat proses penulisan data atau tempat status dari HD44780 dapat dibaca pada saat pembacaan data. Penulisan Data ke Register Perintah Penulisan data ke Register Perintah dilakukan dengan tujuan mengatur

2.7.3. CORAM

CORAM adalah merupakan memori untuk menggambarkan pola sebuah karakter di mana bentuk dari karakter dapat diubah-ubah sesuai keinginan. Namun memori ini akan hilang saat power supply tidak aktif. sehingga pola karakter akan hilang.

2.7.4. CGROM

CGROM adalah merupakan memori untuk menggambarkan pola sebuah karakter di mana pola tersebut sudah ditentukan secara permanen dari HD44780 sehingga penggunaannya tidak dapat merubah lagi. Namun karena ROM bersifat permanen, maka pola karakter tersebut tidak akan hilang walaupun power supply tidak aktif.

2.7.5. Register

HD44780 mempunyai dua buah Register yang akses ke dalam dengan menggunakan kaki RS. Pada saat RS berlogika 0, maka register yang diakses adalah Register Perintah dan pada saat RS berlogika 1, maka register yang diakses adalah Register Data.

2.7.6. Register Perintah

Register ini adalah register di mana perintah-perintah dan mikrokontroler ke HD44780 pada saat proses pemilihan data atau tempat status dari HD44780 dapat dibaca pada saat pembacaan data. Pemilihan Data ke Register Perintah Pemilihan data ke Register Perintah dilakukan dengan mejaga mengatur

tampilan LCD, inisialisasi dan mengatur Address Counter maupun Address Data. Gambar 5 menunjukkan proses penulisan data ke register perintah dengan menggunakan mode 4 bit interface. Kondisi RS berlogika 0 menunjukkan akses data ke Register Perintah. RW berlogika 0 yang menunjukkan proses penulisan data akan dilakukan. Nibble tinggi (bit 7 sampai bit 4) terlebih dahulu dikirimkan dengan diawali pulsa logika 1 pada E Clock. Kemudian Nibble rendah (bit 3 sampai bit 0) dikirimkan dengan diawali pulsa logika 1 pada E Clock lagi. Untuk mode 8 bit interface, proses penulisan dapat langsung dilakukan secara 8 bit (bit 7 ... bit 0) dan diawali sebuah pulsa logika 1 pada E Clock.

2.8. KOMUNIKASI SERIAL

Terdapat dua metode untuk mentransmisiskan data, yaitu metode parallel atau metode serial. Pada metode parallel, data ditansfer sekaligus dalam satu waktu menggunakan saluran transmisi sebanyak bit data yang dikirimkan. Komunikasi serial ialah pengiriman data secara serial (data dikirim satu per satu secara berurutan) sehingga komunikasi serial jauh lebih lambat daripada komunikasi paralel. Serial port lebih sulit ditangani karena peralatan yang dihubungkan ke serial port harus berkomunikasi mengunakan transmisi serial sedangkan data di komputer diolah secara paralel. Oleh karena itu data dari / ke serial port harus dikonversikan ke / dari bentuk paralel untuk bisa digunakan. Kelebihan komunikasi serial ialah jangkauan panjang kabel yang lebih jauh dibandingkan paralel karena serial port mengirimkan logika 1 dengan kisaran tegangan -3 hingga -25 volt dan logika 0 sebagai +3 hingga +25 volt sehingga kehilangan daya karena panjangnya kabel bukan masalah utama

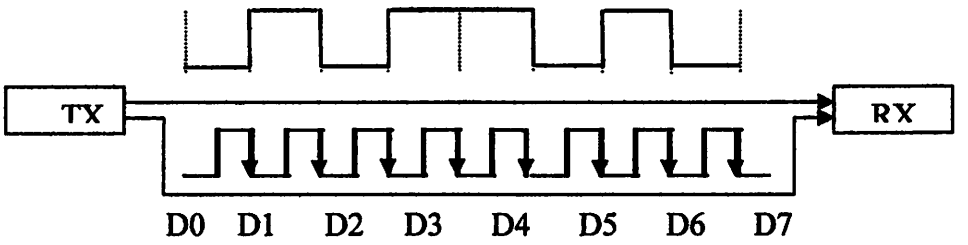
tampilan LCD, inisialisasi dan mengatur Address Counter maupun Address Data. Gambar 5 menunjukkan proses pemulisan data ke register perintah dengan menggunakan mode 4 bit interface. Kondisi RS berlogika 0 menunjukkan akses data ke Register Perintah. R/W berlogika 0 yang menunjukkan proses pemulisan data akan dilakukan. Nibble tinggi (bit 7 sampai bit 4) terlebih dahulu dikirimkan dengan diawali pulsa logika 1 pada E Clock. Kemudian Nibble rendah (bit 3 sampai bit 0) dikirimkan dengan diawali pulsa logika 1 pada E Clock lagi. Untuk mode 8 bit interface, proses pemulisan dapat langsung dilakukan secara 8 bit (bit 7 ... bit 0) dan diawali sebuah pulsa logika 1 pada E Clock.

2.8. KOMUNIKASI SERIAL

Terdapat dua metode untuk mentransmisikan data, yaitu metode parallel atau metode serial. Pada metode parallel, data ditransfer sebanyak delapan saluran atau lebih menggunakan saluran transmisi sebanyak bit data yang dikirimkan. Komunikasi serial ialah pengiriman data secara serial (data dikirim satu per satu secara berurutan) sedangga komunikasi serial jauh lebih lambat daripada komunikasi parallel. Serial port lebih sulit ditangani karena peralatan yang dihubungkan ke serial port harus berkomunikasi menggunakan transmisi serial sedangkan data di komputer diolah secara parallel. Oleh karena itu data dari / ke serial port harus dikonversikan ke / dari bentuk parallel untuk bisa digunakan. Kelebihan komunikasi serial ialah jangkauan panjang kabel yang lebih jauh dibandingkan parallel karena serial port menginimkan logika 1 dengan kisaran tegangan -3 hingga -25 volt dan logika 0 sebagai +3 hingga +25 volt sedangga kebilangan daya karena panjangnya kabel bukan masalah utama

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

2.8.1. KOMUNIKASI SINKRON



Gambar 2.16. Komunikasi Serial dengan Sinyal Sinkronisasi

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

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

2.9.1. KOMUNIKASI SINKRON

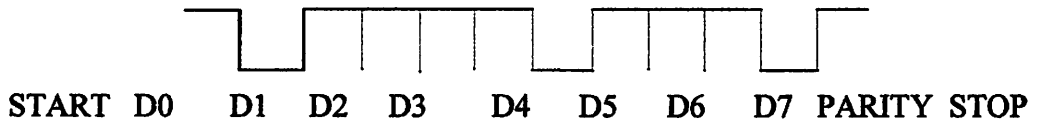
Gambar 2.16. Komunikasi Serial dengan Sinyal Sinkronisasi

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

Pada pengiriman data sinkron sejumlah blok data dikirimkan secara kontinu tanpa bit awal atau akhir. Sinyal clock pada komunikasi sinkron diperlukan oleh peralatan penerima data untuk mengetahui adanya pengiriman setiap bit data. Untuk mendapatkan keadaan yang sesuai, informasi pendetakan harus dikirim lewat jalur bersama-sama dengan data dan memanfaatkan metode penyandian tertentu sehingga informasi pendetakan dapat diikut sertakan.

Data secara kontinu menunjukkan akan dikirimkan terus menerus tanpa adanya pembatas (*gap*). Interval waktu antara bit terakhir dari suatu karakter dengan bit pertama dari karakter berikutnya adalah 0 atau kelipatan bulat dari periode waktu yang diperlukan untuk mengirimkan sebuah karakter. Jika pada data yang dikirimkan terdapat pembatas, pengirim akan menambahkan *byte* tambahan untuk mengganti pembatas tersebut. Sehingga tidak diperlukan adanya bit awal dan bit akhir.

2.8.2. KOMUNIKASI ASINKRON



Gambar 2.17. Format Sinyal Serial Asinkron

Cara kedua dengan komunikasi asinkron, yaitu dengan menetapkan kecepatan bit dan menyisipkan beberapa bit protokol, yaitu bit START, PARITY bit dan STOP seperti diperlihatkan pada gambar 2.17. diatas.

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

Pada pengiriman data sinkron sejumlah blok data dikirimkan secara lorntun tanpa bit awal atau akhir. Sinyal clock pada komunikasi sinkron diperlukan oleh peralatan penerima data untuk mengetahui adanya pengiriman setiap bit data. Untuk mendapatkan keadaan yang sesuai, informasi pendeteksian harus dikirim lewat jalur bersama-sama dengan data dan menambahkan metode penyandian tertentu sehingga informasi pendeteksian dapat difikut sertakan.

Data secara kontinu memanfaatkan akan dikirimkan terus menerus tanpa adanya pembatas (gap). Interval waktu antara bit terakhir dari satu karakter dengan bit pertama dari karakter berikutnya adalah 0 atau kelipatan bulat dari periode waktu yang diperlukan untuk mengirimkan sebuah karakter. Jika pada data yang dikirimkan terdapat pembatas, pengirim akan menambahkan byte tambahan untuk menggenati pembatas tersebut. Sehingga tidak diperlukan adanya bit awal dan bit akhir.

2.8.2. KOMUNIKASI ASINKRON

Gambar 2.17. Format Sinyal Serial Asinkron

Cara kedua dengan komunikasi asinkron, yaitu dengan menciptakan kecepatan bit dan menyisipkan beberapa bit protokol, yaitu bit START, PARITY bit dan STOP seperti diperlihatkan pada gambar 2.17. diatas.

- Kecepatan bit disebut sebagai baud rate atau besar bit disimpan byte (bit per second). pada standar komunikasi ditentukan adalah 1200.

4800, 9600 bps. Sebagai contoh untuk pesat bit 4800 bps berarti lebar pulsa untuk 1 bit adalah 20,84 ms.

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

2.8.3. Arah Pengiriman Data

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

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

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

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

Untuk dapat jelasnya, dapat dilihat pada gambar berikut :

4800, 9600 bps. Sebagai contoh untuk pesan bit 4800 bps bernilai lebar pulsa untuk 1 bit adalah 20.84 ms.

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

2.8.3. Arah Pengiriman Data

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

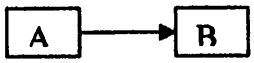
Simplex adalah sistem pemindahan data yg hanya satu arah saja. misalnya dari A ke B. dimana A sebagai pengirim dan B sebagai penerima. dan tidak dapat mengirimkan data dari B ke A.

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

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

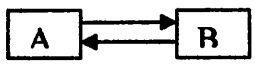
Untuk dapat jelasnya, dapat dilihat pada gambar berikut :

a. Komunikasi data *Simplex*



b. Komunkasi data *Half Duplex*

c. Komunikasi data *Full Duplex*



2.8.4. RS-232

Standart RS232 yang ditetapkan tahun 1962 oleh Electronic Industry Association dan Telecomunication Industry Association sebenarnya standart lama, sebelum era TTL muncul, karena sudah dianggap standart dan kemudian muncul TTL, maka untuk menjembatani antara TTl dan RS232 level, maka dibuat konverter.

Konverter RS232 level akan menganggap tegangan antara +5 hingga +15 Volt sebagai tegangan ' 0 ' sedangkan tegangan -3 hingga -15 Volt dianggap sebagai tegangan ' 1 ', level antara -3 hingga +3 tidak didefinisikan, sebab di daerah ini kemungkinan adalah noise. Level TTL diatas 2 Volt yang dianggap sebagai level ' 1 ' akan dikonversikan ke level RS232 yaitu sebesar -15 Volt, sedangkan level ' 0 ' TTL, yaitu tegangan di bawah 0.8 Volt, akan dikonversikan ke +15Volt, demikian juga pada konversi sebaliknya, level +3

a. Komunikasi data Simplex

b. Komunikasi data Half Duplex

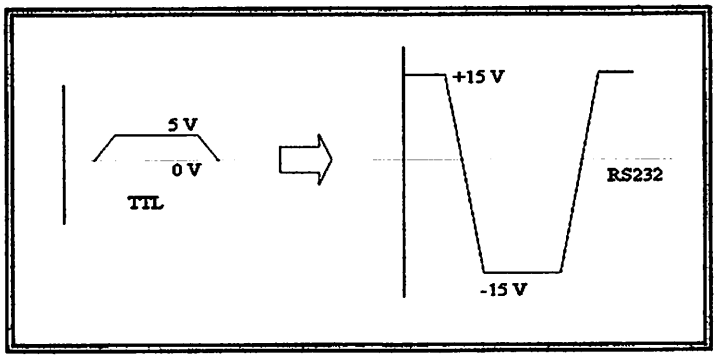
c. Komunikasi data Full Duplex

2.8.4. RS-232

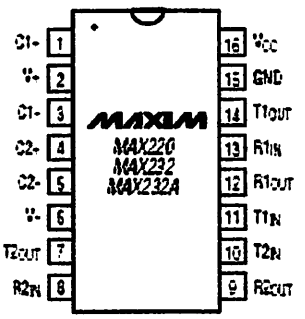
Standart RS232 yang dikeluarkan tahun 1962 oleh Electronic Industry Association dan Telecommunication Industry Association sebenarnya standart lama. sebelum ada TTL manual, karena sudah dianggap standart dan komunikasi manual TTL, maka untuk menjembatani antara TTL dan RS232 level, maka dibuat konverter.

Konverter RS232 level akan menganggap tegangan antara +5 hingga +15 Volt sebagai tegangan " 0 " sedangkan tegangan -3 hingga -15 Volt dianggap sebagai tegangan " 1 ". level antara -3 hingga +3 tidak didefinisikan. sebab di dacrah ini kemungkinan adalah noise. Level TTL dinas 2 Volt yang dianggap sebagai level " 1 " akan dikonversikan ke level RS232 yaitu sebesar -15 Volt. sedangkan level " 0 " TTL yaitu tegangan di bawah 0.8 Volt. akan dikonversikan ke +15Volt. demikian juga pada konversi sebaliknya. level +3

hingga +15 Volt akan dikonversikan ke level TTL 5 Volt dan -3 hingga -15 Volt akan dikonversikan ke 0 Volt.

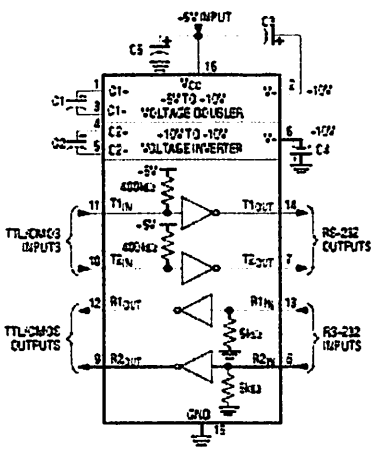


Gambar 2.18. Level Converter TTL Ke RS232 [4]



Gambar 2.19. Konfigurasi Pin IC MAX232 [4]

Rangkaian dasar dari MAX232 dapat dilihat pada gambar berikut ini :



Gambar 2.20. Rangkaian Operasi MAX232 [4]

hingga +15 Volt akan dikonversikan ke level TTL 5 Volt dan -3 hingga -15 Volt akan dikonversikan ke 0 Volt.

Gambar 2.18. Level Converter TTL Ke RS232 [1]

Gambar 2.19. Konfigurasi Pin IC MAX232 [1]

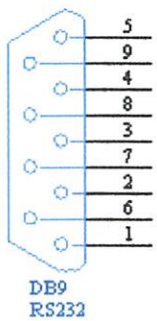
Rangkaian dasar dari MAX232 dapat dilihat pada gambar berikut ini :

Gambar 2.20. Rangkaian Operasi MAX232 [1]

RS-232 adalah standard komunikasi serial antar periperal - periperal. Standar ini menggunakan beberapa piranti dalam implementasinya. Paling umum yang dipakai adalah plug DB9. Untuk RS-232 dengan DB9, biasanya dipakai untuk *serial port* pada personal komputer. Dipakai untuk *port mouse* dan *modem*. Bentuk dan fungsi dari masing - masing pin ditunjukkan pada gambar dibawah :

Pin Number	Signal Name	Abbreviation
1	Carrier Detect	CD
2	Receive Data	RxD
3	Transmit Data	TxD
4	Data Terminal Ready	DTR
5	System Ground	SG
6	Data Set Ready	DSR
7	Request To Send	RTS
8	Clear To Send	CTS
9	Ring Indicator	RI

Tabel 2.11. Fungsi Pin – Pin DB9 [4]



Gambar 2.21. Bentuk Pin DB9 Standar RS-232. [4]

2.8.5. RS-485

RS-485 biasa digunakan untuk menyediakan signal serial yang kuat sehingga sanggup untuk jarak kabel yang panjang (sampai dengan 1,2 km). Mampu melalui kabel yang panjang (1,2 km) pada *baudrate* yang tinggi dalam

RS-232 adalah standard komunikasi serial antar peripheral - peripheral. Standar ini menggunakan beberapa pinout dalam implementasinya. Paling umum yang dipakai adalah plug DB9. Untuk RS-232 dengan DB9, biasanya dipakai untuk serial pada personal komputer. Dipakai untuk port more dan modem. Bentuk dan fungsi dari masing - masing pin ditunjukkan pada gambar dibawah :

Pin Number	Signal Name	Abbreviation
1	[illegible]	[illegible]
2	[illegible]	[illegible]
3	[illegible]	[illegible]
4	[illegible]	[illegible]
5	[illegible]	[illegible]
6	[illegible]	[illegible]
7	[illegible]	[illegible]
8	[illegible]	[illegible]
9	[illegible]	[illegible]

Tabel 2.11. Fungsi Pin – Pin DB9 [4]

Gambar 2.21. Bentuk Pin DB9 Standar RS-232. [4]

2.8.5. RS-485

RS-485 biasa digunakan untuk menyediakan signal serial yang kuat sehingga sanggup untuk jarak kabel yang panjang (sampai dengan 1.2 km). Walaupun modulnya kabel yang panjang (1.2 km) pada kenyataannya yang hanya

lingkungan listrik yang berpotensi menimbulkan *noise* atau interferensi elektromagnetik yang tinggi. Berikut adalah spesifikasi dari RS-485.

SPECIFICATIONS		RS485
Mode of Operation		DIFFERENTIAL
Total Number of Drivers and Receivers on One Line		1 DRIVER 32 RECEIVER
Maximum Cable Length		4000 FT.
Maximum Data Rate		10Mb/s
Maximum Driver Output Voltage		-7V to +12V
Driver Output Signal Level (Loaded Min.)	Loaded	+/-1.5V
Driver Output Signal Level (Unloaded Max)	Unloaded	+/-6V
Driver Load Impedance (Ohms)		54
Max. Driver Current in High Z State	Power On	+/-100uA
Max. Driver Current in High Z State	Power Off	+/-100uA
Slew Rate (Max.)		N/A
Receiver Input Voltage Range		-7V to +12V
Receiver Input Sensitivity		+/-200mV
Receiver Input Resistance (Ohms)		>=12k

Tabel 2.12. Spesifikasi RS – 485 [5]

2.9 Personal Computer

Komputer adalah salah satu dari sekian banyak peralatan yang dibuat manusia guna meningkatkan efisiensi kerja, produktivitas dan tingkat kehidupan manusia. Suatu perjalanan teramat panjang telah dilalui dalam sejarah hingga memungkinkan kita di masa kini dapat menikmati pemakaian komputer yang canggih dalam segala bentuknya seperti sekarang ini.

Komputer harus mempunyai minimal 2 unsur yang harus ada sebagai persyaratan dalam menggunakan komputer. Kedua unsur tersebut adalah :

1. *Hardware* (Perangkat Keras)
2. *Software* (Perangkat Lunak)

lingkungan listrik yang berpotensi menimbulkan noise atau interferensi elektromagnetik yang tinggi. Bentuk adalah spesifikasi dari RS-485.

[illegible]	[illegible]	[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]		[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]		[illegible]
[illegible]		[illegible]

Tabel 2.12. Spesifikasi RS – 485 [2]

2.9 Personal Computer

Komputer adalah salah satu dari sekian banyak peralatan yang dibuat manusia guna meningkatkan efisiensi kerja, produktivitas dan tingkat kehidupan manusia. Suatu perjalanan terkait panjang telah dilalui dalam sejarah hingga menunggkinkan kita di masa kini dapat menikmati pemakaian komputer yang canggih dalam segala bentuknya seperti sekarang ini.

Komputer harus mempunyai minimal 2 unsur yang harus ada sebagai pendukung dalam menggunakan komputer. Kedua unsur tersebut adalah :

1. Hardware (Perangkat Keras)
2. Software (Perangkat Lunak)

2.9.1. Hardware

Hardware berupa peralatan fisik dari sebuah sistem komputer. Umumnya peralatan tersebut harus terdiri dari 3 jenis perangkat yaitu perangkat masukan, perangkat keluaran serta perangkat pengolahan.

Perangkat input berfungsi untuk memasukkan data, baik berupa teks maupun gambar ke dalam komputer. Contoh perangkat input misalnya *mouse*, *keyboard*, dan sebagainya.

Perangkat keluaran digunakan untuk menampung dan menghasilkan data keluaran, misalnya monitor dan *printer*. Sedangkan perangkat pengolah meliputi Unit Pengolah Pusat (*CPU - Central Processing Unit*) dan juga *mikroprosessor*.

2.9.2. Software

Software atau perangkat lunak tidak lain adalah program komputer yang merupakan suatu susunan *instruksi* yang harus diberikan kepada unit pengolah agar komputer dapat menjalankan pekerjaan sesuai dengan yang dikehendaki. Program – program itu ditulis dalam bahasa khusus yang dimengerti oleh mesin.

Penggunaan komputer pada tugas akhir ini berfungsi sebagai media penyimpanan *database*. Hal ini dilatar belakangi karena kapasitas dari komputer itu sendiri sudah mencapai satuan GB, sehingga dapat menampung data dengan jumlah banyak.

2.9.1. Hardware

Hardware berupa peralatan fisik dari sebuah sistem komputer. Umumnya peralatan tersebut harus terdiri dari 3 jenis perangkat yaitu perangkat masukan, perangkat keluaran serta perangkat pengolahan.

Perangkat input berfungsi untuk memasukkan data, baik berupa teks maupun gambar ke dalam komputer. Contoh perangkat input misalnya mouse, keyboard, dan sebagainya.

Perangkat keluaran digunakan untuk menampung dan menghasilkan data keluaran, misalnya monitor dan printer. Sedangkan perangkat pengolah meliputi Unit Pengolah Pusat (CPU - Central Processing Unit) dan juga microprocessor.

2.9.2. Software

Software atau perangkat lunak ialah dalah program komputer yang merupakan suatu susunan instruksi yang harus diberikan kepada unit pengolah agar komputer dapat menjalankan pekerjaan sesuai dengan yang dikehendaki. Program – program itu ditulis dalam bahasa khusus yang dimengerti oleh mesin.

Penggunaan komputer pada tugas akhir ini berfungsi sebagai media penyimpanan database. Hal ini dilatar belakangi karena kapasitas dari komputer itu sendiri sudah mencukupi satuan GB. sehingga dapat menampung data dengan jumlah banyak.

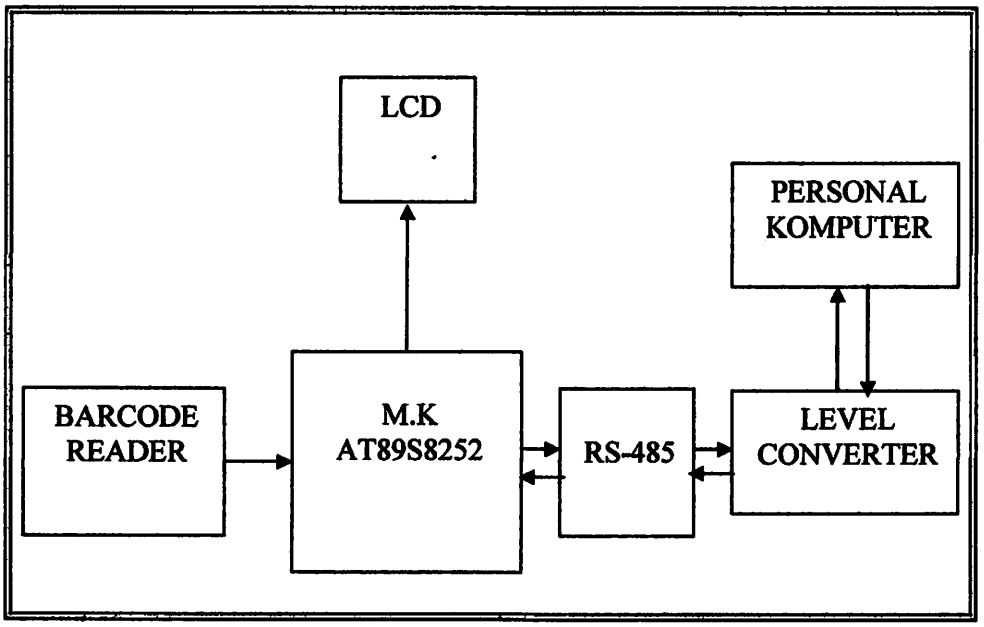
BAB III

PERANCANGAN DAN PEMBUATAN ALAT

3.1. Pendahuluan

Untuk merancang dan membuat alat untuk mendeteksi harga dan jenis barang dengan menggunakan barcode reader berbasis Mikrokontroller AT89S8252 yang terhubung pada komputer yang berfungsi sebagai database harga dan jenis barang, perlu diketahui terlebih dahulu mengenai blok diagram dari sistem tersebut, prinsip kerja sistem secara keseluruhan, dan perancangan perangkat lunak yaitu mengenai flowchart dan software yang digunakan.

3.2. Perancangan Perangkat Keras



Gambar 3.1. Diagram Blok Sistem Keseluruhan

Secara garis besar, blok diagram di atas tersebut dapat dijelaskan sebagai berikut :

1) **Barcode Reader**

   Barcode reader merupakan suatu sensor yang akan digunakan untuk membaca kode – kode yang ada pada kartu yang berisi data barcode dengan *code-39*, kemudian menerjemahkannya kedalam bentuk format scan code yang nilainya sama dengan format data pada keyboard.

2) **LCD**

   LCD berfungsi untuk menampilkan pilihan menu – menu yang akan diakses.

3) **Mikrokontroller AT89S8252**

   Mikrokontroller AT89S8252 berfungsi untuk mengolah data yang diterima, mengontrol dan mengendalikan rangkaian – rangkaian yang dihubungkan dengan Mikrokontroller AT89S8252 tersebut.

4) **RS-485**

   RS-485 adalah teknik komunikasi data serial yang dapat melakukan komunikasi data pada jarak yang cukup jauh yaitu 1,2 Km.

5) **Level Converter**

   Level converter berfungsi untuk mengubah level tegangan TTL menjadi level tegangan yang bisa diterima oleh komputer.

6) **Personal Komputer**

   Personal komputer merupakan tempat akhir penampung data yang nantinya dapat diketahui data harga dan jenis barang tersebut.

Secara garis besar, blok diagram di atas tersebut dapat dijelaskan sebagai berikut :

1) **Barcode Reader**

Barcode reader merupakan suatu sensor yang akan digunakan untuk membaca kode – kode yang ada pada kertas yang berisi data barcode dengan cetak-39. kemudian mengelompokkannya kedalam bentuk format scan code yang nilainya sama dengan format data pada Icy board.

2) **LCD**

LCD berfungsi untuk menampilkan pilihan menu - menu yang akan diakses.

3) **Microkontroller AT89S8252**

Microkontroller AT89S8252 berfungsi untuk mengolah data yang diterima, mengontrol dan mengendalikan rangkaian - rangkaian yang ditunbungkan dengan Mikrokontroller AT89S8252 tersebut.

4) **RS-485**

RS-485 adalah teknik komunikasi data serial yang dapat melakukan komunikasi data pada jarak yang cukup jauh yaitu 1.2 Km.

5) **Level Converter**

Level converter berfungsi untuk mengubah level tegangan TTL menjadi level tegangan yang bisa diterima oleh komputer.

6) **Personal Komputer**

Personal komputer merupakan tempat akhir penampung data yang nantinya dapat diketahui data hasil dan jenis barang tersebut.

Cara kerja Sistem :

Pertama – tama sensor barcode akan membaca data barcode pada label barang yang ingin diketahui spesifikasinya (nama barang, harga barang, netto) dengan jenis barcode tipe code-39. Kemudian data dari barcode ini akan dikirimkan ke mikrokontroler untuk diteruskan ke PC. Didalam PC terdapat database dari berbagai macam barang dengan spesifikasinya, pada PC data dari mikrokontroler ini akan dibandingkan dengan data yang ada di database, apabila data yang dikirimkan oleh mikrokontroler sama dengan database, maka PC akan mengiri data berupa nama barang, netto, harga ke MK dan oleh MK akan ditampilkan pada LCD, sebaliknya apabila data yang dikirimkan mikrokontroler tidak ada pada database PC maka pada LCD akan diampilkan bahwa tidak ada data yang dicari.

3.3. Perencanaan Alat

Untuk merealisasikan perancangan dan pembuatan alat untuk mendeteksi harga dan jenis barang ini, maka perlu direncanakan dan dibuat suatu alat agar menjadi suatu sistem sesuai dengan yang diharapkan. Sistem Yang direncanakan terdiri dari 6 bagian utama, yaitu :

1. Perancangan kartu barcode
2. Sensor barcode
3. Box 1 : LCD, Mikrokontroller AT89S8252, Rangkaian RS-485.
4. Kabel konektor antara box 1 dengan box 2.
5. Box 2 : Level converter (RS232), Rangkaian RS-485.
6. Komputer.

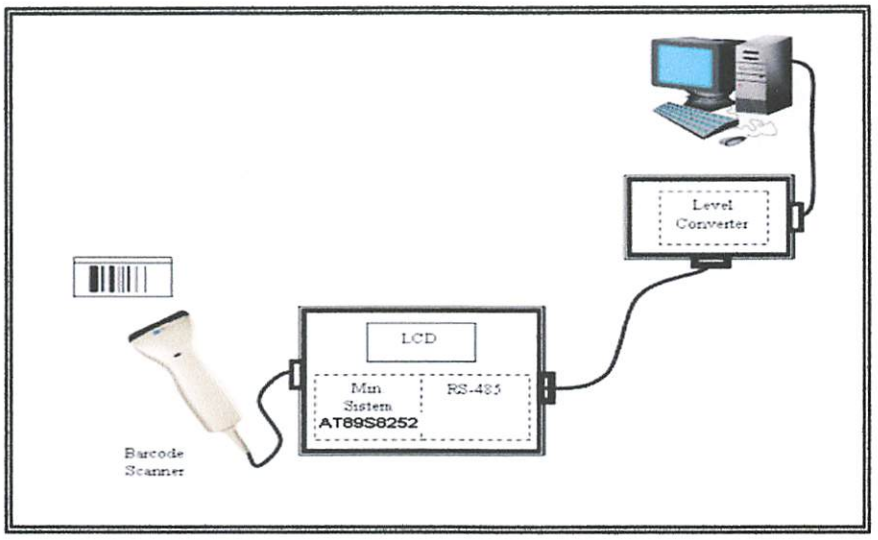
Cara kerja Sistem :

Pertama – tama sensor barcode akan membaca data barcode pada label barang yang ingin dikotahui spesifikasinya (nama barang, harga barang, merk) dengan jenis barcode tipe code-39. Kemudian data dari barcode ini akan dikirimkan ke mikrokontroler untuk diteruskan ke PC. Didalam PC terdapat database dari berbagai macam barang dengan spesifikasinya, pada PC data dari mikrokontroler ini akan dibandingkan dengan data yang ada di database. apabila data yang dikirimkan oleh mikrokontroler sama dengan database, maka PC akan mengirim data berupa nama barang, merk, harga ke MK dan oleh MK akan ditampilkan pada LCD, sebaliknya apabila data yang dikirimkan mikrokontroler tidak ada pada database PC maka pada LCD akan ditampilkan bahwa tidak ada data yang dicari.

3.3. Perancangan Alat

Untuk merealisasikan perancangan dan pembuatan alat untuk mendeteksi harga dan jenis barang ini, maka perlu dirancangkan dan dibuat suatu alat agar menjadi suatu sistem sesuai dengan yang diberapkan. Sistem Yang dirancangkan terdiri dari 6 bagian utama, yaitu :

1. Perancangan kartu barcode
2. Sensor barcode
3. Box 1 : LCD, Mikrokontroller AT89S8252, Rangkaian RS-485.
4. Kabel konektor antara box 1 dengan box 2.
5. Box 2 : Level converter (RS232), Rangkaian RS-485.
6. Komputer.



Gambar 3.2. Bentuk Perencanaan mesin pendeteksi harga dan jenis barang

Barcode Reader

3.3.1. Perancangan Kartu Barcode

Kartu barcode yang dirancang menggunakan format data code-39, yaitu kartu barcode untuk mengecek harga dan jenis barang.

Data yang digunakan pada kartu barcode harga dan jenis barang sebanyak 1 karakter, dimana fungsi dari data tersebut untuk memudahkan penyesuaian data dengan database daftar harga dan jenis barang. Data barcode dirancang dengan tinggi *bar* 3/4 inchi, lebar *bar* 45 *mils*, dan kerapatan 0,0 *mils*. Data barang yang digunakan adalah sebagai berikut :

1. Kode barang 1 : SUSU DANCOW
2. Kode barang 2 : KECAP HITAM
3. Kode barang 3 : ENFAMIL
4. Kode barang 4 : LATOGEN 1

Gambar 3.2. Bentuk Perancangan mesin pendeteksi harga dan jenis barang Barcode Reader

3.3.1. Perancangan Kartu Barcode

Kartu barcode yang dirancang menggunakan format data code-39. yaitu kartu barcode untuk mengecek harga dan jenis barang.

Data yang digunakan pada kartu barcode harga dan jenis barang sebanyak 1 kunci. dimana fungsi dari data tersebut untuk memudahkan pengosuaian dan dengan database daftar harga dan jenis barang. Data barcode dirancang dengan tinggi bar 34 inchi. lebar bar 45 mil. dan kerapatan 0.0 mil.

Data barang yang digunakan adalah sebagai berikut :

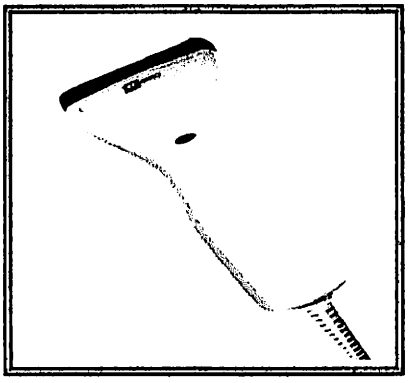
1. Kode barang 1 : SUSU DANCOW
2. Kode barang 2 : KECAP HITAM
3. Kode barang 3 : ENHAMIL
4. Kode barang 4 : LATOGEN 1

Model dari kartu tersebut dapat dilihat pada gambar dibawah ini :

Gambar 3.3. Kartu Harga Dan jenis Barang

3.3.2. Sensor Barcode Reader

Sensor *barcode* yang digunakan adalah jenis *CCD* keluaran metrologic. Penggunaan *barcode* jenis ini dinilai cocok untuk sistem yang akan dirancang, disamping sensor ini hanya memerlukan tegangan sebesar 5V, yang terpenting adalah sensor ini dapat mengkodekan format kode-39 yang digunakan pada pengisian data *barcode*. Selain itu sensor jenis ini juga mudah dijumpai dipasaran dengan harga yang terjangkau.



Gambar 3.4.
CCD *Barcode reader* Metrologic

Model dari kartu tersebut dapat dilihat pada gambar dibawah ini :

Gambar 3.3. Kartu Harga Dan Jenis Barang

3.3.2. Sensor Barcode Reader

Sensor barcode yang digunakan adalah jenis CCD keluaran motorologic. Penggunaan barcode jenis ini dimiliki cocok untuk sistem yang akan dirancang, disamping sensor ini banyak memberikan kegunaan sebesar 3V, yang terpenting adalah sensor ini dapat mengkodekan format kode-39 yang digunakan pada pengisian data barcode. Selain itu sensor jenis ini juga mudah dipunpai dipasarkan dengan harga yang terjangkau.

Gambar 3.4.
CCD Barcode reader Motorologic

3.4. Mikrokontroller AT89S8252

3.4.1. Penggunaan Port Mikrokontroller AT89S8252

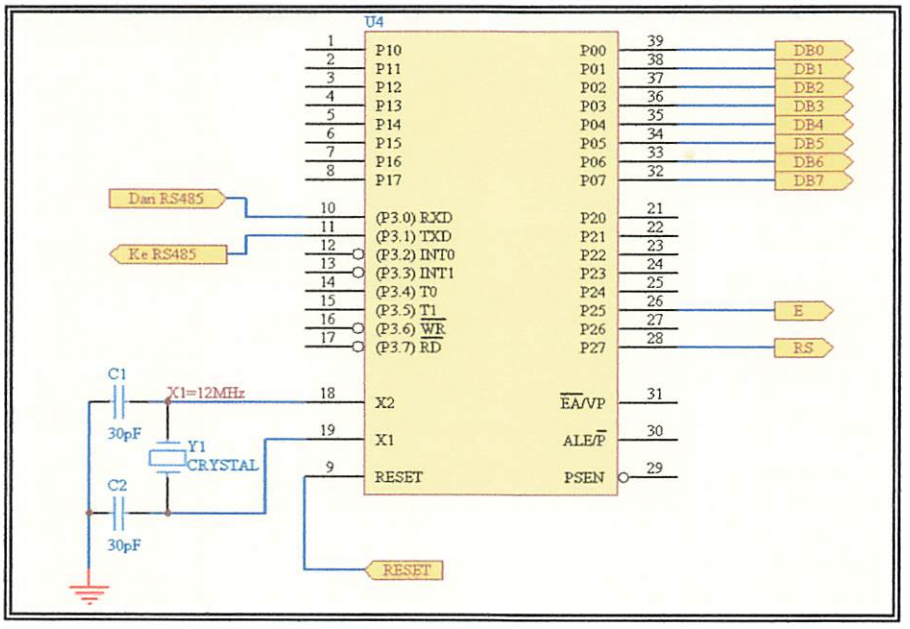
Mikrokontroller AT89S8252 adalah sebuah chip IC yang terdiri dari 40 pin. Dalam perencanaan sistem ini pin – pin yang digunakan adalah sebagai berikut:

Tabel 3.1

Fungsi Port Mikrokontroller AT89S8252 Pada Perancangan Alat

NO PIN	PORT	FUNGSI
9	RST	Digunakan untuk mereset sistem pada Mikrokontroller AT89S8252 agar kembali ke keadaan awalnya.
10	P3.0 (RXD)	Sebagai receiver data, digunakan untuk menerima data dari komputer.
11	P3.1 (TXD)	Sebagai transmitter data, digunakan untuk mengirim data ke komputer.
18	XTAL2	Sebagai pembangkit oscilator (clock) XTAL 2
19	XTAL1	Sebagai pembangkit oscilator (clock) XTAL 1
20	GND	Sebagai masukan catu daya 0 V DC (Ground)
26	P2.5	Sebagai input Enable pada LCD
27	P2.6	Sebagi input Read / Write pada LCD
28	P2.7	Sebagai input Register Select pada LCD
31	EA/VPP	Untuk pilihan program, menggunakan program internal.
32 - 39	P0.0 – P0.7	Sebagai port I/O yang terhubung dengan LCD
40	VCC	Sebagi masukan catu daya +5 Volt DC

Mikrokontroller pada alat ini tidak dapat bekerja sendiri sehingga masih membutuhkan komponen – komponen pendukung lain, komponen – komponen tersebut saling berhubungan secara hardware dan juga sofware. Rangkaian mikrokontroller dapat dilihat pada gambar dibawah ini:

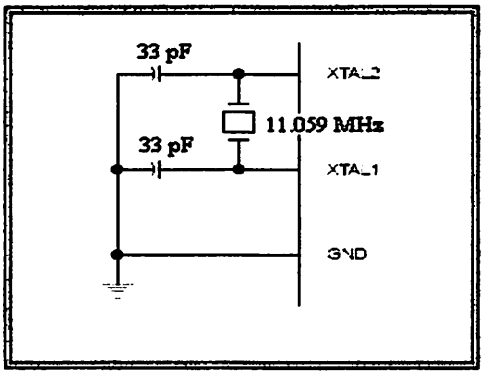


Gambar 3.5. Hubungan Port Mikrokontroller AT89S8252

3.4.1.1. Rangkaian Clock

Kecepatan proses yang dilakukan oleh Mikrokontroller AT89S8252 ditentukan oleh sumber clock (pewaktuan) yang dikendalikan oleh mikrokontroller AT89S8252 tersebut.

Sistem yang dirancang ini akan menggunakan oscilator internal yang telah tersedia di dalam chip AT89S8252, untuk menetukan frekuensi oscilatornya cukup dengan menghubungkan kristal pada pin XTAL1 dan XTAL2 serta dua buah kapasitor ke ground, besar kapasitansinya yaitu 33 pF dan kristal 11.0592 MHz sesuai dengan spesifikasi pada datasheet AT89S8252.

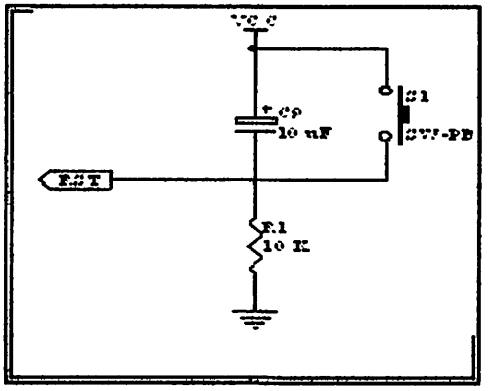


Gambar 3.6. Rangkaian Clock

3.4.1.2 Rangkaian Reset

Untuk melakukan reset sistem (mengembalikan sistem ke keadaan awalnya), maka pin nomor 9 pada Mikrokontroller AT89S8252 dihubungkan dengan rangkaian *reset,* rangkaian ini digunakan bertujuan agar sistem yang dirancang dapat mempunyai kemampuan *power ON Reset*, yaitu Reset terjadi saat *catu daya* diaktifkan.

Dibawah ini adalah adalah rangkaian reset yang digunakan :

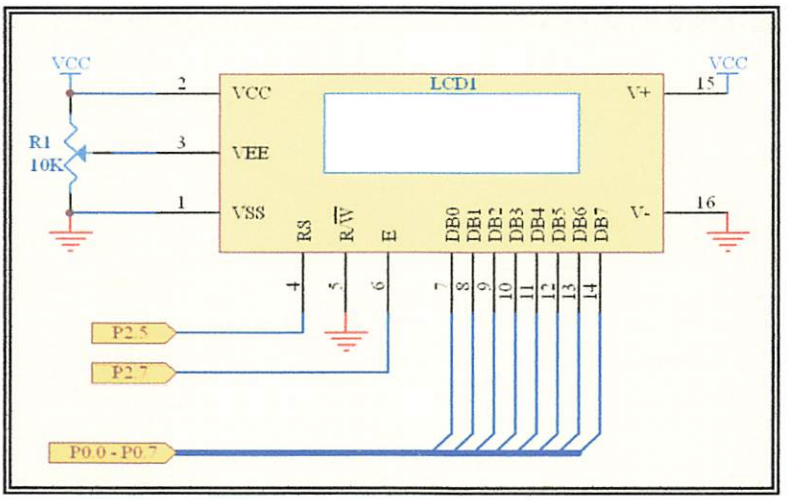


Gambar 3.7. Rangkaian Reset

3.5. LCD (Liquid Crystal Display)

Penggunaan LCD pada perancangan alat pendeteksi harga dan jenis barang ini bertujuan untuk menampilkan instruksi – instruksi dan pesan – pesan yang berhubungan dengan sistem perancangan alat barcode reader ini. Instruksi yang ditampilkan berupa panduan / tuntunan untuk menjalankan sistem

Gambar rangkaian LCD ditunjukkan pada gambar sebagai berikut :



Gambar 3.8. Rangkaian LCD

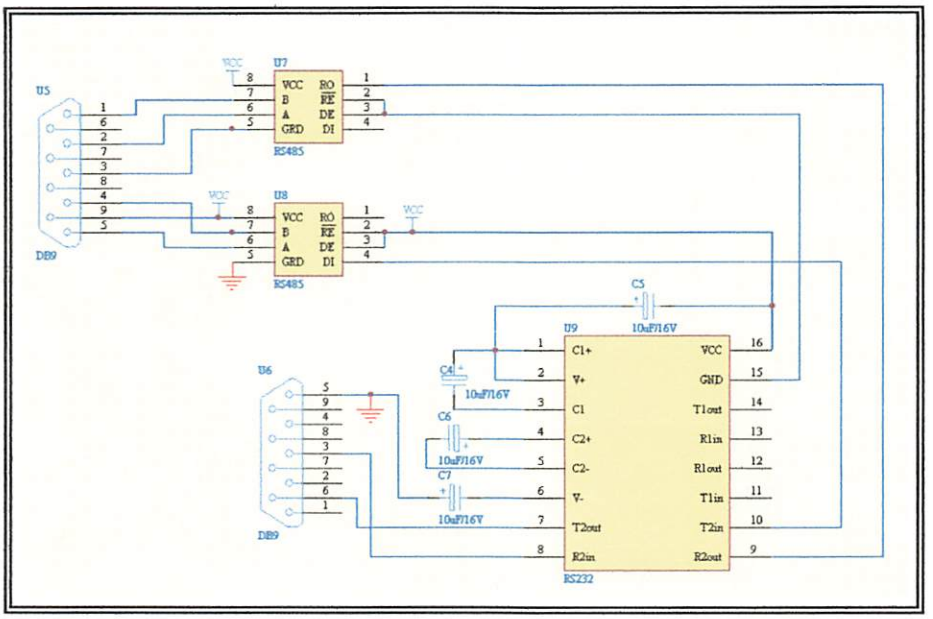
3.6. Komunikasi Serial

3.6.1. Port Serial

Untuk melakukan proses komunikasi data antara Mikrokontroller AT89S8252 dengan PC diperlukan syarat – syarat yang harus dipenuhi, yaitu pengaturan baudrate yang digunakan serta pengaturan secara perangkat lunak. Dimana kecepatan bit yang digunakan pada perancangan sebesar 9600 bps, 1 start bit, 8 bit data, 1 parity bit, dan diakhiri dengan 1 stop bit.

Proses pengiriman dan penerimaan data serial dari mikrokontroller ke komputer maupun sebaliknya, dilakukan dengan cara merubah level tegangan data

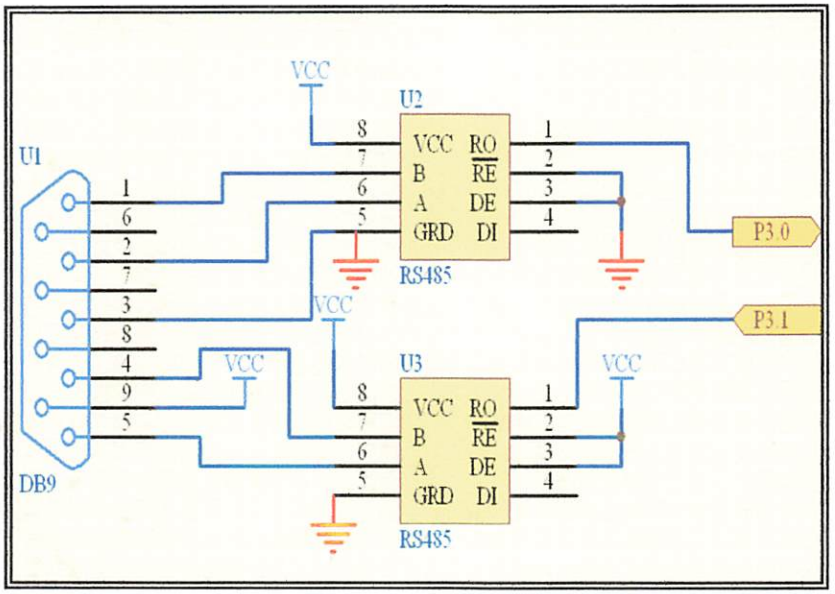
serial terlebih dahulu, yaitu dari level tegangan TTL menjadi level tegangan RS – 232. Penggunaan level converter ini dimaksudkan karena PC tidak dapat mengolah level tegangan TTL.



Gambar 3.9. Rangkaian RS – 232 Dilengkapi DB-9

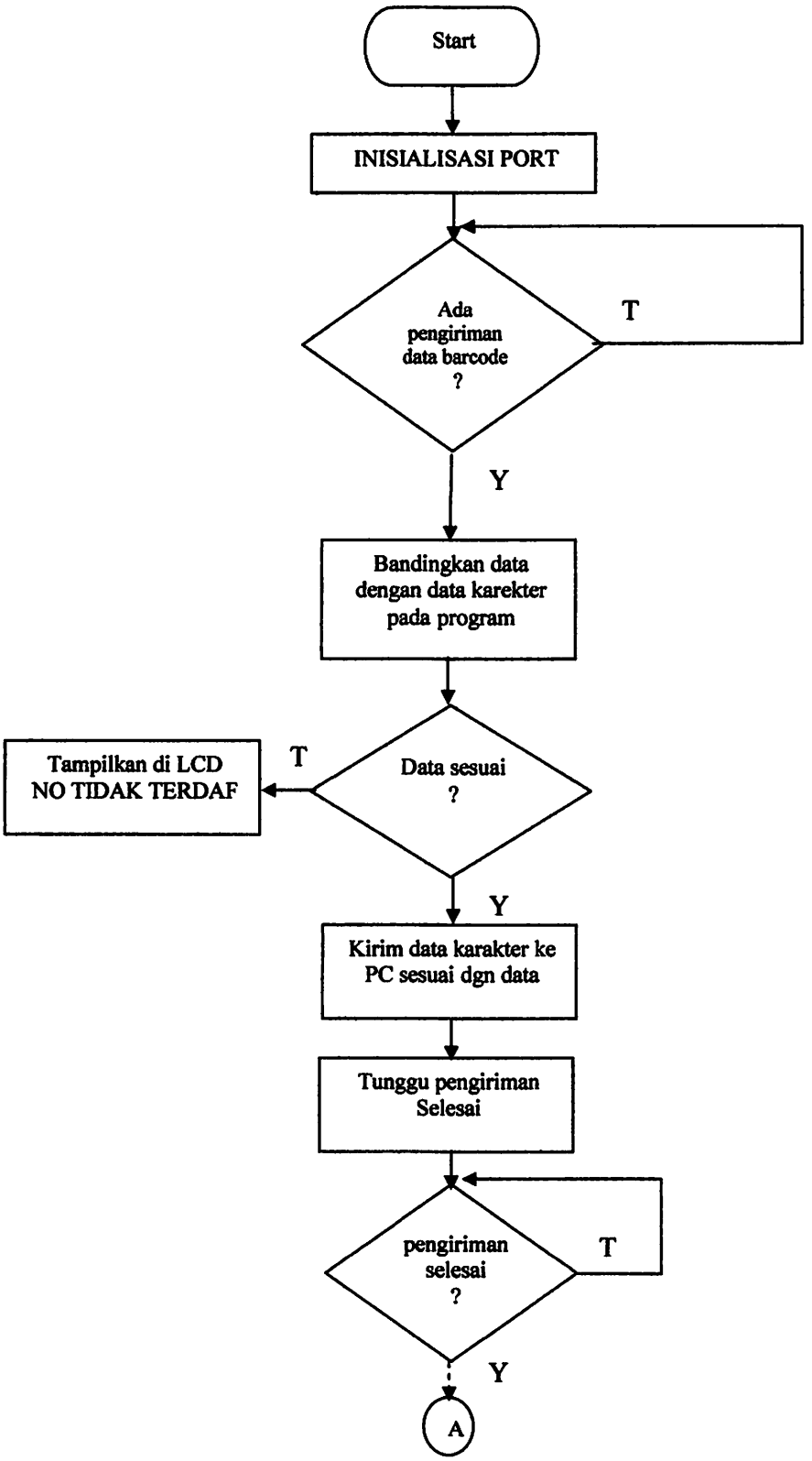
Agar Mikrokontroller AT89S8252 Unit dapat berkomunikasi dengan jarak jauh dengan PC maka dibutuhkan penguat transmisi data serial (buffer) yaitu dengan menggunakan rangkaian RS-485. Rangkaian RS-485 disini digunakan agar data serial yang dikirimkan dari mikrokontroller AT89S8252 dapat menjangkau komputer yang jaraknya ≤1200m. Dalam alat ini RS-485 menggunakan IC MAX RS-485 yang mampu mentransmisikan data serial sampai dengan jarak 1200 m.

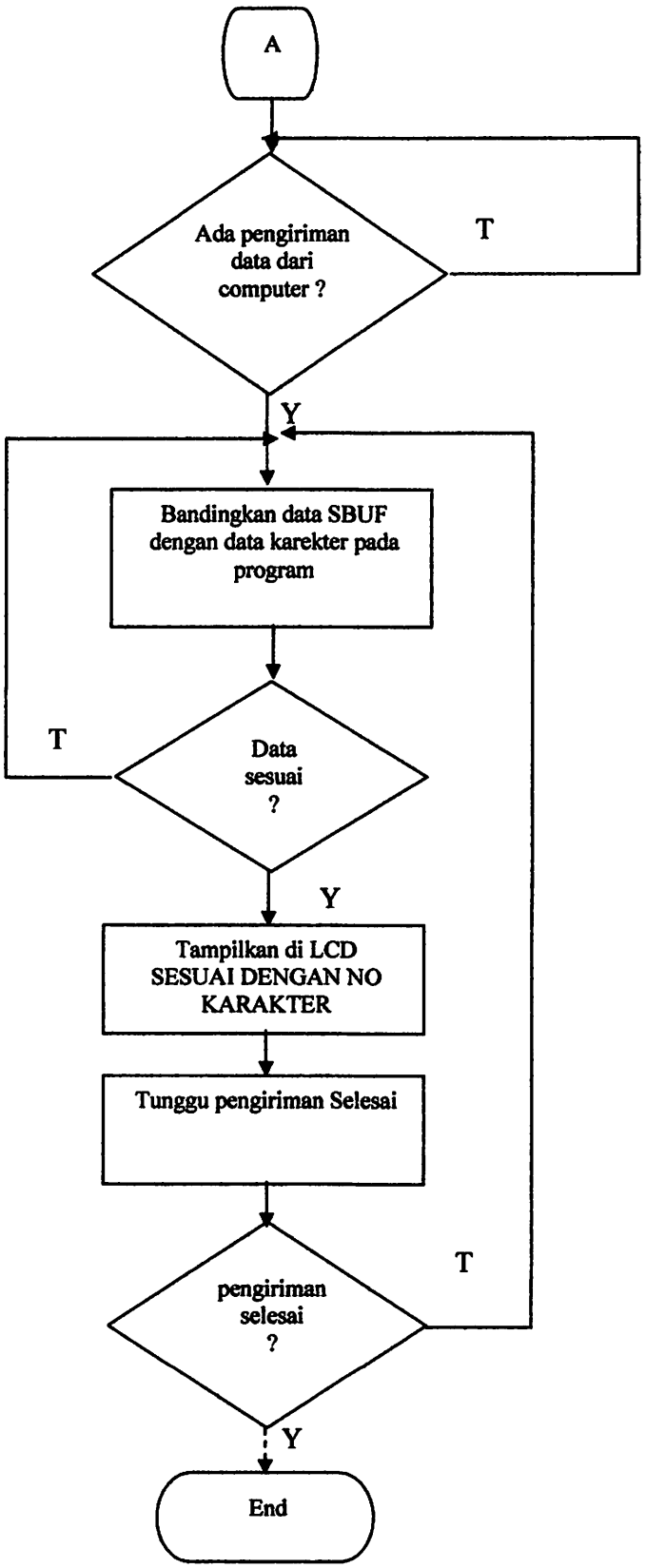
Gambar rangkaian penguat RS-485 yang digunakan pada perancangan alat ini ditunjukkan sebagai berikut :



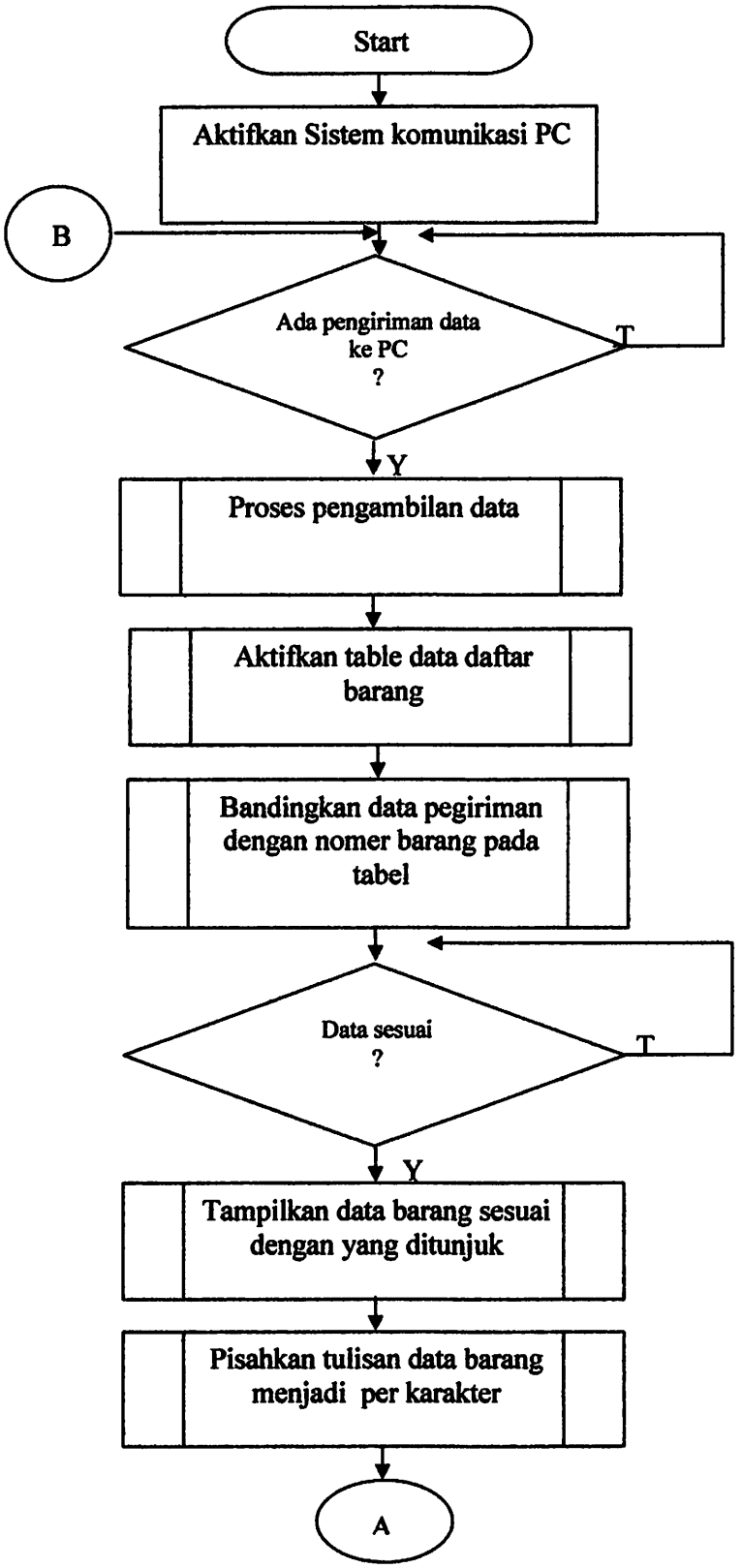
Gambar 3.10. Rangkaian RS-45 Dilengkapi DB-9

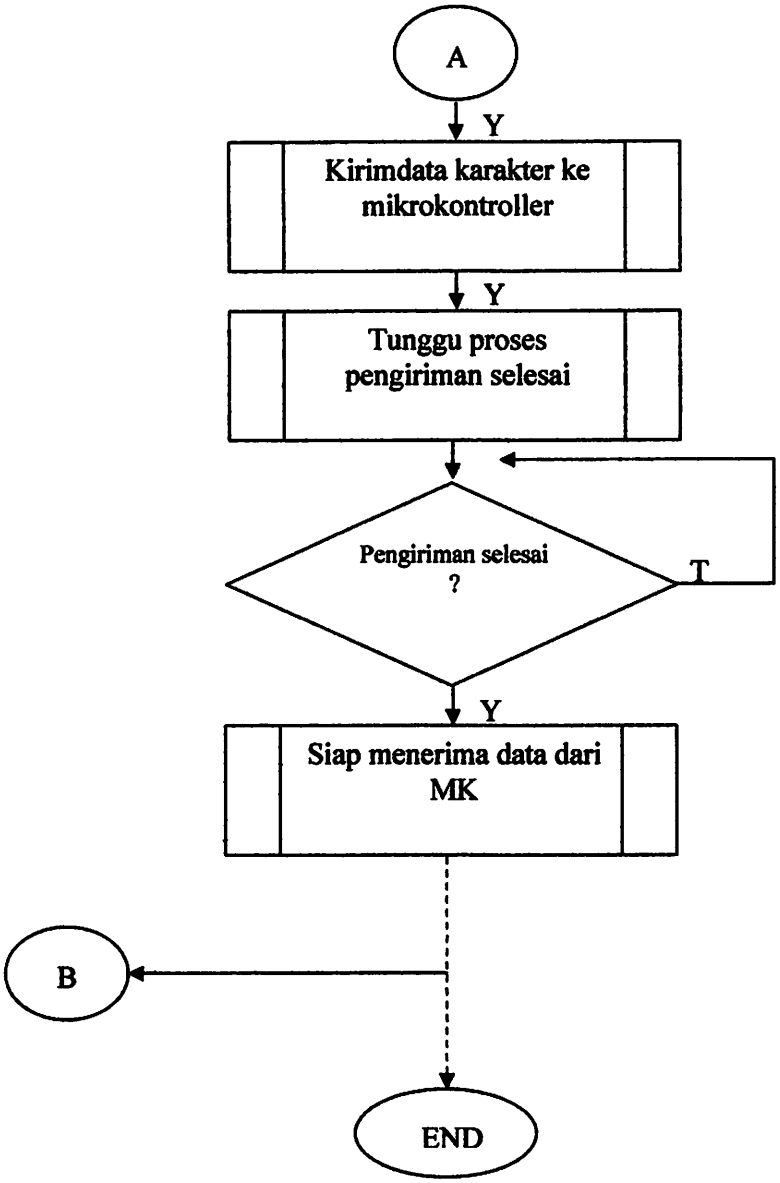
3.7. Flowchart Software Pada Mikrokontroller




A
Ada pengiriman data dari computer ?
T
Y
Bandingkan data SBUF dengan data karekter pada program
Data sesuai ?
T
Y
Tampilkan di LCD SESUAI DENGAN NO KARAKTER
Tunggu pengiriman Selesai
pengiriman selesai ?
T
Y
End

3.8. Flowchart Software Pada Komputer




A
Y
Kirimdata karakter ke mikrokontroller
Y
Tunggu proses pengiriman selesai
Pengiriman selesai ?
T
Y
Siap menerima data dari MK
B
END

BAB IV

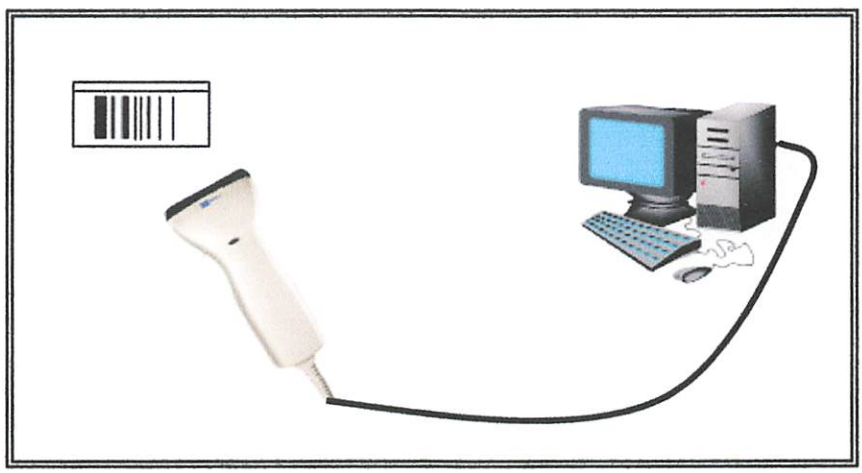
PENGUJIAN ALAT

4.1. Pendahuluan

Pengujian alat ini dilakukan untuk mengetahui kinerja dari keseluruhan sistem rangkaian. Jadi pada tahap ini akan diketahui parameter - parameter dari setiap bagian yang menyusun sistem secara keseluruhan. Pengujian sistem ini dibagi menjadi dua bagian, yang pertama adalah pengujian perangkat keras (hardware) yaitu pengujian pada sensor barcode, dan yang kedua adalah pengujian peragkat lunak (software) yaitu pengujian LCD dan pengujian komunikasi data secara serial.

4.2. Pengujian Perangkat Keras (Hardware)

4.2.1. Pengujian Sensor Barcode



Gambar 4.1. Pengujian Sensor Barcode

BAB IV

PENGUJIAN ALAT

4.1. Pendahuluan

Pengujian alat ini dilakukan untuk mengetahui kinerja dari keseluruhan sistem rangkaian. Jadi pada tahap ini akan dikerahui parameter - parameter dari setiap bagian yang menyusun sistem secara keseluruhan. Pengujian sistem ini dibagi menjadi dua bagian, yang pertama adalah pengujian perangkat keras (hardware) yaitu pengujian pada sensor barcode, dan yang kedua adalah pengujian perangkat lunak (software) yaitu pengujian LCD dan pengujian komunikasi data secara serial.

4.2. Pengujian Perangkat Keras (Hardware)

4.2.1. Pengujian Sensor Barcode

Gambar 4.1. Pengujian Sensor Barcode

1. Tujuan :

   Mengetahui ketelitian pembacaan data barcode pada kartu harga dan jenis barang.

2. Peralatan yang dibutuhkan :

   1) Sensor Barcode
   2) Kartu barcode
   3) Komputer

3. Prosedur Pengujian :

   1) Hubungkan sensor barcode dengan connector keyboard / PS2 pada PC.
   2) Scan data barcode menggunakan barcode scanner hingga terdengar bunyi "beep" yang menandakan data dapat terbaca.
   3) hasil dari proses scanning tadi dapat dilihat pada PC melalui notepad.

4. Hasil Pengujian :

Tabel 4.1. Hasil pengujian Sensor Barcode

NO	5 mm	1 cm	1,5 cm	2 cm	2,5 cm	3 cm
1	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	×
3	✓	✓	✓	×	×	×
4	✓	✓	✓	✓	×	×
5	✓	✓	✓	×	✓	✓
6	✓	✓	✓	✓	×	×
7	✓	✓	✓	✓	✓	✓

8	✓	✓	✓	✓	×	×
9	✓	✓	✓	×	✓	✓
10	✓	✓	✓	✓	×	×
11	✓	✓	✓	✓	✓	×
12	✓	✓	✓	×	×	×
13	✓	✓	✓	✓	✓	✓
14	✓	✓	✓	×	×	×
15	✓	✓	✓	✓	✓	×

Dalam penggunaannya sensor barcode jenis ini dapat membaca dengan jarak maksimal ≤ 2 cm dari kartu barcode. Semakin dekat sensor dengan kartu barcode maka pembacaan dapat lebih maksimal dan jika sensor terlalu jauh dari kartu barcode maka pembacaan tidak dapat maksimal (lambat) dan juga data yang dikirimkan tidak sesuai dengan yang diinginkan.

Dari sampel yang diambil dilakukan sebanyak 15 kali dengan berbagai jarak penembakan sensor barcode mulai dari jarak minimal 5 mm sampai ≤ 3 cm, yaitu pada jarak penembakan sensor barcode 5 mm sampai dengan 1,5 cm tingkat keberhasilan yang diperoleh adalah 100 %, sedangkan pada jarak 2 cm tingkat keberhasilan diperoleh adalah 66,6 %, pada jarak 2,5 cm tingkat keberhasilan diperoleh adalah 53,3 %, pada jarak 3 cm tingkat keberhasilan diperoleh adalah 33,3 % karena jarak optimal pada sensor yang digunakan hanya sampai ≤ 2 cm.

4.3. Pengujian Perangkat Lunak (Software)

4.3.1. Pengujian LCD M1632

1. Tujuan :

   Untuk mengetahui apakah rangkaian LCD yang telah dibuat dapat bekerja sesuai dengan yang direncanakan.

2. Peralatan Yang Dibutuhkan :

   1) Programmer, Emulator dan Evaluation Board (Downloader)

   2) LCD M1632

   3) Komputer

   4) Catu Daya 5 V

3. Prosedur Pengujian :

   1) Hubungkan catu daya dengan rangkaian downloader.

   2) Isikan Program LCD melalui komputer untuk didownloadkan ke mikrokontroller.

   3) Hubungkan LCD dengan mikrokontroller

```
DELAY_INIT_LCD:
MOV R6,#20H

DLY_LCD_LP:
    MOV   R7,#0
    DJNZ  R7,$
    DJNZ  R6,DLY_LCD_LP
    RET

INIT_LCD:
    SETB RS
    CLR E
    MOV A,#DISPCLR
    LCALL CONTROLOUT
    LCALL DELAY_INIT_LCD
    MOV   A,#FUNCSET
    LCALL  CONTROLOUT
    MOV   A,#DISPON
```

```
        MOV   A,#DISPLAY
        LCALL CONTROLOUT
        MOV   A,#CLEAR
        LCALL DELAY_INIT_LCD
        LCALL CONTROLOUT
        MOV A,#DISPSCR
        CLR E
        SETB RS
INIT_LCD:

        RET
        DJNZ  R6,DELAY_LCD_L1
        DJNZ  R7,$
        MOV   R7,#20
DELAY_LCD_L1:

        MOV R6,#50H
        DELAY_INIT_LCD:
```

3) Hubungkan LCD dengan mikrokontroler

mikrokontroler.

5) Isilah Program LCD masukkan kemudian untuk ditransmisikan ke

1) Hubungkan catu daya dengan rangkaian downloader.

2. Prosedur Pengujian :

4) Catu Daya 5 V

3) Komputer

5) LCD M1632

1) Programmer Eksekusi dan Evaluation Board (Downloader)

3. Peralatan yang Digunakan :

sesuai dengan yang ditentukan.

Untuk mengetahui adapun rangkaian LCD yang telah dipasat dapat pakcjh

1. Tujuan :

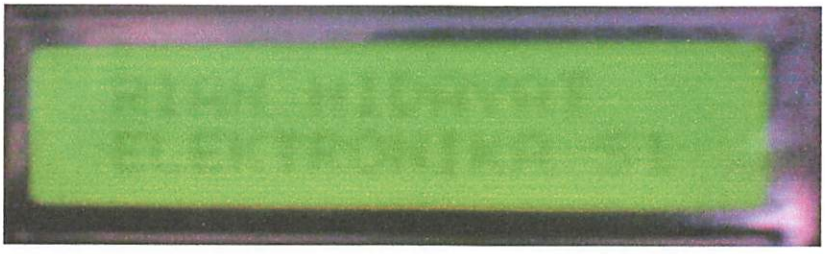
4.3.1. Pengujian LCD M1632

4.3 Pengujian Perangkat Lunak (Software)

```
        LCALL  CONTROLOUT
        MOV    A,#ENTRMOD
        LCALL  CONTROLOUT;
        MOV  DPTR,#NAMA
        LCALL        PRINTSTRING1
        MOV  DPTR,#NIM
        LCALL        PRINTSTRING2
        CALL TUNDA_TAM
        MOV  DPTR,#UNIV
        LCALL        PRINTSTRING1
        MOV  DPTR,#UNIV1
        LCALL        PRINTSTRING2
        CALL TUNDA_TAM

DATA_MATI:
        MOV  LOKASI,#0
        MOV  MY_TEMP,#0
        MOV  MY_COUNTER,#0
        ACALL        TUNDA
        MOV  DPTR,#PRO
        LCALL        PRINTSTRING1
        MOV  DPTR,#PRO1
        LCALL        PRINTSTRING2
        JMP    DATA_MATI

POSISI2_1:
        MOV    A,#1
POSISI2:
        ADD    A,#11000000B
        SJMP  POSISI_SUB

POSISI1_1:
        MOV    A,#1
POSISI1:
        ADD    A,#10000000B
POSISI_SUB:
        DEC    A
        LCALL CONTROLOUT
        RET

PRINTSTRING2:
        LCALL POSISI2_1
        SJMP  PRINTSTRING
PRINTSTRING1:
        LCALL POSISI1_1
PRINTSTRING:
        SJMP  OUTSTRING
PRINTSTRINGLOOP:
```

```
        LCALL  DATAOUT
        INC    DPTR

OUTSTRING:
        CLR    A
        MOVC   A,@A+DPTR
        JNZ    PRINTSTRINGLOOP
        RET

CONTROLOUT:
        CPL RS
        CPL E
        MOV P0,A
        CPL E
        CPL RS
        MOV P0,#0FFH
        SJMP LCD_OUT

DATAOUT:
        CPL E
        MOV P0,A
        CPL E

LCD_OUT:
        MOVX   @DPTR,A

DELAY_LCD:
        PUSH ACC
        MOV    A,#250
        DJNZ   ACC,$
        POP ACC
        RET;

NAMA:    DB ' RIAN HIDAYAT   ',0
NIM:     DB ' ELEKTRONIKA S1 ',0
UNIV:    DB '     ITN      ',0
UNIV1:   DB '     MALANG     ',0
```

4. Hasil Pengujian :



Gambar 4.2. Tampilan Pada LCD M1632

4.3.2. Pengujian Komunikasi Data Serial

1. Tujuan :

   untuk mengetahui apakah mikrokontroller dapat berkomunikasi secara serial dengan PC sesuai dengan yang diharapkan

2. Peralatan yang digunakan :
   1. Komputer
   2. Rangkaian Serial
   3. Kabel Serial

3. Prosedur Pengujian :
   1. Hubungkan kabel serial dengan rangkaian serial (kabel serial dengan connector db-9 female terhubung dengan komputer dan db-9 male terhubung dengan rangkaian serial).
   2. Aktifkan program komunikasi serial pada komputer. Membuat aplikasi tes komunikasi data serial pada delphi 7 dengan program sebagai berikut :

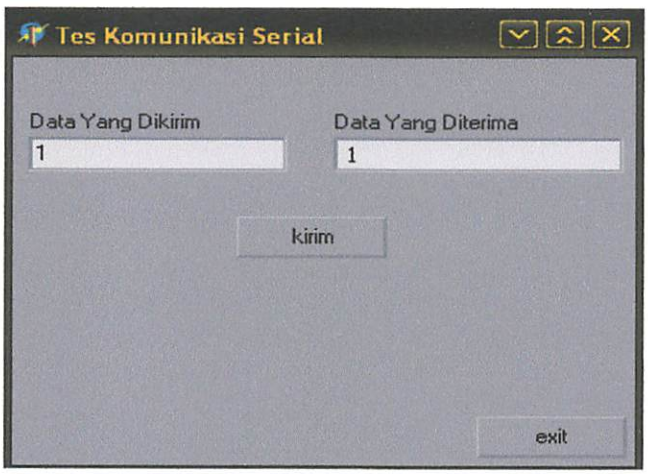
```
unit tes serial;

interface

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

Procedure OutPort(addr:word;data:byte);
function InPort(addr:word):byte;
procedure Delay(lama:longint);



type
  TForm1 = class(TForm)
    Button1: TButton;
```

```
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
Procedure OutPort(addr:word;data:byte);
begin
 asm
  mov  dx,addr
  mov  al,data
  out  dx,al
 end;
end;

function InPort(addr:word):byte;
var m:byte;
begin
 asm
  mov  dx,addr
  in   al,dx
  mov  m,al
 end;
 Inport:=m;
end;

procedure Delay(lama:longint);
var ref:longint;

begin
ref:=GetTickCount;
repeat
application.ProcessMessages ;
until ((gettickCount-ref)>=lama);
```

```
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
application.Terminate;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
SetupSerial(1);

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Outserial(1,strtoint(edit1.text));
Delay(2);
edit2.text:=inttostr(inserial(1));
end;

end.
```

3. Hubungkan pin2 dan pin3 (short) pada rangkaian serial.

4. Hasil Pengujian :

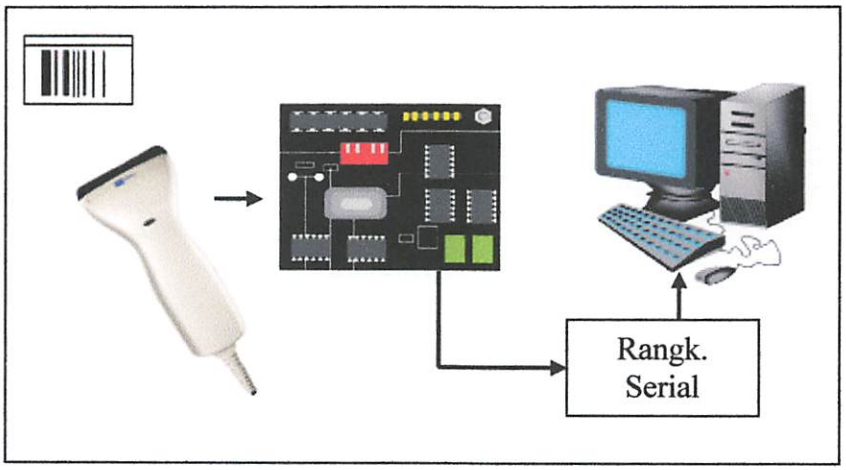


Gambar 4.3. Jendela Hyper Terminal

Dari hasil pengujian komunikasi data serial, apabila kita mengirimkan data numerik 1, maka data yang diterima juga harus bernilai sama dengan data yang

dikirim. Hal ini menandakan bahwa komunikasi data secara serial bekerja sesuai dengan yang diharapkan.

4.4. Pengujian Sistem Keseluruhan



Gambar 4.4. Pengujian Sistem Keseluruhan

1. Tujuan :

   Untuk mengetahui hasil akhir dari sistem yang telah dibuat.

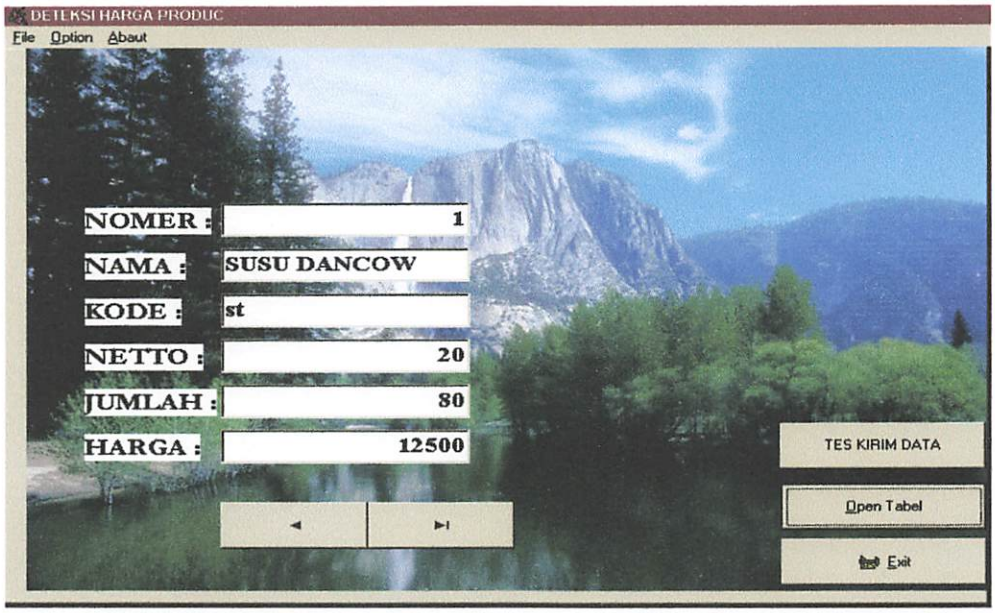
2. Peralatan Yang Dibutuhkan :

   1) Barcode Reader
   2) Mikrokontroller AT89S8252
   3) Rangkaian Serial meliputi rangkaian RS485 dan rangkaian RS232
   4) Komputer

3. Prosedur Pengujian :

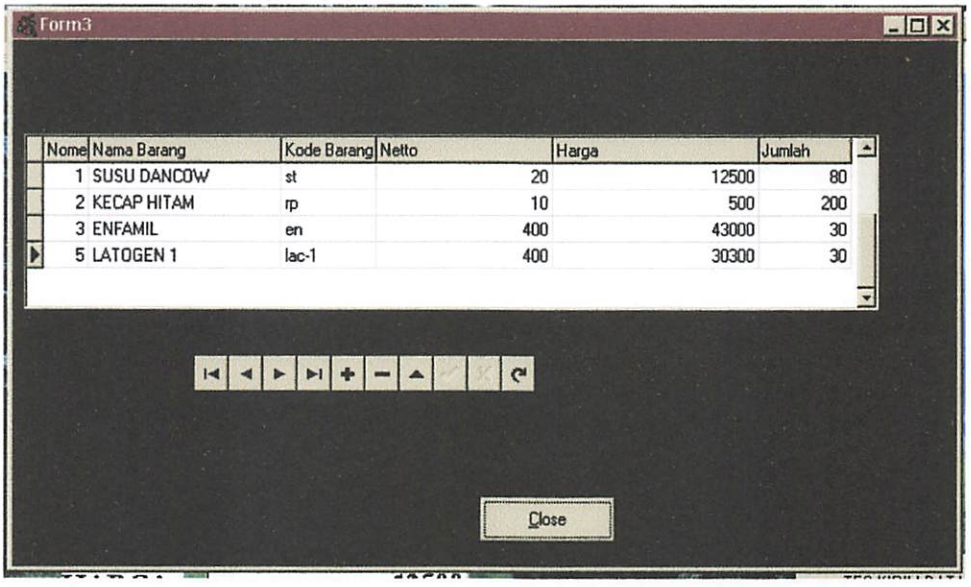
   1) Hubungkan seluruh connector, mulai dari barcode reader, mikrokontroller AT89S8252, rangkaian serial dan komputer.
   2) Scan kartu barcode.

4. Hasil Pengujian :



Gambar 4.5. Hasil Pengujian Sistem Keseluruhan

Untuk proses selanjutnya data dapat dilihat didalam PC itu sendiri berupa database file yang sudah disimpan dalam bentuk database. Jika ada penambahan atau pengurangan database dapat langsung dirubah melalui database itu sendiri. sedangkan bentuk tampilan database pada PC dapat dilihat pada gambar dibawah ini :



Gambar 4.6. Hasil pada database

Gambar 4.5. Hasil Pengujian Sistem Keseluruhan

Untuk proses selanjutnya, data akan dilihat dibalikan PC ke sistem bentuk warna. File yang sudah disimpan dalam bentuk binary. Jika ada pemanfaatan data yang ingin dilihat dapat langsung dibuka terlebih dahulu ke sistem. Selanjutnya bentuk tampilan dari hasil PC. Jika dilihat pada gambar dibawah ini :

Gambar 4.6. Hasil pada database

Dari hasil pengujian rangkaian keseluruhan proses pengiriman dan penerimaan data sudah berjalan tetapi memerlukan proses waktu yang agak lama karena proses pengiriman data yang dikirim perkarakter.

BAB V
PENUTUP

5.1. Kesimpulan

1. Pembacan data pada kartu barcode dapat dilakukan dengan sensor barcode, optimal pada jarak ≤ 2 cm.
2. Pengiriman data barcode harga dan jenis barang dapat diterima oleh komputer dan disimpan sebagai daftar harga dan jenis barang secara otomatis.
3. Menggunakan rangkaian IC RS485 untuk membantu komunikasi serial antara Mikro dan PC agar data yang diterima tidak cacat (dalam simulasi hanya menggunakan kabel berukuran 10 meter).
4. Pada pengujian komunikasi serial dengan menggunakan aplikasi sotfware tes komunikasi serial yang terhubung dengan rangkaian RS – 232 yaitu dengan menghubung singkatkan pin no 2 dan 3 diketahui bahwa data yang dikirim sama dengan data yang diterima ini menunjukkan bahwa komunikasi serial bekerja sesuai dengan yang diharapkan.
5. Dari hasil pengujian rangkaian keseluruhan yang dilakukan sebanyak 15 dengan berbagai jarak penembakan sensor barcode mulai dari jarak minimal 5 mm sampai ≤ 3 cm, yaitu pada jarak penembakan sensor barcode 5 mm sampai dengan 1,5 cm tingkat keberhasilan yang diperoleh adalah 100 %, sedangkan pada jarak 2 cm tingkat keberhasilan diperoleh adalah 66,6 %, pada jarak 2,5 cm tingkat

BAB V
PENUTUP

5.1. Kesimpulan

1. Pembacaan data pada kartu barcode dapat dilakukan dengan sensor barcode. optimal pada jarak ≤ 2 cm.
2. Pengiriman data barcode harga dan jenis barang dapat diterima oleh komputer dan disimpan sebagai daftar harga dan jenis barang secara otomatis.
3. Menggunakan rangkaian IC RS-485 untuk membantu komunikasi serial antara Mikro dan PC agar data yang diterima tidak cacat (dalam simulasi hanya menggunakan kabel berukuran 10 meter).
4. Pada pengujian komunikasi serial dengan menggunakan aplikasi software tes komunikasi serial yang terhubung dengan rangkaian RS – 232 yaitu dengan menghubung singkatkan pin no 2 dan 3 diketahui bahwa data yang dikirim sama dengan data yang diterima ini membuktikan bahwa komunikasi serial bekerja secara dengan yang diharapkan.
5. Dari hasil pengujian rangkaian keseluruhan yang dilakukan sebanyak 15 dengan berbagai jarak pembacaan sensor barcode mulai dari jarak minimal 5 mm sampai ≤ 3 cm. yaitu pada jarak pembacaan sensor barcode 5 mm sampai dengan 1.5 cm tingkat keberhasilan yang diperoleh adalah 100 %. sedangkan pada jarak 2 cm tingkat keberhasilan diperoleh adalah 66.6 %. pada jarak 2.5 cm tingkat

keberhasilan diperoleh adalah 53,3 %, pada jarak 3 cm tingkat keberhasilan diperoleh adalah 33,3 % karena jarak optimal pada sensor yang digunakan hanya sampai ≤ 2 cm

5.2. Saran

1. Database harga dan jenis barang dapat memuat lebih banyak daftar barang-barang.
2. Diharapkan pada pengembangan selanjutnya dapat menggunakan komunikasi data secara wireless.
3. Dapat menggunakan jenis sensor barcode lain dengan keakuratan yang lebih tinggi.

keberhasilan diperoleh adalah 33,3 %. pada jarak 3 cm tingkat keberhasilan diperoleh adalah 33,3 %. karena jarak optimal pada sensor yang digunakan hanya sampai ≤ 2 cm

5.2. Saran

1. Database harga dan jenis barang dapat memuat lebih banyak dalam barang-barang.
2. Dibarapkan pada pengembangan selanjutnya dapat menggunakan komunikasi data secara wireless.
3. Dapat menggunakan jenis sensor barcode lain dengan kecukupan yang lebih tinggi.

DAFTAR PUSTAKA

1. Bar Coding Basics www.taltech.com
2. Datasheet Mikrokontroller AT89S8252 www.atmel.com
3. Data Sheet LCD http://www.delta-electronik.com
4. Datasheet IC Max232 www.alldatasheet.com
5. RS485 Data Interface www.arcelect.com
6. *Barcode Scanner, User Manual*, Metrologic Scanner
7. General Symbology Background Information
   www.barcodeisland.com
8. Code 39 Symbology www.barcodeisland.com
9. Widodo Budiharto,S.Si,M.Kom, Interfacing Komputer dan Mikrokontroler, Elex Media Computindo, 2004
10. RS232 / EIA232 – Null Modem www.arcelect.com
11. RS232 Serial Communication Overview www.quatech.com
12. Datasheet IC Max485 www.alldatasheet.com

DAFTAR PUSTAKA

1. Bar Coding Basics
2. Datasheet Mikrokontroller AT89S8252
3. Data Sheet LCD
4. Datasheet IC Max232
5. RS485 Data Interface
6. Barcode Scanner. User Manual. Motorlogic Scanner
7. General Symbology Background Information
8. Code 39 Symbology
9. Widodo Budiharto,S.Si,M.Kom. Interfacing Komputer dan Mikrokontroler. Elex Media Computindo. 2004
10. RS232 / EIA232 – Null Modem
11. RS232 Serial Communication Overview
12. Datasheet IC Max485

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

LEMBAR BIMBINGAN SKRIPSI

Nama	: Rian Hidayat
NIM	: 01.17.010
Jurusan	: Teknik Elektro S-1
Program Studi	: Elektronika
Judul Skripsi	: Perancangan dan pembuatan alat pendeteksi harga dan jenis barang di supermarket menggunakan barcode berbasis MC AT89S8252 yang terhubung dengan pc sebagai database
Tanggal Pengajuan Skripsi	: 7 Desember 2005
Selesai Penulisan Skripsi	: Maret 2006
Dosen Pembimbing	: Ir. F. Yudi Limpraptono,MT
Telah Dievaluasi Dengan Nilai	: 90

Mengetahui
Ketua Jurusan Elektro S-1

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

Diperiksa dan disetujui
Dosen Pembimbing

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

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Bimbingan Skripsi

NAMA : RIAN HIDAYAT
NIM : 0117010
Masa Bimbingan : 7 Desember 2005 – 8 Mei 2006
Judul Skripsi : **PERANCANGAN DAN PEMBUATAN ALAT PENDETAKSI HARGA DAN JENIS BARANG DI SUPERMARKET MENGGUNAKAN BARCODE BERBASIS MC AT89S8252 YANG TERHUBUNG DENGAN PC SEBAGAI DATA BASE**

No	Tanggal	Uraian	Paraf Pembimbing
1	11/2 2006	Bab I, II, III perlu diperbaiki tata tulis	
2	15/2 2006	Bab IV & V perbaiki kesimpulan	
3	21/2 2006	Proposal Hasil	
4		Acc	
5	13/3 06	Bab I – V	
6	13/3 06	Kata abstrak	
7			
8			
9			
10			

Malang, 2006
Dosen Pembimbing

Ir. F. YUDI LIMPRAPTONO, MT
NIP.Y. 1039500274

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

FORMULIR PERBAIKAN SKRIPSI

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

Hari : Senin
Tanggal : 20 Maret 2006

Telah dilakukan perbaikan Skripsi oleh :

Nama : Rian Hidayat
NIM : 01.17.010
Jurusan : Teknik Elektro S-1
Program Studi : Elektronika
Judul Skripsi : Perancangan dan pembuatan alat pendeteksi harga dan jenis barang di supermarket menggunakan barcode berbasis MC AT89S8252 yang terhubung dengan pc sebagai database

No.	Materi Perbaikan	Keterangan
1.	Gambar 4.2	
2.	Kesimpulan	
3.	Flowchart	
4.	Database	
5.	Proses registrasi barcode ke system	

Disetujui
Penguji Kedua

M. Ashar,ST,MT

Mengetahui
Dosen Pembimbing

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

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

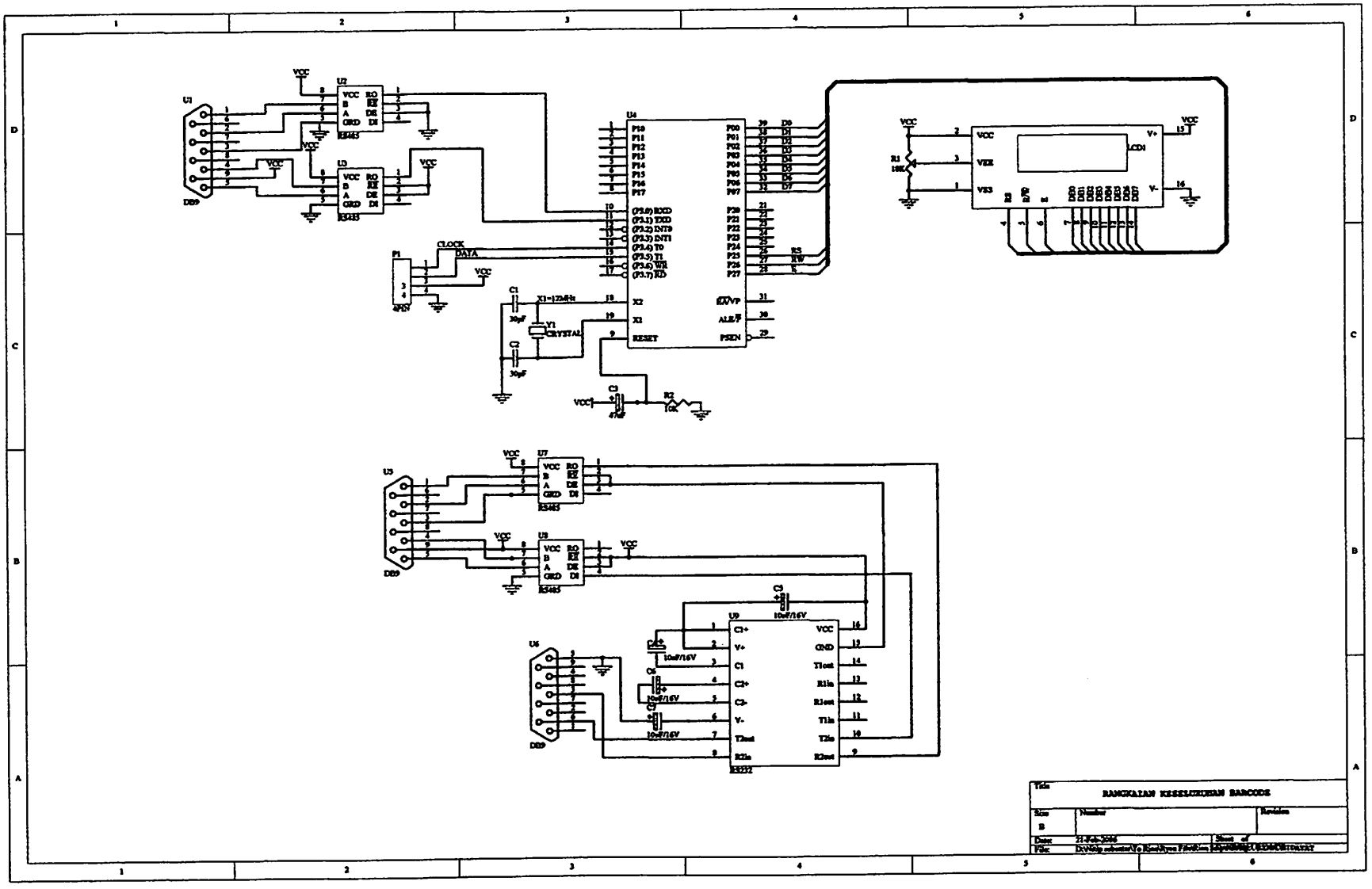
Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentasi T. Energi Listrik / T. Elektronika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Rian hidayat.
N I M : 01/7010
Perbaikan meliputi :

- Gambar 4.2.
- Database
- Kesimpulan
- Proses Registrasi Barcode ke system.
- Flowchart

Malang,

(M. ASHAN)


RANGKAIAN KESELURUHAN BARCODE
Size
B
Number
Revision
Date: 21-Feb-2010
Sheet of
File:
VCC
GND
CLOCK
DATA
CRYSTAL
LCD1
RS485
DB9
10uF/16V
30pF
4k7
10K

LAMPIRAN

MIKROKONTROLER.Rian

```
; LCD CONSTANTA
DISPCLR  EQU 00000001B
BLINK    EQU 00001101B
ENTRMOD  EQU 00000110B
DISPON   EQU 00001100B
CURSOR   EQU 00011100B
FUNCSET  EQU 00111000B
;
;DEVICE ADDRESS [LCD]
RS       BIT P2.6  ;LCD
E        BIT P2.7  ;LCD
MY_DATA      BIT P3.4
MY_COUNTER EQU 31H
MY_TEMP      EQU 32H
KEYDATA      EQU 33H

COD1     EQU 27H
COD2     EQU 28H
COD3     EQU 29H
DATAKEY0 EQU 34H
DATAKEY1 EQU 35H
DATAKEY2 EQU 36H
DATAKEY3 EQU 37H
DATAKEY4 EQU 38H
DATAKEY5 EQU 39H
DATAKEY6 EQU 40H
DATAKEY7 EQU 41H
DATAKEY8 EQU 42H
DATAKEY9 EQU 43H
DATAKEY10 EQU 44H
DATAKEY11 EQU 45H
DATAKEY12 EQU 46H
LOKASI       EQU 47H
DATAKEY13 EQU 48H
DATAKEY14 EQU 49H
DATAKEY15 EQU 50H
DATAKEY16 EQU 51H
DATAKEY17 EQU 52H
DATAKEY18 EQU 53H
DATAKEY19 EQU 54H
DATAKEY20 EQU 55H
DATAKEY21 EQU 56H
DATAKEY22 EQU 57H
DATAKEY23 EQU 58H
DATAKEY24 EQU 59H
DATAKEY25 EQU 60H
DATAKEY26 EQU 61H

        ORG   0000H          ; Mulai dari alamat reset
        AJMP  MULAI          ; Loncat ke label mulai
        ORG   0013H          ; Alamat interupt
        AJMP  DATA_BARCODE   ; Jalankan Instruksi DATA_BARCODE

        ORG   23H            ; alamat untuk serial comm. interupt
        CLR   ES             ; matikan interupt serial
        JNB   RI,$           ; tunggu sampai selesai menerima
        CLR   RI             ; clear receive interupt flag
        MOV   R1,SBUF
        MOV   B,R1
```

```
MOV   A,LOKASI
      INC   A
      MOV   LOKASI,A
PIL0:
      CJNE  A,#00000001B,PILL1
      MOV   DATAKEY1,B
      MOV A,#1
      LCALL POSISI1
      MOV A,DATAKEY1
      AJMP TAMPILL
PILL1:
      CJNE  A,#00000010B,PIL2
      MOV   DATAKEY2,B
      MOV A,#2
      LCALL POSISI1
      MOV A,DATAKEY2
      AJMP TAMPILL
PIL2:
      CJNE  A,#00000011B,PIL3
      MOV   DATAKEY3,B
      MOV A,#3
      LCALL POSISI1
      MOV A,DATAKEY3
      AJMP TAMPILL
PIL3:
      CJNE  A,#00000100B,PIL4
      MOV   DATAKEY4,B
      MOV A,#4
      LCALL POSISI1
      MOV A,DATAKEY4
      AJMP TAMPILL
PIL4:
      CJNE  A,#00000101B,PIL5
      MOV   DATAKEY5,B
      MOV A,#5
      LCALL POSISI1
      MOV A,DATAKEY5
      AJMP TAMPILL
PIL5:
      CJNE  A,#00000110B,PILL6
      MOV   DATAKEY6,B
      MOV A,#6
      LCALL POSISI1
      MOV A,DATAKEY6
      AJMP TAMPILL
PILL6:
      CJNE  A,#00000111B,PIL7
      MOV   DATAKEY7,B
      MOV A,#7
      LCALL POSISI1
      MOV A,DATAKEY7
      AJMP TAMPILL
PIL7:
      CJNE  A,#00001000B,PIL8
      MOV   DATAKEY8,B
      MOV A,#8
      LCALL POSISI1
      MOV A,DATAKEY8
      AJMP TAMPILL
PIL8:
      CJNE  A,#00001001B,PIL10
      MOV   DATAKEY9,B
      MOV A,#9
      LCALL POSISI1
      MOV A,DATAKEY9
      AJMP TAMPILL
PIL10:
      CJNE  A,#00001010B,PIL11
      MOV   DATAKEY10,B
      MOV A,#10
      LCALL POSISI1
      MOV A,DATAKEY10
      AJMP TAMPILL
PIL11:
      CJNE  A,#00001011B,PIL12
      MOV   DATAKEY11,B
      MOV A,#11
      LCALL POSISI1
      MOV A,DATAKEY11
      AJMP TAMPILL
PIL12:
      CJNE  A,#00001100B,PIL13
      MOV   DATAKEY12,B
      MOV A,#12
      LCALL POSISI1
      MOV A,DATAKEY12
      AJMP TAMPILL
PIL13:
      CJNE  A,#00001101B,PIL14
      MOV   DATAKEY13,B
      MOV A,#13
      LCALL POSISI1
      MOV A,DATAKEY13
      AJMP TAMPILL
PIL14:
      CJNE  A,#00001110B,PIL15
      MOV   DATAKEY14,B
      MOV A,#14
      LCALL POSISI1
      MOV A,DATAKEY14
      AJMP TAMPILL
PIL15:
      CJNE  A,#00001111B,PIL16
      MOV   DATAKEY15,B
      MOV A,#15
      LCALL POSISI1
      MOV A,DATAKEY15
      AJMP TAMPILL
PIL16:
      CJNE  A,#00010000B,PIL17
      MOV   DATAKEY16,B
      MOV A,#16
      LCALL POSISI1
      MOV A,DATAKEY16
      AJMP TAMPILL

PIL17:
      CJNE  A,#00010001B,PIL18
      MOV   DATAKEY17,B
      MOV A,#3
```

```
MOV   A,LOKASI
INC   A
MOV   LOKASI,A
PIL0:
CJNE  A,#00000010B,PIL1
MOV   DATAKEY1,B
MOV A,#1
LCALL POSISI
MOV A,DATAKEY1
AJMP TAMPIL
PIL1:
CJNE  A,#00000010B,PIL2
MOV   DATAKEY2,B
MOV A,#2
LCALL POSISI
MOV A,DATAKEY2
AJMP TAMPIL
PIL2:
CJNE  A,#00000011B,PIL3
MOV   DATAKEY3,B
MOV A,#3
LCALL POSISI
MOV A,DATAKEY3
AJMP TAMPIL
PIL3:
CJNE  A,#00001000B,PIL4
MOV   DATAKEY4,B
MOV A,#4
LCALL POSISI
MOV A,DATAKEY4
AJMP TAMPIL
PIL4:
CJNE  A,#00001001B,PIL5
MOV   DATAKEY5,B
MOV A,#5
LCALL POSISI
MOV A,DATAKEY5
AJMP TAMPIL
PIL5:
CJNE  A,#00001010B,PIL6
MOV   DATAKEY6,B
MOV A,#6
LCALL POSISI
MOV A,DATAKEY6
AJMP TAMPIL
PIL6:
CJNE  A,#00001011B,PIL7
MOV   DATAKEY7,B
MOV A,#7
LCALL POSISI
MOV A,DATAKEY7
AJMP TAMPIL
PIL7:
CJNE  A,#00001000B,PIL8
MOV   DATAKEY8,B
MOV A,#8
LCALL POSISI
MOV A,DATAKEY8
AJMP TAMPIL
PIL8:

PIL17:
CJNE  A,#00010000B,PIL18
MOV   DATAKEY17,B
MOV A,#9
CJNE  A,#00010010B,PIL19
MOV   DATAKEY9,B
MOV A,#9
LCALL POSISI
MOV A,DATAKEY9
AJMP TAMPIL
PIL10:
CJNE  A,#00010100B,PIL11
MOV   DATAKEY10,B
MOV A,#10
LCALL POSISI
MOV A,DATAKEY10
AJMP TAMPIL
PIL11:
CJNE  A,#00010110B,PIL12
MOV   DATAKEY11,B
MOV A,#11
LCALL POSISI
MOV A,DATAKEY11
AJMP TAMPIL
PIL12:
CJNE  A,#00011000B,PIL13
MOV   DATAKEY12,B
MOV A,#12
LCALL POSISI
MOV A,DATAKEY12
AJMP TAMPIL
PIL13:
CJNE  A,#00011010B,PIL14
MOV   DATAKEY13,B
MOV A,#13
LCALL POSISI
MOV A,DATAKEY13
AJMP TAMPIL
PIL14:
CJNE  A,#00011100B,PIL15
MOV   DATAKEY14,B
MOV A,#14
LCALL POSISI
MOV A,DATAKEY14
AJMP TAMPIL
PIL15:
CJNE  A,#00011110B,PIL16
MOV   DATAKEY15,B
MOV A,#15
LCALL POSISI
MOV A,DATAKEY15
AJMP TAMPIL
PIL16:
CJNE  A,#00100000B,PIL17
MOV   DATAKEY16,B
MOV A,#16
LCALL POSISI
MOV A,DATAKEY16
AJMP TAMPIL
```

```
        LCALL POSISI2
        MOV A,DATAKEY17
        AJMP TAMPILL
PIL18:
        CJNE    A,#00010010B,PIL19
        MOV     DATAKEY18,B
        MOV A,#4
        LCALL POSISI2
        MOV A,DATAKEY18
        AJMP TAMPILL
PIL19:
        CJNE    A,#00010011B,PIL20
        MOV     DATAKEY19,B
        MOV A,#5
        LCALL POSISI2
        MOV A,DATAKEY19
        AJMP TAMPILL

PIL20:
        CJNE    A,#00010100B,PIL21
        MOV     DATAKEY20,B
        MOV A,#11
        LCALL POSISI2
        MOV A,DATAKEY20
        AJMP TAMPILL
PIL21:
        CJNE    A,#00010101B,PIL22
        MOV     DATAKEY21,B
        MOV A,#12
        LCALL POSISI2
        MOV A,DATAKEY21
        AJMP TAMPILL


PIL22:
        CJNE    A,#00010110B,PIL23
        MOV     DATAKEY22,B
        MOV A,#13
        LCALL POSISI2
        MOV A,DATAKEY22
        AJMP TAMPILL
PIL23:
        CJNE    A,#00010111B,PIL24
        MOV     DATAKEY23,B
        MOV A,#14
        LCALL POSISI2
        MOV A,DATAKEY23
        AJMP TAMPILL
PIL24:
        CJNE    A,#00011000B,PIL25
        MOV     DATAKEY24,B
        MOV A,#15
        LCALL POSISI2
        MOV A,DATAKEY24
        AJMP TAMPILL
PIL25:
        CJNE    A,#00011001B,PIL26
        MOV     DATAKEY25,B
        MOV A,#16
        LCALL POSISI2
```

```
        MOV A,DATAKEY25
        AJMP TAMPILL

PIL26:
        CJNE    A,#00011010B,PIL27
        MOV     DATAKEY26,B
        MOV A,#16
        LCALL POSISI1
        MOV     A,#' '
        LCALL DATAOUT
        ACALL TUNDA_TAM
        RETI

PIL27:
        AJMP PIL0



TAMPILL:
        ACALL DATAOUT
        ACALL TUNDA_T
        SETB    ES
        RETI

MULAI:
        MOV     P1,#00H         ; Mulai Port1 dengan alamat 00H
        ACALL INIT_INT          ; Jalankan prosedur INIT_INT
        ACALL init_serial
        ACALL INIT_LCD

DATA_BARCODE:
        PUSH    DPH
        PUSH    DPL
        MOV LOKASI,#0
        INC     MY_COUNTER ; Tambahkan prosedur MY_COUNTER dengan 1
        MOV     A,MY_COUNTER        ; Pindahkan isi data MY_COUNTER ke Akumulator
        CJNE    A,#1,CEK_PARITY
        AJMP    NT_EXIT_DATAK

CEK_PARITY:
        CJNE    A,#10,CEK_STOPBIT
        AJMP    NT_EXIT_DATAK       ; Jalankan prosedur NT_EXIT_DATAK

CEK_STOPBIT:
        CJNE    A,#11,DATABIT
        AJMP    EXIT_DATAK

DATABIT:
        JB      MY_DATA,MY_LOGIC1

MY_LOGIC0:
        CLR     C               ;
        MOV     A,MY_TEMP
        RRC     A
        MOV     MY_TEMP,A
        AJMP    NT_EXIT_DATAK

MY_LOGIC1:
        SETB    C
```

```
        MOV   A,MY_TEMP
        RRC   A
        MOV   MY_TEMP,A
        AJMP  NT_EXIT_DATAK

EXIT_DATAK:
        MOV LOKASI,#0
        MOV   MY_COUNTER,#0       ; Isi MY_COUNTER dengan data O
        MOV   A,MY_TEMP    ; Pindahkan MY_TEMP ke akumulator
        MOV   MY_TEMP,#0   ; Isi MY_TEMP dengan data 0
ANGKA0:
        CJNE  A,#45H,ANGKA1         ;
        MOV   A,#0
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA     ; Jalankan prosedur SENDATA

ANGKA1:
        CJNE  A,#16H,ANGKA2
        MOV   A,#1
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA     ; Jalankan prosedur SENDATA


ANGKA2:
        CJNE  A,#1EH,ANGKA3
        MOV   A,#2
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA     ; Jalankan prosedur SENDATA


ANGKA3:
        CJNE  A,#26H,ANGKA4
        MOV   A,#3
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA     ; Jalankan prosedur SENDATA


ANGKA4:
        CJNE  A,#25H,ANGKA5
        MOV   A,#4
```

```
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA


ANGKA5:
        CJNE  A,#2EH,ANGKA6
        MOV   A,#5
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA


ANGKA6:
        CJNE  A,#36H,ANGKA7
        MOV   A,#6
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA

ANGKA7:
        CJNE  A,#3DH,ANGKA8
        MOV   A,#7
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA


ANGKA8:
        CJNE  A,#3EH,ANGKA9
        MOV   A,#8
        MOV   KEYDATA,A
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA


ANGKA9:
        CJNE  A,#46H,ANGKA_MATI
        MOV   A,#9
        MOV   KEYDATA,A
```

```
        MOV   LOKASI,#0
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2
        AJMP  SENDDATA      ; Jalankan prosedur SENDATA

ANGKA_MATI:
        MOV   DPTR,#NO_MAHA
        LCALL PRINTSTRING2

SENDDATA:
KIRIM: CLR    ES       ; matikan serial interupt saat mengirim
        MOV   A,KEYDATA
        MOV   SBUF,A ; isi serial buffer dengan data yg dikirim
        JNB   TI,$     ; tunggu pengiriman selesai
        CLR   TI       ; clear transmit interupt flag
        MOV   50H,#50
TUNDAL1:
        MOV   51H,#50
        DJNZ  51H,$
        DJNZ  50H,TUNDAL1
        SETB  ES       ; hidupkan kembali serial interupt
        SETB  EA

NT_EXIT_DATAK:
        POP   DPL
        POP   DPH
;       POP   A
        RETI

TUNDA_T:
        MOV   R7,#25
LOOP10:       MOV   R6,#25
LOOP9: MOV    R5,#25
        DJNZ  R5,$
        DJNZ  R6,LOOP9
        DJNZ  R7,LOOP10
        RET
TUNDA_TAM:
        MOV   R7,#100
LOP10: MOV    R6,#100
LOP9:  MOV    R5,#100
        DJNZ  R5,$
        DJNZ  R6,LOP9
        DJNZ  R7,LOP10
        RET

INIT_INT:
        MOV LOKASI,#0
        MOV   MY_COUNTER,#0
        SETB  IT0
        SETB  IT1
        MOV IE,#94H
        RET

init_serial:
        MOV   SCON,#50H
        MOV   TMOD,#20H
        MOV   TL1,#0FDH
```

```
        MOV   TH1,#0FDH
        SETB  TR1
        SETB  EA              ; enable interupt
        SETB  ES              ; enable serial interupt
        RET


INISIALISASI:

;**************************
;* INISIALISASI LCD    *
;**************************
DELAY_INIT_LCD:
        MOV R6,#20H

DLY_LCD_LP:

        MOV   R7,#0
        DJNZ  R7,$
        DJNZ  R6,DLY_LCD_LP
        RET

INIT_LCD:
        SETB RS

        CLR E
        MOV A,#DISPCLR
        LCALL CONTROLOUT
        LCALL DELAY_INIT_LCD
        MOV   A,#FUNCSET
        LCALL  CONTROLOUT
        MOV   A,#DISPON
        LCALL  CONTROLOUT
        MOV   A,#ENTRMOD
        LCALL  CONTROLOUT
        ;
        MOV   DPTR,#NAMA
        LCALL PRINTSTRING1
        MOV   DPTR,#NIM
        LCALL PRINTSTRING2
        CALL  TUNDA_TAM
        MOV   DPTR,#UNIV
        LCALL PRINTSTRING1
        MOV   DPTR,#UNIV1
        LCALL PRINTSTRING2
        CALL  TUNDA_TAM


DATA_MATI:
        MOV   LOKASI,#0
        MOV   MY_TEMP,#0
        MOV   MY_COUNTER,#0

        ACALL TUNDA
        MOV   DPTR,#PRO
        LCALL PRINTSTRING1
        MOV   DPTR,#PRO1
        LCALL PRINTSTRING2

        JMP   DATA_MATI
```

```
;*********************************
;* KUMPULAN RUTIN PELAYANAN LCD
*
;*********************************

POSISI2_1:
        MOV   A,#1
POSISI2:
        ADD   A,#11000000B
        SJMP  POSISI_SUB

POSISI1_1:
        MOV   A,#1
POSISI1:
        ADD   A,#10000000B
POSISI_SUB:
        DEC   A
        LCALL CONTROLOUT
        RET

PRINTSTRING2:
        LCALL POSISI2_1
        SJMP  PRINTSTRING

PRINTSTRING1:
        LCALL POSISI1_1

PRINTSTRING:
        SJMP  OUTSTRING
PRINTSTRINGLOOP:
        LCALL DATAOUT
        INC   DPTR

OUTSTRING:
        CLR   A
        MOVC  A,@A+DPTR
        JNZ   PRINTSTRINGLOOP
        RET

CONTROLOUT:
        CPL RS

        CPL E
        MOV P0,A
        CPL E

        CPL RS
        MOV P0,#0FFH
        SJMP LCD_OUT

DATAOUT:

        CPL E
        MOV P0,A
        CPL E


LCD_OUT:
        MOVX  @DPTR,A

DELAY_LCD:
        PUSH ACC
        MOV   A,#250
        DJNZ  ACC,$
        POP ACC
        RET
;

NAMA:    DB ' RIAN HIDAYAT  ',0
NIM:     DB ' ELEKTRONIKA S1 ',0
UNIV:    DB '    ITN     ',0
UNIV1:   DB '    MALANG    ',0

NO_MAHA: DB 'NO TIDAK TERDAF ',0

PRO:     DB '            ',0
PRO1:    DB 'N:  g Rp:     ',0


TUNDA:
        MOV   R7,#100
OOPA:   MOV   R6,#100
OOPB:   MOV   R5,#100
        DJNZ  R5,$
        DJNZ  R6,OOPB
        DJNZ  R7,OOPA
        RET

        END
```

DELPHI.Rian

```
unit F1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls,Unit_Serial, Buttons, Menus,Registry,Unit_Com,
  MMSystem, Db, DBTables,Datcontrol, ImgList, Mask, DBCtrls;


Const
 Com1=1; com2=2;com3=3;com4=4;

Procedure OutPort(addr:word;data:byte);
function InPort(addr:word):byte;


type
  TForm1 = class(TForm)
    BitBtn1: TBitBtn;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Option1: TMenuItem;
    Comunication1: TMenuItem;
    Abaut1: TMenuItem;
    Button1: TButton;
    Button2: TButton;
    Timer2: TTimer;
    Timer1: TTimer;
    Panel3: TPanel;
    Edit10: TEdit;
    Timer3: TTimer;
    Timer4: TTimer;
    Timer5: TTimer;
    Timer6: TTimer;
    BitBtn2: TBitBtn;
    Panel1: TPanel;
    BitBtn3: TBitBtn;
    Button3: TButton;
    Image1: TImage;
    Table1: TTable;
    DataSource1: TDataSource;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBNavigator1: TDBNavigator;
    DBEdit6: TDBEdit;
    Table1Nomer: TFloatField;
    Table1NamaBarang: TStringField;
    Table1KodeBarang: TStringField;
    Table1Netto: TFloatField;
    Table1Jumlah: TFloatField;
    Table1Harga: TFloatField;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Edit1: TEdit;
    Button4: TButton;
    Timer7: TTimer;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
```

```
    Panel2: TPanel;
    Edit11: TEdit;
    Edit12: TEdit;
    Edit13: TEdit;
    Edit14: TEdit;
    Edit15: TEdit;
    Edit16: TEdit;
    Edit17: TEdit;
    Edit18: TEdit;
    Button5: TButton;
    Button6: TButton;
    Edit19: TEdit;
    Edit20: TEdit;
    Edit21: TEdit;
    Edit22: TEdit;
    Edit23: TEdit;
    Edit24: TEdit;
    Edit25: TEdit;
    Edit26: TEdit;
    Edit27: TEdit;
    Edit28: TEdit;
    Edit29: TEdit;
    Edit30: TEdit;
    Edit31: TEdit;
    Edit32: TEdit;
    Edit33: TEdit;
    Edit34: TEdit;
    Edit35: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Comunication1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
    procedure FormPaint(Sender: TObject);

  private
    uses Unit4 ;

{$R *.DFM}
Procedure OutPort(addr:word;data:byte);
begin
 asm
  mov   dx,addr
  mov   al,data
  out   dx,al
 end;
end;

function InPort(addr:word):byte;
var m:byte;
begin
 asm
  mov   dx,addr
  in    al,dx
  mov   m,al
 end;
 Inport:=m;
 END;

procedure TForm1.Delay(lama:longint);
var ref:longint;
begin
ref:=GetTickCount;
repeat
application.ProcessMessages ;
until ((gettickCount-ref) >= lama);
end;

procedure TForm1.FormCreate(Sender: TObject);
var
```

```
 Reg  : TRegistry;
 kodeBintang :string;
begin
lantaike:=1;
FORM1.Edit1.Hide;
FORM1.Edit2.Hide;
FORM1.Edit3.Hide;
FORM1.Edit4.Hide;
FORM1.Edit5.Hide;
FORM1.Edit6.Hide;
FORM1.Edit7.Hide;
FORM1.Edit8.Hide;
FORM1.Edit9.Hide;
FORM1.Edit11.Hide;
FORM1.Edit12.Hide;
FORM1.Edit13.Hide;
FORM1.Edit14.Hide;
FORM1.Edit15.Hide;
FORM1.Edit16.Hide;
FORM1.Edit17.Hide;
FORM1.Edit18.Hide;
FORM1.Edit19.Hide;
FORM1.Edit20.Hide;
FORM1.Edit21.Hide;
FORM1.Edit22.Hide;
FORM1.Edit23.Hide;
FORM1.Edit24.Hide;
FORM1.Edit25.Hide;
FORM1.Edit26.Hide;
FORM1.Edit27.Hide;
FORM1.Edit28.Hide;
FORM1.Edit29.Hide;
FORM1.Edit30.Hide;
FORM1.Edit31.Hide;
FORM1.Edit32.Hide;
FORM1.Edit33.Hide;
FORM1.Edit34.Hide;
FORM1.Edit35.Hide;
FORM1.Edit35.Hide;

{FORM1.Button4.Hide;}
FORM1.Button5.Hide;
FORM1.Button6.Hide;
FORM1.Panel2.Hide;
Timer2.Enabled :=false;
 Timer3.Enabled :=false;
  Timer4.Enabled :=false;
   Timer5.Enabled :=false;

procedure TForm1.Timer1Timer(Sender: TObject);
var dataseri : byte;
    waktu     :byte;
    x:integer;
    caridata:String;
      kodeBintang :string;
begin
Timer1.Enabled :=false;
waktu:=0;
  repeat
  inc(waktu);
  Delay(10);
  until (LineStatus(addrcom)) or (waktu= 255);
  begin
  if (LineStatus(addrcom)) then
   Begin
   dataseri:=inserial(addrcom);
   if dataseri in [1,2,3,4,5,6,7,8,9,10,ord('A')] then
    begin

    if dataseri = 1 then
     begin
      FORM1.Edit1.Text := '1';
      kodeBintang := FORM1.Edit1.Text;
      if kodeBintang <>'' then
```

```
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 end
else
if dataseri = 2 then
Begin
 FORM1.Edit1.Text := '2' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
else
if dataseri = 3 then
Begin
 FORM1.Edit1.Text := '3' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
else
if dataseri = 4 then
Begin
 FORM1.Edit1.Text := '4' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
 else
if dataseri = 5 then
Begin
 FORM1.Edit1.Text := '5' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
 else
if dataseri = 6 then
Begin
 FORM1.Edit1.Text := '6' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
 else
if dataseri = 7 then
Begin
 FORM1.Edit1.Text := '7' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
 else
if dataseri = 8 then
Begin
 FORM1.Edit1.Text := '8' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
 END
 else
if dataseri = 0 then
 Begin
 FORM1.Edit1.Text := '10' ;
 kodeBintang := FORM1.Edit1.Text;
 if kodeBintang <>'' then
 table1.FindNearest([kodeBintang]);
 Timer3.Enabled :=true;
```

```
     END
    else
    if dataseri = 9 then
    Begin
     FORM1.Edit1.Text := '9' ;
     kodeBintang := FORM1.Edit1.Text;
    if kodeBintang <>'' then
     table1.FindNearest([kodeBintang]);
     Timer3.Enabled :=true;
    END ;
    END

END;
    timer1.Enabled :=true;
end;
END;

procedure TForm1.Comunication1Click(Sender: TObject);
var x:integer;
begin
x:=form2.ShowModal;
if x = mrOk then
 begin
 addrcom:=form2.RadioGroup1.ItemIndex + 1;
 SetupSerial(addrcom);
 end;
Timer1.Enabled:=True;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
application.Terminate ;
end;

procedure TForm1.Button2Click(Sender: TObject);

begin
form3.showmodal;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin

  Timer3.Enabled :=true;
 end;

procedure TForm1.Timer2Timer(Sender: TObject);
var
weleh,weleh1,weleh2,weleh3,weleh4,weleh5,welehh: STRING;
weleh6,weleh7,weleh8,weleh9,weleh10,weleh11,weleh12: STRING;
weleh13,weleh14,weleh15,weleh16,weleh17,weleh18,weleh19: STRING;
weleh20,weleh21,weleh22,weleh23,weleh24,weleh25,weleh26: STRING;

    DATASERI : byte;
     waktu     :byte;
 Jam,Menit,Detik,MiliDetik : Word;
 Jam1,menit1,detik1,msec1:word;
  Totalwaktu        :TDateTime;
begin
weleh1 := FORM1.Edit4.Text;
weleh2 := FORM1.Edit5.Text;
weleh3 := FORM1.Edit6.Text;
weleh4 := FORM1.Edit7.Text;
weleh5 := FORM1.Edit8.Text;
weleh6 := FORM1.Edit9.Text;
weleh7 := FORM1.Edit11.Text;
weleh8 := FORM1.Edit12.Text;
weleh9 := FORM1.Edit13.Text;
weleh10 := FORM1.Edit14.Text;
weleh11 := FORM1.Edit15.Text;
weleh12 := FORM1.Edit16.Text;
weleh13 := FORM1.Edit17.Text;
weleh14 := FORM1.Edit18.Text;
weleh15 := FORM1.Edit19.Text;
```

```
weleh16 := FORM1.Edit20.Text;
weleh17 := FORM1.Edit21.Text;
weleh18 := FORM1.Edit22.Text;
weleh19 := FORM1.Edit23.Text;
weleh20 := FORM1.Edit24.Text;
weleh21 := FORM1.Edit25.Text;
weleh22 := FORM1.Edit26.Text;
weleh23 := FORM1.Edit27.Text;
weleh24 := FORM1.Edit28.Text;
weleh25 := FORM1.Edit29.Text;
weleh26 := FORM1.Edit30.Text;

 DecodeTime(Time, Jam , Menit, Detik, MiliDetik);
 inc (datajam);

 Panel2.Caption := IntToStr(Jam) +':'+
                   IntToStr(Menit)+':'+
                   IntToStr(Detik);
  inc   (perwaktu);

 form1.Edit31.Text := IntToStr(perwaktu);;
 if form1.Edit31.Text = form1.Edit32.Text then
 begin
 perwaktu := 0;
 Timer2.Enabled :=false;
 Timer3.Enabled :=false;
 END
 ELSE
 if PERWAKTU = 1 then
 BEGIN
  IF WELEH1 = 'A' THEN
  BEGIN
  Outserial(addrcom,ORD ('A'));
  END

  ELSE
  IF WELEH1 = 'B' THEN
  BEGIN
  Outserial(addrcom,ORD ('B'));
  END

  ELSE
  IF WELEH1 = 'C' THEN
  BEGIN
  Outserial(addrcom,ORD ('C'));
  END

  ELSE
  IF WELEH1 = 'D' THEN
  BEGIN
  Outserial(addrcom,ORD ('D'));
  END

  ELSE
  IF WELEH1 = 'E' THEN
  BEGIN
  Outserial(addrcom,ORD ('E'));
  END

  ELSE
  IF WELEH1 = 'F' THEN
  BEGIN
  Outserial(addrcom,ORD ('F'));
  END

  ELSE
  IF WELEH1 = 'G' THEN
  BEGIN
  Outserial(addrcom,ORD ('G'));
  END

  ELSE
  IF WELEH1 = 'H' THEN
  BEGIN
```

```
Outserial(addrcom,ORD ('H'));
END

ELSE
IF WELEH1 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END

ELSE
IF WELEH1 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END

ELSE
IF WELEH1 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END

ELSE
IF WELEH1 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END

ELSE
IF WELEH1 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END

ELSE
IF WELEH1 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END

ELSE
IF WELEH1 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END

ELSE
IF WELEH1 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END

ELSE
IF WELEH1 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END

ELSE
IF WELEH1 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END

ELSE
IF WELEH1 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END

ELSE
IF WELEH1 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
```

```
ELSE
IF WELEH1 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END

ELSE
IF WELEH1 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END

ELSE
IF WELEH1 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END

ELSE
IF WELEH1 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END

ELSE
IF WELEH1 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END

ELSE
IF WELEH1 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END

ELSE
IF WELEH1 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END

ELSE
IF WELEH1 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END

ELSE
IF WELEH1 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END

ELSE
IF WELEH1 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END

ELSE
IF WELEH1 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END

ELSE
IF WELEH1 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END

ELSE
```

```
 IF WELEH1 = '7' THEN
 BEGIN
 Outserial(addrcom,ORD ('7'));
 END

 ELSE
 IF WELEH1 = '8' THEN
 BEGIN
 Outserial(addrcom,ORD ('8'));
 END

 ELSE
 IF WELEH1 = '9' THEN
 BEGIN
 Outserial(addrcom,ORD ('9'));
 END

 ELSE
 IF WELEH1 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH10 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END

END
ELSE
if PERWAKTU = 2 then
BEGIN
 IF WELEH2 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH2 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH2 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH2 = 'D' THEN
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH2 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
 IF WELEH2 = 'F' THEN
 BEGIN
 Outserial(addrcom,ORD ('F'));
 END
 ELSE
 IF WELEH2 = 'G' THEN
 BEGIN
 Outserial(addrcom,ORD ('G'));
 END
 ELSE
 IF WELEH2 = 'H' THEN
 BEGIN
 Outserial(addrcom,ORD ('H'));
 END
 ELSE
 IF WELEH2 = 'I' THEN
 BEGIN
```

```
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH2 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH2 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH2 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH2 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH2 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH2 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH2 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH2 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH2 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH2 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH2 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH2 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH2 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH2 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH2 = 'X' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH2 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH2 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH2 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH2 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH2 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH2 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH2 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH2 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH2 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH2 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH2 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH2 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH2 = ' ' THEN
BEGIN
Outserial(addrcom,ORD (' '));
END;

END
ELSE
if PERWAKTU = 3 then
BEGIN
 IF WELEH3 = 'A' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('A'));
END
ELSE
IF WELEH3 = 'B' THEN
BEGIN
Outserial(addrcom,ORD ('B'));
END
ELSE
IF WELEH3 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH2 = 'D' THEN
BEGIN
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH3 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH3 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH3 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH3 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH3 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH3 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH3 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH3 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH3 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH3 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH3 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
```

```
IF WELEH3 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH3 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH3 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH3 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH3 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH3 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH3 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH3 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH3 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH3 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH3 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH3 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH3 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH3 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH3 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
```

```
 ELSE
 IF WELEH3 = '5' THEN
 BEGIN
 Outserial(addrcom,ORD ('5'));
 END
  ELSE
 IF WELEH3 = '6' THEN
 BEGIN
 Outserial(addrcom,ORD ('6'));
 END
  ELSE
 IF WELEH3 = '7' THEN
 BEGIN
 Outserial(addrcom,ORD ('7'));
 END
  ELSE
 IF WELEH3 = '8' THEN
 BEGIN
 Outserial(addrcom,ORD ('8'));
 END
 ELSE
  IF WELEH3 = '9' THEN
 BEGIN
 Outserial(addrcom,ORD ('9'));
 END
 ELSE
 IF WELEH3 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH3 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;

 END
 ELSE

if PERWAKTU = 4 then
BEGIN
 IF WELEH4 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH4 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH4 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH4 = 'D' THEN
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH4 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
 IF WELEH4 = 'F' THEN
 BEGIN
 Outserial(addrcom,ORD ('F'));
 END
 ELSE
 IF WELEH4 = 'G' THEN
 BEGIN
 Outserial(addrcom,ORD ('G'));
```

```
END
ELSE
IF WELEH4 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH4 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH4 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH4 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH4 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH4 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH4 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH4 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH4 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH4 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH4 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH4 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH4 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH4 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH4 = 'V' THEN
BEGIN
```

```
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH4 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH4 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH4 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH4 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH4 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH4 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH4 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH4 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH4 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH4 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH4 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH4 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH4 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH4 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH4 = ' ' THEN
```

```
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;

 END
 ELSE

 if PERWAKTU = 5 then
BEGIN
 IF WELEH5 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH5 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH5 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH1 = 'D' then
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH5 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
 IF WELEH5 = 'F' THEN
 BEGIN
 Outserial(addrcom,ORD ('F'));
 END
 ELSE
 IF WELEH5 = 'G' THEN
 BEGIN
 Outserial(addrcom,ORD ('G'));
 END
 ELSE
 IF WELEH5 = 'H' THEN
 BEGIN
 Outserial(addrcom,ORD ('H'));
 END
 ELSE
 IF WELEH5 = 'I' THEN
 BEGIN
 Outserial(addrcom,ORD ('I'));
 END
 ELSE
 IF WELEH5 = 'J' THEN
 BEGIN
 Outserial(addrcom,ORD ('J'));
 END
 ELSE
 IF WELEH5 = 'K' THEN
 BEGIN
 Outserial(addrcom,ORD ('K'));
 END
 ELSE
 IF WELEH5 = 'L' THEN
 BEGIN
 Outserial(addrcom,ORD ('L'));
 END
 ELSE
 IF WELEH5 = 'M' THEN
 BEGIN
 Outserial(addrcom,ORD ('M'));
 END
 ELSE
```

```
IF WELEH5 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH5 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH5 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH5 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH5 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH5 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH5 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH5 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH5 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH5 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH5 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH5 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH5 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH5 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH5 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
```

```
 ELSE
 IF WELEH5 = '3' THEN
 BEGIN
 Outserial(addrcom,ORD ('3'));
 END
  ELSE
 IF WELEH5 = '4' THEN
 BEGIN
 Outserial(addrcom,ORD ('4'));
 END
 ELSE
 IF WELEH5 = '5' THEN
 BEGIN
 Outserial(addrcom,ORD ('5'));
 END
  ELSE
 IF WELEH5 = '6' THEN
 BEGIN
 Outserial(addrcom,ORD ('6'));
 END
  ELSE
 IF WELEH5 = '7' THEN
 BEGIN
 Outserial(addrcom,ORD ('7'));
 END
  ELSE
 IF WELEH5 = '8' THEN
 BEGIN
 Outserial(addrcom,ORD ('8'));
 END
 ELSE
  IF WELEH5 = '9' THEN
 BEGIN
 Outserial(addrcom,ORD ('9'));
 END
 ELSE
 IF WELEH5 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH5 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;
 END
 ELSE
if PERWAKTU = 6 then
 BEGIN
 IF WELEH6 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH6 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH6 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH6 = 'D' THEN
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH6 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
```

```
IF WELEH6 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH6 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH6 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH6 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH6 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH6 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH6 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH6 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH6 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH6 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH6 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH6 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH6 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH6 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH6 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
```

```
ELSE
IF WELEH6 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH6 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH6 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH6 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH6 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH6 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH6 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH6 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH6 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH6 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH6 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH6 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH6 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH6 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH6 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
```

```
 END
 ELSE
 IF WELEH6 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH6 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;

 END
 ELSE
if PERWAKTU = 7 then
 BEGIN
 IF WELEH7 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH7 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH7 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH7 = 'D' THEN
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH7 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
 IF WELEH7 = 'F' THEN
 BEGIN
 Outserial(addrcom,ORD ('F'));
 END
 ELSE
 IF WELEH7 = 'G' THEN
 BEGIN
 Outserial(addrcom,ORD ('G'));
 END
 ELSE
 IF WELEH7 = 'H' THEN
 BEGIN
 Outserial(addrcom,ORD ('H'));
 END
 ELSE
 IF WELEH7 = 'I' THEN
 BEGIN
 Outserial(addrcom,ORD ('I'));
 END
 ELSE
 IF WELEH7 = 'J' THEN
 BEGIN
 Outserial(addrcom,ORD ('J'));
 END
 ELSE
 IF WELEH7 = 'K' THEN
 BEGIN
 Outserial(addrcom,ORD ('K'));
 END
 ELSE
 IF WELEH7 = 'L' THEN
 BEGIN
 Outserial(addrcom,ORD ('L'));
```

```
END
ELSE
IF WELEH7 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH7 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH7 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH7 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH7 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH7 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH7 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH7 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH7 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH7 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH7 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH7 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH7 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH7 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH7 = 'I' THEN
BEGIN
```

```
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH7 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH7 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH7 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH7 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH7 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH7 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH7 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH7 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH7 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH7 = ' ' THEN
BEGIN
Outserial(addrcom,ORD (' '));
END;

END
ELSE
if PERWAKTU = 8 then
 BEGIN
IF WELEH8 = 'A' THEN
BEGIN
Outserial(addrcom,ORD ('A'));
END
ELSE
IF WELEH8 = 'B' THEN
BEGIN
Outserial(addrcom,ORD ('B'));
END
ELSE
IF WELEH8 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH8 = 'D' THEN
BEGIN
```

```
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH8 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH8 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH8 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH8 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH8 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH8 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH8 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH8 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH8 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH8 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH8 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH8 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH8 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH8 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH8 = 'S' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH8 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH8 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH8 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH8 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH8 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH8 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH8 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH8 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH8 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH8 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH8 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH8 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH8 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH8 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
```

```
 IF WELEH8 = '8' THEN
 BEGIN
 Outserial(addrcom,ORD ('8'));
 END
 ELSE
  IF WELEH8 = '9' THEN
 BEGIN
 Outserial(addrcom,ORD ('9'));
 END
 ELSE
 IF WELEH8 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH8 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;

 END
 ELSE
if PERWAKTU = 9 then
 BEGIN
 IF WELEH9 = 'A' THEN
 BEGIN
 Outserial(addrcom,ORD ('A'));
 END
 ELSE
 IF WELEH9 = 'B' THEN
 BEGIN
 Outserial(addrcom,ORD ('B'));
 END
 ELSE
 IF WELEH9 = 'C' THEN
 BEGIN
 Outserial(addrcom,ORD ('C'));
 END
 ELSE
 IF WELEH9 = 'D' THEN
 BEGIN
 Outserial(addrcom,ORD ('D'));
 END
 ELSE
 IF WELEH9 = 'E' THEN
 BEGIN
 Outserial(addrcom,ORD ('E'));
 END
 ELSE
 IF WELEH9 = 'F' THEN
 BEGIN
 Outserial(addrcom,ORD ('F'));
 END
 ELSE
 IF WELEH9 = 'G' THEN
 BEGIN
 Outserial(addrcom,ORD ('G'));
 END
 ELSE
 IF WELEH9 = 'H' THEN
 BEGIN
 Outserial(addrcom,ORD ('H'));
 END
 ELSE
 IF WELEH9 = 'I' THEN
 BEGIN
 Outserial(addrcom,ORD ('I'));
 END
 ELSE
 IF WELEH9 = 'J' THEN
 BEGIN
 Outserial(addrcom,ORD ('J'));
 END
 ELSE
```

```
IF WELEH9 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH9 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH9 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH9 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH9 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH9 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH9 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH9 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH9 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH9 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH9 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH9 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH9 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH9 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH9 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
```

```
  ELSE
  IF WELEH9 = 'Z' THEN
  BEGIN
  Outserial(addrcom,ORD ('Z'));
  END
  ELSE
  IF WELEH9 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH9 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH9 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH9 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH9 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH9 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH9 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH9 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH9 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH9 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH9 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 10 then
  BEGIN
  IF WELEH10 = 'A' THEN
  BEGIN
  Outserial(addrcom,ORD ('A'));
  END
  ELSE
  IF WELEH10 = 'B' THEN
  BEGIN
  Outserial(addrcom,ORD ('B'));
  END
```

```
ELSE
IF WELEH10 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH10 = 'D' THEN
BEGIN
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH10 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH10 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH10 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH10 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH10 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH10 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH10 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH10 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH10 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH10 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH10 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH10 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH10 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
```

```
END
ELSE
IF WELEH10 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH10 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH10 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH10 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH10 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH10 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH10 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH10 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH10 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH10 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH10 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH10 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH10 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH10 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH10 = '6' THEN
BEGIN
```

```
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH10 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH10 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH10 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH10 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH10 = ' ' THEN
BEGIN
Outserial(addrcom,ORD (' '));
END;

END
ELSE
if PERWAKTU = 11 then
BEGIN
IF WELEH11 = 'A' THEN
BEGIN
Outserial(addrcom,ORD ('A'));
END
ELSE
IF WELEH11 = 'B' THEN
BEGIN
Outserial(addrcom,ORD ('B'));
END
ELSE
IF WELEH11 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH11 = 'D' THEN
BEGIN
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH11 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH11 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH11 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH11 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH11 = 'I' THEN
BEGIN
```

```
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH11 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH11 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH11 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH11 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH11 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH11 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH11 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH11 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH11 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH11 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH11 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH11 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH11 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH11 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH11 = 'X' THEN
```

```
  BEGIN
  Outserial(addrcom,ORD ('X'));
  END
  ELSE
  IF WELEH11 = 'Y' THEN
  BEGIN
  Outserial(addrcom,ORD ('Y'));
  END
  ELSE
  IF WELEH11 = 'Z' THEN
  BEGIN
  Outserial(addrcom,ORD ('Z'));
  END
  ELSE
  IF WELEH11 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH11 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH11 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH11 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH11 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH11 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH11 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH11 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH11 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH11 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH11 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 12 then
  BEGIN
  IF WELEH12 = 'A' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('A'));
END
ELSE
IF WELEH12 = 'B' THEN
BEGIN
Outserial(addrcom,ORD ('B'));
END
ELSE
IF WELEH12 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH12 = 'D' THEN
BEGIN
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH12 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH12 = 'F' THEN
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH12 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH12 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH12 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH12 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH12 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH12 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH12 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH12 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH12 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
```

```
IF WELEH12 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH12 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH12 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH12 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH12 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH12 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH12 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH12 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH12 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH12 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH12 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH12 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH12 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH12 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH12 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
```

```
  ELSE
  IF WELEH12 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH12 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH12 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH12 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH12 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH12 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH12 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 13 then
 BEGIN
  IF WELEH13 = 'A' THEN
  BEGIN
  Outserial(addrcom,ORD ('A'));
  END
  ELSE
  IF WELEH13 = 'B' THEN
  BEGIN
  Outserial(addrcom,ORD ('B'));
  END
  ELSE
  IF WELEH13 = 'C' THEN
  BEGIN
  Outserial(addrcom,ORD ('C'));
  END
  ELSE
  IF WELEH13 = 'D' THEN
  BEGIN
  Outserial(addrcom,ORD ('D'));
  END
  ELSE
  IF WELEH13 = 'E' THEN
  BEGIN
  Outserial(addrcom,ORD ('E'));
  END
  ELSE
  IF WELEH13 = 'F' THEN
  BEGIN
  Outserial(addrcom,ORD ('F'));
  END
  ELSE
  IF WELEH13 = 'G' THEN
  BEGIN
  Outserial(addrcom,ORD ('G'));
  END
```

```
ELSE
IF WELEH13 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH13 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH13 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH13 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH13 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH13 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH13 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH13 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH13 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH13 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH13 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH13 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH13 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH13 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH13 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
```

```
END
ELSE
IF WELEH13 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH13 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH13 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH13 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH13 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH13 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH13 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH13 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH13 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH13 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH13 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH13 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH13 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH13 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH13 = ' ' THEN
BEGIN
```

```
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 14 then
  BEGIN
  IF WELEH14 = 'A' THEN
  BEGIN
  Outserial(addrcom,ORD ('A'));
  END
  ELSE
  IF WELEH14 = 'B' THEN
  BEGIN
  Outserial(addrcom,ORD ('B'));
  END
  ELSE
  IF WELEH14 = 'C' THEN
  BEGIN
  Outserial(addrcom,ORD ('C'));
  END
  ELSE
  IF WELEH14 = 'D' THEN
  BEGIN
  Outserial(addrcom,ORD ('D'));
  END
  ELSE
  IF WELEH14 = 'E' THEN
  BEGIN
  Outserial(addrcom,ORD ('E'));
  END
  ELSE
  IF WELEH14 = 'F' THEN
  BEGIN
  Outserial(addrcom,ORD ('F'));
  END
  ELSE
  IF WELEH14 = 'G' THEN
  BEGIN
  Outserial(addrcom,ORD ('G'));
  END
  ELSE
  IF WELEH14 = 'H' THEN
  BEGIN
  Outserial(addrcom,ORD ('H'));
  END
  ELSE
  IF WELEH14 = 'I' THEN
  BEGIN
  Outserial(addrcom,ORD ('I'));
  END
  ELSE
  IF WELEH14 = 'J' THEN
  BEGIN
  Outserial(addrcom,ORD ('J'));
  END
  ELSE
  IF WELEH14 = 'K' THEN
  BEGIN
  Outserial(addrcom,ORD ('K'));
  END
  ELSE
  IF WELEH14 = 'L' THEN
  BEGIN
  Outserial(addrcom,ORD ('L'));
  END
  ELSE
  IF WELEH14 = 'M' THEN
  BEGIN
  Outserial(addrcom,ORD ('M'));
  END
  ELSE
  IF WELEH14 = 'N' THEN
  BEGIN
```

```
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH14 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH14 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH14 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH14 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH14 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH14 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH14 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH14 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH14 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH14 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH14 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH14 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH14 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH14 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH14 = '3' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH14 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH14 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH14 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH14 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH14 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH14 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH14 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH14 = ' ' THEN
BEGIN
Outserial(addrcom,ORD (' '));
END;

END
ELSE
if PERWAKTU = 15 then
 BEGIN
IF WELEH15 = 'A' THEN
BEGIN
Outserial(addrcom,ORD ('A'));
END
ELSE
IF WELEH15 = 'B' THEN
BEGIN
Outserial(addrcom,ORD ('B'));
END
ELSE
IF WELEH15 = 'C' THEN
BEGIN
Outserial(addrcom,ORD ('C'));
END
ELSE
IF WELEH15 = 'D' THEN
BEGIN
Outserial(addrcom,ORD ('D'));
END
ELSE
IF WELEH15 = 'E' THEN
BEGIN
Outserial(addrcom,ORD ('E'));
END
ELSE
IF WELEH15 = 'F' THEN
```

```
BEGIN
Outserial(addrcom,ORD ('F'));
END
ELSE
IF WELEH15 = 'G' THEN
BEGIN
Outserial(addrcom,ORD ('G'));
END
ELSE
IF WELEH15 = 'H' THEN
BEGIN
Outserial(addrcom,ORD ('H'));
END
ELSE
IF WELEH15 = 'I' THEN
BEGIN
Outserial(addrcom,ORD ('I'));
END
ELSE
IF WELEH15 = 'J' THEN
BEGIN
Outserial(addrcom,ORD ('J'));
END
ELSE
IF WELEH15 = 'K' THEN
BEGIN
Outserial(addrcom,ORD ('K'));
END
ELSE
IF WELEH15 = 'L' THEN
BEGIN
Outserial(addrcom,ORD ('L'));
END
ELSE
IF WELEH15 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH15 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH15 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH15 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH15 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH15 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH15 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH15 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
```

```
IF WELEH15 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH15 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH15 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH15 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH15 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH15 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH15 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH15 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH15 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH15 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH15 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH15 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH15 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH15 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH15 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
```

```
  ELSE
  IF WELEH15 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH15 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 16 then
 BEGIN
  IF WELEH16 = 'A' THEN
  BEGIN
  Outserial(addrcom,ORD ('A'));
  END
  ELSE
  IF WELEH16 = 'B' THEN
  BEGIN
  Outserial(addrcom,ORD ('B'));
  END
  ELSE
  IF WELEH16 = 'C' THEN
  BEGIN
  Outserial(addrcom,ORD ('C'));
  END
  ELSE
  IF WELEH16 = 'D' THEN
  BEGIN
  Outserial(addrcom,ORD ('D'));
  END
  ELSE
  IF WELEH16 = 'E' THEN
  BEGIN
  Outserial(addrcom,ORD ('E'));
  END
  ELSE
  IF WELEH16 = 'F' THEN
  BEGIN
  Outserial(addrcom,ORD ('F'));
  END
  ELSE
  IF WELEH16 = 'G' THEN
  BEGIN
  Outserial(addrcom,ORD ('G'));
  END
  ELSE
  IF WELEH16 = 'H' THEN
  BEGIN
  Outserial(addrcom,ORD ('H'));
  END
  ELSE
  IF WELEH16 = 'I' THEN
  BEGIN
  Outserial(addrcom,ORD ('I'));
  END
  ELSE
  IF WELEH16 = 'J' THEN
  BEGIN
  Outserial(addrcom,ORD ('J'));
  END
  ELSE
  IF WELEH16 = 'K' THEN
  BEGIN
  Outserial(addrcom,ORD ('K'));
  END
  ELSE
  IF WELEH16 = 'L' THEN
  BEGIN
  Outserial(addrcom,ORD ('L'));
  END
```

```
ELSE
IF WELEH16 = 'M' THEN
BEGIN
Outserial(addrcom,ORD ('M'));
END
ELSE
IF WELEH16 = 'N' THEN
BEGIN
Outserial(addrcom,ORD ('N'));
END
ELSE
IF WELEH16 = 'O' THEN
BEGIN
Outserial(addrcom,ORD ('O'));
END
ELSE
IF WELEH16 = 'P' THEN
BEGIN
Outserial(addrcom,ORD ('P'));
END
ELSE
IF WELEH16 = 'Q' THEN
BEGIN
Outserial(addrcom,ORD ('Q'));
END
ELSE
IF WELEH16 = 'R' THEN
BEGIN
Outserial(addrcom,ORD ('R'));
END
ELSE
IF WELEH16 = 'S' THEN
BEGIN
Outserial(addrcom,ORD ('S'));
END
ELSE
IF WELEH16 = 'T' THEN
BEGIN
Outserial(addrcom,ORD ('T'));
END
ELSE
IF WELEH16 = 'U' THEN
BEGIN
Outserial(addrcom,ORD ('U'));
END
ELSE
IF WELEH16 = 'V' THEN
BEGIN
Outserial(addrcom,ORD ('V'));
END
ELSE
IF WELEH16 = 'W' THEN
BEGIN
Outserial(addrcom,ORD ('W'));
END
ELSE
IF WELEH16 = 'X' THEN
BEGIN
Outserial(addrcom,ORD ('X'));
END
ELSE
IF WELEH16 = 'Y' THEN
BEGIN
Outserial(addrcom,ORD ('Y'));
END
ELSE
IF WELEH16 = 'Z' THEN
BEGIN
Outserial(addrcom,ORD ('Z'));
END
ELSE
IF WELEH16 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
```

```
 END
 ELSE
 IF WELEH16 = '2' THEN
 BEGIN
 Outserial(addrcom,ORD ('2'));
 END
 ELSE
 IF WELEH16 = '3' THEN
 BEGIN
 Outserial(addrcom,ORD ('3'));
 END
  ELSE
 IF WELEH16 = '4' THEN
 BEGIN
 Outserial(addrcom,ORD ('4'));
 END
 ELSE
 IF WELEH16 = '5' THEN
 BEGIN
 Outserial(addrcom,ORD ('5'));
 END
  ELSE
 IF WELEH16 = '6' THEN
 BEGIN
 Outserial(addrcom,ORD ('6'));
 END
  ELSE
 IF WELEH16 = '7' THEN
 BEGIN
 Outserial(addrcom,ORD ('7'));
 END
  ELSE
 IF WELEH16 = '8' THEN
 BEGIN
 Outserial(addrcom,ORD ('8'));
 END
 ELSE
  IF WELEH16 = '9' THEN
 BEGIN
 Outserial(addrcom,ORD ('9'));
 END
 ELSE
 IF WELEH16 = '0' THEN
 BEGIN
 Outserial(addrcom,ORD ('0'));
 END
 ELSE
 IF WELEH16 = ' ' THEN
 BEGIN
 Outserial(addrcom,ORD (' '));
 END;

 END
 ELSE
if PERWAKTU = 17 then
 BEGIN
 IF WELEH17 = '1' THEN
 BEGIN
 Outserial(addrcom,ORD ('1'));
 END
 ELSE
 IF WELEH17 = '2' THEN
 BEGIN
 Outserial(addrcom,ORD ('2'));
 END
 ELSE
 IF WELEH17 = '3' THEN
 BEGIN
 Outserial(addrcom,ORD ('3'));
 END
  ELSE
 IF WELEH17 = '4' THEN
 BEGIN
 Outserial(addrcom,ORD ('4'));
```

```
  END
  ELSE
  IF WELEH17 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH17 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH17 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH17 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH17 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH17 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH17 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 18 then
 BEGIN
  IF WELEH18 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH18 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH18 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH18 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH18 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH18 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH18 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
```

```
END
 ELSE
IF WELEH18 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH18 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH18 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
ELSE
IF WELEH18 = '-' THEN
BEGIN
Outserial(addrcom,ORD (' '));
END;
END
ELSE
if PERWAKTU = 19 then
BEGIN
IF WELEH19 = '1' THEN
BEGIN
Outserial(addrcom,ORD ('1'));
END
ELSE
IF WELEH19 = '2' THEN
BEGIN
Outserial(addrcom,ORD ('2'));
END
ELSE
IF WELEH19 = '3' THEN
BEGIN
Outserial(addrcom,ORD ('3'));
END
 ELSE
IF WELEH19 = '4' THEN
BEGIN
Outserial(addrcom,ORD ('4'));
END
ELSE
IF WELEH19 = '5' THEN
BEGIN
Outserial(addrcom,ORD ('5'));
END
 ELSE
IF WELEH19 = '6' THEN
BEGIN
Outserial(addrcom,ORD ('6'));
END
 ELSE
IF WELEH19 = '7' THEN
BEGIN
Outserial(addrcom,ORD ('7'));
END
 ELSE
IF WELEH19 = '8' THEN
BEGIN
Outserial(addrcom,ORD ('8'));
END
ELSE
 IF WELEH19 = '9' THEN
BEGIN
Outserial(addrcom,ORD ('9'));
END
ELSE
IF WELEH19 = '0' THEN
BEGIN
Outserial(addrcom,ORD ('0'));
END
```

```
  ELSE
  IF WELEH19 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 20 then
  BEGIN
  IF WELEH20 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH20 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH20 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH20 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH20 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH20 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH20 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH20 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH20 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH20 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH20 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 21 then
  BEGIN
  IF WELEH21 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
```

```
  IF WELEH21 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH21 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH21 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH21 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH21 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH21 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH21 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH21 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH21 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH21 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 22 then
 BEGIN
  IF WELEH22 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH22 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH22 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH22 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
```

```
  IF WELEH22 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH22 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH22 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH22 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH22 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH22 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH22 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 23 then
 BEGIN
  IF WELEH23 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH23 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH23 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH23 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH23 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH23 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH23 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
```

```
  IF WELEH23 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH23 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH23 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH23 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 24 then
 BEGIN
  IF WELEH24 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH24 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH24 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH24 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH24 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH24 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH24 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH24 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH24 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH24 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
```

```
  IF WELEH24 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 25 then
  BEGIN
  IF WELEH25 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH25 = '2' THEN
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH25 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH25 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH25 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH25 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH25 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH25 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH25 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH25 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH25 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END
  ELSE
if PERWAKTU = 26 then
  BEGIN
  IF WELEH26 = '1' THEN
  BEGIN
  Outserial(addrcom,ORD ('1'));
  END
  ELSE
  IF WELEH26 = '2' THEN
```

```
  BEGIN
  Outserial(addrcom,ORD ('2'));
  END
  ELSE
  IF WELEH26 = '3' THEN
  BEGIN
  Outserial(addrcom,ORD ('3'));
  END
   ELSE
  IF WELEH26 = '4' THEN
  BEGIN
  Outserial(addrcom,ORD ('4'));
  END
  ELSE
  IF WELEH26 = '5' THEN
  BEGIN
  Outserial(addrcom,ORD ('5'));
  END
   ELSE
  IF WELEH26 = '6' THEN
  BEGIN
  Outserial(addrcom,ORD ('6'));
  END
   ELSE
  IF WELEH26 = '7' THEN
  BEGIN
  Outserial(addrcom,ORD ('7'));
  END
   ELSE
  IF WELEH26 = '8' THEN
  BEGIN
  Outserial(addrcom,ORD ('8'));
  END
  ELSE
   IF WELEH26 = '9' THEN
  BEGIN
  Outserial(addrcom,ORD ('9'));
  END
  ELSE
  IF WELEH26 = '0' THEN
  BEGIN
  Outserial(addrcom,ORD ('0'));
  END
  ELSE
  IF WELEH26 = ' ' THEN
  BEGIN
  Outserial(addrcom,ORD (' '));
  END;

  END;
  END;




procedure TForm1.Timer3Timer(Sender: TObject);
var
    Hasil : array [1..16] of integer;
    Data,data1,data2 : string;
    k,a,b,JUMKARAK,JUMKARAK1,JUMKARAK2 : integer;
     lembaran,lembaran1,lembaran2 : string;

begin
        edit4.Text := '';
        edit5.Text := '';
        edit6.Text := '';
        edit7.Text := '';
        edit8.Text := '';
        edit9.Text := '';
        edit11.Text := '';
        edit12.Text := '';
        edit13.Text := '';
        edit14.Text := '';
```

```
      edit15.Text :=' ';
      edit16.Text :=' ';
      edit17.Text :=' ';
      edit18.Text :=' ';
      edit19.Text :=' ';
      edit20.Text :=' ';
      edit21.Text :=' ';
      edit22.Text :=' ';
      edit23.Text :=' ';
      edit24.Text :=' ';
      edit25.Text :=' ';
      edit26.Text :=' ';
      edit27.Text :=' ';
      edit28.Text :=' ';
      edit29.Text :=' ';
      edit30.Text :=' ';
FORM1.Edit33.Text := FORM1.DBEdit2.Text;
data := edit33.Text;
JUMKARAK := LENGTH (EDIT33.Text);

 Begin
   for k := 1 to JUMKARAK do
   Begin
     lembaran := Data[k];
     case k of
     1: edit4.Text :=lembaran;
     2: edit5.Text :=lembaran;
     3: edit6.Text :=lembaran;
     4: edit7.Text :=lembaran;
     5: edit8.Text :=lembaran;
     6: edit9.Text :=lembaran;
     7: edit11.Text :=lembaran;
     8: edit12.Text :=lembaran;
     9: edit13.Text :=lembaran;
     10: edit14.Text :=lembaran;
     11: edit15.Text :=lembaran;
     12: edit16.Text :=lembaran;
     13: edit17.Text :=lembaran;
     14: edit18.Text :=lembaran;
     15: edit19.Text :=lembaran;
     16: edit20.Text :=lembaran;
    End;
    end;
     FORM1.Edit34.Text := FORM1.DBEdit4.Text;
     data1 := edit34.Text;
    JUMKARAK1 := LENGTH (EDIT34.Text);
  begin
    for a := 1 to jumkarak1 do
  Begin
    lembaran1 := Data1[a];
    case a of
    1: edit21.Text :=lembaran1;
    2: edit22.Text :=lembaran1;
    3: edit23.Text :=lembaran1;
  end;
end;
FORM1.Edit35.Text := FORM1.DBEdit6.Text;
    data2 := edit35.Text;
   JUMKARAK2 := LENGTH (EDIT35.Text);
  begin
    for b := 1 to jumkarak2 do
  Begin
    lembaran2 := Data2[b];
    case b of
    1: edit24.Text :=lembaran2;
    2: edit25.Text :=lembaran2;
    3: edit26.Text :=lembaran2;
    4: edit27.Text :=lembaran2;
    5: edit28.Text :=lembaran2;
    6: edit29.Text :=lembaran2;
    7: edit30.Text :=lembaran2;
  end;
end;
 End;
```

```
end;
   Timer3.Enabled :=false;
  Timer2.Enabled :=true;
end;
end;



procedure TForm1.FormPaint(Sender: TObject);
var
row, Ht:word;
begin
Ht:=(clientheight + 255) div 256;
for row:=0 to 255 do
with canvas do begin
brush.Color:=RGB(row,110,190);
fillrect(rect(0,row * Ht, clientwidth,(row + 1)* Ht));
end;
end;
end.
```

CODE 39 SYMBOLOGY

See Also: Symbology Index

Quick Link: Background | Check Digit | Encoding | Structure | Encoding Table | Example | Full ASCI

CODE 39 BACKGROUND INFORMATION

Code 39, the first alpha-numeric symbology to be developed, is still widely used–especially in non-ret *environments. It is the standard bar code used by the United States Department of Defense, and is alsc* by the Health Industry Bar Code Council (HIBCC). Code 39 is also known as "3 of 9 Code" and "USI

A typical Code 39 barcode is:

Code 39 is a discrete, variable-length symbology. It is self-checking in that a single print defect cannc transpose one character into another valid character.

COMPUTING THE CHECKSUM DIGIT

Since Code 39 is self checking, a check digit normally isn't necessary. However, in applications that r an extremely high level of accuracy a modulo 43 checksum digit may be added.

To calculate the optional checksum digit, follow the following steps.

1. Take the value (0 through 42) of each character in the barcode. The start and stop characters a included in the checksum calculation.
2. Sum the value of each of the values of each of the characters described in step 1.
3. Divide the result from step 2 by 43.
4. The remainder from the division in step 3 is the checksum character that will be appended to t message before the stop character.

ENCODING THE SYMBOL

Once the checksum digit has been calculated we know the entire message which must be encoded in t and spaces. Continuing with our example, we will encode, from zero, the Code 39 barcode we used in example above: **HI345678** with a checksum digit of 67.

In the following text, we will discuss the encoding of the barcode by considering that the number "1" represents a "dark" or "bar" section of the barcode whereas a "0" represents a "light" or "space" sectio the barcode. Thus the numbers 1101 represents a double-wide bar (11), followed by a single-wide spa followed by a single-wide bar (1). This would be printed in the barcode as:

STRUCTURE OF A CODE 39 BARCODE

A Code 39 barcode has the following structure:

1. A start character - the asterisk (*) character.
2. Any number of characters encoded from the table below.
3. An optional checksum digit calculated as described above and encoded from the table below.
4. A stop character, which is a second asterisk character.

CODE 39 ENCODING TABLE

This table indicates how to encode each digit of a Code 39 barcode. Note that the "Width Encoding" colur is expressed as "N" for narrow and "W" for wide while the "Barcode Encoding" column represents how th barcode will actually be encoded as described above in "Encoding the Symbol."

Keep in mind that each character begins and ends with a bar, thus the "barcode encoding" always starts an ends with a "1".

CHECK VALUE	ASCII CHAR	WIDTH ENCODING	BARCODE ENCODING	CHECK VALUE	ASCII CHAR	WIDTH ENCODING	BARCODI ENCODIN
0	0	NNNWWNWNNN	101001101101	22	M	WNWNNNNWN	1101101010
1	1	WNNWNNNNW	110100101011	23	N	NNNNWNNWW	1010110100
2	2	NNWWNNNNW	101100101011	24	O	WNNNWNNWN	1101011010
3	3	WNWWNNNNN	110110010101	25	P	NNWNWNNWN	1011011010
4	4	NNNWWNNNW	101001101011	26	Q	NNNNNNWWW	1010101100
5	5	WNNWWNNNN	110100110101	27	R	WNNNNNWWN	1101010110
6	6	NNWWWNNNN	101100110101	28	S	NNWNNNWWN	1011010110
7	7	NNNWNNWNW	101001011011	29	T	NNNNWNWWN	1010110110
8	8	WNNWNNWNN	110100101101	30	U	WWNNNNNNW	1100101010
9	9	NNWWNNWNN	101100101101	31	V	NWWNNNNNW	1001101010
10	A	NNWWNNWNN	110101001011	32	W	WWWNNNNNN	1100110101
11	B	NNWNNWNNW	101101001011	33	X	NWNNWNNNW	1001011010
12	C	WNWNNWNNN	110110100101	34	Y	WWNNWNNNN	1100101101
13	D	NNNNWWNNW	101011001011	35	Z	NWWNWNNNN	1001101101
14	E	WNNNWWNNN	110101100101	36	-	NWNNNNWNW	1001010110
15	F	NNWNWWNNN	101101100101	37	.	WWNNNNWNN	1100101011
16	G	NNNNNWWNW	101010011011	38	SPACE	NWWNNNWNN	1001101011
17	H	WNNNNWWNN	110101001101	39	$	NWNWNWNNN	1001001001
18	I	NNWNNWWNN	101101001101	40	/	NWNWNNNWN	1001001010
19	J	NNNNWWWNN	101011001101	41	+	NWNNNWNWN	1001010010
20	K	WNNNNNNWW	110101010011	42	%	NNNWNWNWN	1010010010

the buffer and the system will wait for additional barcode(s). If a Code 39 barcode doesn't start with a the barcode will be appended to any previous code 39 barcodes and the entire message will be deliver the applicat.on.

In other words, if a code 39 barcode has additional barcodes to follow, it must start with a space-if the barcode is the last barcode in the message it must not start with a space.

CODE 39 ENCODING EXAMPLE

We will now code the example we used above, **TEST8052**. In this case we will not use a check digit.

1. The START character (*): **100101101101**.
2. The digit "T": enocded as **101011011001**.
3. The digit "E": enocded as **110101100101**.
4. The digit "S": enocded as **101101011001**.
5. The digit "T": enocded as **101011011001**.
6. The digit "8": enocded as **110100101101**.
7. The digit "0": enocded as **101001101101**.
8. The digit "5": enocded as **110100110101**.
9. The digit "2": enocded as **101100101011**.
10. The STOP character (*): **100101101101**.

This is shown in the following graphical representation where the barcode has been sectioned-off into that reflect each of the 10 components just mentioned.

NOTE: In the above encoding example note that there is an inter-character space between eac character. This is not listed in the list of 10 components, but there is an inter-character space b each character. This inter-character space is represented in the graphic by the white space sepa the grey areas.

EXTENDED CODE 39 ENCODING TABLE

It is possible, using Code 39's "Full ASCII Mode" to encode all 128 ASCII characters. This is accomp by using the $, /, %, and + symbols as "shift" characters. Those characters combined with the single character that follows indicate which Full ASCII character is to be used.

ASCII	ENCODING	ASCII	ENCODING	ASCII	ENCODING	ASCII	ENCODING
NUL	%U	SP	Space	@	%V	`	%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	"	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	'	/G	G	G	g	+G
BS	$H	(	/H	H	H	H	H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	,	/L	L	L	l	+L

CR	$M	-	-	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	0	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W
CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<<	%G	\	%L	|	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	^	%N	~	%S
YS	%E	?	%J	_	%O	DEL	%T, %X, %Y, %Z

ASCII CHAR	WIDTH ENCODING	BARCODE ENCODING	ASCII CHAR	WIDTH ENCODING	BARCODE ENCODING
0	NNNWWNWNN	101001101101	M	WNWNNNNWN	110110101001
1	WNNWNNNNW	110100101011	N	NNNNWNNWW	101011010011
2	NNWWNNNNW	101100101011	O	WNNNWNNWN	110101101001
3	WNWWNNNNN	110110010101	P	NNWNWNNWN	101101101001
4	NNNWWNNNW	101001101011	Q	NNNNNNWWW	101010110011
5	WNNWWNNNN	110100110101	R	WNNNNNWWN	110101011001
6	NNWWWNNNN	101100110101	S	NNWNNNWWN	101101011001
7	NNNWNNWNW	101001011011	T	NNNNWNWWN	101011011001
8	WNNWNNWNN	110100101101	U	WWNNNNNNW	110010101011
9	NNWWNNWNN	101100101101	V	NWWNNNNNW	100110101011
A	NNWWNNWNN	110101001011	W	WWWNNNNNN	110011010101
B	NNWNNWNNW	101101001011	X	NWNNWNNNW	100101101011
C	WNWNNWNNN	110110100101	Y	WWNNWNNNN	110010110101
D	NNNNWWNNW	101011001011	Z	NWWNWNNNN	100110110101
E	WNNNWWNNN	110101100101	-	NWNNNNWNW	100101011011
F	NNWNWWNNN	101101100101	.	WWNNNNWNN	110010101101
G	NNNNNWWNW	101010011011	SPACE	NWWNNNWNN	100110101101
H	WNNNNWWNN	110101001101	$	NWNWNWNNN	100100100101
I	NNWNNWWNN	101101001101	/	NWNWNNNWN	100100101001
J	NNNNWWWNN	101011001101	+	NWNNNWNWN	100101001001
K	WNNNNNNWW	110101010011	%	NNNWNWNWN	101001001001
L	NNWNNNNWW	101101010011	*	NWNNWNWNN	100101101101

eatures

- Compatible with MCS®51 Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low-power Idle and Power-down Modes
- nterrupt Recovery from Power-down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power-off Flag

escription

e AT89S8252 is a low-power, high-performance CMOS 8-bit microcontroller with 8K tes of downloadable Flash programmable and erasable read-only memory and 2K tes of EEPROM. The device is manufactured using Atmel's high-density nonvolatile mory technology and is compatible with the industry-standard 80C51 instruction t and pinout. The on-chip downloadable Flash allows the program memory to be rogrammed In-System through an SPI serial interface or by a conventional nonvol-le memory programmer. By combining a versatile 8-bit CPU with downloadable ish on a monolithic chip, the Atmel AT89S8252 is a powerful microcontroller, which vides a highly-flexible and cost-effective solution to many embedded control plications.

e AT89S8252 provides the following standard features: 8K bytes of downloadable ish, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog er, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt hitecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, AT89S8252 is designed with static logic for operation down to zero frequency and pports two software selectable power saving modes. The Idle Mode stops the CPU ile allowing the RAM, timer/counters, serial port, and interrupt system to continue ctioning. The Power-down mode saves the RAM contents but freezes the oscillator, abling all other chip functions until the next external interrupt or hardware reset.

e downloadable Flash can be changed a single byte at a time and is accessible ough the SPI serial interface. Holding RESET active forces the SPI bus into a serial gramming interface and allows the program memory to be written to or read from ess lock bits have been activated.

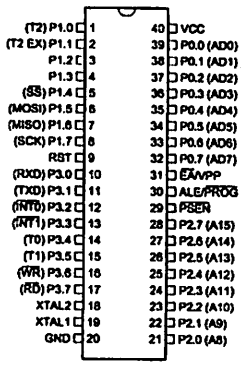
8-bit Microcontroller with 8K Bytes Flash

AT89S8252

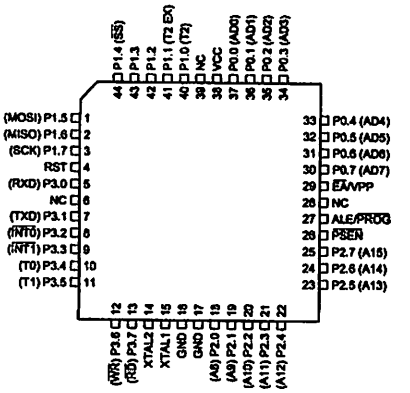
0401F–MICRO–11/03

in Configurations

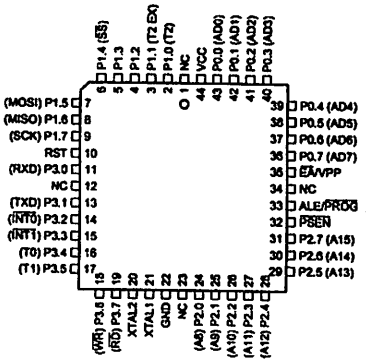
PDIP



TQFP



PLCC



n Description

C

Supply voltage.

ID

Ground.

rt 0

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

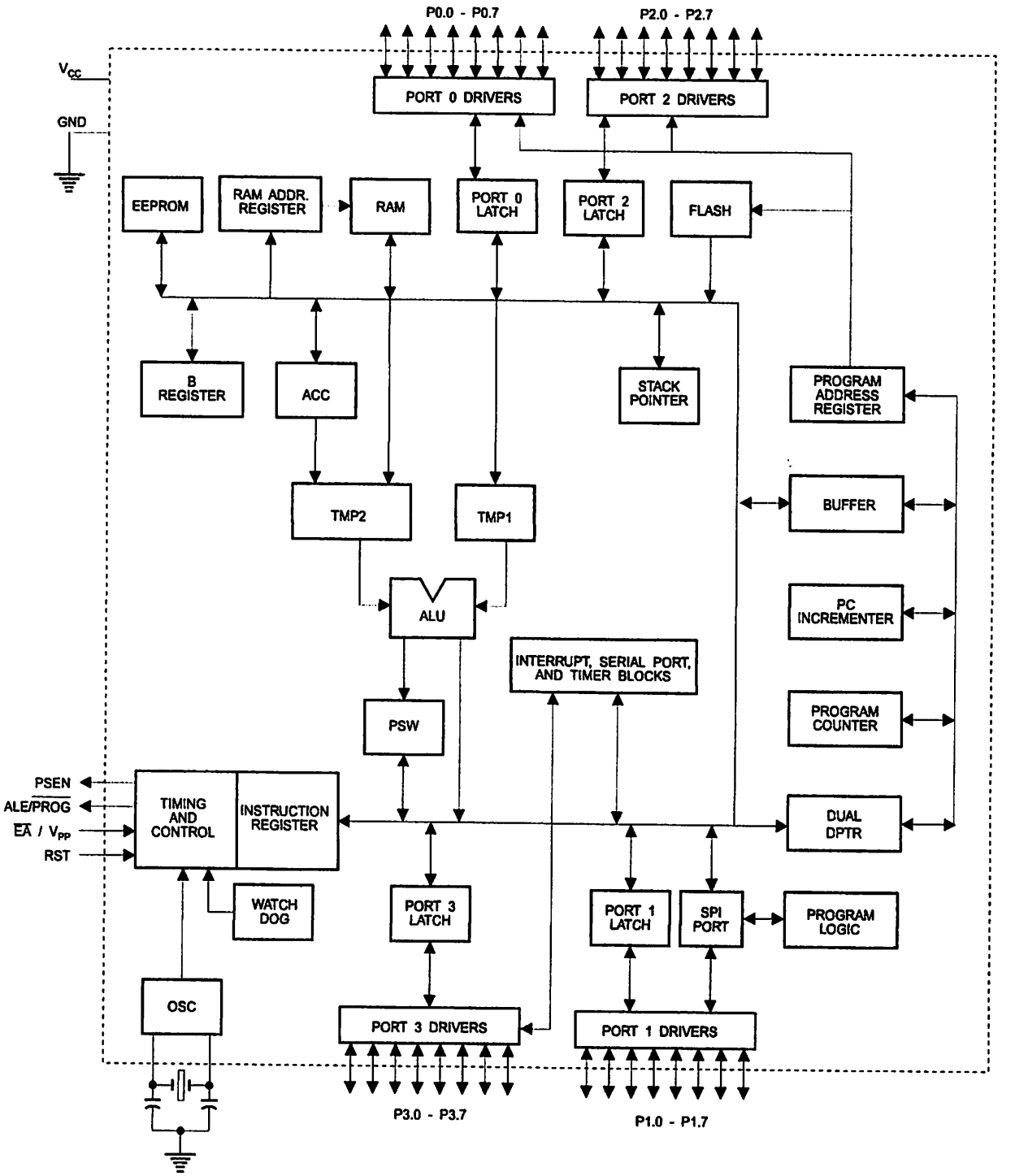
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups.

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

rt 1

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

lock Diagram



Some Port 1 pins provide additional functions. P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

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

ort 2

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

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

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

ort 3

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

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

Port 3 also serves the functions of various special features of the AT89S8252, as shown in the following table.

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

ST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

LE/$\overline{PROG}$

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

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

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

$\overline{SEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89S8252 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{A}$/VPP

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

$\overline{EA}$ should be strapped to V_{CC} for internal program executions. This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming when 12-volt programming is selected.

TAL1

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

TAL2

Output from the inverting oscillator amplifier.

pecial Function egisters

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

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

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

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 9) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

ble 1. AT89S8252 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DF H
0D0H	PSW 00000000					SPCR 000001XX		0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		0CF H
0C0H								0C7H
0B8H	IP XX000000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX					0AFH
0A0H	P2 11111111							0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111						WMCON 00000010	97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

ıble 2. T2CON – Timer/Counter 2 Control Register

'2CON Address = 0C8H — Reset Value = 0000 0000B

3it Addressable

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

ymbol	Function
F2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
XF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
CLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflows to be used for the receive clock.
CLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
XEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
R2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
P/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

atchdog and Memory Control Register The WMCON register contains control bits for the Watchdog Timer (shown in ible 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The PS bit selects one of two DPTR registers available.

ible 3. WMCON—Watchdog and Memory Control Register

VMCON Address = 96H — Reset Value = 0000 0010B

	PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN
lit	7	6	5	4	3	2	1	0

ymbol	Function
S2 S1 S0	Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms.
EMWE	EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed.
EMEN	Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory.
PS	Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1
/DTRST DY/$\overline{\text{BSY}}$	Watchdog Timer Reset and EEPROM Ready/$\overline{\text{Busy}}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{\text{BSY}}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{\text{BSY}}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{\text{BSY}}$ bit equals "0" and is automatically reset to "1" when programming is completed.
/DTEN	Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer.

SPI Registers Control and status bits for the Serial Peripheral Interface are contained in registers SPCR (shown in Table 4) and SPSR (shown in Table 5). The SPI data bits are contained in the SPDR register. Writing the SPI data register during serial data transfer sets the Write Collision bit, WCOL, in the SPSR register. The SPDR is double buffered for writing and the values in SPDR are not changed by Reset.

Interrupt Registers The global interrupt enable bit and the individual interrupt enable bits are in the IE register. In addition, the individual interrupt enable bit for the SPI is in the SPCR register. Two priorities can be set for each of the six interrupt sources in the IP register.

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

Power Off Flag The Power Off Flag (POF) is located at bit_4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and reset under software control and is not affected by RESET.

ble 4. SPCR – SPI Control Register

PCR Address = D5H — Reset Value = 0000 01XXB

Bit	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
	7	6	5	4	3	2	1	0

ymbol	Function
PIE	SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts.
PE	SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel.
ORD	Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission.
STR	Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode.
POL	Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control.
PHA	Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control.
PR0 PR1	SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 1 1 128

ible 5. SPSR – SPI Status Register

SPSR Address = AAH — Reset Value = 00XX XXXXB

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	–	–	–	–	–	–

Symbol	Function
SPIF	SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then reading/writing the SPI data register.
WCOL	Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register.

ible 6. SPDR – SPI Data Register

SPDR Address = 86H — Reset Value = unchanged

Bit	7	6	5	4	3	2	1	0
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0

ata Memory – EPROM and RAM

The AT89S8252 implements 2K bytes of on-chip EEPROM for data storage and 256 bytes of RAM. The upper 128 bytes of RAM occupy a parallel space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

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

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

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

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

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

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

The on-chip EEPROM data memory is selected by setting the EEMEN bit in the WMCON register at SFR address location 96H. The EEPROM address range is from 000H to 7FFH. The MOVX instructions are used to access the EEPROM. To access off-chip data memory with the MOVX instructions, the EEMEN bit needs to be set to "0".

The EEMWE bit in the WMCON register needs to be set to "1" before any byte location in the EEPROM can be written. User software should reset EEMWE bit to "0" if no further EEPROM write is required. EEPROM write cycles in the serial programming mode are self-timed and typically take 2.5 ms. The progress of EEPROM write can be monitored by reading the RDY/$\overline{BSY}$ bit (read-only) in SFR WMCON. RDY/$\overline{BSY}$ = 0 means

programming is still in progress and RDY/$\overline{BSY}$ = 1 means EEPROM write cycle is completed and another write cycle can be initiated.

In addition, during EEPROM programming, an attempted read from the EEPROM will fetch the byte being written with the MSB complemented. Once the write cycle is completed, true data are valid at all bit locations.

rogrammable /atchdog Timer

The programmable Watchdog Timer (WDT) operates from an independent internal oscillator. The prescaler bits, PS0, PS1 and PS2 in SFR WMCON are used to set the period of the Watchdog Timer from 16 ms to 2048 ms. The available timer periods are shown in the following table and the actual timer periods (at V_{CC} = 5V) are within ±30% of the nominal.

The WDT is disabled by Power-on Reset and during Power-down. It is enabled by setting the WDTEN bit in SFR WMCON (address = 96H). The WDT is reset by setting the WDTRST bit in WMCON. When the WDT times out without being reset or disabled, an internal RST pulse is generated to reset the CPU.

Table 7. Watchdog Timer Period Selection

WDT Prescaler Bits			Period (nominal)
PS2	PS1	PS0	
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

mer 0 and 1

Timer 0 and Timer 1 in the AT89S8252 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers' operation, refer to the Atmel web site (http://www.atmel.com). From the home page, select "Products", then "Microcontrollers, then "8051-Architecture". Click on "Documentation", then on "Other Documents". Open the document "AT89 Series Hardware Description".

mer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 8.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected.

Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

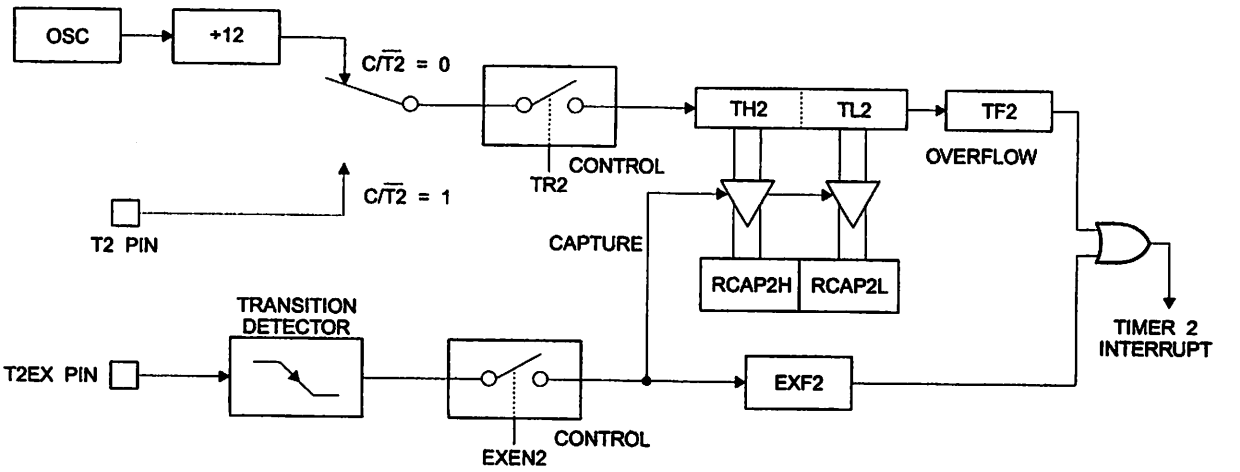
Table 8. Timer 2 Operating Modes

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

apture Mode

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

gure 1. Timer 2 in Capture Mode



uto-reload (Up or Down ounter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 9). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

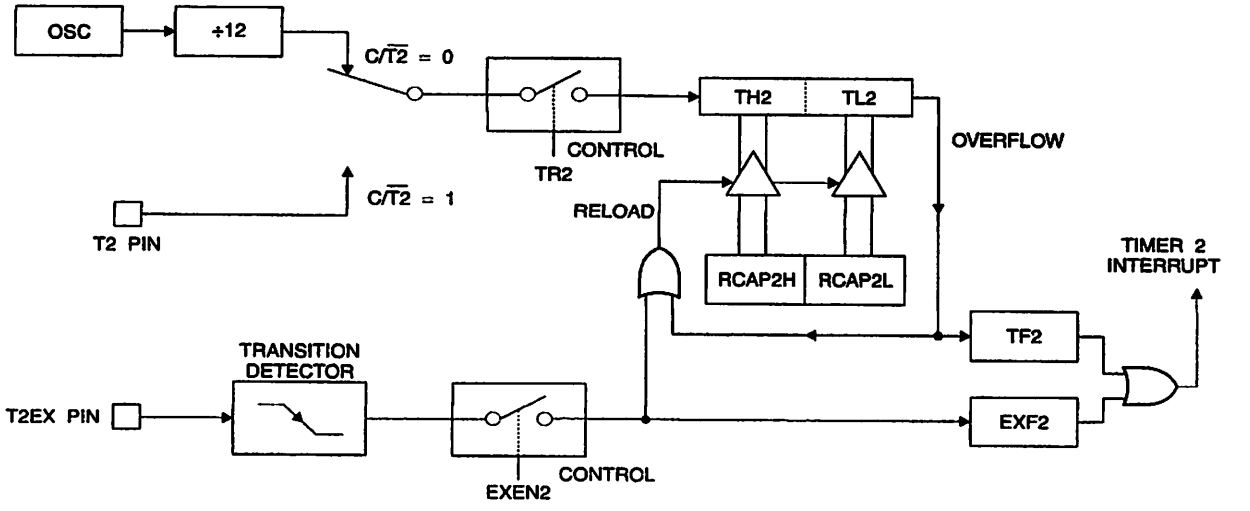
Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H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

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

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

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

gure 2. Timer 2 in Auto Reload Mode (DCEN = 0)



ble 9. T2MOD – Timer 2 Mode Control Register

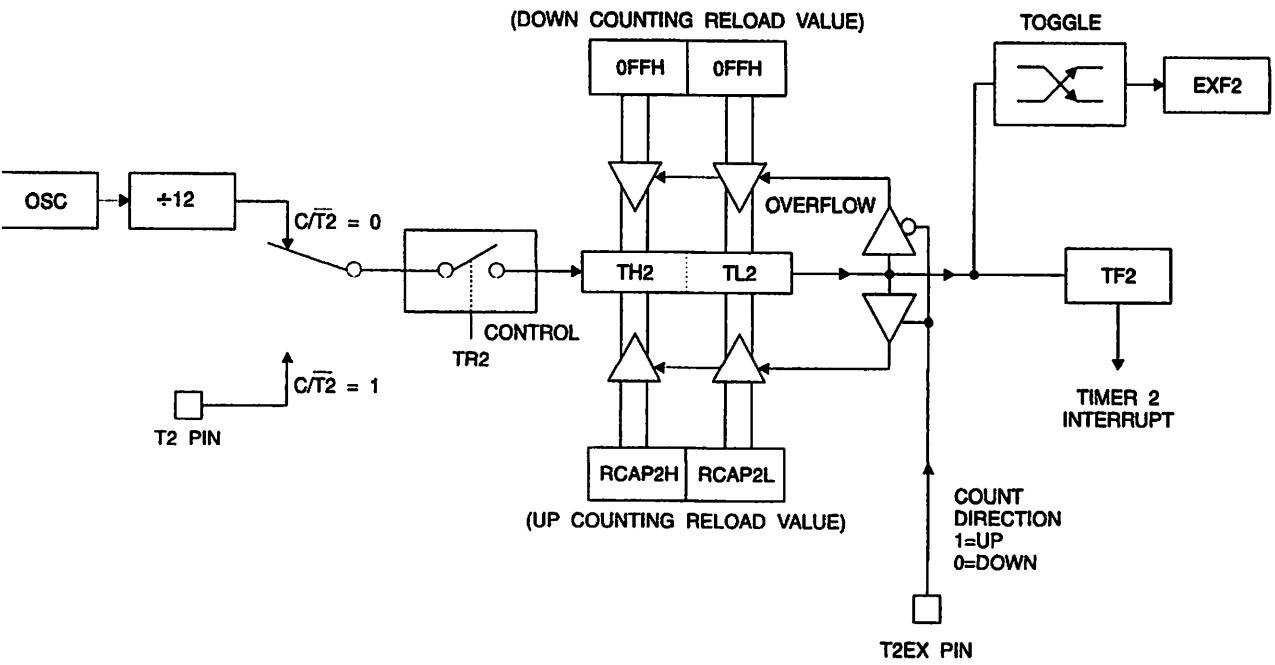
2MOD Address = 0C9H — Reset Value = XXXX XX00B

ot Bit Addressable

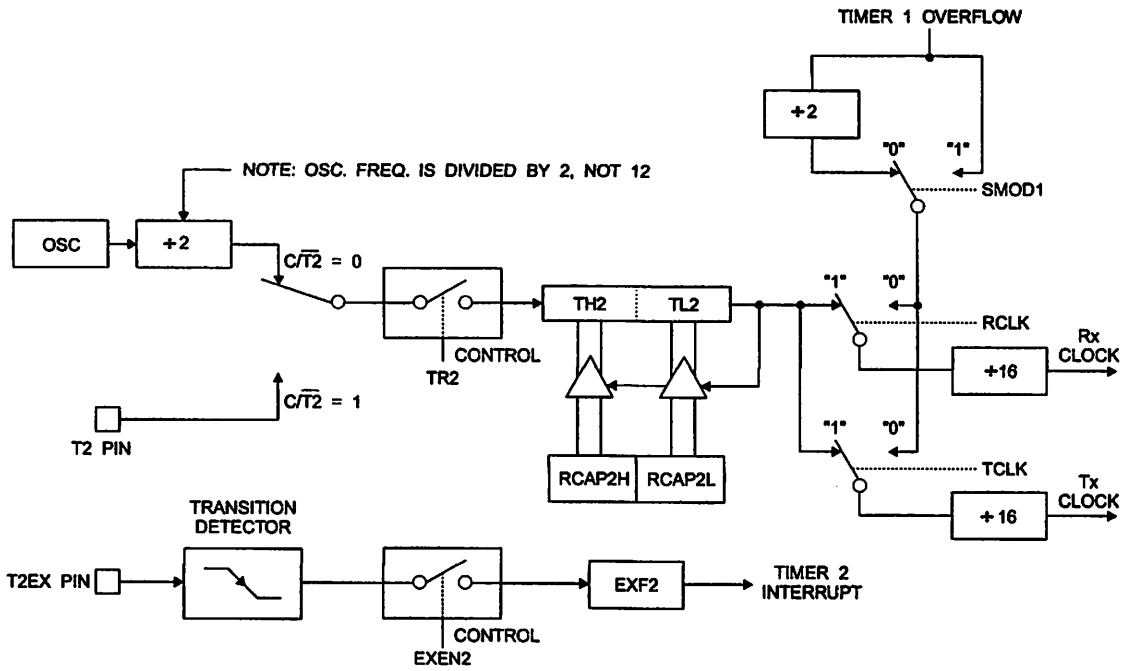
Bit	–	–	–	–	–	–	T2OE	DCEN
	7	6	5	4	3	2	1	0

ymbol	Function
	Not implemented, reserved for future use.
2OE	Timer 2 Output Enable bit.
CEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

jure 3. Timer 2 Auto Reload Mode (DCEN = 1)



ıure 4. Timer 2 in Baud Rate Generator Mode



ıud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 4.

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

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

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

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

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

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

Timer 2 as a baud rate generator is shown in Figure 4. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

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

rogrammable lock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/0 pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz (for a 16-MHz operating frequency).

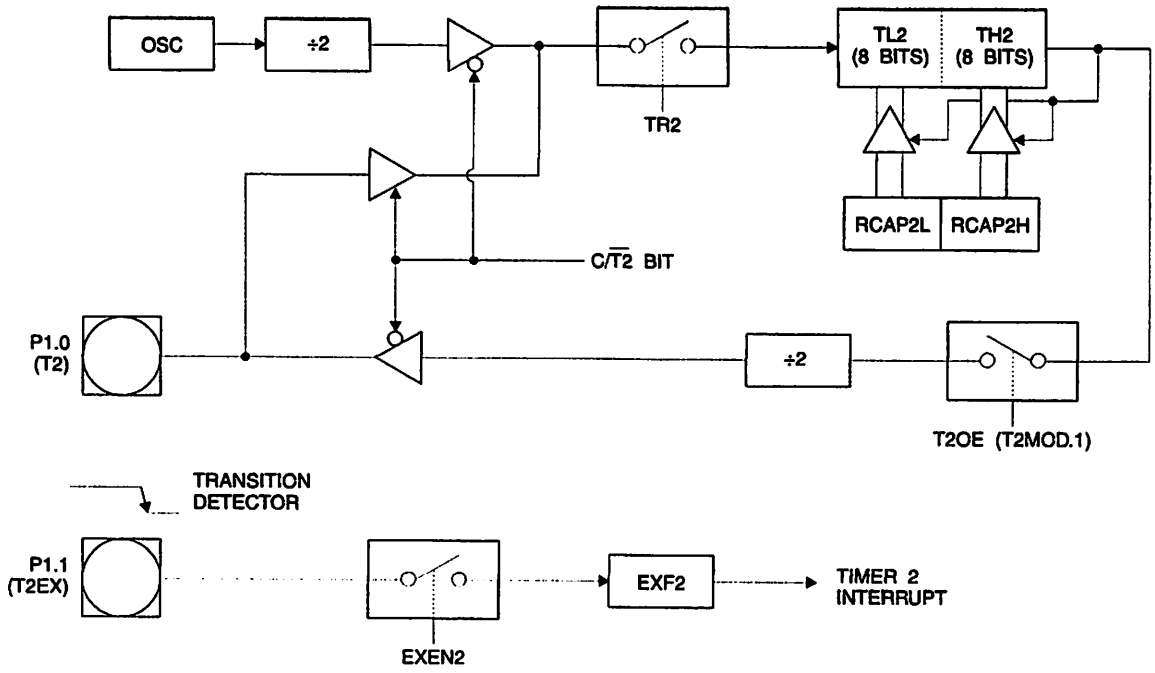
To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

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

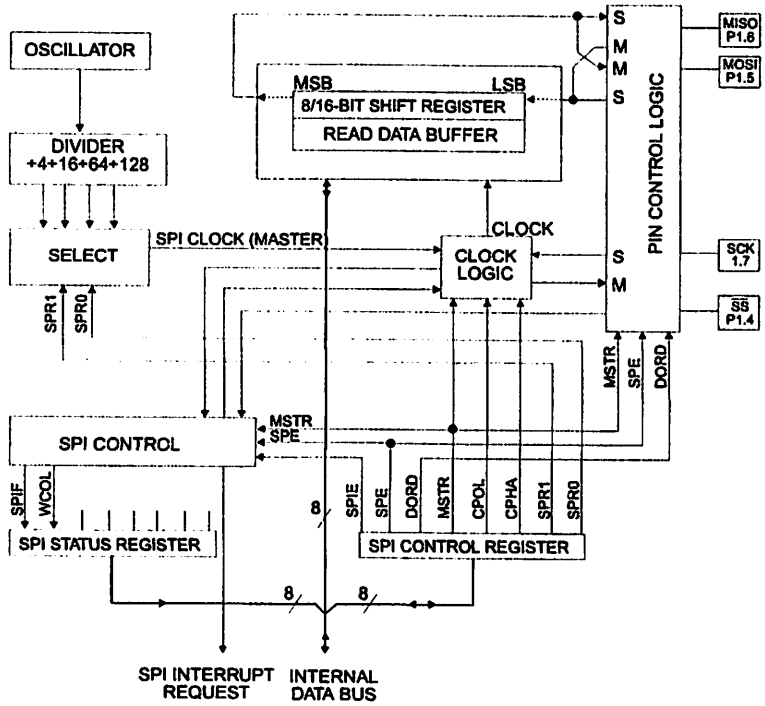
$$\text{Clock Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

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

ure 5. Timer 2 in Clock-out Mode



ure 6. SPI Block Diagram



ART

The UART in the AT89S8252 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, refer to the Atmel web site (http://www.atmel.com). From the home page, select "Products", then "Microcontrollers, then "8051-Architecture". Click on "Documentation", then on "Other Documents". Open the document "AT89 Series Hardware Description".

erial Peripheral terface

The serial peripheral interface (SPI) allows high-speed synchronous data transfer between the AT89S8252 and peripheral devices or between several AT89S8252 devices. The AT89S8252 SPI features include the following:

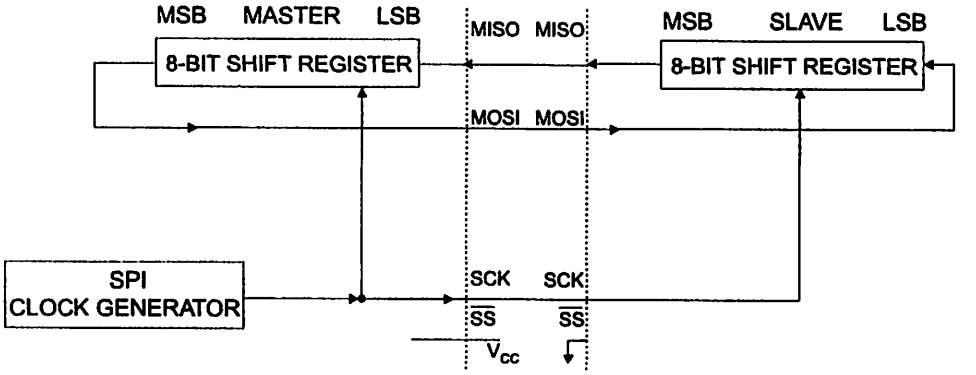
- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 1.5 MHz Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

The interconnection between master and slave CPUs with SPI is shown in the following figure. The SCK pin is the clock output in the master mode but is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the MOSI pin and into the MOSI pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If both the SPI interrupt enable bit (SPIE) and the serial port interrupt enable bit (ES) are set, an interrupt is requested.

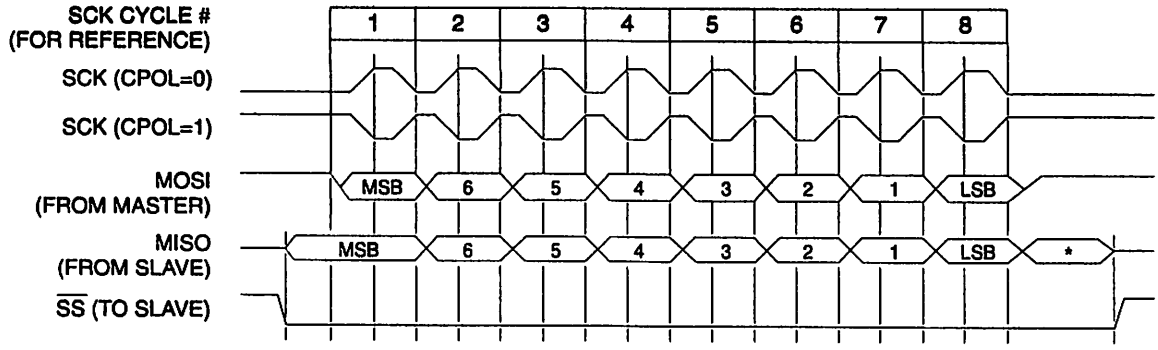
The Slave Select input, $\overline{SS}$/P1.4, is set low to select an individual SPI device as a slave. When $\overline{SS}$/P1.4 is set high, the SPI port is deactivated and the MOSI/P1.5 pin can be used as an input.

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 8 and Figure 9.

ure 7. SPI Master-slave Interconnection

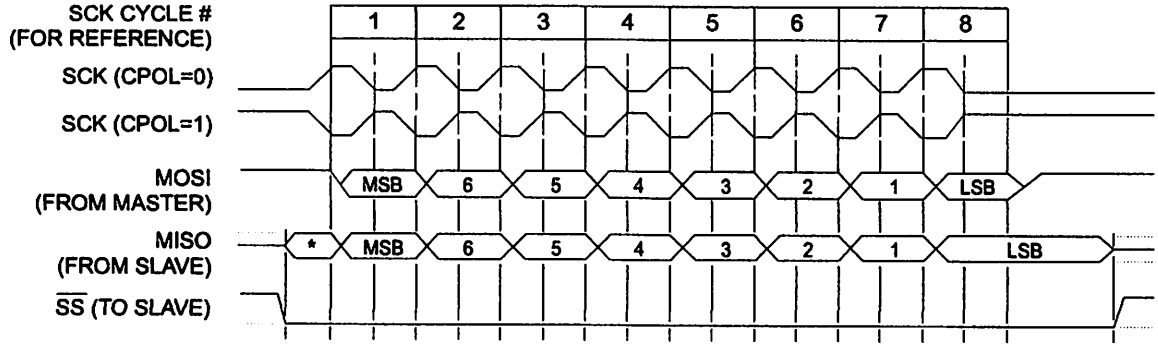


ure 8. SPI transfer Format with CPHA = 0



te: *Not defined but normally MSB of character just received

ure 9. SPI Transfer Format with CPHA = 1



te: *Not defined but normally LSB of previously transmitted character.

terrupts

The AT89S8252 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 10.

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

Note that Table 10 shows that bit position IE.6 is unimplemented. In the AT89C51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

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

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

ble 10. Interrupt Enable (IE) Register

MSB)(LSB)

EA	–	ET2	ES	ET1	EX1	ET0	EX0

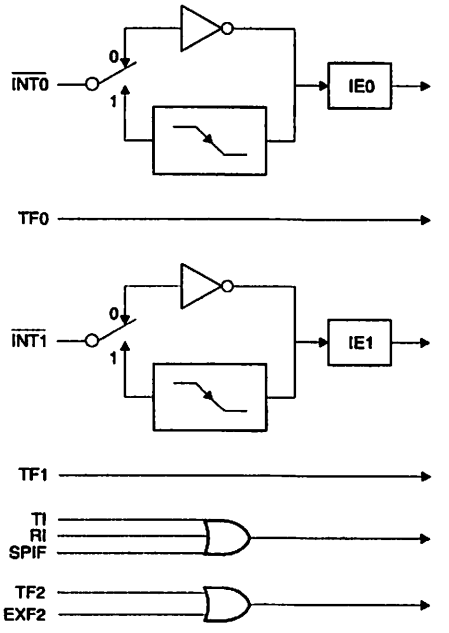
Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	SPI and UART interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

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

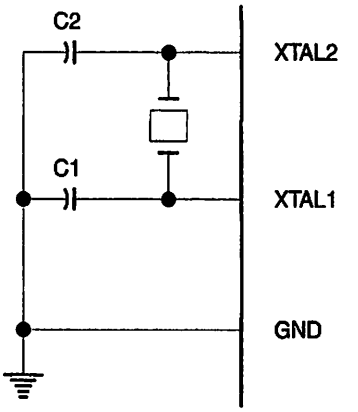
gure 10. Interrupt Sources



scillator
ıaracteristics

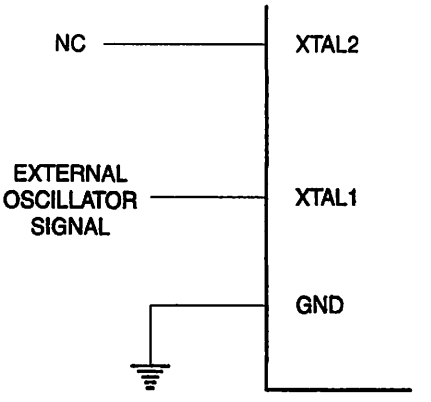
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 12. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 11. Oscillator Connections



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

Figure 12. External Clock Drive Configuration



le Mode

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

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

Status of External Pins During Idle and Power-down Modes

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

ɔwer-down Mode

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

To exit power-down via an interrupt, the external interrupt must be enabled as level sensitive before entering power-down. The interrupt service routine starts at 16 ms (nominal) after the enabled interrupt pin is activated.

ʳogram Memory ɔck Bits

The AT89S8252 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

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

Once programmed, the lock bits can only be unprogrammed with the Chip Erase operations in either the parallel or serial modes.

ɔck Bit Protection Modes[1][2]

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

tes: 1. U = Unprogrammed
2. P = Programmed

rogramming the lash and EEPROM

Atmel's AT89S8252 Flash Microcontroller offers 8K bytes of in-system reprogrammable Flash Code memory and 2K bytes of EEPROM Data memory.

The AT89S8252 is normally shipped with the on-chip Flash Code and EEPROM Data memory arrays in the erased state (i.e. contents = FFH) and ready to be programmed. This device supports a High-voltage (12-V V_{PP}) Parallel programming mode and a Low-voltage (5-V V_{CC}) Serial programming mode. The serial programming mode provides a convenient way to reprogram the AT89S8252 inside the user's system. The parallel programming mode is compatible with conventional third party Flash or EPROM programmers.

The Code and Data memory arrays are mapped via separate address spaces in the serial programming mode. In the parallel programming mode, the two arrays occupy one contiguous address space: 0000H to 1FFFH for the Code array and 2000H to 27FFH for the Data array.

The Code and Data memory arrays on the AT89S8252 are programmed byte-by-byte in either programming mode. An auto-erase cycle is provided with the self-timed programming operation in the serial programming mode. There is no need to perform the Chip Erase operation to reprogram any memory location in the serial programming mode unless any of the lock bits have been programmed.

In the parallel programming mode, there is no auto-erase cycle. To reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

Parallel Programming Algorithm: To program and verify the AT89S8252 in the parallel programming mode, the following sequence is recommended:

1. Power-up sequence:
   Apply power between V_{CC} and GND pins.
   Set RST pin to "H".
   Apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Set $\overline{PSEN}$ pin to "L"
   ALE pin to "H"
   $\overline{EA}$ pin to "H" and all other pins to "H".
3. Apply the appropriate combination of "H" or "L" logic levels to pins P2.6, P2.7, P3.6, P3.7 to select one of the programming operations shown in the Flash Programming Modes table.
4. Apply the desired byte address to pins P1.0 to P1.7 and P2.0 to P2.5.
   Apply data to pins P0.0 to P0.7 for Write Code operation.
5. Raise $\overline{EA}/V_{PP}$ to 12V to enable Flash programming, erase or verification.
6. Pulse ALE/$\overline{PROG}$ once to program a byte in the Code memory array, the Data memory array or the lock bits. The byte-write cycle is self-timed and typically takes 1.5 ms.
7. To verify the byte just programmed, bring pin P2.7 to "L" and read the programmed data at pins P0.0 to P0.7.
8. Repeat steps 3 through 7 changing the address and data for the entire 2K or 8K bytes array or until the end of the object file is reached.
9. Power-off sequence:
   Set XTAL1 to "L".
   Set RST and $\overline{EA}$ pins to "L".
   Turn V_{CC} power off.

In the parallel programming mode, there is no auto-erase cycle and to reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

Data Polling: The AT89S8252 features DATA Polling to indicate the end of a byte write cycle. During a byte write cycle in the parallel or serial programming mode, an attempted read of the last byte written will result in the complement of the written datum on P0.7 (parallel mode), and on the MSB of the serial output byte on MISO (serial mode). Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. DATA Polling may begin any time after a write cycle has been initiated.

Ready/Busy: The progress of byte programming in the parallel programming mode can also be monitored by the RDY/BSY output signal. Pin P3.4 is pulled Low after ALE goes High during programming to indicate BUSY. P3.4 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed Code or Data byte can be read back via the address and data lines for verification. The state of the lock bits can also be verified directly in the parallel programming mode. In the serial programming mode, the state of the lock bits can only be verified indirectly by observing that the lock bit features are enabled.

Chip Erase: Both Flash and EEPROM arrays are erased electrically at the same time. In the parallel programming mode, chip erase is initiated by using the proper combination of control signals and by holding ALE/PROG low for 10 ms. The Code and Data arrays are written with all "1"s in the Chip Erase operation.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 16 ms.

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

Serial Programming Fuse: A programmable fuse is available to disable Serial Programming if the user needs maximum system security. The Serial Programming Fuse can only be programmed or erased in the Parallel Programming Mode.

The AT89S8252 is shipped with the Serial Programming Mode enabled.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows:

(030H) = 1EH indicates manufactured by Atmel
(031H) = 72H indicates 89S8252

ogramming terface

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

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

ɛrial Downloading

Both the Code and Data memory arrays can be programmed using the serial SPI bus while RST is pulled to V_{CC}. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

An auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction unless any of the lock bits have been programmed. The Chip Erase operation turns the content of every memory location in both the Code and Data arrays into FFH.

The Code and Data memory arrays have separate address spaces:

0000H to 1FFFH for Code memory and 000H to 7FFH for Data memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/40 of the crystal frequency. With a 24 MHz oscillator clock, the maximum SCK frequency is 600 kHz.

ɪrial Programming gorithm

To program and verify the AT89S8252 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

   If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
3. The Code or Data array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. The selected memory location is first automatically erased before new data is written. The write cycle is self-timed and typically takes less than 2.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal operation.
6. Power-off sequence (if needed):

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

   Set RST to "L".

   Turn V_{CC} power off.

ɛrial Programming struction

The Instruction Set for Serial Programming follows a 3-byte protocol and is shown in the following table:

ɜtruction Set

ɪstruction	Input Format			Operation
	Byte 1	Byte 2	Byte 3	
ʳrogramming Enable	1010 1100	0101 0011	xxxx xxxx	Enable serial programming interface after RST goes high.
ːhip Erase	1010 1100	xxxx x100	xxxx xxxx	Chip erase both 8K & 2K memory arrays.
ːead Code Memory	aaaa a001	low addr	xxxx xxxx	Read data from Code memory array at the selected address. The 5 MSBs of the first byte are the high order address bits. The low order address bits are in the second byte. Data are available at pin MISO during the third byte.
Ⱶrite Code Memory	aaaa a010	low addr	data in	Write data to Code memory location at selected address. The address bits are the 5 MSBs of the first byte together with the second byte.
ːead Data Memory	00aa a101	low addr	xxxx xxxx	Read data from Data memory array at selected address. Data are available at pin MISO during the third byte.
Ⱶrite Data Memory	00aa a110	low addr	data in	Write data to Data memory location at selected address.
Ⱶrite Lock Bits	1010 1100	LB1 LB2 LB3 x x111	xxxx xxxx	Write lock bits. Set LB1, LB2 or LB3 = "0" to program lock bits.

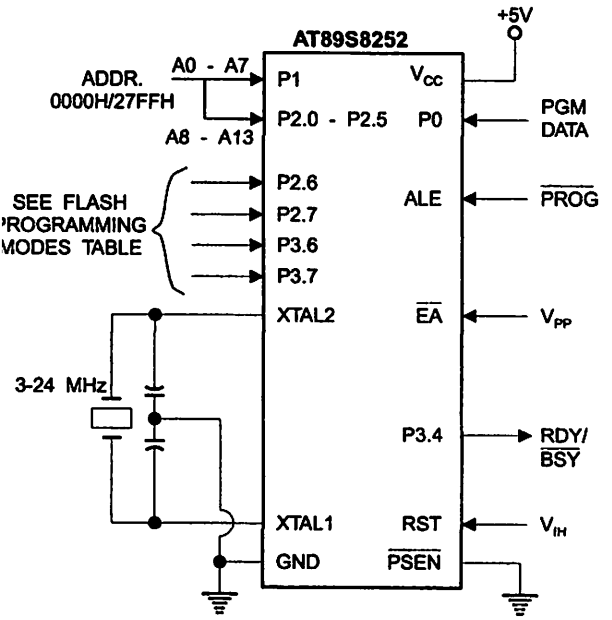
tes:
1. DATA polling is used to indicate the end of a byte write cycle which typically takes less than 2.5 ms at 5V.
2. "aaaaa" = high order address.
3. "x" = don't care.

lash and EEPROM Parallel Programming Modes

lode	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7	Data I/O P0.7:0	Address P2.5:0 P1.7:0
erial Prog. Modes	H	h(1)	h(1)	x						
hip Erase	H	L	(2)	12V	H	L	L	L	X	X
Vrite (10K bytes) Memory	H	L		12V	L	H	H	H	DIN	ADDR
ead (10K bytes) Memory	H	L	H	12V	L	L	H	H	DOUT	ADDR
Vrite Lock Bits:	H	L		12V	H	L	H	L	DIN	X
Bit - 1									P0.7 = 0	X
Bit - 2									P0.6 = 0	X
Bit - 3									P0.5 = 0	X
ead Lock Bits:	H	L	H	12V	H	H	L	L	DOUT	X
Bit - 1									@P0.2	X
Bit - 2									@P0.1	X
Bit - 3									@P0.0	X
ead Atmel Code	H	L	H	12V	L	L	L	L	DOUT	30H
ead Device Code	H	L	H	12V	L	L	L	L	DOUT	31H
erial Prog. Enable	H	L	(2)	12V	L	H	L	H	P0.0 = 0	X
erial Prog. Disable	H	L	(2)	12V	L	H	L	H	P0.0 = 1	X
ead Serial Prog. Fuse	H	L	H	12V	H	H	L	H	@P0.0	X

tes:
1. "h" = weakly pulled "High" internally.
2. Chip Erase and Serial Programming Fuse require a 10 ms PROG pulse. Chip Erase needs to be performed first before reprogramming any byte with a content other than FFH.
3. P3.4 is pulled Low during programming to indicate RDY/BSY.
4. "X" = don't care

ɟure 13. Programming the Flash/EEPROM Memory



ɟure 14. Verifying the Flash/EEPROM Memory

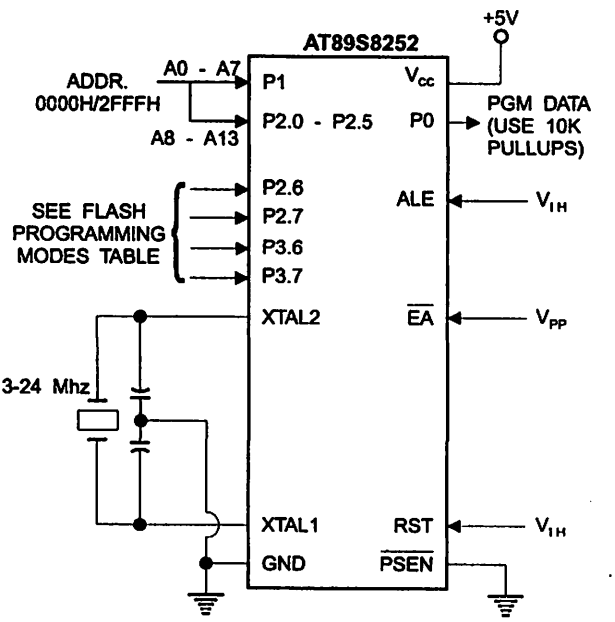
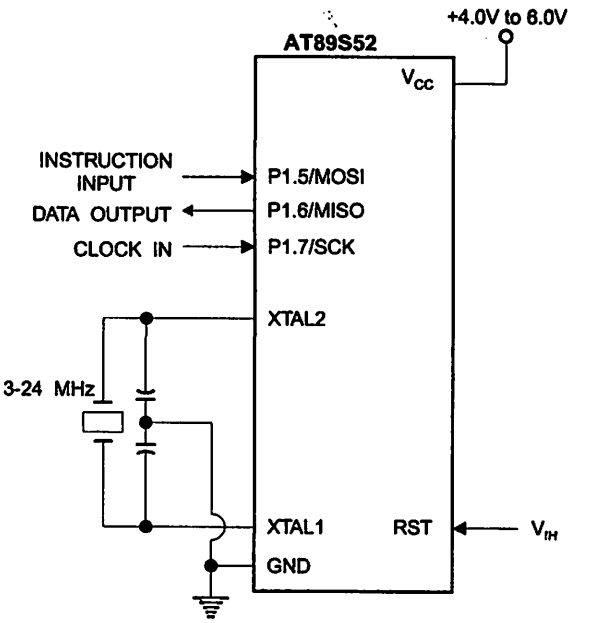


Figure 15. Flash/EEPROM Serial Downloading

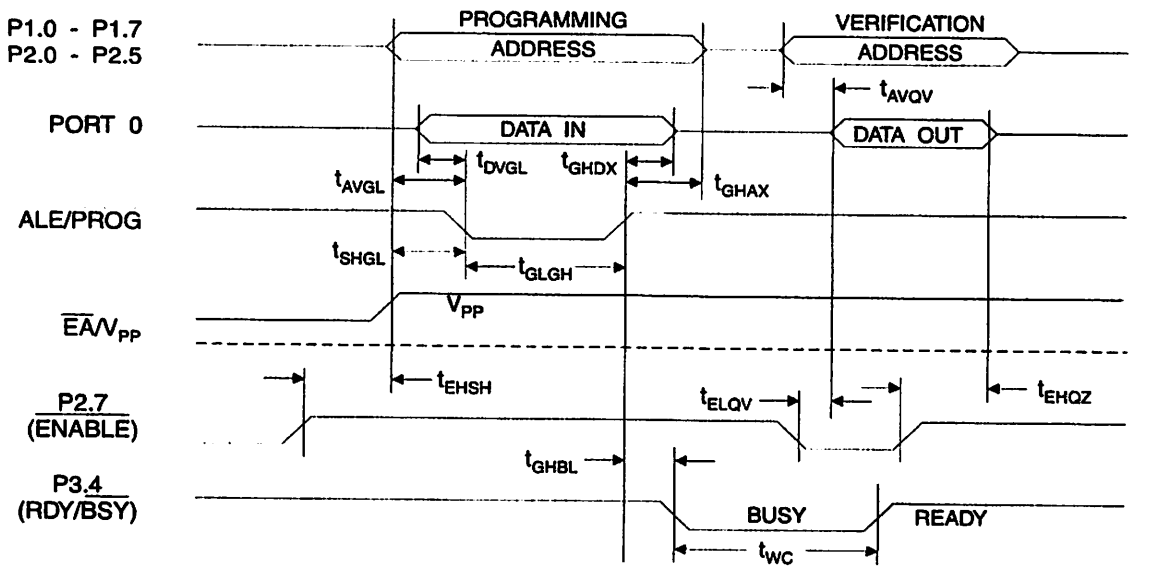


lash Programming and Verification Characteristics – Parallel Mode

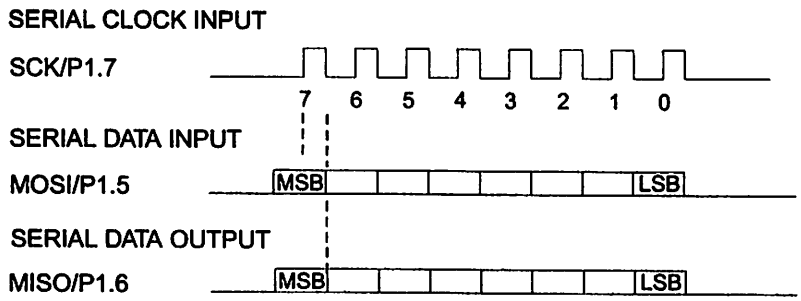
= 0°C to 70°C, V_{CC} = 5.0V ±10%

ymbol	Parameter	Min	Max	Units
PP	Programming Enable Voltage	11.5	12.5	V
PP	Programming Enable Current		1.0	mA
/tCLCL	Oscillator Frequency	3	24	MHz
AVGL	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
GHAX	Address Hold after $\overline{PROG}$	$48t_{CLCL}$		
DVGL	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
GHDX	Data Hold after $\overline{PROG}$	$48t_{CLCL}$		
EHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
SHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
GLGH	$\overline{PROG}$ Width	1	110	µs
AVQV	Address to Data Valid		$48t_{CLCL}$	
ELQV	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
EHQZ	Data Float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
GHBL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
WC	Byte Write Cycle Time		2.0	ms

ash/EEPROM Programming and Verification Waveforms – Parallel Mode

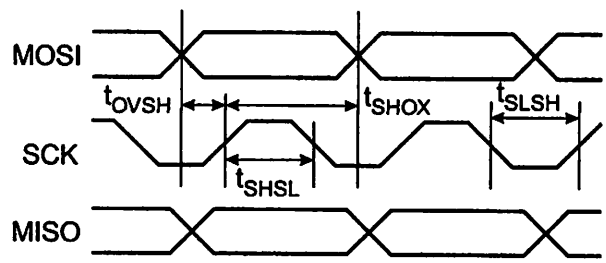


erial Downloading Waveforms



erial Programming Characteristics

gure 16. Serial Programming Timing



ble 11. Serial Programming Characteristics, T_A = -40° C to 85° C, V_{CC} = 4.0 - 6.0V (Unless Otherwise Noted)

ymbol	Parameter	Min	Typ	Max	Units
t_{CLCL}	Oscillator Frequency	0		24	MHz
LCL	Oscillator Period	41.6			ns
HSL	SCK Pulse Width High	24 t_{CLCL}			ns
LSH	SCK Pulse Width Low	24 t_{CLCL}			ns
VSH	MOSI Setup to SCK High	t_{CLCL}			ns
HOX	MOSI Hold after SCK High	2 t_{CLCL}			ns

bsolute Maximum Ratings*

perating Temperature	-55°C to +125°C
torage Temperature	-65°C to +150°C
oltage on Any Pin ith Respect to Ground	-1.0V to +7.0V
laximum Operating Voltage	6.6V
C Output Current	15.0 mA

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

C Characteristics

e values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ± 20%, unless otherwise noted.

ymbol	Parameter	Condition	Min	Max	Units
IL	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
IL1	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
IH	Input High-voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
IH1	Input High-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
OL	Output Low-voltage (1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.5	V
OL1	Output Low-voltage (1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.5	V
OH	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
OH1	Output High-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
.	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
L	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RST	Reset Pull-down Resistor		50	300	KΩ
IO	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
C	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode (2)	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

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

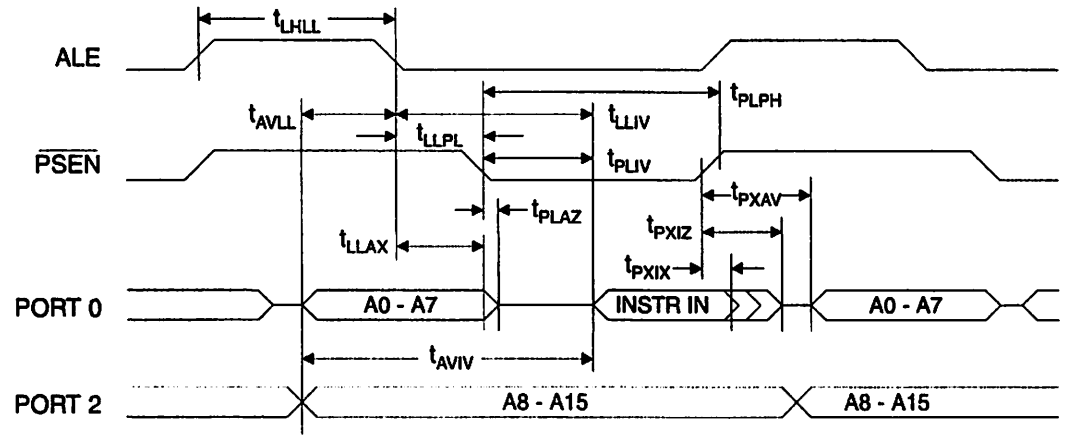
C Characteristics

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

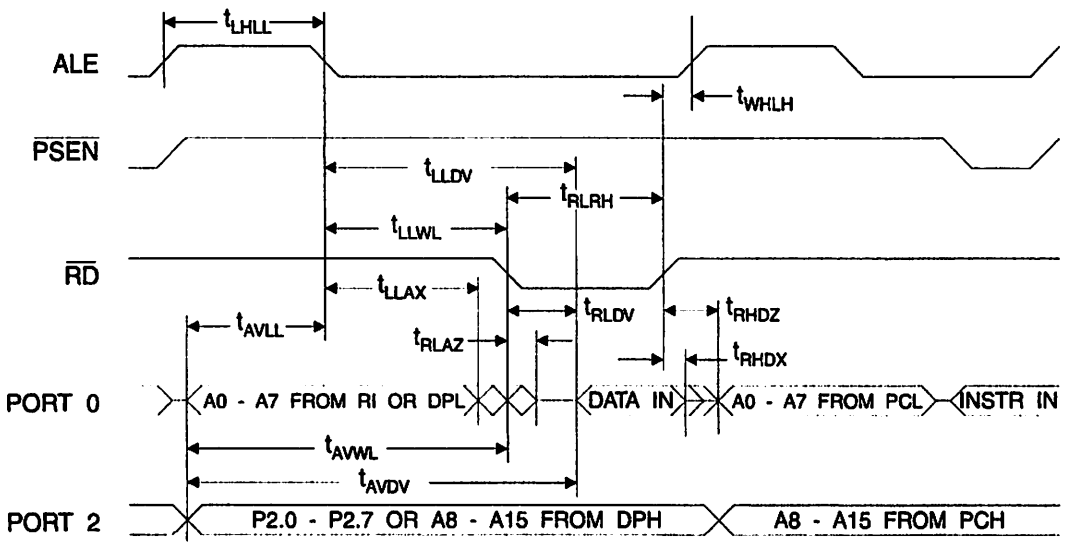
ternal Program and Data Memory Characteristics

ymbol	Parameter	Variable Oscillator		Units
		Min	Max	
f_{CLCL}	Oscillator Frequency	0	24	MHz
HLL	ALE Pulse Width	$2t_{CLCL}$ - 40		ns
VLL	Address Valid to ALE Low	t_{CLCL} - 13		ns
LAX	Address Hold after ALE Low	t_{CLCL} - 20		ns
LIV	ALE Low to Valid Instruction In		$4t_{CLCL}$ - 65	ns
LPL	ALE Low to $\overline{PSEN}$ Low	t_{CLCL} - 13		ns
PLPH	$\overline{PSEN}$ Pulse Width	$3t_{CLCL}$ - 20		ns
PLIV	$\overline{PSEN}$ Low to Valid Instruction In		$3t_{CLCL}$ - 45	ns
PXIX	Input Instruction Hold after $\overline{PSEN}$	0		ns
PXIZ	Input Instruction Float after $\overline{PSEN}$		t_{CLCL} - 10	ns
PXAV	$\overline{PSEN}$ to Address Valid	t_{CLCL} - 8		ns
AVIV	Address to Valid Instruction In		$5t_{CLCL}$ - 55	ns
PLAZ	$\overline{PSEN}$ Low to Address Float		10	ns
RLRH	$\overline{RD}$ Pulse Width	$6t_{CLCL}$ - 100		ns
WLWH	$\overline{WR}$ Pulse Width	$6t_{CLCL}$ - 100		ns
RLDV	$\overline{RD}$ Low to Valid Data In		$5t_{CLCL}$ - 90	ns
RHDX	Data Hold after $\overline{RD}$	0		ns
RHDZ	Data Float after $\overline{RD}$		$2t_{CLCL}$ - 28	ns
LDV	ALE Low to Valid Data In		$8t_{CLCL}$ - 150	ns
AVDV	Address to Valid Data In		$9t_{CLCL}$ - 165	ns
LWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$3t_{CLCL}$ - 50	$3t_{CLCL}$ + 50	ns
AVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	$4t_{CLCL}$ - 75		ns
QVWX	Data Valid to $\overline{WR}$ Transition	t_{CLCL} - 20		ns
QVWH	Data Valid to $\overline{WR}$ High	$7t_{CLCL}$ - 120		ns
WHQX	Data Hold after $\overline{WR}$	t_{CLCL} - 20		ns
RLAZ	$\overline{RD}$ Low to Address Float		0	ns
WHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{CLCL} - 20	t_{CLCL} + 25	ns

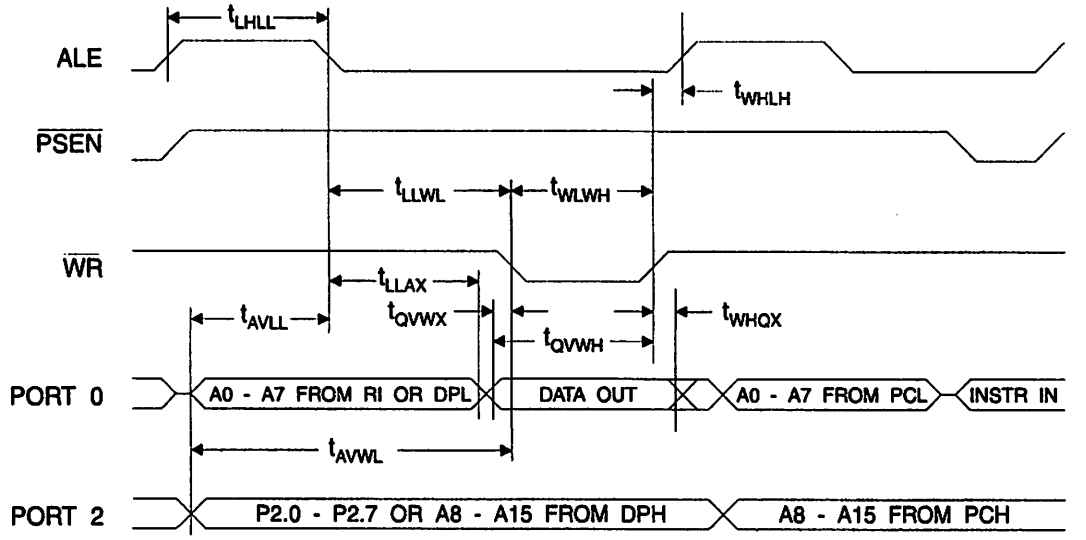
xternal Program Memory Read Cycle



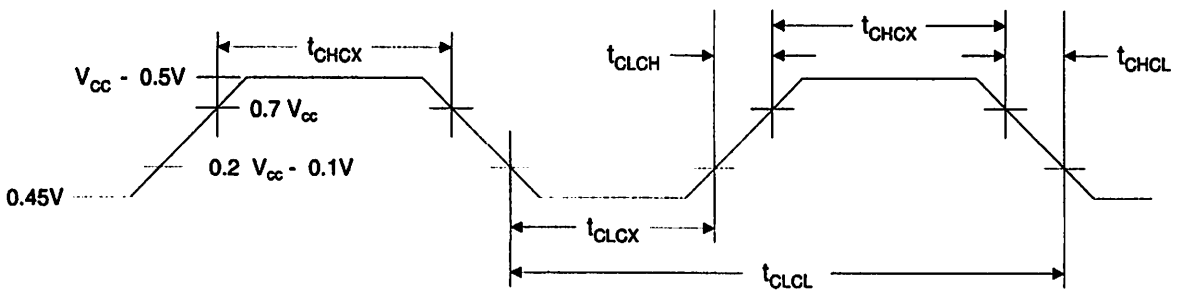
xternal Data Memory Read Cycle



xternal Data Memory Write Cycle



xternal Clock Drive Waveforms



xternal Clock Drive

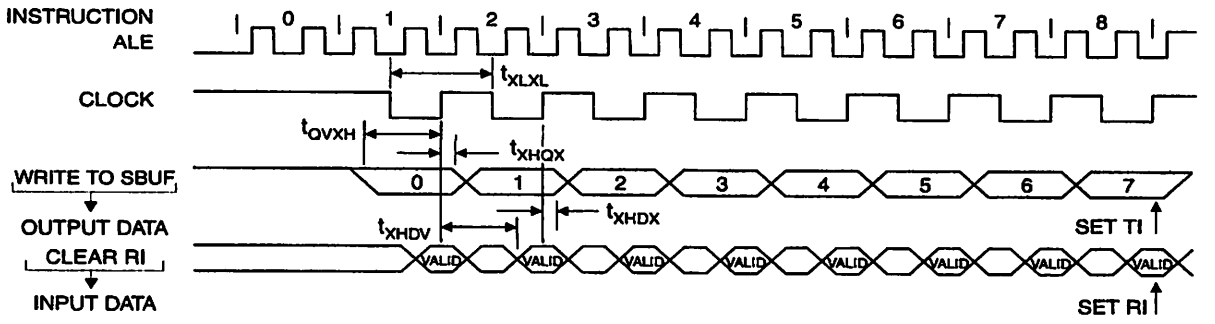
ymbol	Parameter	V_{CC} = 4.0V to 6.0V		Units
		Min	Max	
/t_{CLCL}	Oscillator Frequency	0	24	MHz
:LCL	Clock Period	41.6		ns
:HCX	High Time	15		ns
:LCX	Low Time	15		ns
:LCH	Rise Time		20	ns
:HCL	Fall Time		20	ns

erial Port Timing: Shift Register Mode Test Conditions

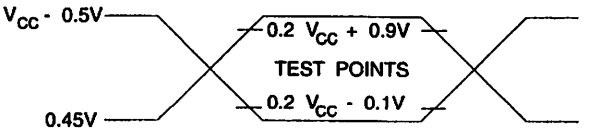
e values in this table are valid for V_{CC} = 4.0V to 6V and Load Capacitance = 80 pF.

ymbol	Parameter	Variable Oscillator		Units
		Min	Max	
LXL	Serial Port Clock Cycle Time	$12t_{CLCL}$		µs
VXH	Output Data Setup to Clock Rising Edge	$10t_{CLCL}$ - 133		ns
HQX	Output Data Hold after Clock Rising Edge	$2t_{CLCL}$ - 117		ns
HDX	Input Data Hold after Clock Rising Edge	0		ns
HDV	Clock Rising Edge to Input Data Valid		$10t_{CLCL}$ - 133	ns

hift Register Mode Timing Waveforms

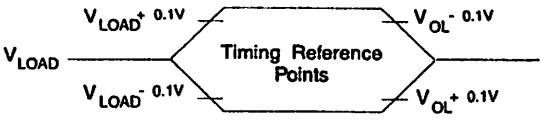


C Testing Input/Output Waveforms[1]



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

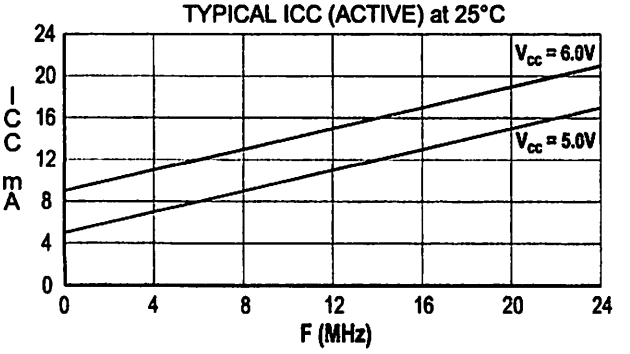
loat Waveforms[1]



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

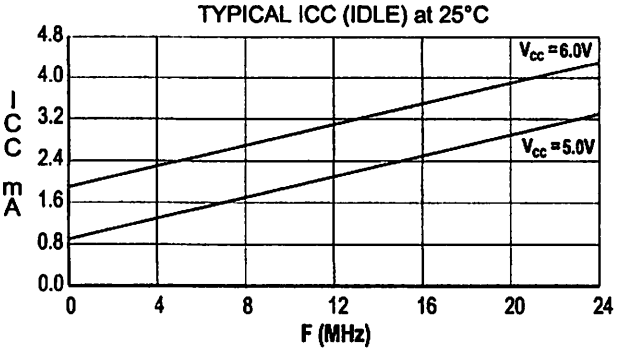
AT89S8252

TYPICAL ICC (ACTIVE) at 25°C



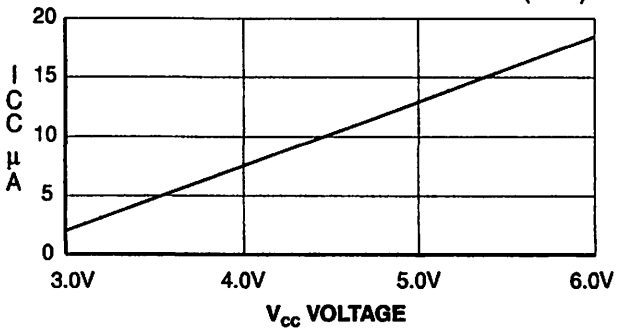
AT89S8252

TYPICAL ICC (IDLE) at 25°C



AT89S8252

TYPICAL ICC vs. VOLTAGE - POWER DOWN (85°C)



Notes: 1. XTAL1 tied to GND for Icc (power-down)
2. Lock bits programmed

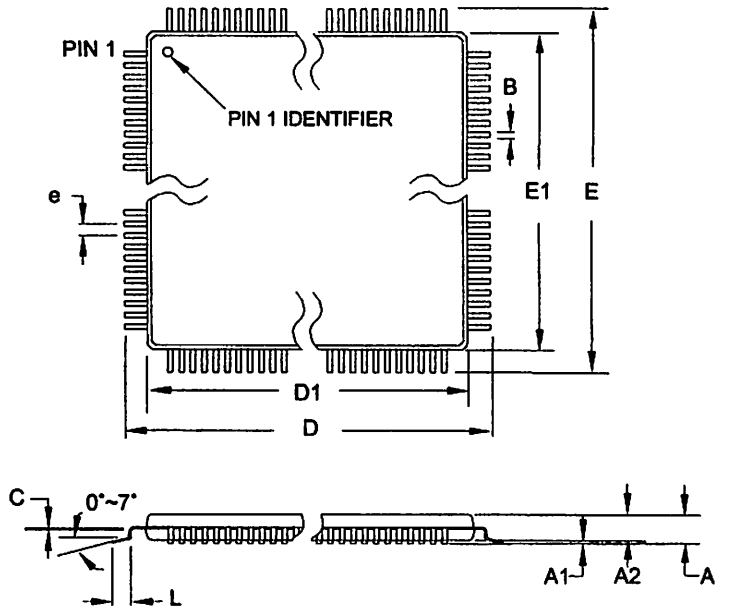
rdering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 6.0V	AT89S8252-24AC AT89S8252-24JC AT89S8252-24PC	44A 44J 40P6	Commercial (0° C to 70° C)
	4.0V to 6.0V	AT89S8252-24AI AT89S8252-24JI AT89S8252-24PI	44A 44J 40P6	Industrial (-40° C to 85° C)

Package Type	
4A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
4J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
0P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)

ackaging Information

A – TQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

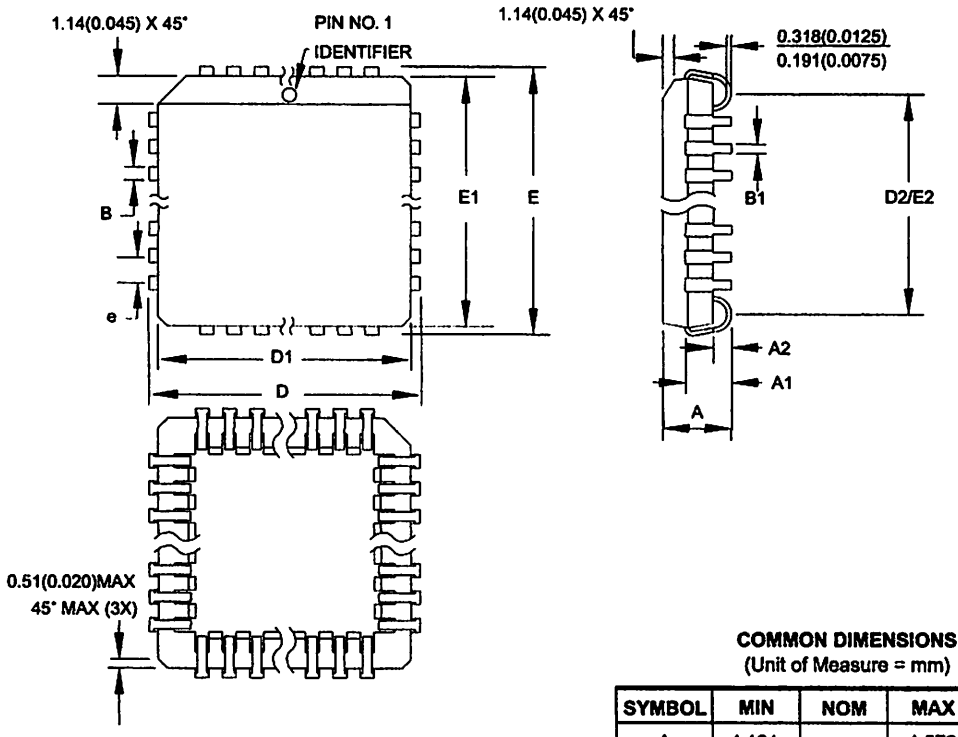
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

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

10/5/2001

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

J – PLCC



COMMON DIMENSIONS
(Unit of Measure = mm)

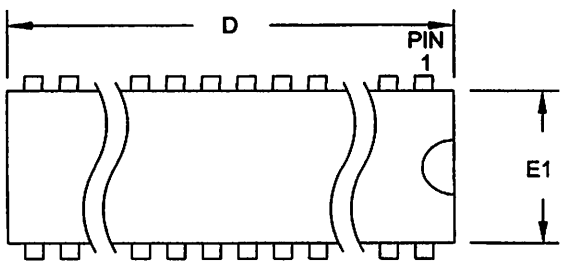
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

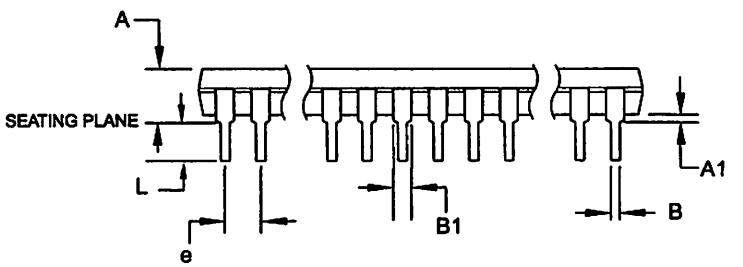
Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting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

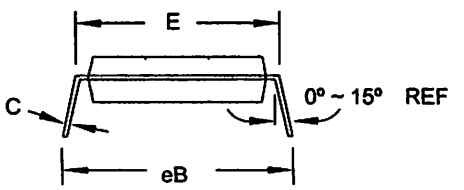
10/04/01

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

P6 – PDIP







COMMON DIMENSIONS
(Unit of Measure = mm)

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

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

09/28/01

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

ıtmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

ıegional Headquarters

ıurope
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

ısia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

ıpan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

0401F–MICRO–11/03 xM

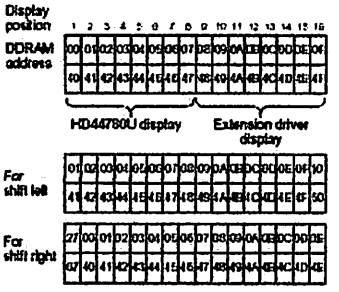
M1632 MODULE LCD 16 X 2 BARIS (M1632)

Deskripsi:

M1632 adalah merupakan modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya y rendah. Modul ini dilengkapi dengan mikrokontroler yang didisain khusus untuk mengendalikan L(Mikrokontroler HD44780 buatan Hitachi yang berfungsi sebagai pengendali LCD ini mempunyai CGR((Character Generator Read Only Memory), CGRAM (Character Generator Random Access Memory) DDRAM (Display Data Random Access Memory).

DDRAM

DDRAM adalah merupakan memori tempat karakter yang ditampilkan berada. Contoh, ur karakter 'A' atau 41H yang ditulis pada alamat 00, maka karakter tersebut akan tampil pada baris pert dan kolom pertama dari LCD. Apabila karakter tersebut ditulis di alamat 40, maka karakter tersebut a tampil pada baris kedua kolom pertama dari LCD.



Gambar 1
DDRAM M1632 (diambil dari data sheet HD44780)

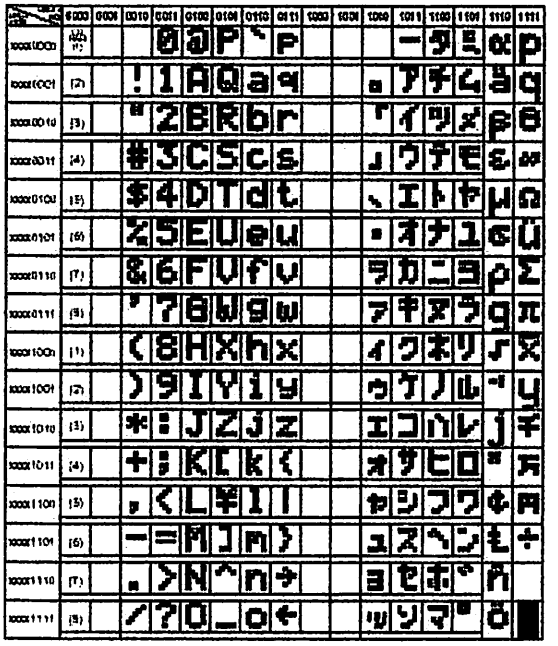
CGRAM

CGRAM adalah merupakan memori untuk menggambarkan **pola sebuah karakter** di m bentuk dari karakter dapat diubah-ubah sesuai keinginan. Namun memori ini akan hilang saat power su tidak aktif, sehingga pola karakter akan hilang.

CGROM

CGROM adalah merupakan memori untuk menggambarkan **pola sebuah karakter** di mana p tersebut sudah ditentukan secara permanen dari HD44780 sehingga pengguna tidak dapat mengubah l Namun karena ROM bersifat permanen, maka pola karakter tersebut tidak akan hilang walaupun po supply tidak aktif

Pada gambar 2, tampak terlihat pola-pola karakter yang tersimpan dalam lokasi-lokasi tert dalam CGROM. Pada saat HD44780 akan menampilkan data 41H yang tersimpan pada DDRAM, m HD44780 akan mengambil data di alamat 41H (0100 0001) yang ada pada CGROM yaitu pola karakter A

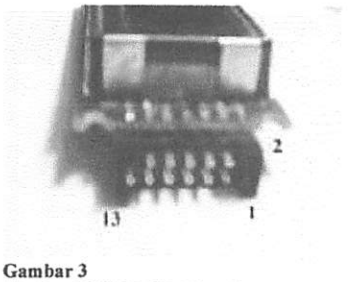


Gambar 2
Hubungan antara CGROM dan DDRAM (diambil dari data sheet HD44780)

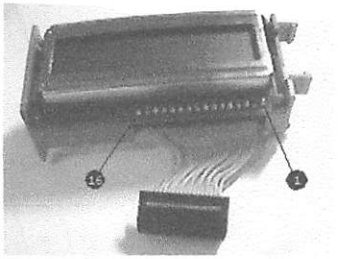
Pin Out

No	Nama Pin	Deskripsi
1	VCC	+5V
2	GND	0V
3	VEE	Tegangan Kontras LCD
4	RS	Register Select, 0 = Register Perintah, 1 = Register I
5	R/W	1 = Read, 0 = Write
6	E	Enable Clock LCD, logika 1 setiap kali pengiriman : pembacaan data
7	D0	Data Bus 0
8	D1	Data Bus 1
9	D2	Data Bus 2
10	D3	Data Bus 3
11	D4	Data Bus 4
12	D5	Data Bus 5
13	D6	Data Bus 6
14	D7	Data Bus 7
15	Anoda (Kabel coklat untuk LCD Hitachi)	Tegangan positif backlight
16	Katoda (Kabel merah untuk LCD Hitachi)	Tegangan negatif backlight

DELTA ELECTRONIC
http://www.delta-electronic.com



Gambar 3
Pin Out M1632 LCD Hitachi



Gambar 4
Pin Out LCD M1632 Standard

Register

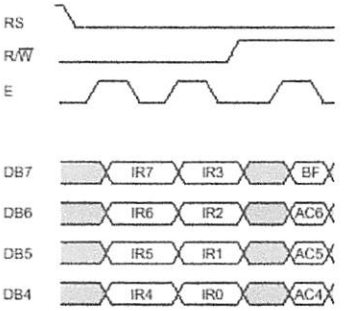
HD44780, mempunyai dua buah Register yang aksesnya diatur dengan menggunakan kaki Pada saat RS berlogika 0, maka register yang diakses adalah Register Perintah dan pada saat RS berlog 1, maka register yang diakses adalah Register Data

Register Perintah

Register ini adalah register di mana perintah-perintah dari mikrokontroler ke HD44780 pada : proses penulisan data atau tempat status dari HD44780 dapat dibaca pada saat pembacaan data.

Penulisan Data ke Register Perintah

Penulisan data ke Register Perintah dilakukan dengan tujuan mengatur tampilan LCD, inisiali dan mengatur Address Counter maupun Address Data. Gambar 5 menunjukkan proses penulisan data register perintah dengan menggunakan mode 4 bit interface. Kondisi RS berlogika 0 menunjukkan al data ke Register Perintah. RW berlogika 0 yang menunjukkan proses penulisan data akan dilakukan. Nib tinggi (bit 7 sampai bit 4) terlebih dahulu dikirimkan dengan diawali pulsa logika 1 pada E Cl Kemudian Nibble rendah (bit 3 sampai bit 0) dikirimkan dengan diawali pulsa logika 1 pada E Clock l Untuk mode 8 bit interface, proses penulisan dapat langsung dilakukan secara 8 bit (bit 7 ... bit 0) diawali sebuah pulsa logika 1 pada E Clock.



Gambar 5
Timing diagram Penulisan Data ke Register Perintah Mode 4 bit Interface

Tabel 1
Perintah-perintah M1632

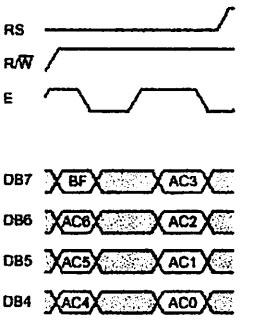
Perintah	D7	D6	D5	D4	D3	D2	D1	D0	Deskripsi
Hapus Display	0	0	0	0	0	0	0	1	Hapus Display dan DDRAM
Posisi Awal	0	0	0	0	0	0	1	X	Set Alamat DDRAM di 0
Set Mode	0	0	0	0	0	1	I/D	S	Atur arah pergeseran cursor dan displ

Display On/OFF	0	0	0	0	1	D	C	B	Atur display (D) On/OFF, cursor ON/OFF, Blinking (B)
Geser Cursor/Display	0	0	0	1	S/C	R/L	X	X	Geser Cursor atau display tanpa men alamat DDRAM
Set Fungsi	0	0	1	DL	N	F	X	X	Atur panjang data, jumlah baris tampil, dan font karakter
Set Alamat CGRAM	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Data dapat dibaca atau ditulis se alamat diatur
Set Alamat DDRAM	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Data dapat dibaca atau ditulis se alamat diatur

X = diabaikan
I/D 1=Increment, 0=Decrement
S 0=Display tidak geser
S/C 1=Display Shift, 0=Geser Cursor
R/L 1=Geser Kiri, 0=Geser Kanan
DL 1=8 bit, 0=4bit
N 1=2 baris, 0=1 baris
F 1=5x10, 0=5x8
D 0=Display OFF, 1=Display ON
C 0=Cursor OFF, 1=Cursor ON
B 0=Blinking OFF, 1=Blinking ON

Pembacaan Data dari Register Perintah

Proses pembacaan data pada register perintah biasa digunakan untuk melihat status busy dari L atau membaca Address Counter. RS diatur pada logika 0 untuk akses ke Register Perintah, R/W diatur p logika 1 yang menunjukkan proses pembacaan data. 4 bit nibble tinggi dibaca dengan diawali pulsa log 1 pada E Clock dan kemudian 4 bit nibble rendah dibaca dengan diawali pulsa logika 1 pada E Cl Untuk Mode 8 bit interface, pembacaan 8 bit (nibble tinggi dan rendah) dilakukan sekaligus dengan diav sebuah pulsa logika 1 pada E Clock.



Gambar 6
Timing Diagram Pembacaan Register Perintah Mode 4 bit Interface

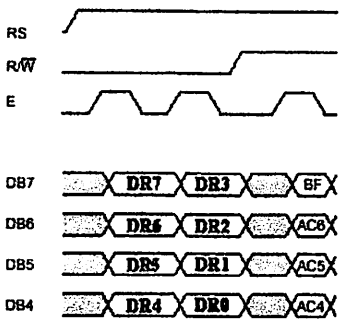
Register Data

Register ini adalah register di mana mikrokontroler dapat menuliskan atau membaca data ke dari DDRAM. Penulisan data pada register ini akan menempatkan data tersebut ke DDRAM sesuai den alamat yang telah diatur sebelumnya

Penulisan Data ke Register Data

Penulisan data pada Register Data dilakukan untuk mengirimkan data yang akan ditampilkan pa LCD. Proses diawali dengan adanya logika 1 pada RS yang menunjukkan akses ke Register Data, kondisi R/W diatur pada logika 0 yang menunjukkan proses penulisan data. Data 4 bit nibble tinggi (bit 7 hingga

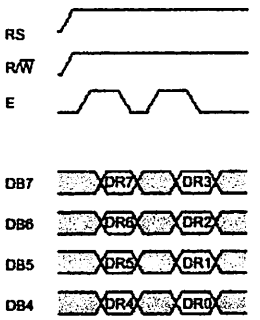
bit 4) dikirim dengan diawali pulsa logika 1 pada sinyal E Clock dan kemudian diikuti 4 bit nibble rendah (bit 3 hingga bit 0) yang juga diawali pulsa logika 1 pada sinyal E Clock.



Gambar 7
Timing Diagram Penulisan Data ke Register Data Mode 4 bit Interface

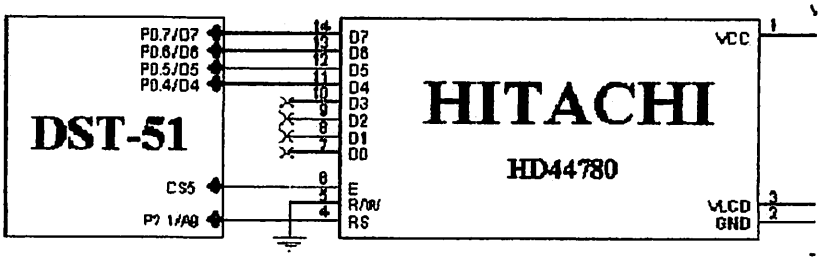
Pembacaan Data dari Register Data

Pembacaan data dari Register Data dilakukan untuk membaca kembali data yang tampil p LCD. Proses dilakukan dengan mengatur RS pada logika 1 yang menunjukkan adanya akses ke Regi Data. Kondisi R/W diatur pada logika tinggi yang menunjukkan adanya proses pembacaan data. Data 4 nibble tinggi (bit 7 hingga bit 4) dibaca dengan diawali adanya pulsa logika 1 pada E Clock dan dilanjut dengan data 4 bit nibble rendah (bit 3 hingga bit 0) yang juga diawali dengan pulsa logika 1 pada E Clock

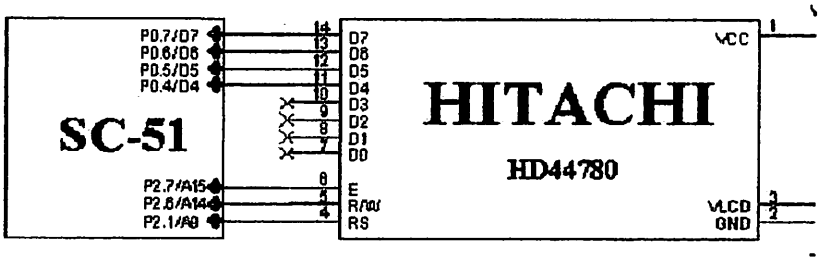


Gambar 8
Timing Diagram Pembacaan Data dari Register Data Mode 4 bit Interface

Antar muka LCD dengan mikrokontroler



Gambar 9
Antar muka dengan Modul DST-51



Gambar 10
Antar Muka dengan Modul SC-51 atau AT8951

Program
Rutin-rutin Program untuk DST-51 yang diassembly dengan ALDS atau ASM51
Rutin-rutin Program untuk SC-51/AT8951 yang diassembly dengan ALDS atau ASM51
Rutin delay yang diassembly dengan ALDS atau ASM51
Datasheet HD44780

HARRIS SEMICONDUCTOR

ICL232

+5V Powered Dual RS-232 Transmitter/Receiver

December 1993

Features

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

Applications

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

Description

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

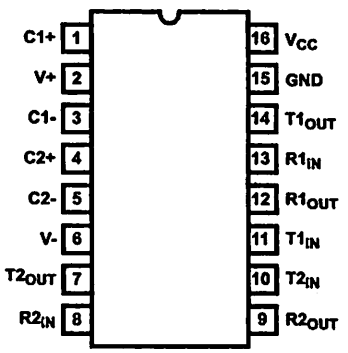
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

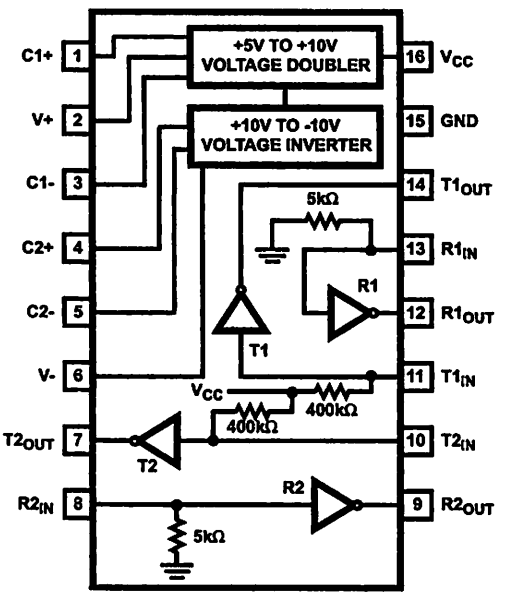
PART NUMBER	TEMPERATURE RANGE	PACKAGE
ICL232CPE	0°C to +70°C	16 Lead Plastic DIP
ICL232CJE	0°C to +70°C	16 Lead Ceramic DIP
ICL232CBE	0°C to +70°C	16 Lead SOIC (W)
ICL232IPE	-40°C to +85°C	16 Lead Plastic DIP
ICL232IJE	-40°C to +85°C	16 Lead Ceramic DIP
ICL232IBE	-40°C to +85°C	16 Lead SOIC (W)
ICL232MJE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

Pinouts

ICL232
(PDIP, CDIP, SOIC)
TOP VIEW



Functional Diagram



File Number 3020.2

Absolute Maximum Ratings

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

Thermal Information

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

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

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

Electrical Specifications

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

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

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

Typical Performance Curves

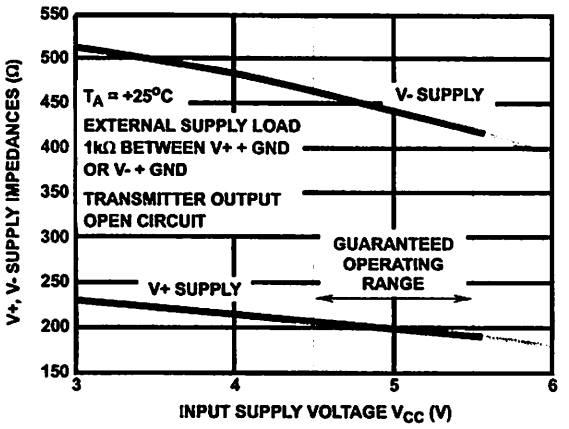


FIGURE 1. V+, V- OUTPUT IMPEDANCES vs V_{CC}

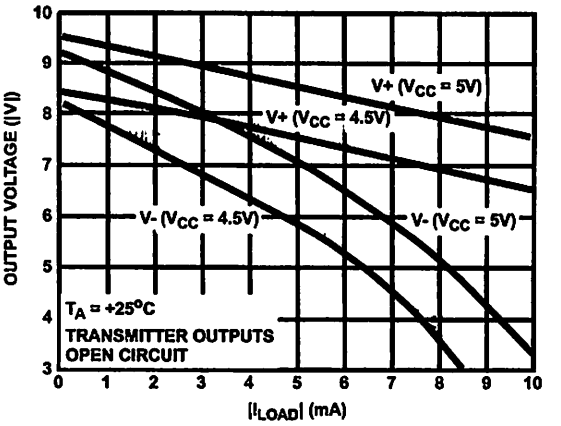


FIGURE 2. V+, V- OUTPUT VOLTAGES vs LOAD CURRENT

Pin Descriptions

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

Detailed Description

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

Voltage Converter

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

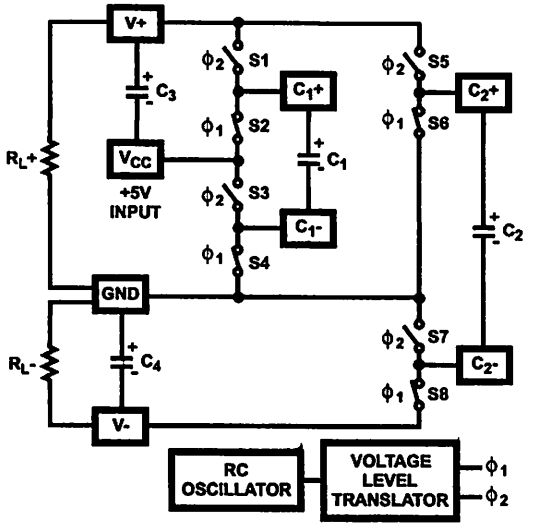
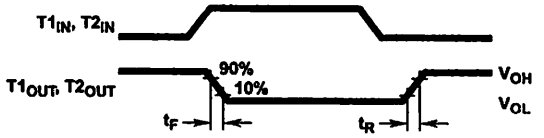


FIGURE 3. DUAL CHARGE PUMP

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



Instantaneous Slew Rate (SR) = $\frac{(0.8)(V_{OH} - V_{OL})}{t_R}$ or $\frac{(0.8)(V_{OL} - V_{OH})}{t_F}$

FIGURE 4. SLEW RATE DEFINITION

Transmitters

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

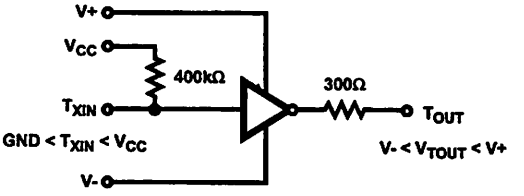


FIGURE 5. TRANSMITTER

Receivers

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

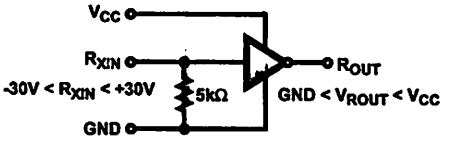
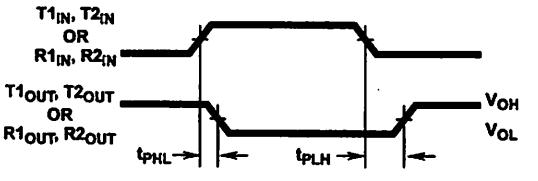


FIGURE 6. RECEIVER



Average Propagation Delay = $\frac{t_{PHL} + t_{PLH}}{2}$

FIGURE 7. PROPAGATION DELAY DEFINITION

Test Circuits

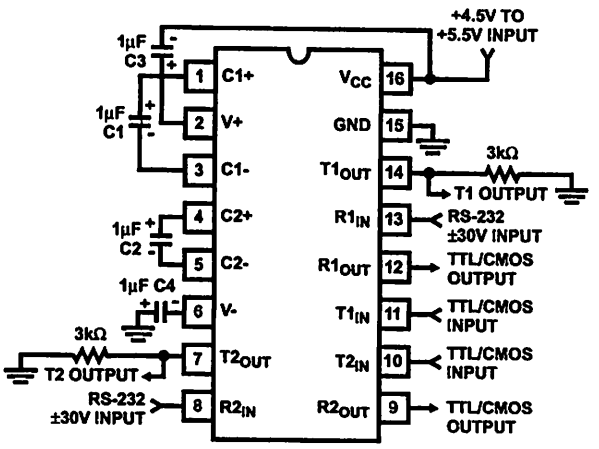


FIGURE 8. GENERAL TEST CIRCUIT

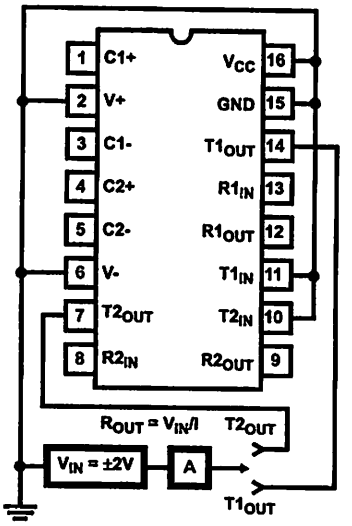


FIGURE 9. POWER-OFF SOURCE RESISTANCE CONFIGURATION

Applications

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

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

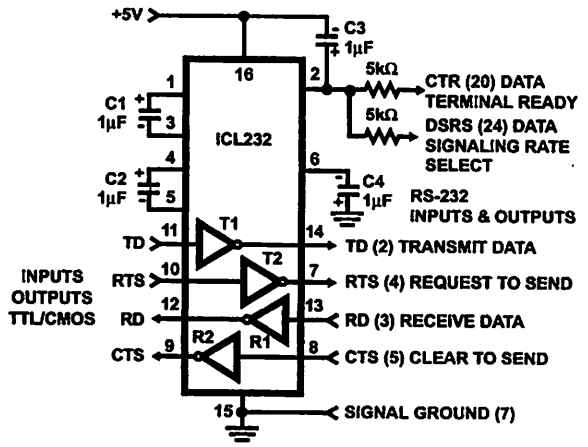


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

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

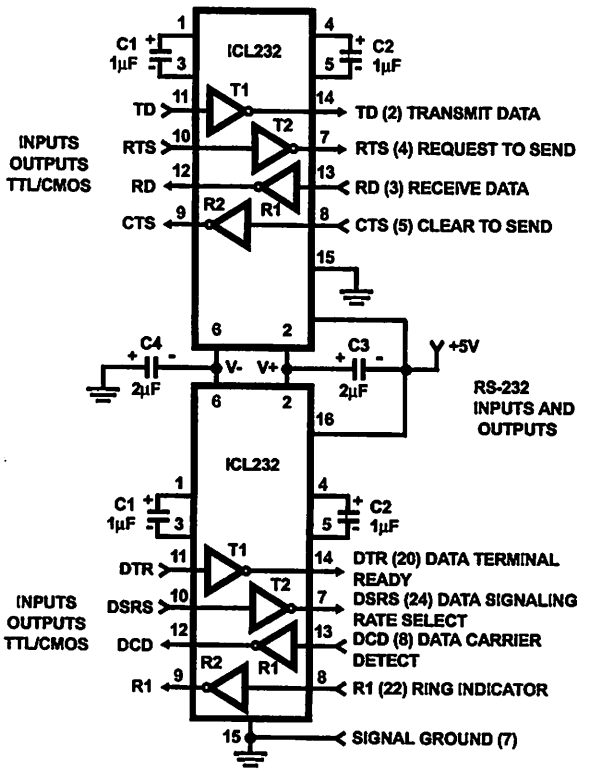


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS

19-0122; Rev 5; 2/96

MAXIM

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

General Description

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

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

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

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

Applications

Low-Power RS-485 Transceivers

Low-Power RS-422 Transceivers

Level Translators

Transceivers for EMI-Sensitive Applications

Industrial-Control Local Area Networks

Features

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

Ordering Information

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

Ordering Information continued at end of data sheet.

* *Contact factory for dice specifications.*

Selection Table

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

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC}) 12V
Control Input Voltage ($\overline{RE}$, DE) ..... -0.5V to (V_{CC} + 0.5V)
Driver Input Voltage (DI) ..... -0.5V to (V_{CC} + 0.5V)
Driver Output Voltage (A, B) ..... -8V to +12.5V
Receiver Input Voltage (A, B) ..... -8V to +12.5V
Receiver Output Voltage (RO) ..... -0.5V to (V_{CC} +0.5V)
Continuous Power Dissipation (T_A = +70°C)
8-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 727mW
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) ..800mW
8-Pin SO (derate 5.88mW/°C above +70°C) 471mW
14-Pin SO (derate 8.33mW/°C above +70°C) 667mW
8-Pin µMAX (derate 4.1mW/°C above +70°C) 830mW
8-Pin CERDIP (derate 8.00mW/°C above +70°C) 640mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
Operating Temperature Ranges
MAX4_ _C_ _/MAX1487C_ A ..... 0°C to +70°C
MAX4_ _E_ _/MAX1487E_ A ..... -40°C to +85°C
MAX4_ _MJ_/MAX1487MJA ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

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

DC ELECTRICAL CHARACTERISTICS

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

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

DC ELECTRICAL CHARACTERISTICS (continued)

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

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

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

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

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

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

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

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

SWITCHING CHARACTERISTICS—MAX483, MAX487/MAX488/MAX489

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

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

NOTES FOR ELECTRICAL/SWITCHING CHARACTERISTICS

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

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

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

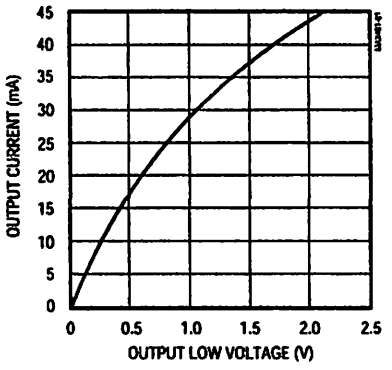
Note 4: Applies to peak current. See *Typical Operating Characteristics.*

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

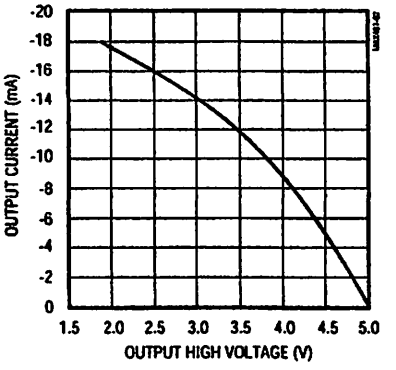
Typical Operating Characteristics

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

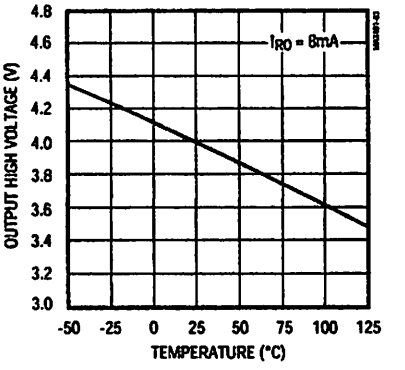
OUTPUT CURRENT vs. RECEIVER OUTPUT LOW VOLTAGE



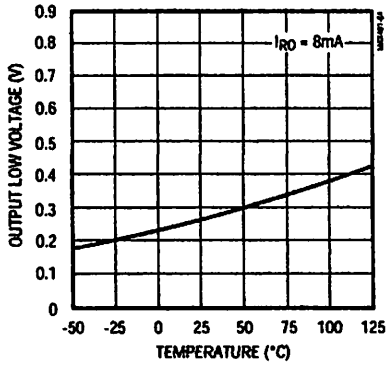
OUTPUT CURRENT vs. RECEIVER OUTPUT HIGH VOLTAGE



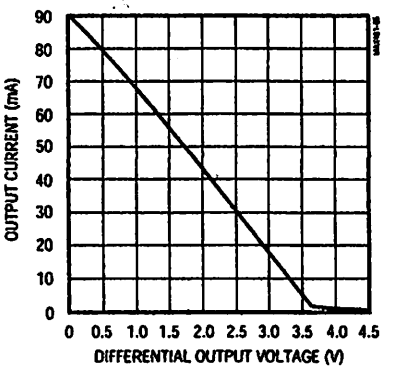
RECEIVER OUTPUT HIGH VOLTAGE vs. TEMPERATURE



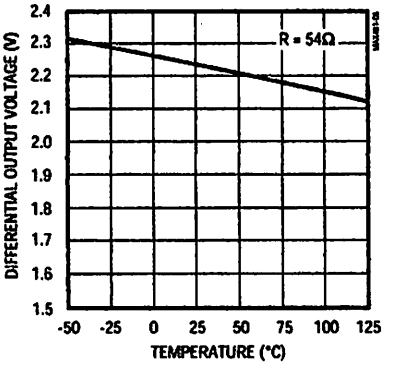
RECEIVER OUTPUT LOW VOLTAGE vs. TEMPERATURE



DRIVER OUTPUT CURRENT vs. DIFFERENTIAL OUTPUT VOLTAGE



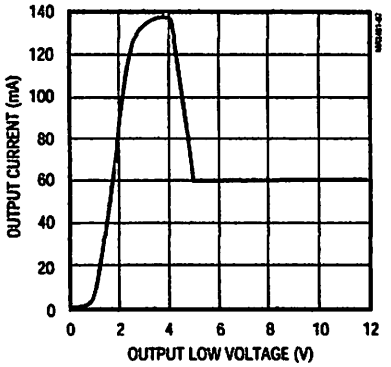
DRIVER DIFFERENTIAL OUTPUT VOLTAGE vs. TEMPERATURE



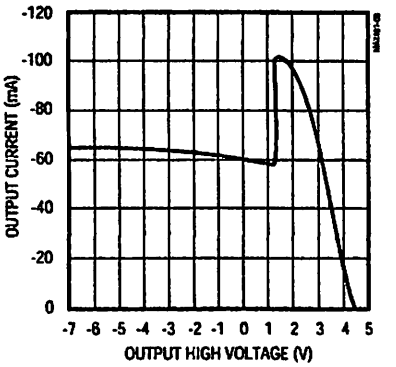
Typical Operating Characteristics (continued)

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

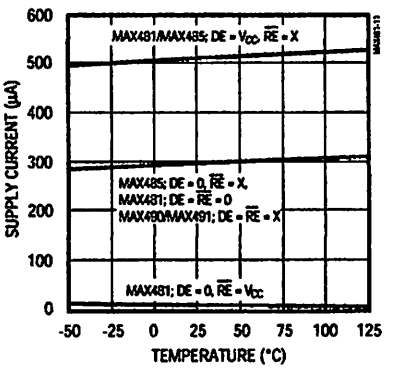
OUTPUT CURRENT vs.
DRIVER OUTPUT LOW VOLTAGE



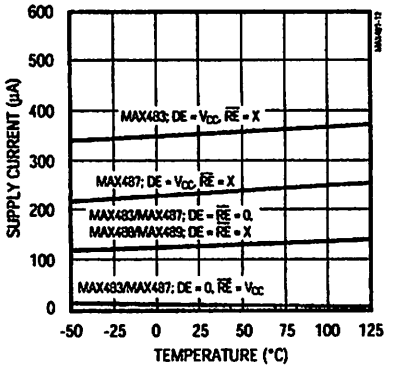
OUTPUT CURRENT vs.
DRIVER OUTPUT HIGH VOLTAGE



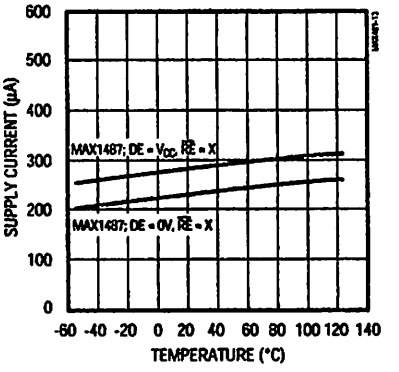
MAX481/MAX485/MAX490/MAX491
SUPPLY CURRENT vs. TEMPERATURE



MAX483/MAX487–MAX489
SUPPLY CURRENT vs. TEMPERATURE



MAX1487
SUPPLY CURRENT vs. TEMPERATURE



Pin Description

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

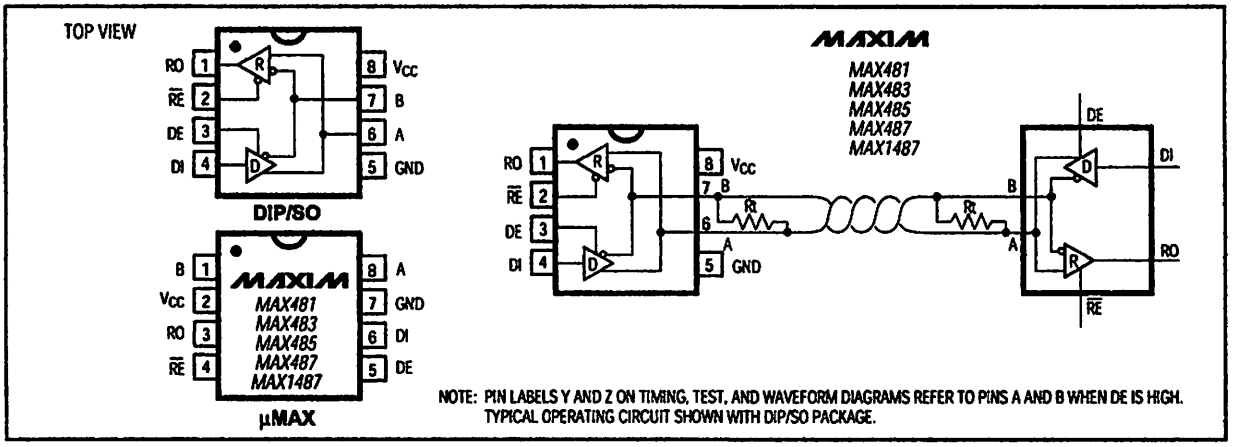


Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit

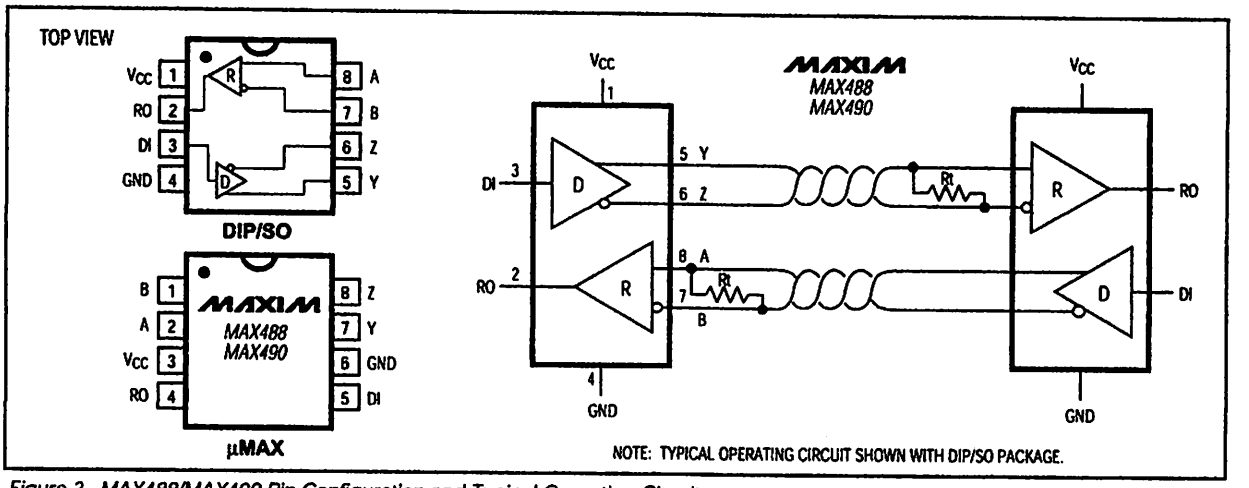


Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

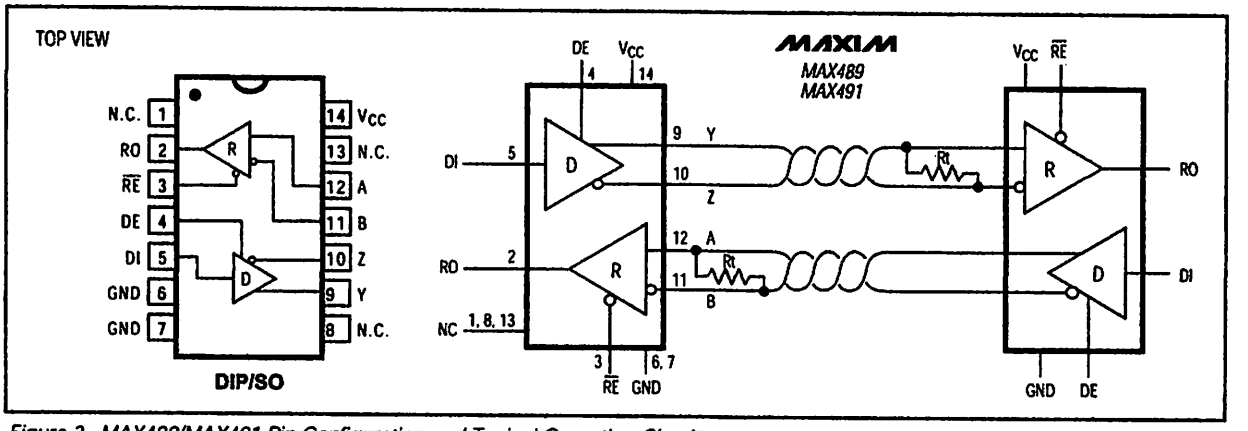


Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

Applications Information

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

MAX487/MAX1487: 128 Transceivers on the Bus

The 48kΩ, 1/4-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12kΩ input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488–MAX491 have standard 12kΩ Receiver Input impedance.

Test Circuits

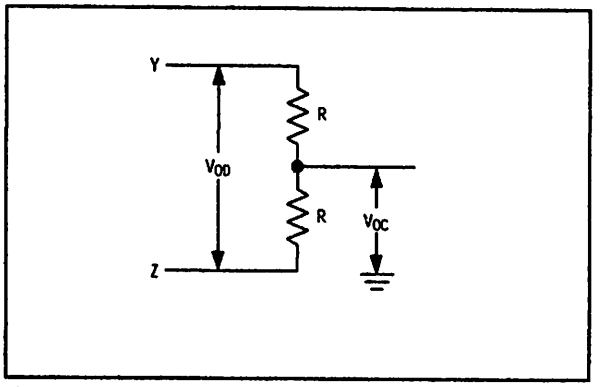


Figure 4. Driver DC Test Load

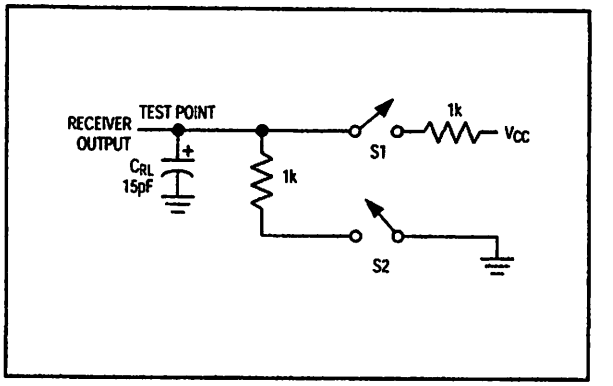


Figure 5. Receiver Timing Test Load

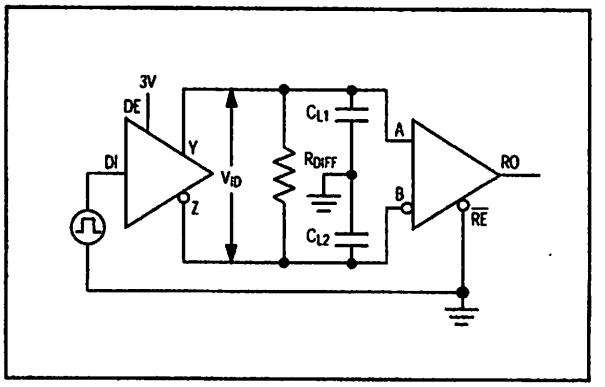


Figure 6. Driver/Receiver Timing Test Circuit

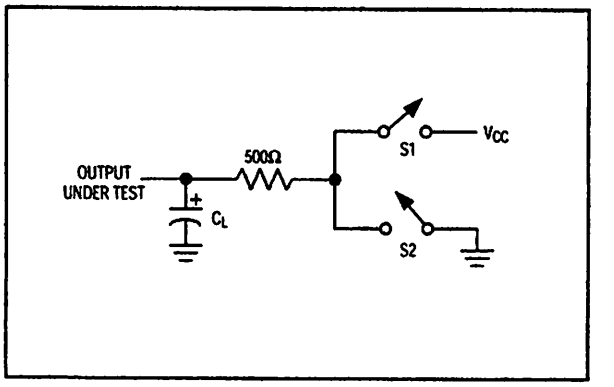


Figure 7. Driver Timing Test Load

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

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

Switching Waveforms

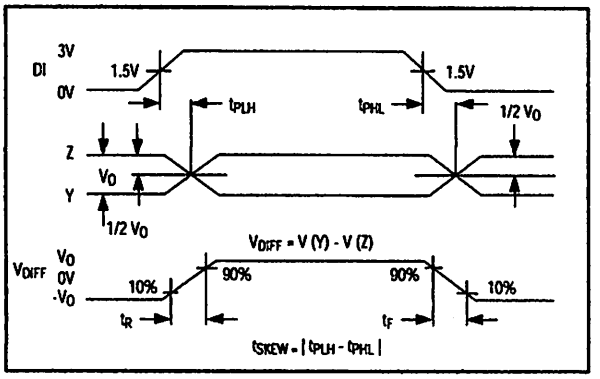


Figure 8. Driver Propagation Delays

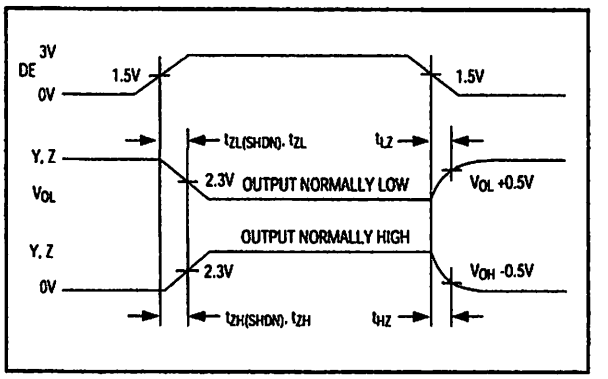


Figure 9. Driver Enable and Disable Times (except MAX488 and MAX490)

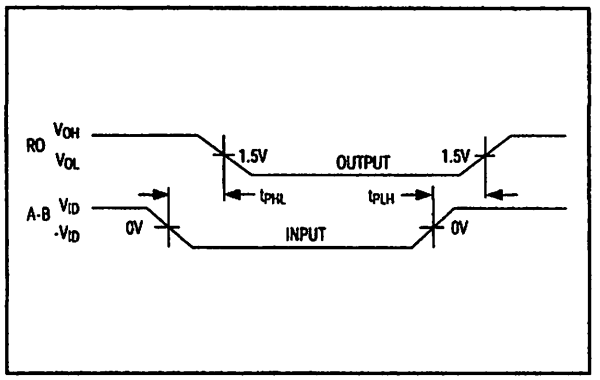


Figure 10. Receiver Propagation Delays

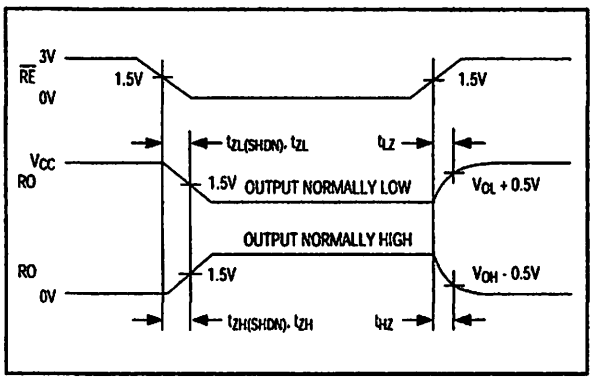


Figure 11. Receiver Enable and Disable Times (except MAX488 and MAX490)

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

Table 1. Transmitting

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

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

Table 2. Receiving

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

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

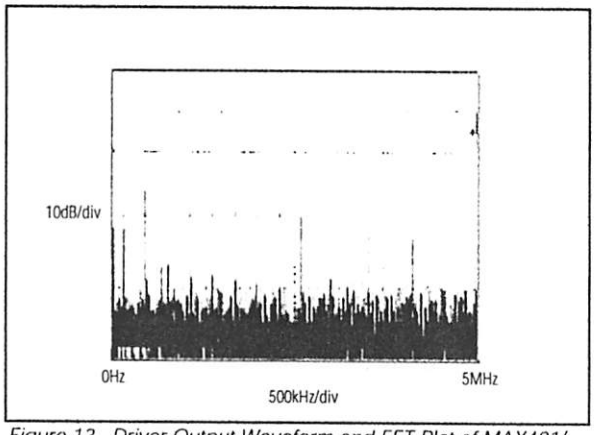


Figure 12. Driver Output Waveform and FFT Plot of MAX481/MAX485/MAX490/MAX491/MAX1487 Transmitting a 150kHz Signal

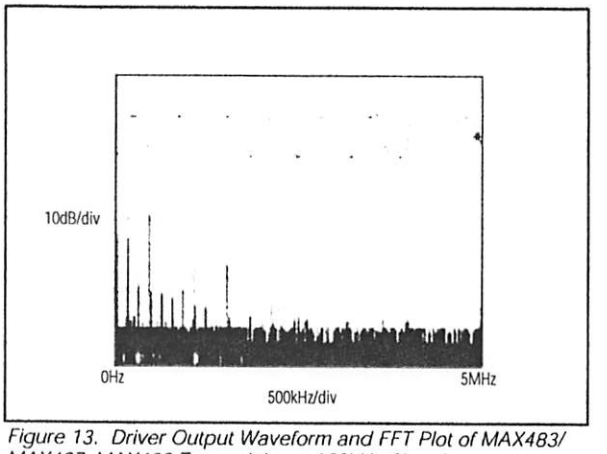


Figure 13. Driver Output Waveform and FFT Plot of MAX483/MAX487–MAX489 Transmitting a 150kHz Signal

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

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

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

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

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

Driver Output Protection

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

Propagation Delay

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

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

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

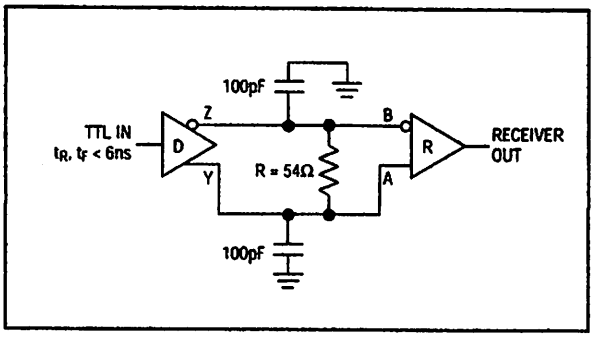


Figure 14. Receiver Propagation Delay Test Circuit

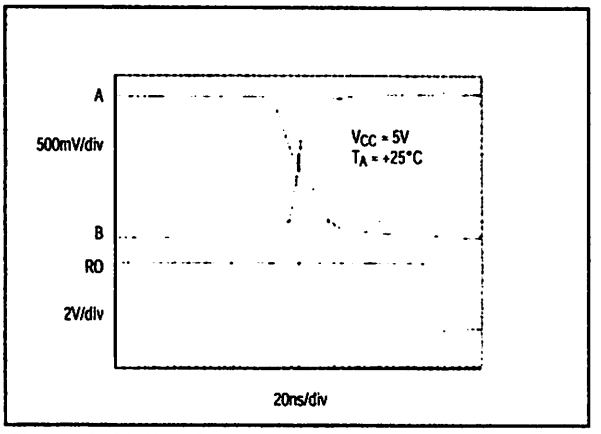


Figure 15. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PHL}

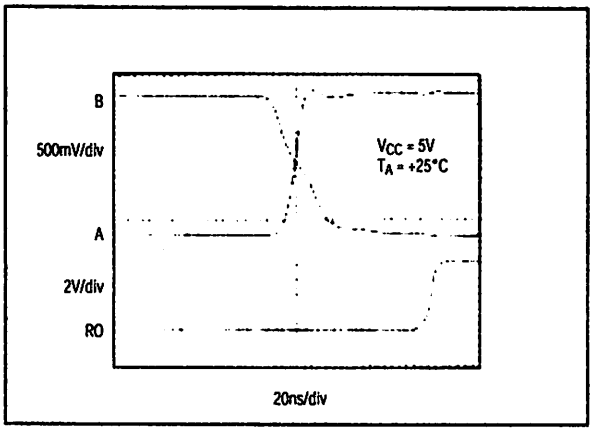


Figure 16. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PLH}

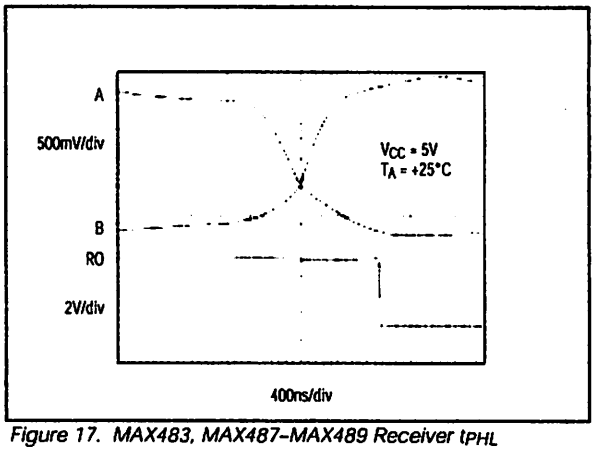


Figure 17. MAX483, MAX487–MAX489 Receiver t_{PHL}

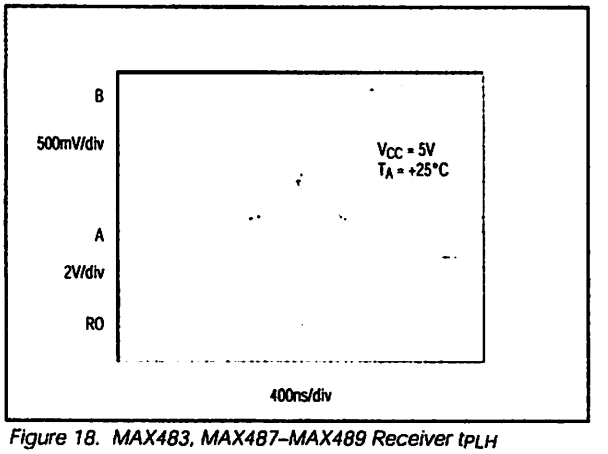


Figure 18. MAX483, MAX487–MAX489 Receiver t_{PLH}

Line Length vs. Data Rate

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

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

Typical Applications

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

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

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

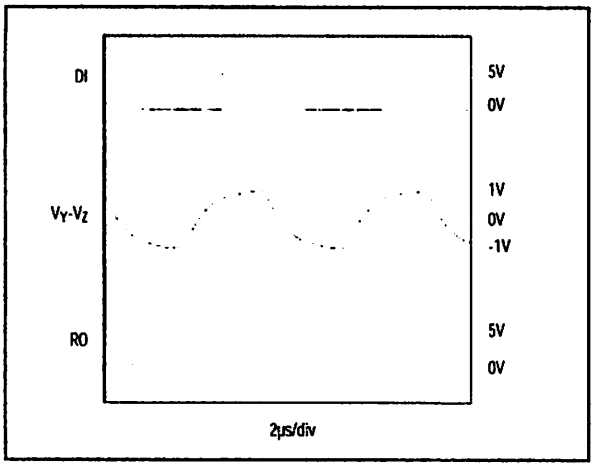


Figure 19. MAX481/MAX485/MAX490/MAX491/MAX1487 System Differential Voltage at 110kHz Driving 4000ft of Cable

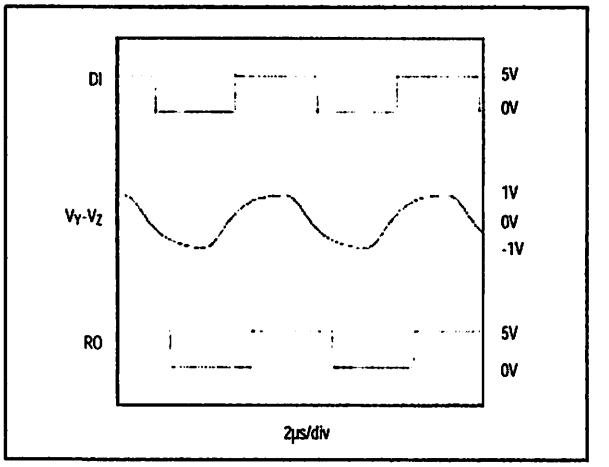


Figure 20. MAX483, MAX487–MAX489 System Differential Voltage at 110kHz Driving 4000ft of Cable

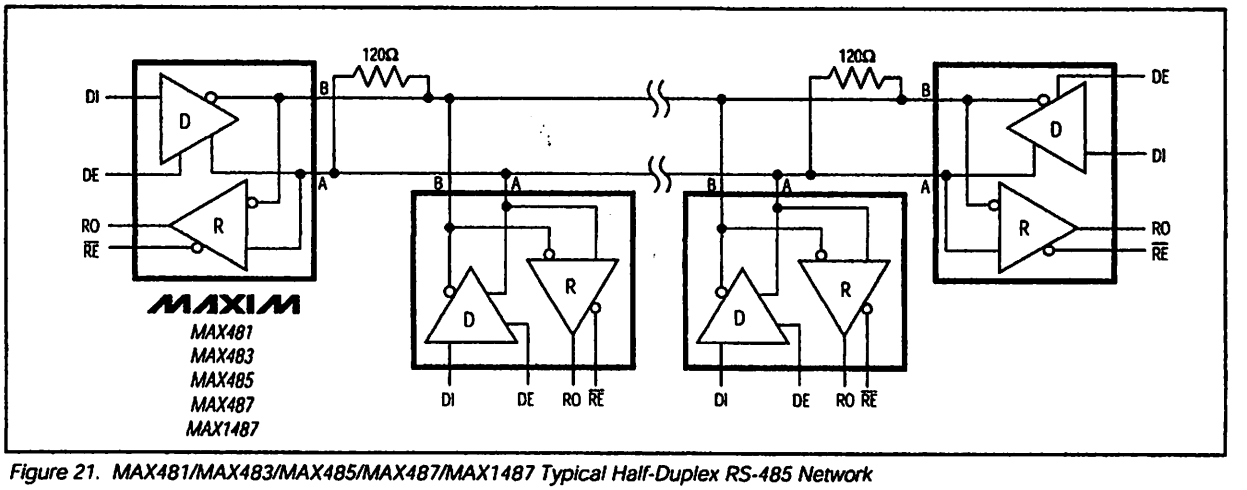
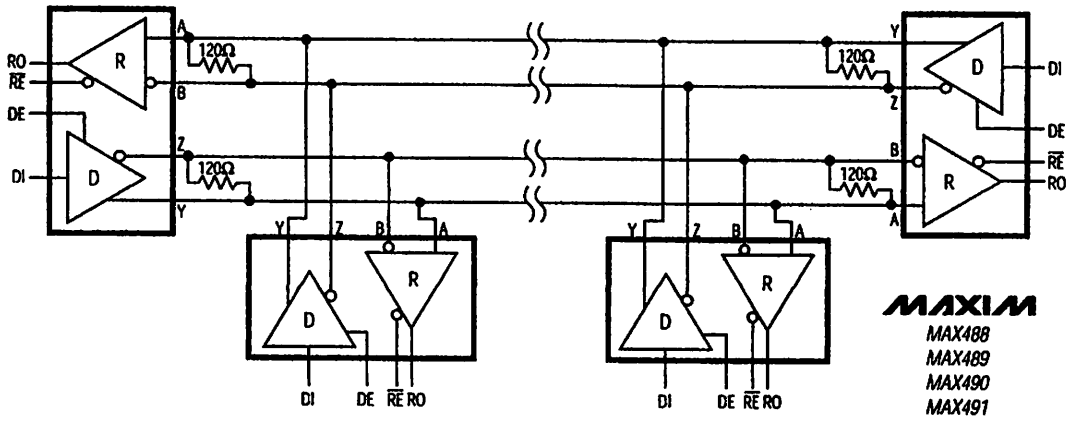
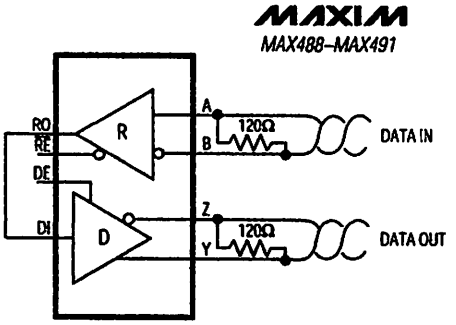


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network



NOTE: $\overline{RE}$ AND DE ON MAX489/MAX491 ONLY.

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



NOTE: $\overline{RE}$ AND DE ON MAX489/MAX491 ONLY.

Figure 23. Line Repeater for MAX488–MAX491

Isolated RS-485

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

Ordering Information (continued)

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

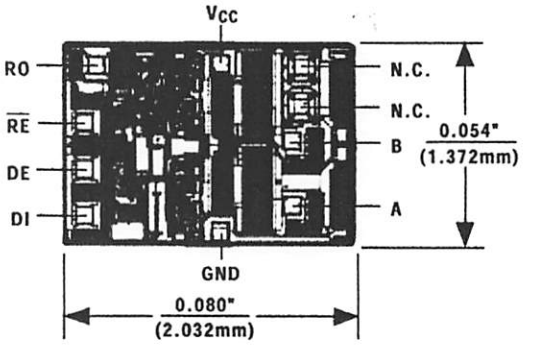
Ordering Information (continued)

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

* *Contact factory for dice specifications.*

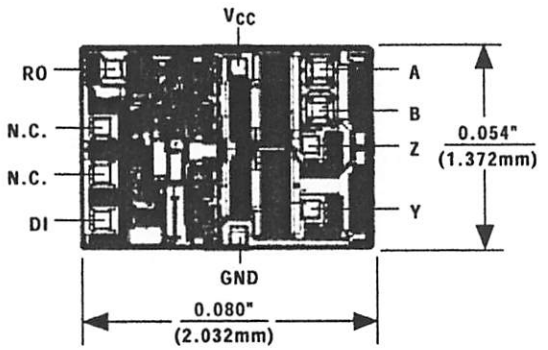
Chip Topographies

MAX481/MAX483/MAX485/MAX487/MAX1487

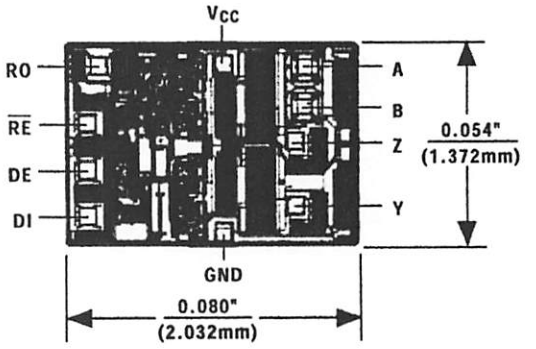


Chip Topographies (continued)

MAX488/MAX490

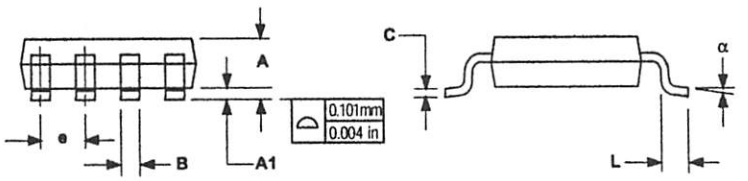


MAX489/MAX491



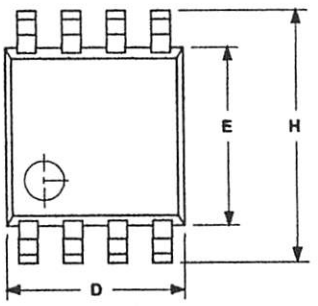
TRANSISTOR COUNT: 248
SUBSTRATE CONNECTED TO GND

Package Information



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

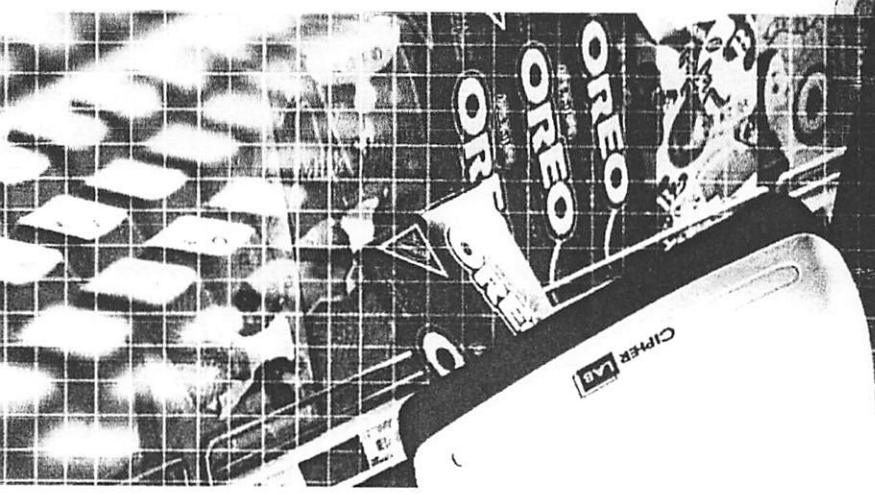
21-0036D



**8-PIN μMAX
MICROMAX SMALL-OUTLINE
PACKAGE**

1000
CCD BARCODE SCANNER

Best Price
High Performance



Features & Benefits
- Built-in Keyboard Emulator for laptop
- Excelling in densities starting at 0.125 and lower
- 16bit CMOS CPU
- 100 scans/second
- Ergonomic Design
- OEM Welcome

Accessories
Power Supply for RS232 Model

Interface
- Keyboard-AT/PS2
- Serial - RS232
- USB

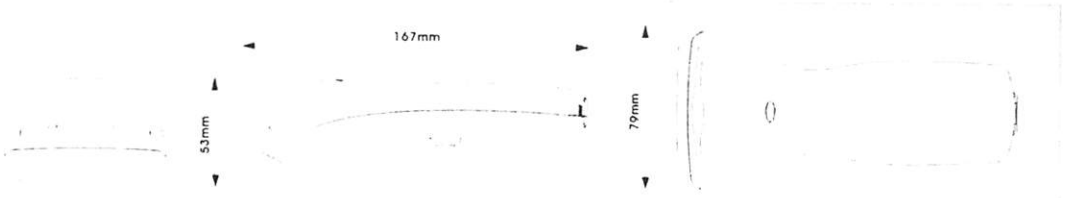
CIPHER LAB

1000 CCD BARCODE SCANNER

Specifications

Models		1000K (K-B Wedge)	1000R (RS-232)	1000U (USB)
Electrical	Voltage	5V +/- 10% **(at scanner end)**		
	Power Consumption			
	Operation (Maximum)	85 mA	90 mA	110mA
	(Typical)	75 mA	80 mA	100mA
Environmental	Temperature	0°C~50°C (operating) / -20°C~60°C (storage)		
	Humidity	20%~90% non-condensing (operating) / 10%~95% non-condensing (storage)		
	Shock Resistance	100cm drops onto a concrete surface		
Physical	Weight	250g	260g	265g
	Housing	ABS plastic		
	Switch	Rubber switch		
	Color	Ivory		
Optical	Optical Sensor	2048 Pixel		
	Light Source	Red LED (660 nm)		
	Resolution	0.125mm		
	Depth of Field	0~10mm		
	Width of Field	67mm		
	PCS	0.45 or more		
	Scanning Engine	100 scans/sec		
	Scanning Angle	Front 40° ; rear 70°		
		Left & right 25° to 25°		
	Ambient Light Rejection	1200 lux(Direct sun-light)		
		2500 lux(Fluorescent light)		
Features	Keyboard Auto-detection	Built -in keyboard emulator for laptop (Phony Keyboard included)		
	Symbology Supported	Code 39/Full ASCII, Codabar, Italy Pharma Code, Industrial 2 of 5, Interleave 2 of 5, Matrix 2 of 5, Code 93, Code 128, EAN 128, MSI, EAN 8 & 13, UPCA & E, Plessey		
	Decoder	16-bit CMOS CPU		
	Indicator Buzzer	Frequency programmable and can be disabled.		
	LED	Red Color for Good Read; Green Color for Status		
	Programmable Features	Code ID, Prefix Code, Postfix Code, Length Code, Barcode type, Keyboard layout, ISBN/ISSN/UK Plessey conversion, Inter-character delay, Scanning mode, Terminal type, Language type, Time out, Baud rate, Data bit, Full/Half duplex.		
	Data Editing Mode	Three editing formats are supported. The following conditions can be programmed: code type, data length and location. Data can be divided into 6 Matching string fields. Five additional fields can be programmed for each editing format. User programmable field transmission sequence.		
	Languages Support	English(US), English(UK), French, Italian, Belgian , Norwegian/Swedish, Spanish, Portuguese, German.		
Safety & Regulatory Approvals		FCC class-A, CE, C-tick		
Warranty	1 Year			

Dimensions



TÜV SÜD
DIN EN ISO 9001

Headquarter
Syntech Information Co., Ltd.
5F, No. 196-3, Ta-Tung Road
Sec. 3, Hsi-Chih
Taipei Hsien, Taiwan
TEL: +886 2 8647 3300
FAX: +886 2 8647 1155
www.cipherlab.com.tw

USA
Syntech West Inc.
3404 57th Street NW Suite B
Gig Harbor, WA 98335
U.S.A.
TEL: +1 253 858 7906
FAX: +1 253 858 7713
www.cipherlab.com

:: Drivers, Firmware & Manuals
:: FAQ
□ Technology Overviews
:: Product Spec Sheets
:: Discontinued Products
:: Web Seminars
:: White Papers

Support Request Form
Warranty Policy
RMA Request Form

RS-232 SERIAL COMMUNICATION OVERVIEW

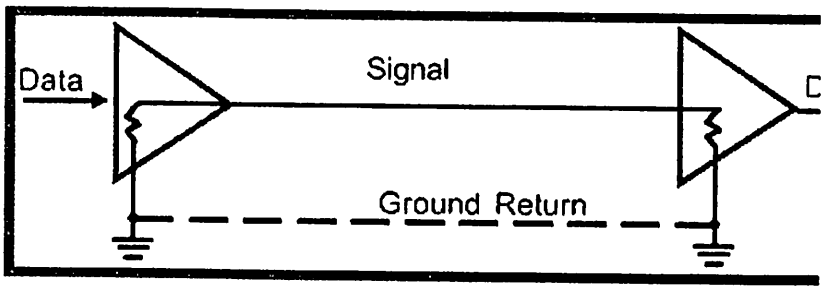
RS-232 SERIAL COMMUNICATION OVERVIEW

Data is typically transmitted between two points either asynchronously or synchronously. However, what happens to the data between point A and point another discussion. This in-between area is a cable made up of wires through v data travels. Specifications for this cable were developed to maximize signal in (to limit the possible degradation that could be caused by external noise or gro shifts). Quatech supports three major protocols for asynchronous and synchror communication: RS-232, RS-422 and RS-485. Differences between the three a highlighted in the chart below, and in the sections which follow.

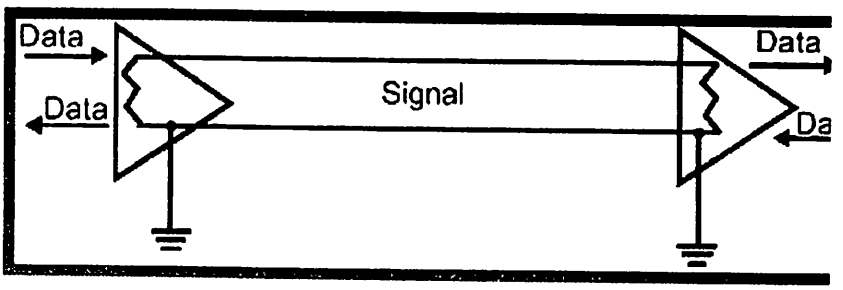
	RS-232	RS-422	RS-485
Mode of Operation	single ended	differential	differentia
Drivers per Line	1	1	32
Receivers per Line	1	10	32
Maximum Cable Length	50 feet	4000 feet	4000 feet
Maximum Data Rate	20 kbps	10 Mbps	10 Mbps
Driver Output Maximum Voltage	±25V	-0.25 to +6V	-7 to +12V
Driver Output Signal Level (loaded)	±5V	±2V	±1.5V
Driver Output Signal Level (unloaded)	±15V	±5V	±5V
Driver Load Impedance	3kΩ to 7kΩ	100kΩ	54kΩ
Max. Driver Output Current (Power on)	n/a	n/a	±100µA
Max. Driver Output Current (Power off)	$V^{MAX/300\Omega}$	±100µA	±100µA
Slew Rate	30V/µs max.	n/a	n/a
Receiver Input Voltage Range	±15V	-7V to +7V	-7V to +12V
Receiver Input Sensitivity	±3V	±200mV	±200mV
Receiver Input Resistance	3kΩ to 7kΩ	4kΩ	12kΩ

THE FIRST STANDARD

RS-232 was introduced in 1960, and is currently the most widely used commur protocol. It is simple, inexpensive to implement, and though relatively slow, it i than adequate for most simple serial communication devices such as keyboard: mice. RS-232 is a single-ended data transmission system, which means that it single wire for data transmission. (Since useful communication is generally two two-wire system is employed, one to transmit and one to receive.) Signals are processed by determining whether they are positive or negative when compare ground. Because signals traveling this single wire are vulnerable to degradatior 232 systems are recommended for communication over short distances (up to and at relatively slow data rates (up to 20 kbps). However, in practice, these li be exceeded.



Unbalanced Single-Wire Transmission



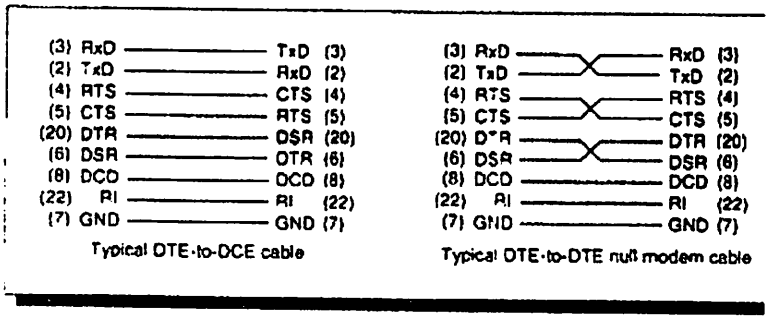
Unbalanced Two-Wire Transmission

DTE AND DCE: SERIAL COMMUNICATION PARTNERS

A typical system is made up of two types of device, data communication equipı (DCE) and data terminal equipment (DTE). Typically DTE is defined as the communication source, and DCE is defined as the device that provides a comm channel between two DTE-type devices.

In order for DCE and DTE devices to communicate with each other, two wires r used--one for transmission and the other for reception--and both devices must the same wire for the same purpose. Should this happen, nothing would get communicated because both devices would be talking on the same line and list a line on which nothing is transmitted. To solve this problem, DTE and DCE de\ have complementary pinouts to allow terminals and modems to be connected c using a one-to-one cable (see Figure below).

Situations arise in which no DCE device is needed, such as a desktop computer communicating with a laptop. In this case, a null modem cable or modem elimi cable is used to connect the two DTE devices. This cable effectively changes th the second device uses for transmission and reception (see below-left) to assur both sides can communicate.



DTE and DCE Cable Configurations

CONNECTORS: THE COMMUNICATION CONDUIT

As discussed in the asynchronous and synchronous communication sections, th more to a serial transmission than simply data. Additional information must be transmitted between both ends of a conversation to make sure that when point sends, point B is listening, and vice versa. This is called handshaking. These handshaking lines take up considerable space on a serial connector.

Connectors for RS-232 devices are always constructed using standard assignm the wires in a RS-232 cable in order to maintain the DTE-DCE relationship desc above. These connectors can be modular (phone jack) or male D-shell (pins co in a rough "D" shape which fit into sockets on the device). The signal assignme the RS-232 wires follow. The descriptions reference the standard DB-9 and DB- connectors used by most serial devices, however the definitions are applicable modular connectors as well (though in modular connections not all of the hand: signals are implemented). See ESC-100M for Quatech modular adapters for PC include modular connector pinout information.

VIEW FIGURE

RS-232 SIGNAL DESCRIPTIONS

Abbreviations used in the following definitions of RS-232 wire functions will be throughout Quatech's website.

DTR: Data Terminal Ready--Used by a DTE to signal that it is plugged in and a to begin communication.

DSR: Data Set Ready--Sister signal to DTR, it is used by the DCE to indicate it to begin communication.

CTS: Clear to Send--Used by DCE to signal it is available to send data, and use response to a RTS request for data.

RTS: Request to Send--Used by a DTE to indicate that it wants to send data. A multi-drop network, used to turn carrier on the modem on and off.

DCD: Data Carrier Detect--Used by a DCE to indicate to the DTE that it has rec carrier signal from the modem and that real data is being transmitted.

RI: Ring Indicator--Used by DCE modem to tell the DTE that the phone is ringi that data will be forthcoming.

TxD: Transmit Data--This wire is used for sending data.

RxD: Receive Data--This line is used for receiving data.

GND: Signal Ground--This pin is the same for DTE and DCE devices, and it pro return path for both data and hand-shake signals.

SYNCHRONOUS COMMUNICATION ONLY

TxCLK: Transmit Signal Element Timing--Used by DTE to provide DCE with tim information for data transfer.

RxCLK: Receiver Signal Element Timing--Used by DCE to provide DTE with tim information for data transfer.

LLBK: Local Loopback--Used by DTE to make sure the local transmit and recei interface is functioning properly.

RLBK: Remote Loopback--Used by DTE to make sure a remote transmit/receiv interface is functioning properly.

TEST MODE: Test Mode--Used by DCE to indicate that it is testing itself in resp a local or remote loopback signal from a DTE.

- Data Communication Overview
- Bus Options (ISA, MCA, PCI, cPCI, PCMCIA,CompactFlash, USB, Ethern
- Serial Communication (Asynchronous, Synchronous, Isochronous; RS-2 RS-422, RS-485)
- Parallel Communication
- Bluetooth Communication
- GPS (Global Positioning System) Communication
- Application Examples
- Product Selection Guide

Introduction to Serial Communications

All IBM PC and compatible computers are typically equipped with two serial ports and one parallel port. Although these two types of ports are used for communicating with external devices, they work in different ways.

A parallel port sends and receives data eight bits at a time over 8 separate wires. This allows data to be transferred very quickly; however, the cable required is more bulky because of the number of individual wires it must contain. Parallel ports are typically used to connect a PC to a printer and are rarely used for much else. A serial port sends and receives data one bit at a time over one wire. While it takes eight times as long to transfer each byte of data this way, only a few wires are required. In fact, two-way (full duplex) communications is possible with only three separate wires - one to send, one to receive, and a common signal ground wire.

--

Bi-Directional Communications

The serial port on your PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receiving data. Some types of serial devices support only one-way communications and therefore use only two wires in the cable - the transmit line and the signal ground.
Back to Top

--

Communicating by Bits

Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number you have selected. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate. Almost all devices transmit data using either 7 or 8 databits.

Notice that when only 7 data bits are employed, you cannot send ASCII values greater than 127. Likewise, using 5 bits limits the highest possible value to 31. After the data has been transmitted, a stop bit is sent. A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit's duration. Stop bits can be 1, 1.5, or 2 bit periods in length.
Back to Top

--

The Parity Bit

Besides the synchronization provided by the use of start and stop bits, an additional bit called a parity bit may optionally be transmitted along with the data. A parity bit affords a small amount of error checking, to help detect data corruption that might occur during transmission. You can choose either even parity, odd parity, mark parity, space parity or none at all. When even or odd parity is being used, the number of marks (logical 1 bits) in each data byte are counted, and a single bit is transmitted following the data bits to indicate whether the number of 1 bits just sent is even or odd.

For example, when even parity is chosen, the parity bit is transmitted with a value of 0 if the number of preceding marks is an even number. For the binary value of 0110 0011 the parity bit would be 0. If even parity were in effect and the binary number 1101 0110 were sent, then the parity bit would be 1. Odd parity is just the opposite, and the parity bit is 0 when the number of mark bits in the preceding word is an odd number. Parity error checking is very rudimentary. While it will tell you if there is a single bit error in the character, it doesn't show which bit was received in error. Also, if an even number of bits are in error then the parity bit would not reflect any error at all.

Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Since these two parity options serve no useful purpose whatsoever, they are almost never used.
Back to Top

--

RS-232C

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.
Back to Top

--

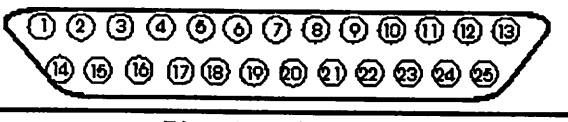
DCE and DTE Devices

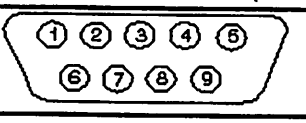
Two terms you should be familiar with are DTE and DCE. DTE stands for Data Terminal Equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices are usually DCE devices.

If you have trouble keeping the two straight then replace the term "DTE device" with "your PC" and the term "DCE device" with "remote device" in the following discussion.

The RS-232 standard states that DTE devices use a 25-pin male connector, and DCE devices use a 25-pin female connector. You can therefore connect a DTE device to a DCE using a straight pin-for-pin connection. However, to connect two like devices, you must instead use a null modem cable. Null modem cables cross the transmit and receive lines in the cable, and are discussed later in this chapter. The listing below shows the connections and signal directions for both 25 and 9-pin connectors.

	25 Pin Connector on a DTE device (PC connection)

Male RS232 DB25	
Pin Number	**Direction of signal:**
1	Protective Ground
2	Transmitted Data (TD) Outgoing Data (from a DTE to a DCE)
3	Received Data (RD) Incoming Data (from a DCE to a DTE)
4	Request To Send (RTS) Outgoing flow control signal controlled by DTE
5	Clear To Send (CTS) Incoming flow control signal controlled by DCE
6	Data Set Ready (DSR) Incoming handshaking signal controlled by DCE
7	Signal Ground Common reference voltage
8	Carrier Detect (CD) Incoming signal from a modem
20	Data Terminal Ready (DTR) Outgoing handshaking signal controlled by DTE
22	Ring Indicator (RI) Incoming signal from a modem

	9 Pin Connector on a DTE device (PC connection)
Male RS232 DB9	
Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

The TD (transmit data) wire is the one through which data from a DTE device is transmitted to a DCE device. This name can be deceiving, because this wire is used by a DCE device to receive its data. The TD line is kept in a mark condition by the DTE device when it is idle. The RD (receive data) wire is the one on which data is received by a DTE device, and the DCE device keeps this line in a mark condition when idle.

RTS stands for **Request To Send**. This line and the CTS line are used when "hardware flow control" is enabled in both the DTE and DCE devices. The DTE device puts this line in a mark condition to tell the remote device that it is ready and able to receive data. If the DTE device is not able to receive data (typically because its receive buffer is almost full), it will put this line in the space condition as a signal to the DCE to stop sending data. When the DTE device is ready to receive more data (i.e. after data has been removed from its receive buffer), it will place this line back in the mark condition. The complement of the RTS wire is CTS, which stands for Clear To Send. The DCE device puts this line in a mark condition to tell the DTE device that it is ready to receive the data. Likewise, if the DCE device is unable to receive data, it will place this line in the space condition. Together, these two lines make up what is called RTS/CTS or "hardware" flow control. The Software Wedge supports this type of flow control, as well as Xon/XOff or "software" flow control. Software flow control uses special control characters transmitted from one device to another to tell the other device to stop or start sending data. With software flow control the RTS and CTS lines are not used.

DTR stands for **Data Terminal Ready**. Its intended function is very similar to the RTS line. DSR (Data Set Ready) is the companion to DTR in the same way that CTS is to RTS. Some serial devices use DTR and DSR as signals to simply confirm that a device is connected and is turned on. The Software Wedge sets DTR to the mark state when the serial port is opened and leaves it in that state until the port is closed. The DTR and DSR lines were originally designed to provide an alternate method of hardware handshaking. It would be pointless to use both RTS/CTS and DTR/DSR for flow control signals at the same time. Because of this, DTR and DSR are rarely used for flow control.

CD stands for **Carrier Detect**. Carrier Detect is used by a modem to signal that it has a made a connection with another modem, or has detected a carrier tone.

The last remaining line is **RI** or **Ring Indicator**. A modem toggles the state of this line when an incoming call rings your phone.

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.
Back to Top

--

9 to 25 Pin Adapters

The following table shows the connections inside a standard 9 pin to 25 pin adapter.

9-Pin Connector	25 Pin Connector
Pin 1 DCD	Pin 8 DCD
Pin 2 RD	Pin 3 RD
Pin 3 TD	Pin 2 TD
Pin 4 DTR	Pin 20 DTR
Pin 5 GND	Pin 7 GND
Pin 6 DSR	Pin 6 DSR
Pin 7 RTS	Pin 4 RTS
Pin 8 CTS	Pin 5 CTS
Pin 9 RI	Pin 22 RI

Back to Top

--

Baud vs. Bits per Second

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (BPS). If you connect two serial devices together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 19200 BPS, then the line is also changing states 19200 times per second. But when considering modems, this isn't the case.

Because modems transfer signals over a telephone line, the baud rate is actually limited to a maximum of 2400 baud. This is a physical restriction of the lines provided by the phone company. The increased data throughput achieved with 9600 or higher baud modems is accomplished by using sophisticated phase modulation, and data compression techniques.
Back to Top

--

Cables, Null Modems, and Gender Changers

In a perfect world, all serial ports on every computer would be DTE devices with 25-pin male "D" connectors. All other devices would be DCE devices with 25-pin female connectors. This would allow you to use a cable in which each pin on one end of the cable is connected to the same pin on the other end. Unfortunately, we don't live in a perfect world. Serial ports use both 9 and 25 pins, many devices can be configured as either DTE or DCE, and - as in the case of many data collection devices - may use completely non-standard or proprietary pin-outs. Because of this lack of standardization, special cables called null modem cables, gender changers and custom made cables are often required.
Back to Top

--

Cables Lengths

The RS-232C standard imposes a cable length limit of 50 feet. You can usually ignore this "standard", since a cable can be as long as 10000 feet at baud rates up to 19200 if you use a high quality, well shielded cable. The external environment has a large effect on lengths for unshielded cables. In electrically noisy environments, even very short cables can pick up stray signals. The following chart offers some reasonable guidelines for 24 gauge wire under typical conditions. You can greatly extend the cable length by using additional devices like optical isolators and signal boosters. Optical isolators use LEDs and Photo Diodes to isolate each line in a serial cable including the signal ground. Any electrical noise affects all lines in the optically isolated cable equally - including the signal ground line. This causes the voltages on the signal lines relative to the signal ground line to reflect the true voltage of the signal and thus canceling out the effect of any noise signals.

Baud Rate	Shielded Cable Length	Unshielded Cable Length
110	5000	1000
300	4000	1000
1200	3000	500
2400	2000	500
4800	500	250
9600	250	100

--

Gender Changers

A problem you may encounter is having two connectors of the same gender that must be connected. You can purchase gender changers at any computer or office supply store for under $5.

Note: The parallel port on a PC uses a 25 pin female connector which sometimes causes confusion because it looks just like a serial port except that it has the wrong gender. Both 9 and 25 pin serial ports on a PC will always have a male connector.
Back to Top

--

Null Modem Cables and Null Modem Adapters

If you connect two DTE devices (or two DCE devices) using a straight RS232 cable, then the transmit line on each device will be connected to the transmit line on the other device and the receive lines will likewise be connected to each other. A Null Modem cable or Null Modem

adapter simply crosses the receive and transmit lines so that transmit on one end is connected to receive on the other end and vice versa. In addition to transmit and receive, DTR & DSR, as well as RTS & CTS are also crossed in a Null Modem connection.

Null Modem adapters are available at most computer and office supply stores for under $5.
Back to Top

--

Synchronous and Asynchronous Communications

There are two basic types of serial communications, synchronous and asynchronous. With synchronous communications, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. Even when data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each character that is sent is either actual data or an idle character. Synchronous communications allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required. The serial ports on IBM-style PCs are asynchronous devices and therefore only support asynchronous serial communications.

Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits causes asynchronous communication to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters.

An asynchronous line that is idle is identified with a value of 1 (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0 (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to be sent.
Back to Top

See Also:

Sartorius Cable Pinouts
Pinouts for other common types of cable
Connecting Serial Devices Via USB

QUICK REFERENCE FOR RS485, RS422, RS232 AND RS423

HOME PAGE (rs485.com)

INTRODUCTION

Line drivers and receivers are commonly used to exchange data between two or more points (nodes) on a network. Reliable data communications can be difficult in the presence of induced noise, ground level differences, impedance mismatches, failure to effectively bias for idle line conditions, and other hazards associated with installation of a network.

The connection between two or more elements (drivers and receivers) should be considered a transmission line if the rise and/or fall time is less than half the time for the signal to travel from the transmitter to the receiver.

Standards have been developed to insure compatibility between units provided by different manufacturers, and to allow for reasonable success in transferring data over specified distances and/or data rates. The Electronics Industry Association (EIA) has produced standards for RS485, RS422, RS232, and RS423 that deal with data communications. Suggestions are often made to deal with practical problems that might be encountered in a typical network. EIA standards where previously marked with the prefix "RS" to indicate recommended standard; however, the standards are now generally indicated as "EIA" standards to identify the standards organization. While the standards bring uniformity to data communications, many areas are not specifically covered and remain as "gray areas" for the user to discover (usually during installation) on his own.

SINGLE-ENDED DATA TRANSMISSION

Electronic data communications between elements will generally fall into two broad categories: single-ended and differential. RS232 (single-ended) was introduced in 1962, and despite rumors for its early demise, has remained widely used through the industry. The specification allows for data transmission from one transmitter to one receiver at relatively slow data rates (up to 20K bits/second) and short distances (up to 50Ft. @ the maximum data rate).

Independent channels are established for two-way (full-duplex) communications. The RS232 signals are represented by voltage levels with respect to a system common (power / logic ground). The "idle" state (MARK) has the signal level negative with respect to common, and the "active" state (SPACE) has the signal level positive with respect to common. RS232 has numerous handshaking lines (primarily used with modems), and also specifies a communications protocol. In general if you are not connected to a modem the handshaking lines can present a lot of problems if not disabled in software or accounted for in the hardware (loop-back or pulled-up). RTS (Request to send) does have some utility in certain applications. RS423 is another single ended specification with enhanced operation over RS232; however, it has not been widely used in the industry.

DIFFERENTIAL DATA TRANSMISSION

When communicating at high data rates, or over long distances in real world environments, single-ended methods are often inadequate. Differential data transmission (balanced differential signal) offers superior performance in most applications. Differential signals can help nullify the effects of ground shifts and induced noise signals that can appear as common mode voltages on a network.

RS422 (differential) was designed for greater distances and higher Baud rates than RS232. In its simplest form, a pair of converters from RS232 to RS422 (and back again) can be used to form an "RS232 extension cord." Data rates of up to 100K bits / second and distances up to 4000 Ft. can be accommodated with RS422. RS422 is also specified for multi-drop (party-line) applications where only one driver is connected to, and transmits on, a "bus" of up to 10 receivers.

While a multi-drop "type" application has many desirable advantages, RS422 devices cannot be used to construct a truly multi-point network. A true multi-point network consists of multiple drivers and receivers connected on a single bus, where any node can transmit or receive data.

"Quasi" multi-drop networks (4-wire) are often constructed using RS422 devices. These networks are often used in a half-duplex mode, where a single master in a system sends a command to one of several "slave" devices on a network. Typically one device (node) is addressed by the host computer and a response is received from that device. Systems of this type (4-wire, half-duplex) are often constructed to avoid "data collision" (bus contention) problems on a multi-drop network (more about solving this problem on a two-wire network in a moment).

RS485 meets the requirements for a truly multi-point communications network, and the standard specifies up to 32 drivers and 32 receivers on a single (2-wire) bus. With the introduction of "automatic" repeaters and high-impedance drivers / receivers this "limitation" can be extended to hundreds (or even thousands) of nodes on a network. RS485 extends the common mode range for both drivers and receivers in the "tri-state" mode and with power off. Also, RS485 drivers are able to withstand "data collisions" (bus contention) problems and bus fault conditions.

To solve the "data collision" problem often present in multi-drop networks hardware units (converters, repeaters, micro-processor controls) can be constructed to remain in a receive mode until they are ready to transmit data. Single master systems (many other communications schemes are available) offer a straight forward and simple means of avoiding "data collisions" in a typical 2-wire, half-duplex, multi-drop system. The master initiates a communications request to a "slave node" by addressing that unit. The hardware detects the start-bit of the transmission and automatically enables (on the fly) the RS485 transmitter. Once a character is sent the hardware reverts back into a receive mode in about 1-2 microseconds (at least with R.E. Smith converters, repeaters, and remote I/O boards).

Any number of characters can be sent, and the transmitter will automatically re-trigger with each new character (or in many cases a "bit-oriented" timing scheme is used in conjunction with network biasing for fully automatic operation, including any Baud rate and/or any communications specification, eg. 9600,N,8,1). Once a "slave" unit is addressed it is able to respond immediately because of the fast transmitter turn-off time of the automatic device. It is NOT necessary to introduce long delays in a network to avoid "data collisions." Because delays are NOT required, networks can be constructed, that will utilize the data communications bandwidth with up to 100% through put.

Below are the specifications for RS232, RS423, RS422, and RS485. Please give us a call at **513-874-4796** if further information is required. We have solutions to most problems that are encountered in this area. Any comments and/or corrections would be appreciated.

Thanks, Ron Smith

SPECIFICATIONS		RS232	RS423	RS422	RS485
Mode of Operation		SINGLE -ENDED	SINGLE -ENDED	DIFFERENTIAL	DIFFERENTIAL
Total Number of Drivers and Receivers on One Line (One driver active at a time for RS485 networks)		1 DRIVER 1 RECVR	1 DRIVER 10 RECVR	1 DRIVER 10 RECVR	32 DRIVER 32 RECVR
Maximum Cable Length		50 FT.	4000 FT.	4000 FT.	4000 FT.
Maximum Data Rate (40ft. - 4000ft. for RS422/RS485)		20kb/s	100kb/s	10Mb/s-100Kb/s	10Mb/s-100Kb/s
Maximum Driver Output Voltage		+/-25V	+/-6V	-0.25V to +6V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	Loaded	+/-5V to +/-15V	+/-3.6V	+/-2.0V	+/-1.5V
Driver Output Signal Level (Unloaded Max)	Unloaded	+/-25V	+/-6V	+/-6V	+/-6V
Driver Load Impedance (Ohms)		3k to 7k	>=450	100	54
Max. Driver Current in High Z State	Power On	N/A	N/A	N/A	+/-100uA
Max. Driver Current in High Z State	Power Off	+/-6mA @ +/-2v	+/-100uA	+/-100uA	+/-100uA
Slew Rate (Max.)		30V/uS	Adjustable	N/A	N/A
Receiver Input Voltage Range		+/-15V	+/-12V	-10V to +10V	-7V to +12V
Receiver Input Sensitivity		+/-3V	+/-200mV	+/-200mV	+/-200mV
Receiver Input Resistance (Ohms), (1 Standard Load for RS485)		3k to 7k	4k min.	4k min.	>=12k

Please call us at: 513-874-4796

Contact Information:
R.E. Smith, 4311 R. E. Smith Dr., Hamilton, Ohio 45011
513-874-4796 Phone, 513-874-1236 Fax., rs485.com

Back **Go Back (Previous view)**

Print **Print Page (Select Landscape)**

HOME PAGE (rs485.com)