TUGAS AKHIR

PERANCANGAN DAN PEMBUATAN KASIR MENU CEPAT SAJI DI RESTORAN DENGAN AUTO DEBET MENGGUNAKAN KARTU RFID BERBASIS MIKROKONTROLLER RENESAS R8C/13 TINY

Disusun Oleh :

FERDY RIDWAN DINATA

01.17.030

KONSENTRASI TEKNIK ELEKTRONIKA

JURUSAN TEKNIK ELEKTRONIKA S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL

MALANG

2007

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN KASIR MENU CEPAT SAJI DI RESTORAN DENGAN AUTO DEBET MENGGUNAKAN KARTU (RFID) BERBASIS MIKROKONTROLER RENESAS R8C/13 TINY

SKRIPSI

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

Disusun oleh :
Nama : Ferdy Ridwan Dinata
Nim : 01.17.030

Dosen Pembimbing I	Dosen Pembimbing II
Joseph Dedy Irawan,ST,MT	**I Komang Somawirata,ST,MT**
NIP.131574847	**NIP.10301000361**

Mengetahui
Ketua Jurusan Teknik Elektro S-1

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

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

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNIK INDUSTRI
JURUSAN TEKNIK ELEKTRO

BERITA ACARA UJIAN SKRIPSI

FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	: Ferdy Ridwan Dinata
NIM	: 01.17.030
Jurusan	: Teknik Elektro S1
Konsentrasi	: Teknik Elektronika
Judul Skripsi	: Perancangan dan Pembuatan Kasir Menu Cepat Saji Dengan Auto Debet Menggunakan Kartu RFID Berbasis Renesas R8C/13 Tiny

Dipertahankan dihadapan team penguji Skripsi jenjang Sarjana (S1) pada :

Hari	: Jum'at
Tanggal	: 16 Maret 2007
Dengan Nilai	: A (82,5)

PANITIA UJIAN SKRIPSI

KETUA

Ir. Mochtar Asroni, MSME
NIP.Y 1018100036

SEKRETARIS

Ir Yudi Limpraptono, MT
NIP.Y 1039500274

ANGGOTA PENGUJI

PENGUJI I

Ir Widodo Pudji M, MT
NIP.Y 1028700171

PENGUJI II

DR. Cahyo Crysdian, Msc
NIP. 1030400412

ABSTRAKSI

PERENCANAAN DAN PEMBUATAN KASIR MENU CEPAT SAJI DIRESTORAN DENGAN MENGGUNAKAN KARTU RFID BERBASIS MIKROKONTROLLER RENESAS R8C/13 TINY

Ferdy Ridwan Dinata

Nim : 01.17.030 Tenik Elektro/Elektronika S-1

Dosen Pembimbing I : Yoseph Deddy Irawan.ST,MT

Dosen Pembimbing II : I Komang Somawirata.ST,MT

Kata Kunci : Mikrokontroller Renesas R8C/13 Tiny, RFID reader, Tag RFID, LCD, PC.

RFID adalah Proses identifikasi obyek dengan menggunakan frekuensi, tranmisi radio. RFID menggunakan frekuensi radio untuk informasi dari sebuah devais kecil yang disebut tag atau tranponder (trnsmiter+responder). Tag RFID akan mengenali dirisendiri ketika mendeteksi sinyal dari devais yang kompatibel, yaitu RFID (RFID reader) dengan e kisaran pembacaan 8cm serta bekerja pada frekuensi 125khz.

Tag berfungsi sebagai inputan data dari pelanggan, data tersebut akan dikenali oleh RFID Reader dan sebagai tanda beep akan mengeluarkan bunyi setelah tag dikenali oleh reader dan data akan dikirim ke PC untuk diproses,maka LCD akan menampilkan menu yang tersedia.

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat-Mu Ya Allah yang telah memberikan rahmat dan hidayah-Nya, sehingga dapat menyelesaikan skripsi yang berjudul "Perancangan Dan Pembuatan Kasir menu cepat saji di restoran auto debet dengan menggunakan kartu RFID berbasis Renesas R8C Tiny R5F21134FP" ini dengan lancar. Skripsi ini merupakan persyaratan kelulusan Studi di Jurusan Teknik Elektro S-1 Konsentrasi Teknik Elektronika ITN Malang dan untuk mencapai gelar Sarjana Teknik.

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

1. Bapak Ir. F. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S-1.
2. Bapak Joseph Deddy Irawan.ST,MT. selaku Dosen Pembimbing I dan Ka. Laboratorium Elektronika Digital.
3. Bapak I Komang Somawirata.ST,MT. selaku Dosen Pembimbing II.
4. Ayah dan Ibu serta saudara-saudara kami yang telah memberikan do'a restu, dorongan, semangat, dan biaya.
5. Rekan-rekan Instruktur di Laboratorium Perancangan Elektronika.
6. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penyusun telah berusaha semaksimal mungkin dan meyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu

penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

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

Malang, Maret 2007

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

BAB III PERANCANGAN DAN PEMBUATAN ALAT

BAB IV PENGUJIAN ALAT

DAFTAR TABEL

BAB II LANDASAN TEORI

BAB III PERANCANGAN DAN PEMBUATAN ALAT

BAB IV PENGUJIAN ALAT

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangn jaman maka aktivitas kehidupan manusia semakin sibuk dan padat. Banyak orang yang menginginkan segala sesusatu dilakukan dengan cepat dan efisien untuk menghemat waktu. Untuk pergi ke restoran misalnya.

Selain itu pula apabila pada suatu restoran pada saat kasirnya terlihat antrian panjang, maka mahasiswa yang akan memesan tentunya memilih restoran yang lain yang tidak sampai mengantri dalam pelayanannya. Dengan demikian restoran tersebut akan kehilangan *customers*.

Melihat perkembangan teknologi komunikasi dan elektronika maka penulis berencana suatu alat pelayanan kasir restoran cepat saji dengan Sistem Debet menggunakan RFID Berbasis Renesas tiny R8C. Alat ini adalah alat kasir elektronik yang diletakkan pada masing-masing meja kasir utama dan dapur sehingga setiap *customers* tidak perlu untuk berdiri mengantri untuk memesan menu. Untuk system pembayarannya setiap *customers* cukup dengan membeli kartu debet dengan nilai rupiah tertentu yang nantinya kartu debet tersebut digunakan sebagai akses dalam pelayanan menu di masing-masing meja kasir Dengan adanya alat ini nantinya pelanggan dapat langsung memilih dan menunggu pesanannya.

1.2. Rumusan Masalah

Pembuatan alat ini meliputi perencanaan hardware dan software.

- Bagaimana merencanakan dan membuat alat pelayanan kasir restoran cepat saji dengan system debet menggunakan RFID berbasis RENESAS R8C/13 TINY.
- Bagaimana merancang system konunikasi serial RS-232 yang terhubung dengan kasir utama.
- Bagaimana merancang keseluruhan system pengontrol oleh RENESAS R8C/13 TINY.

1.3. Batasan Masalah

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

- Perencanaan dan perealisasian system RENESAS R8C/13 TINY.
- Tidak membahas tentang catu daya.
- Tidak membahas tentang frekuensi RFID.
- Tidak memakai pin/password.

1.4. Tujuan

Tujuan dari penulisan tugas akhir perencanaan dan pembuatan alat Pelayanan kasir restoran cepat saji dengan sistem Kartu Debet menggunakan Kartu RFID Berbasis RENESAS R8C/13 TINY adalah :

- Memberi kemudahan *customers* dalam memilih menu di restoran dan menunggu pesanannya.

- Memberi kemudahan bagi mahasiswa atau pelanggan dengan tidak perlu membawa uang tunai cukup dengan kartu debet RFID.

1.5 Metodologi

Untuk mewujudkan alat Pelayanan kasir cepat saji dengan sistem Kartu Debet menggunakan Kartu RFID Berbasis RENESAS R8C/13 TINY sebagai berikut :

- Studi literature diperlukan untuk mempelajari dasar teori yang berhubungan dengan RENESAS R8C/13 TINY, dan komponen pendukung system yang lain serta pengumpulan data-data mengenai karakteristik komponen yang akan digunakan. Data tersebut digunakan sebagai dasar perencanaan dan pembuatan alat.
- Perencanaan dan pembuatan alat terlebih dahulu dilakukan penentuan spesifikasi alat yang akan dirancang.
- Pengujian alat untuk mengetahui kesesuaian alat yang akan dirancang dengan spesifikasi yang telah ditentukan sebelumnya.
- Penyusunan laporan skripsi.

1.6 Sistematika Penulisan

Pembahasan laporan skripsi ini akan diuraikan dan dijabarkan dalam setiap bab, dengan pembagian sebagai berikut :

BAB I : PENDAHULUAN

Terdiri dari latar belakang, rumusan masalah, tujuan, batasan masalah, metologi dan sistematika pembahasan.

BAB II : LANDASAN TEORI

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat yang digunakan.

BAB III : PERENCANAN DAN PEMBUATAN ALAT

Berisi tentang proses perencanaan dan pembuatan sistem Pelayanan kasir cepat saji dengan sistem Kartu Debet menggunakan Kartu RFID.

BAB IV : PENGUJIAN ALAT

Berisi tentang pengujian terhadap alat yang telah dibuat.

BAB V : PENUTUP

Memuat tentang kesimpulan–kesimpulan yang diperoleh dari hasil pembuatan laporan skripsi ini dan saran untuk pengembangan lanjut.

BAB II

PERENCANAAN DAN PEMBUATAN KASIR RESTORAN CEPAT SAJI DENGAN SISTEM DEBET MENGGUNAKAN (RFID) BERBASIS RENESAS R8C/13 TINY

2.1. Pendahuluan

Pada bagian ini akan dibahas teori penunjang untuk peralatan yang akan dirancang yaitu teori dasar Mikrokontroller Renesas R8C/13 TINY, RFID dengan ID-10 RFID reader, LCD, RS 232, limit switch, bahasa pemogram Delphi.

2.2. Mikrokontroller Renesas R8C/13 Tiny (R5F21134FP)

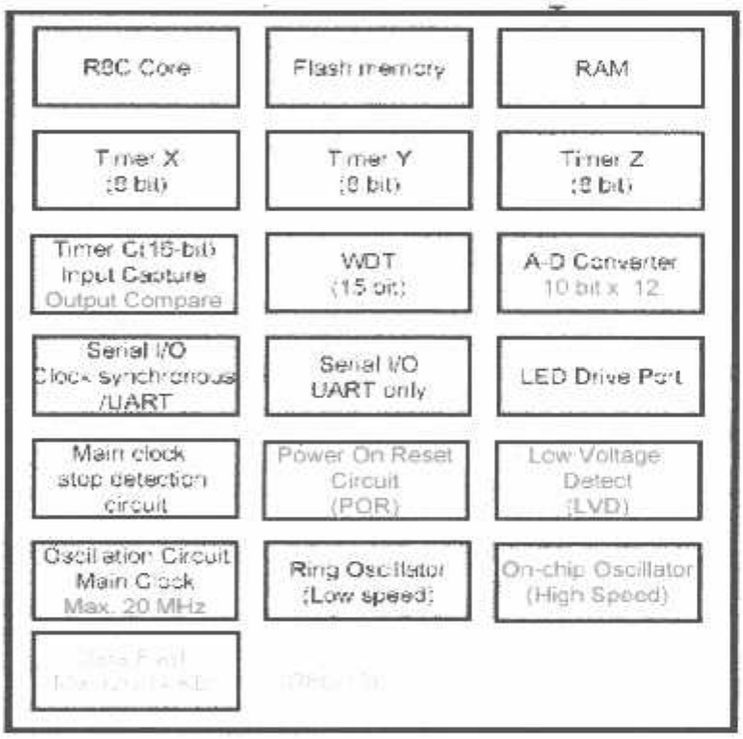
Renesas Technology adalah produsen semikonduktor tingkat internasional. Renesas terbangun dari gabungan dua produsen semikonduktor, yaitu Mitsubishi dan Hitachi. Sebagai produsen semikonduktor, renesas juga mengeluarkan berbagai jenis keluarga mikrokontroler (MK).

Renesas R8C adalah salah satu jenis seri dalam keluarga MK M16C. CPU R8C sama dengan CPU CISC 16-*bit* M16C, hanya saja lebar jalur data R8C adalah 8-*bit*. Karena menggunakan CPU yang sama maka R8C memiliki *instruction set* hampir sama dengan M16C. Perbedaannya hanya terletak pada 2 instruksi, yaitu R8C tidak memiliki instruksi JMPS *(Jump Special Page)* dan JSRS *(Jump Subroutine Special Page)*. R8C/13 adalah salah satu tipe MK dalam seri R8C. MK ini memiliki kemasan 32-*pin* LQFP. Dalam perancangan pada skripsi ini menggunakan menggunakan MK seri R5F21134, yaitu R8C/13 yang memiliki Flash ROM 16 KB (1000 E/W *cycles*) dan RAM sebesar 1 KB.

2.2.1. Spesifikasi R5F21134FP

Berikut ini adalah spesifikasi *R5F21134FP* dengan peta peripheral dan memori-memorinya.

- Mempunyai *CPU Core* (16-bit) 1 – 20 MHz, 3.0 – 5.5 Volt dan 1 – 10MHz 2.7 – 5.5 Volt.
- Rangkaian Clock, kecepatan *Low/High On-Chip Oscillator, Clock* utama dengan Xin/Xout.
- Memory (ROM/SRAM) 16 Kbytes / 1 Kbytes, 2 x 2 K Bytes Data Flash pada R8C/12, 13.
- Kemasan 32 pin LQFP (7mm x 7mm)



Gambar 2-1 Blok Diagram R8C/11, 13 dan Peta *Peripheral*-nya [1]
Sumber data sheet renesas R8C/13 Tiny

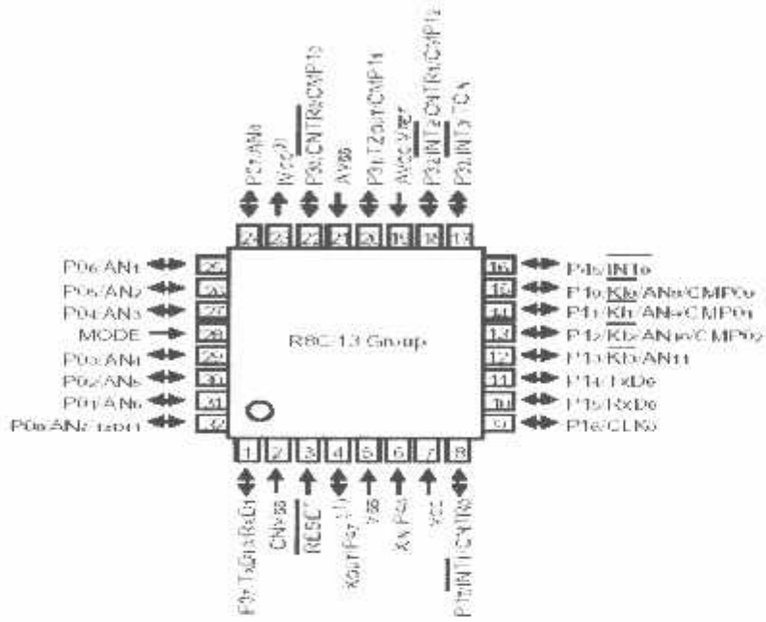
2.2.2. Kelebihan Kunci R8C/Tiny

Banyak kelebihan-kelebihan yang dimiliki R8C/Tiny diantaranya adalah :

Kompatibel dengan M16C yaitu kompatibel dalam instruksi dan kode.

- *Peripheral* lebih terintegrasi jadi lebih hemat.
- *Electromagnetic Compatibility* (EMC) mempunyai EMI rendah, EMS tinggi.
- *Development Tool* (*Compiler* dan *Debugger*) didapat dengan murah dan difasilitasi *On-Chip Debugger*
- Mempunyai fitur *fail-safe* yaitu pengamanan terhadap kegagalan sistem.
- Konsumsi daya rendah.
- 16-bit CISC CPU dengan kecepatan maksimal 20 MHz (1:1).
- 89 instruksi CISC lebih hemat ROM kira-kra 20 %, RAM sampai 1 KB.
- Waktu konversi ADC hanya 3 uS.

2.2.3. Konfigurasi Pin R8C R5F21134FP



Gambar 2-2 Konfigurasi Pin R8C R5F21134FP [1]
Sumber data sheet renesas R8C/13 Tiny

Gambar diatas adalah konfigurasi pin-pin dari *R8C R5F21134FP* untuk lebih jelasnya dapat diamati pada tabel dekripsi pin-pin berikut ini :

Tabel 2-1 Konfigurasi pin-pin dari *R8C R5F21134FP* [1]

Nama Sinyal	Nama Pin	Type I/O	Fungsi
Masukan Catu Daya	Vcc, Vss	I	Tegangan 2.7 V – 5.5 V pada pin Vcc. Tegangan 0 V pada Vss pin
1 Vcc	Ivcc	O	Pin ini untuk men-stabilkan catu daya *internal*, pin ini dihubungkan pada Vss melalui kapasitor 100nF. Jangan dihubungkan pada Vcc.
Input Catu Daya Analog	Avcc, Avss	I	Ini adalah untuk catu daya pada ADC. Avcc dihubungkan pada Vcc, A Vss dihubungkan ke Vss. Dianjurkan untuk menghubungkan kapasitor diantara pin A Vcc dan A Vss.
Input Reset	RESET	I	"L" untuk masukan ini mereset MCU

CNVss	CNVss	I	Pin ini dihubungkan pada Vss melalui sebuah resistor.
MODE	MODE	I	Pin ini dihubungkan pada Vcc melalui sebuah resistor.
Input Clock Utama	Xin	I	Pin-pin ini disediakan untuk membangkitkan rangkaian I/O Clock Utama. Dihubungkan dengan sebuah keramik resonator atau kristal diantara pin Xin dan Xout. Jika digunakan clock internal maka pin Xin dan Xout dalam keadaan terbuka.
Output Clock Utama	Xout	O	
Input Interupsi	INT0 –NT3	I	Pin ini sebagai masukan interupsi.
Input Kunci Interupsi	KI0 – KI3	I	Pin ini sebagai masukan kunci interupsi.
Timer X	CNTR 0	I/O	Pin I/O ini adalah untuk Timer X .
	CNTR 0	O	Pin Ouput untuk Timer X.
Timer Y	CNTR 1	I/O	Pin I/O untuk Timer Y.
Timer Z	TZout	O	Pin Ouput untuk Timer Z.
Timer C	TC in	I	Pin Input untuk Timer C.
	CMP00 – CMP03, CMP10 CMP13	O	Pin Output untuk Timer C.
Serial Interface	CLK 0	I/O	Pin I/O untuk memindahkan Clock.
	RXD0, RXD1	I	Pin input untuk data Serial.
	TXD0, TXD10, TXD11	O	Pin output untuk data Serial.
Input Tegangan Referensi	Vref	I	Tegangan referensi input ini untuk ADC. Vref pin dihubungkan ke Vcc.
ADC, pengubah dari analog ke digital	AN0–N11	I	Pin analog input pada ADC.
Port I/O	P00-P07, P10-P17, P30-P33, P37, P45	I/O	Merupakan port I/O CMOS 8-bit . Setiap port mempunyai pilihan register pengarah sebagai input atau output. Tiap Port dapat dialamati per bit. Dapat di-set menggunakan pull up resistor dengan

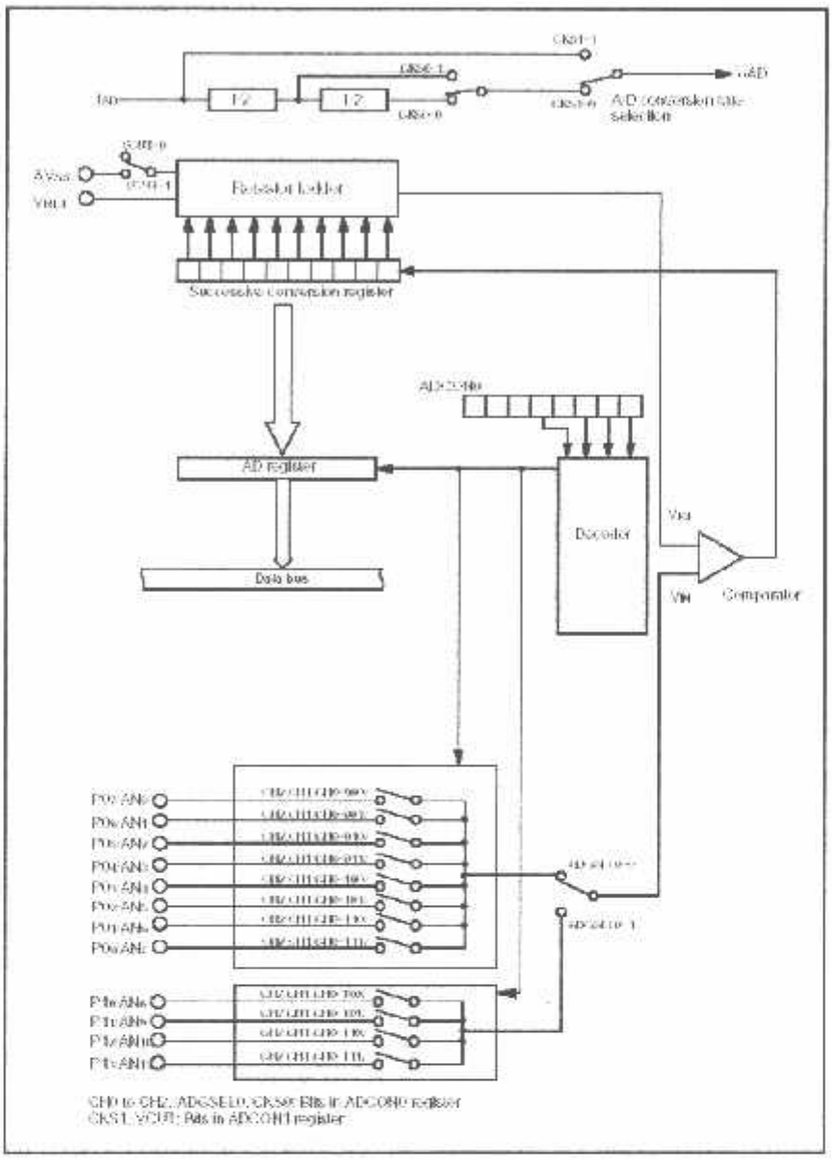
			program. P10 – P17 mempunyai driver transistor.
Port Input	P46, P47	I	Pin ini hanya bisa digunakan sebagai input.

2.2.4. *Peripheral* R8C R5F21134FP

Mikrokontroler *R8C R5F21134FP* mempunyai beberapa *peripheral-peripheral* yang banyak digunakan pada beberapa aplikasi-aplikasi penting, diantaranya adalah sebagai berikut :

- ***Analog To Digital Converter* (ADC)**

  Dengan 12 SAR ADC S/H yang mempunyai resolusi 8-bit atau 10-bit. Mode Operasinya menggunakan *One-Shot dan Repeat* dengan waktu konversi 2.8 uS (pada clock 10 MHz). Berikut gambar diagram blok ADC *built in* pada mikrokontroler ini :



Gambar 2-3 Diagram blok ADC [1]

Sumber data sheet renesas R8C/13 Tiny

- ***Timer Mode***

Mempunyai timer sebanyak 4 yaitu timer X, Y, Z, C. Berikut adalah mode-mode timernya :

Tabel 2-2 Mode-mode Timer [1]

Item		Timer X	Timer Y	Timer Z	Timer C
Configuration		8-bit timer with 8-bit prescaler	8-bit timer with 8-bit prescaler	8-bit timer with 8-bit prescaler	16-bit timer
Count		Down	Down	Down	Up
Count source		•f1 •f2 •f8 •f32	•f1 •f8 •fRING •Input from CNTR1 pin	•f1 •f2 •f8 •Timer Y underflow	•f1 •f8 •f32
Function	Timer mode	provided	provided	provided	not provided
	Pulse output mode	provided	not provided	not provided	not provided
	Event counter mode	provided	provided[1]	not provided	not provided
	Pulse width measurement mode	provided	not provided	not provided	not provided
	Pulse period measurement mode	provided	not provided	not provided	not provided
	Programmable waveform generation mode	not provided	provided	provided	not provided
	Programmable one-shot generation mode	not provided	not provided	provided	not provided
	Programmable wait one-shot generation mode	not provided	not provided	provided	not provided
	Capture	not provided	not provided	not provided	provided
Input pin		CNTR0	CNTR1	$\overline{INT0}$	TCIN
Output pin		CNTR0 $\overline{CNTR0}$	CNTR1	TZOUT	not provided
Related interrupt		Timer X int $\overline{INT1}$ int	Timer Y int $\overline{INT2}$ int	Timer Z int $\overline{INT0}$ int	Timer C int $\overline{INT3}$ int
Timer stop		provided	provided	provided	provided

- ***Low Voltage Detect*** **(LVD)**

  LVD adalah untuk mendeteksi Vcc krang dari 3.8 V (± 0.5 V)

- ***Watchdog Timer***

  Watchdog berfungsi untuk mendeteksi ketika program diluar kontrol.

- ***On Chip Debugger***

Fasilitas ini mempunyai fungsi untuk dapat di-*debug* pada waktu mikro sedang berjalan. Antara PC dan MK dapat berkomunikasi, PC akan mengetahui aktivitas MK saat itu. Syarat-syarat *On Chip Debugger* adalah

- Vektor *Address Match interrupt* harus dihindari.
- *Single step interrupt* tidak dapat digunakan bersamaan interrupt lain.
- *UART1* tidak boleh dipakai.
- Instruksi BRK tidak boleh dipakai.
- Flash Address C000H – C7FFH.
- PD 3.7 harus "0".
- B5 FMR 0 harus "1"
- Menyiapkan 8 Byte untuk Stack.
- *On Chip Debugger* berpengaruh pada *timing run.*

- ***Rangkaian Osilator***

  Pada osilator utama menggunakan kristal luar sampai dengan 20 MHz, dengan memiliki fitur *Clock Stop Detect*. Kemudian untuk *On Chip* Osilator disediakan kecepatan *Low* 125 KHz dan *High* 8 MHz. Saat setelah reset, default clock adalah kecepatan rendah *On Chip* osilator 125 KHz.

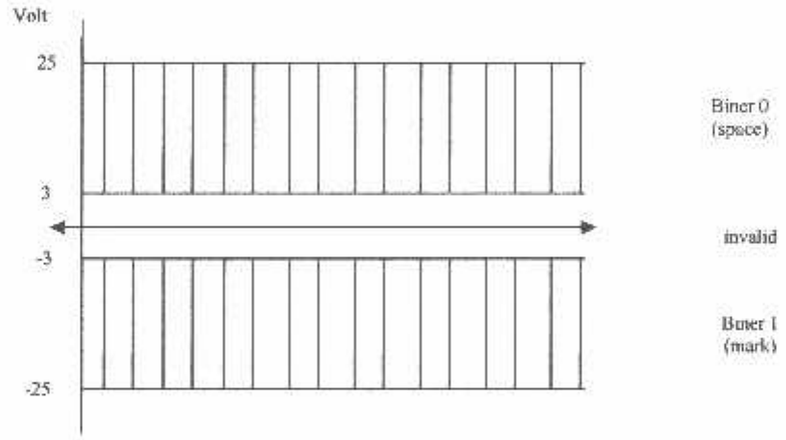
2.3. RS-232

RS-232 (*Recommended Standart number 232*) adalah suatu standar antar muka yang dipublikasikan oleh EISA (*Electronic Industries Association*).

Antarmuka RS-232 digunakan dalam banyak aplikasi komunikasi data. Hal-hal yang perlu diketahui dalam standar antarmuka RS232 antara lain: tegangan standar bagi data *biner* 0 dan 1, sinyal-sinyal yang dipergunakan, dan cara interkoneksi antar RS-232.

2.3.1. Tegangan RS-232

Untuk menampilkan data biner dibutuhkan dua besaran tegangan. *Biner* 1 atau disebut *mark* dinyatakan dengan tegangan antara –3 V sampai dengan –25 V, sedang biner 0 atau yang disebut *space* dinyatakan tegangan antara +3 V sampai +25 V. tegangan antara –3 V sampai dengan +3 adalah tegangan *invalid*. Tegangan yang mewakili *biner* nol dan satu ditunjukkan dalam gambar 2.5 berikut ini:



Gambar 2-4 Tegangan Yang Mewakili Biner 0 dan 1.

2.2.2. Sinyal-sinyal RS-232

Standar antarmuka RS-232 mendefinisikan sinyal-sinyal yang dipakai baik untuk mentransmisikan/ menerima data maupun untuk proses jabat tangan

(*handshaking*). Dalam aplikasi sinyal-sinyal RS-232 ini dihubungkan dengan konektor 25 pin (DB-25) atau 9 pin (DB-9). Pada tabel 2-1 berikut ini diberikan nama sinyal penting beserta hubungannya dengan konektor DB-9 maupun DB-25.

Tabel 2-3 Sinyal-Sinyal Untuk Konektor DB-9 Dan DB-25 [4]

Nomor Pin		Nama Sinyal
DB 9	DB 25	
1	8	DCD (*Data Carrier Detect*)
2	3	RD (*Receive Data*)
3	2	TD (*Transmit Data*)
4	20	DTR (*Data Teminal Ready*)
5	7	SG (*Signal Ground*)
6	6	DSR (*Data Set Ready*)
7	4	RTS (*Request To Send*)
8	5	CTS (*Clear To Send*)

2.3.3. Antarmuka Serial RS-232 Pada IBM PC

Pada IBM PC terdapat antarmuka serial yang mengikuti standar antarmuka RS-232. antarmuka ini menggunakan rangkaian terintegrasi UART (*Universal Asyncronous Reciever/ Transmitter*). IBM PC dapat mempunyai beberapa antarmuka serial, dua diantaranya yang paling penting adalah *primary asynchronous communication adaptor* yang disebut COM 1 dan *secondary asynchronous adaptor* yang disebut COM 2.

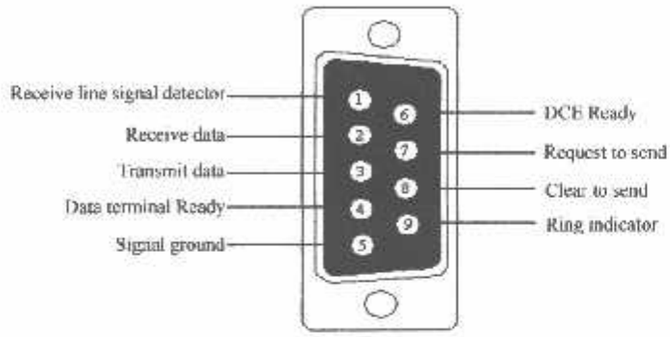
Tabel 2-4 Address Register RS-232 [4]

Nama Register	COM1	COM2
TX Buffer (Transmit Buffer)	03F8H	02F8H
RX Buffer (Receive Buffer)	03F8H	02F8H
Baud rate devisor latch LSB	03F8H	02F8H
Baud rate devisor latch MSB	03F9H	02F9H
Interrupt Enable Register	03F9H	02F9H
Interrupt identification Register	03FAH	02FAH
Line Control Register	03FBH	02FBH
Modem Control Register	03FCH	02FCH
Line Status Rgister	03FDH	02FDH
Modem Status Register	03FEH	02FEH

Fungsi dari beberapa register diantaranya adalah sebagai berikut:

- TX *buffer*: menampung dan menyimpan data yang akan dikirim keluar. Data ini dikirim oleh CPU ke TX *buffer* setelah mengecek kepastian tentang diperbolehkannya melakukan pengiriman.
- RX *buffer* : menampung dan menyimpan data yang diterima dari luar. Data itu harus dibaca oleh CPU setelah mengecek kepastian tentang masukannya data.
- *Baud rate devisor last significant bit*: menampung angka byte bobot rendah untuk pembagi clock yang akan dimasukkan agar didapat baud rate yang diplih. Angka pembagi dapat dipilih antara 01H hingga FFH.

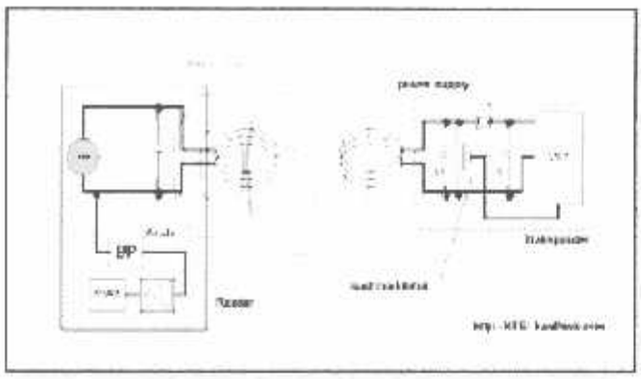
- *Baud rate devisor most significant bit*: menampung angka *byte* bobot tinggi untuk pembagi *clock* yang akan dimasukkan agar didapat *baud rate* yang dipilih. Angka pembagi dapat dipilih antara 00H hingga FFH.



Gambar 2-5 Skematik DB 9.[4]
Sumber data sheet MAX RS-232

2.4. RFID

RFID adalah proses identifikasi seseorang atau objek dengan menggunakan frekuensi transmisi radio. RFID menggunakan frekuensi radio untuk membaca informasi dari sebuah devais kecil yang disebut tag atau *transponder* (*Transmitter* + *Responder*). Tag RFID akan mengenali diri sendiri ketika mendeteksi sinyal dari *devais* yang kompatibel, yaitu pembaca RFID (RFID *Reader*) dengan *range* kisaran pembacaan 8 cm serta bekerja pada frekuensi 125 KHz.



Gambar 2-6 Komunikasi Antara *Reader* dan *Transmiter* (Tag) [2]
Sumber data sheet RFID Reader ID 10

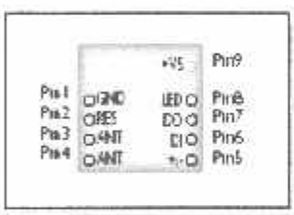
RFID dapat disediakan dalam piranti (devais) yang hanya dapat dibaca saja (*Read Only*) atau dapat dibaca dan ditulis (*Read/Write*), tidak memerlukan kontak langsung maupun jalur cahaya untuk dapat beroperasi, dapat berfungsi pada berbagai variasi kondisi lingkungan, dan menyediakan tingkat integritas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

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

Sistem RFID terdiri dari empat komponen, di antaranya seperti dapat dilihat pada gambar 2-6 :

- Tag : Ini adalah devais yang menyimpan informasi untuk identifikasi objek. Tag RFID sering juga disebut sebagai *transponder*. Format dari tag pada perancangan ini adalah EM4001 atau tag kompatibel lainnya.
- Antena : untuk mentransmisikan sinyal frekuensi radio antara pembaca RFID dengan tag RFID.
- Pembaca RFID : adalah *devais* yang kompatibel dengan tag RFID yang akan berkomunikasi secara *wireless* dengan tag. Digunakan Tipe ID-10 sebagai RFID *reader* pada perancangan ini.
- Software Aplikasi : adalah aplikasi pada sebuah *workstation* atau PC yang dapat membaca data dari tag melalui pembaca RFID. Baik tag dan pembaca RFID diperlengkapi dengan antena sehingga dapat menerima dan memancarkan gelombang elektromagnetik.

Berikut ini adalah konfigurasi pin yang terdapat pada RFID seperti yang terlihat pada gambar 2-7 di bawah ini :



Gambar 2-7 Konfigurasi Pin ID-10 (RFID Reader) [2]
sumber data sheet reader ID 10

2.5. Limit Swicth

Limit swicth merupakan sebuah saklar yang bekerja karena ada suatu sentuhan atau tekanan, sehingga *limit switch* akan berfungsi apabila tersentuh atau tertekan sesuatu, maka *motor* pada pembawa *box* barang akan mati apabila limit swicth tersentuh atau kondisi *off*, selanjutnya motor pengangkat tuas akan hidup dan menuju ke *limit switch* batas areal rak sampai proses selanjutnya.

Ada beberapa tipe *limit switch* yaitu *limit swicth* yang merupakan kontak NC (*normally Closed*) dan NO (*Normally Open*). Limit yang merupakan kontak NO berfungsi sebagai penghubung sedangkan yang kontak NC berfungsi sebagai pemutus.

Adapun simbol dari *limit swicth* atau *push botton* dari yang NO dan NC adalah sebagai berikut:

Pushbutton make No (normally Open) ***Pushbutton break NC (normally Closed)***

Gambar 2-8 Simbol *Limit Switch*.

2.6. Bahasa Pemrograman Borland Delphi

Delphi adalah perangkat lunak untuk menyusun program aplikasi yang berdasarkan pada bahasa pemrograman bahasa *pascal* dan bekerja dalam lingkungan sistem operasi *Windows*. Dengan Delphi diperoleh kemudahan dalm menyusun program aplikasi, karena Delphi menggunakan komponen-komponen

yang akan menghemat penulisan program dengan fasilitas *VCL (Visual Component Library)*.

Dalam pembuatan sebuah program, *Delphi* menggunakan sistem yang disebut *RAD (Rapid Aplication Development)*. Sistem ini memanfaatkan bahasa pemrograman visual yang membuat seorang programmer lebih mudah mendesain tampilan program (*User Interface*. Cara ini bermanfaat untuk membuat program yang bekerja pada sistem *Windows* yang memang tampilan layarnya lebih rumit (sekaligus dapat dilihat dengan indah) dibanding dengan sistem DOS.

Aplikasi dalam tatanan *GUI (Graphical User Interface)* yaitu karakter program aplikasi yang menggunakan sarana perantara grafis dapat dibentuk dengan Delphi. Seperti kotak dialog (*dialog box*), tombol (*button, menu*) dan lain sebagainya. Contoh program GUI adalah program-program *Windows*. Dengan *Delphi* sebuah *Windows* yang mengandung tombol-tombol, kotak cek, tombol pilihan panel dan komponen lainnya dapat dengan mudah diciptakan.

2.6.1. IDE (*Integrated Development Environment*) Delphi

IDE adalah suatu lingkungan dimana sebuah tools yang diperlukan untuk desain, menjalankan dan mengetes sebuah aplikasi disajikan dan terhubung dengan baik sehimgga memudahkan pengembangan program. Pada *Delphi* terdiri dari *main Windows, Component Palette, Toolbar, Form designer, Code Editor, Code Explorer*. Intergrasi ini memberikan kemudahan dalam mengembangkan aplikasi yang kompleks.

- ***Main Windows* (jendela utama)**

Main Widows adalah bagian utama dari IDE. *Main Windows* mempunyai semua fungsi utama dari program-program *Windows* lainnya. *Main Windows* dibagi tiga yaitu menu utama, *toolbar*, dan *component palette*.

Menu utama. Seperti program *Windows* lainnya, menu utama dipakai untuk membuat dan menyimpan file memanggil *wziard*, menampilkan jendela lain, mengubah option dan lain sebagainya setiap pilihan pada menu juga dapat dipanggil dengan sebuah tombol pada *toolbar*.

Toolbar. Beberapa operasi pada menu utama dapat dilakukan melalui *toolbar*. Setiap tombol pada toolbar mempunyai sebuah *tooltip* yang berisi informasi mengenai fungsi dari tombol tersebut. Selain *component palette*, ada 5 *toolbar* terpisah yaitu *debug, desktop, standart, view, dan custom.*

Component Palette adalah toolbar dengan ketinggian ganda, yaitu berisi page kontrol dengan semua komponennnya. Urutan dan tampilan dari *page* dan komponen pada komponen *palette* dapat diatur dengan klik kanan atau dengan memilih menu *component configure component* dari menu utama.

- ***Form designer***

Diawali dengan jendela kosong yang memungkinkan untuk merancang aplikasi *Windows*. Dari sini dapat ditentukan tampilan aplikasi sesuai dengan yang diingnkan. Berinteraksi dengan *form designer* dengan cara memilih komponen *palette* dan meletakkannya ke dalam *form*, posisi dan ukuran dapat diubah-ubah

dengan menggunakan *mouse*. Untuk mengubah tampilan dan perilaku komponen maka digunakan *object inspector* dan *code editor*.

- **Object Editor**

Object Editor terdiri dari dua tab yaitu *tab properties* dan *tab events*. *Tab properties* memberi fasilitas untuk melihat dan mengubah *property* dari setiap item. Klik pada sebuah form kosong, dan perhatikan atribut-atribut yang ada. Jika terdapat tanda + disamping property maka *property* tersebut berarti mempunyai *sub property*. Contohnya *property font*, jika diklik ganda pada *property font* maka akan ditampilkan *sub property*nya seperti *color, height,name* dan lain-lain. *Tab Event* berisi *event-event* yang dapat direspon oleh sebuah obyek. Klik tab *event* disebelah kanan tab *properties*. Misalnya ingin sesuatu dikerjakan pada saat form ditutup, maka tindakan tersebut (berupa sebuah *procedure*) pada *Onclose*.

- **Struktur Menu Delphi**

Struktur menu Delphi memberikan tools untuk mengakses lingkungan Delphi.

1. **File**

Menu file adalah menu paling penting dan akan dijabarkan pada bagian berikut:

New. Digunakan untuk memulai obyek baru.

New Application. Dengan memilih menu ini, berarti akan membuat *project* baru. Jika belum membuka sebuah *project* atau *object* yang dibuka sudah

disimpan ke disk. *Delphi* akan menutup *project* tersebut dan akan membuat *project* baru, termasuk membuat jendela *editor* baru dengan nama file UNIT.PAS, *form baru* (form 1) dan menampilkan *object inspector*.

New Form. Menu ini dipakai untuk membuat form baru.

New frame. Untuk membuat frame kosong dan menambahkannya ke dalam *project*.

Open. Menyatakan pada *Delphi* bahwa akan dibuka sebuah object dapat berupa sebuah program atau seluruh *project*.

Open Project. Untuk membuka sebuah *project*.

Reopen. Menu ini dipakai untuk membuka *object favorit* yang sudah pernah dibuka.

Save. Menu ini dipakai untuk menyimpan *module* yang sedang aktif.

Save as. Dipakai untuk menyimpan *module* dengan nama lain.

Save project as. Menu ini dipakai untuk menyimpan project dengan nama baru.

Save all. Menyimpan sebuah object yang dibuka.

Close. Untuk menutup module program dengan formnya. Jika *module* tersebut belum disimpan, saat menutup maka Delphi akan menanyakan apakah modul tersebut akan disimpan.

Close all. Menutup project.

Use Unit. Delphi akan menambahkan *klauda uses* pada program yang dibuat. Artinya sebuah unit akan dipakai dalam project.

Print. Mencetak item Delphi yang telah dipilih.

Exit. Keluar dari aplikasi Delphi.

2. ***Edit***

   Dipakai untuk menyuting program.

3. ***Search***

   Dipakai untuk mencari dan mengganti kata-kata pada saat menyuting program.

4. ***View***

   Dipakai untuk menampilkan atau menyembunyikan jendela-jendela tertentu, misalnya *object inspector, code explorer, debug* dan lain-lain.

5. ***Project***

   Dipakai untuk mengelola *project. Form dapay* ditambah dan dibuang dari object, mengkompilasi project dan lain-lain.

6. ***Run***

   Menu ini dipakai untuk menjalankan program dan memantau jalannya program. Pada saat di run, apabila terjadi salah tulis akan dapat diketahui.

7. Component

Dengan menu ini komponen baru dapat ditambah atau diinstal.

2.7. Liquid Crystal Display (LCD)

2.7.1. Konfigurasi LCD

Liquid Crystal Display adalah modul tampilan berkonsumsi daya yang relatif rendah dan terdapat sebuah kontroller CMOS didalamnya. Kontroler tersebut sebagai pembangkit karakter dari ROM/RAM dan display data RAM. Semua fungsi tampilan dikontrol oleh suatu intruksi dan modul LCD dapat dengan mudah untuk diinterfacekan dengan mikroprossor/mikrokontroller. Input yang diperlukan untuk mengendalikan modul ini berupa bus data yang termultipleks dengan bus alamat dan 3 bit sinyal kontrol. Pengendali dot matrik LCD dilakukan secara internal pada modul LCD sendiri.

LCD merupakan suatu bentuk kristal cair yang akan beremulsi apabila dikenakan tegangan padanya. Tampilannya ini berupa dot matrik 5 x LCD sehingga jenis huruf yang dapat ditampilkan akan lebih banyak dan lebih baik resolusiny dibandingkan dengan segment 7

Gambar 2-9. Deskripsi pin pada LCD Tipe YJ-162A [3]
sumber data sheet LCD YJ-162

LCD tipe YJ-162A memiliki ciri-ciri sebagai berikut :

- LCD ini terdiri dari 32 karakter dengan 2 baris masing-masing 16 karakter dengan display dot matrik 5 x 7
- Karakter generator Rom dengan 192 tipe karakter
- Karakter generator RAM dengan 8 tipe karakter
- 80 x 8 display data RAM
- Dapat diinterfacekan ke MPU 8 atau 4
- Dilengkapi fungsi tambahan : *display clear, cursor hpme, display ON/OFF, cursor ON/OFF, display character blink, cursor shift, dan display shift.*
- Internal Data
- Internal Otomatis, reset pada saat power ON
- +5 volt PSU Tunggal

Tabel 2-5. Konfigurasi Pin-pin LCD [3]

NO	SYMBOL	LEVEL	FUNCTION
1	Vss	-	0 Ground
2	Vcc	-	5 V + 10%
3	Vee	-	LCD Drive
4	RS	H/L	H : Data Input
			L : Intruksi Input
5	R/W	H/L	H : Read L : Write
6	E	H/L	Enable Signal
7-14	DB0-DB7	H/L	Data Bus
15	Light LCD	-	Menyalakan lampu LCD max 200
16	Light LCD	-	Ground

2.7.2. Interuksi Operasi Dasar

2.7.2.1. Register

Kontroller dari LCD mempunyai dua buah register 8 bit yaitu register intruksi (IR) dan register data (RD). IR menyimpan intruksi seperti *display clear, cursor shift* dan *display data* (DD RAM) serta *character generator* (CG RAM). DR menyimpan data untuk ditulis di DD RAM atau CG RAM atau membaca data dari DD RAM atau CG RAM. Ketika data ditulis ke DD RAM atau CG RAM maka DR akan secara otomatis menulis data ke DD RAM atau CG RAM dan data pada DD RAM atau CG RAM hendak dibaca maka alamat data ditulis pada IR

sedangkan data alamat dimasukkan melalui DR dan mikroprosesor membaca data dari DR.

Tabel 2-6. Tabel Register Seleksi

RS	R/W	OPERASI
0	0	Seleksi IR, IR Write Display Clear
0	0	Busy Flag (DB7) @counter (DB0-DB7) Read
1	0	Seleksi DR, DR Write
1	1	Seleksi DR, DR Read

2.7.2.2. Busy Flag

Busy Flag menunjukkan bahwa modul siap untuk menerima instruksi selanjutnya. Sebagaimana yang terlihat pada table register seleksi sinyal akan melalui DB7. Jika RS = 0 dan R/W = 1. Jika bernilai 1 maka modul sedang melakukan kerja internal dan intruksi tidak dapat diterima. Sehingga status dari flag ini harus diperiksa sebelum melaksanakan intruksi selanjutnya.

2.7.2.3. Address Counter

AC menunjukkan lokasi memori dalam modul LCD. Pemilihan lokasi alamat itu diberikan lewat registerintruksi (IR). Ketika data ada pada A, maka AC secara otomatis menaikkan atau menurunkan alamat tergantung dari *Entry Mode Set*.

R/W	1	I	MPU	Seleksi Sinyal 0 = write 1 = read
RS	1	I	MPU	Seleksi Register
VLS	1	-	PS	0 = intruksi reg (wr) Busy flag addr counter (rd) 1 = data reg (wr dan rd)
7	1	-	PS	Mengatur Tampilan LCD
Vss	1	-	PS	+5 volt

2.7.2.4. Display Data RAM (DD RAM)

Pada LCD masing-masing line mempunyai range alamat tersendiri. Alamat ini diekspresikan dengan bilangan *Hexadecimal* Untuk itu 1 range Alamat berkisar 00H-0FH sedangkan untuk line 2 range alamat berkisar antara 40H-4FH.

2.7.2.5. Character Generator ROM (CG ROM)

CG ROM mempunyai tipe dot matrik 5 x 7 dan data pada LCD telah tersedia ROM sebagai pembangkit Character dalam kode ASCII.

2.7.2.6. Character Generator RAM (CG RAM)

CG RAM digunakan untuk pembuatan karakter tersendiri melalui program.

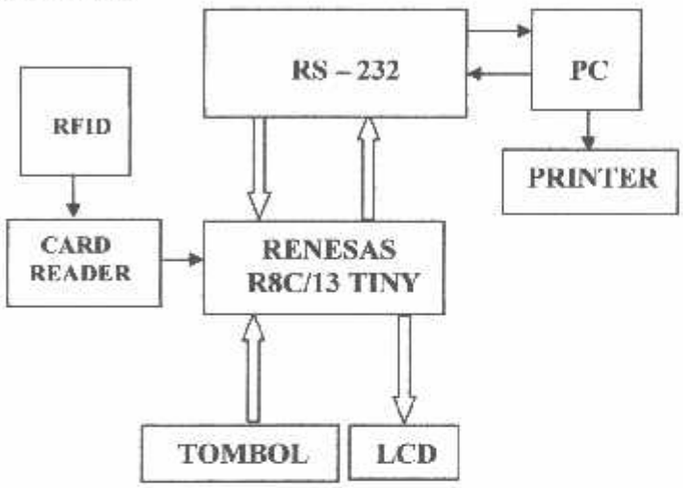
Tabel 2-7. Fungsi Terminal Pada LCD [3]

Nama Signal	Jml Term	I/O	Tujuan	Fungsi
DB0-DB3	4	I/O	MPU	Sebagai lalu lintas data dan intruksi ke atau dari MPU Low Byte
DB4-DB7	4	I/O	MPU	Sebagai lalu lintas data atau intruksi 2 arah upper byte.
E	1	I	MPU	Sinyal Start (read/write)

BAB III
PERENCANAAN DAN PEMBUATAN ALAT

3.1. Pendahuluan

Bagian ini menjelaskan tentang perencanaan dan pembuatan kasir restoran cepat saji Dengan Sistem Debet menggunakan kartu (RFID) Berbasis RENESAS R8C/13 TINY.



Gambar 3-1 Blok diagram

Sistem pelayanan menu makanan dan minuman restoran secara umum dapat di bagi menjadi dua bagian besar, yaitu PC sebagai pusat pelayanan dan sistem-sistem renesas sebai *client*. Masing-masing bagian mempunyai peranan berbeda yang memiliki hubungan timbal balik. Komputer pusat pelayanan direncanakan memiliki fungsi sebagai berikut :

1. Menerima data pesanan menu yang masuk dari *client* pelanggan.

2. Mengirim data kalau pesanan sudah diterima.

Memiliki fungsi sebagai berikut :

1. Menerima order pesanan yang dimasukkan melalui tombol dari tamu yang terlebih dahulu dimasukkan kartu debet.
2. Mengirimkan order pesanan ke computer utama.
3. Menerima data dari PC kalau pesanan sudah dikirim.

Fungsi dari masing-masing blok diagram :

RFID (radio frekuenci identification) : berfungsi sebagai pengidentifikasian kartu debet.

CARD READER :Berfungsi sebagai pembaca kartu yang berisi kode bit RFID.

LCD (liquid crystal diplay) : sebagai tampilan informasi yang di minta.

TOMBOL: untuk masukan inputan ke renesas.

RENESAS R8C/13 TINY : Digunakan untuk mengolah data yang di terima, mengontrol dan mengendalikan rangkaian-rangkaian yang di hubungkan dengan renesas tersebut.

RS 232 : Sebagai penghubung antara renesas dengan PC.

PC (Personal Computer) : Merupakan tempat penyimpan data atau data base dari account yang ada.

PRINTER : Sebagai pencetak hasil transaksi.

Sistem kerja blok diagram secara keseluruhan :

pelanggan yang datang harus mempunyai kartu debet. Kartu debet tersebut dengan cara membeli kartu debet di kasir dengan nilai nominal tertentu dari kartu yang bisa di pilih sebagai kartu akses untuk masuk ke menu-menu yang dipilih. Sehingga apabila pelanggan tidak memiliki kartu debet tersebut maka pelanggan tidak bisa memesan menu-menu yang ada di restoran.

Setelah memiliki kartu debet, Maka kartu tersebut dimasukkan atau digesek pada card reader, dan mulai memesan menu. Program akan menunggu pilihan menu yang dipilih dan jumlah pesanan, apabila pelanggan telah mengisi lengkap maka tinggal menunggu konfirmasi dari computer dan menunggu *struk* (hasil cetak trnsaksi).

Setelah pesanan diterima oleh computer, Maka pesanan tersebut telah masuk ke data base computer dan jika pesanan tersebut oleh pelayan maka kasir akan memberitahukan pelanggan bahwa pesanan segera dikirim. Data-data order dari pelanggan akan tersimpan di computer dan masuk ke database laporan kas harian. Apabila pemesan ingin menambah pesanan makanan maka pelanggan pemesan tinggal menggesek kartu debet tersebut di kasir lalu menentukan pesanan yang akan di pesan, Setelah pemesan selesai memesan dan tinggal menunggu *struk* (hasil cetak trnsaksi). Begitulah untuk pesanan selanjutnya.

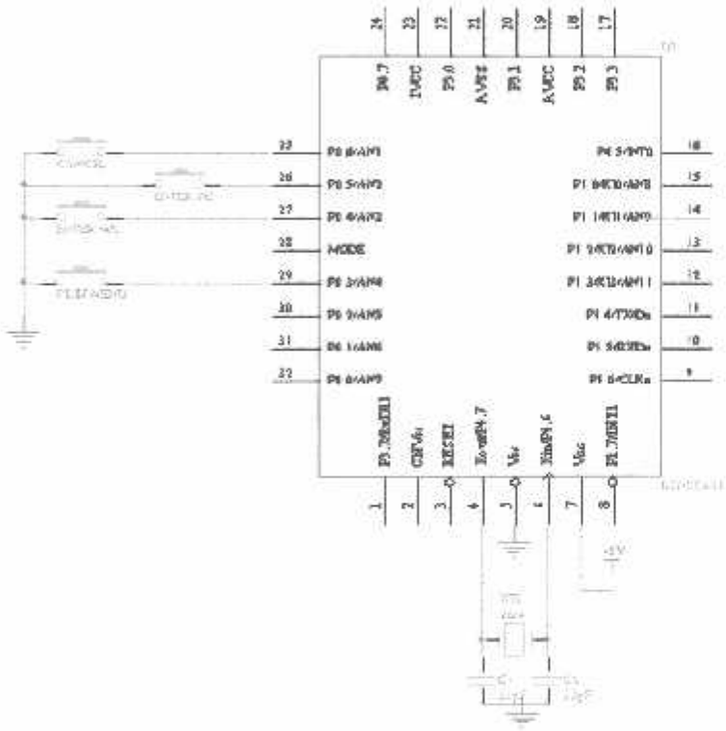
3.2. Perancangan Perangkat Keras

Bagian ini menguraikan perencanaan perangkat keras yang meliputi perencanaan :

1. Sistem mikrokontroller renesas R8C/13 tiny

2. Antarmuka renesas R8C/13 tiny ke modul LCD
3. Antarmuka renesas R8C/13 tiny ke kartu RFID
4. Antarmuka saluran tranmisi standart RS-232 pada titik renesas R8C/13 tiny dan pada titik komputer

3.2.1. Perancangan Mikrokontroler Master Renesas R08C/13 Tiny R5F21134FP.



Gambar 3-2 Minimum sistem Mikrokontroller Renesas R8C/13 Tiny R5F21134FP.

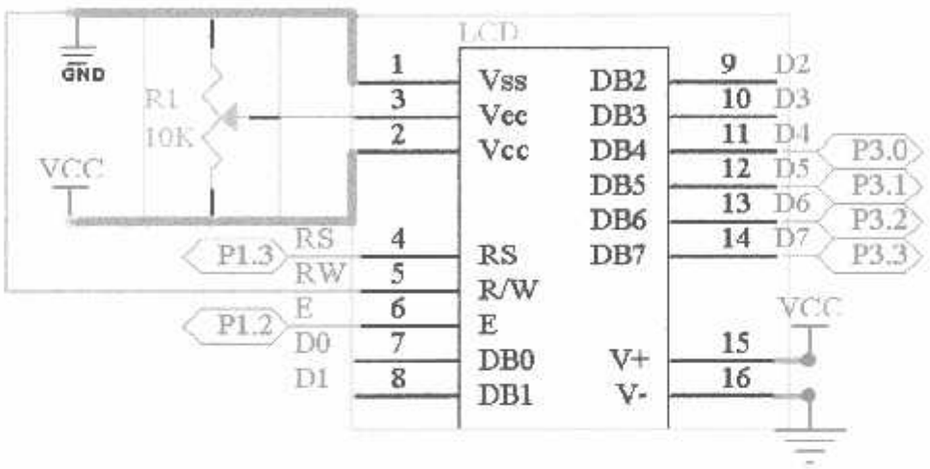
Mikrokontroler *Master* merupakan mikrokontroler pengendali utama. Dalam hal ini menggunakan mikrokontroler Renesas R08C Tiny R5F21134FP karena banyak dengan mempertimbangkan keunggulan-keunggulannya dan fasilitas-fasilitas yang dimiliki IC ini. Pembahasan ini telah dijelaskan pada bab-bab sebelumnya. Mikrokontroler *master* ini mempunyai I/O Port yaitu P0.0 – P0.7, P1.0 – P1.7, P3.0 – P3.3, P3.7 dan P4.5 sedangkan P4.6, P4.7 hanya bisa digunakan sebagai *input* saja, bila konfigurasi kristal memakai kristal *internal*. Dalam hal ini yang digunakan Port I/O saja. Berikut ini konfigurasi pin-pin mikrokontroller master :

1. Untuk AN0 – AN1 digunakan sebagai pin analog input pada ADC
2. Untuk Port 3.0 – Port 3.3 digunakan sebagai output instruksi ke LCD
3. Untuk Pin 7 dihubungkan ke Vcc
4. Pin 12 dan Pin 13 dihubungkan ke RS dan E pada LCD
5. Untuk Pin 5 dihubungkan ke Vss sebagai Ground

3.2.2. Perancangan Rangkaian LCD

LCD (*Liquid Crystal Display*) digunakan sebagai tampilan status alarm. Sinyal-sinyal yang dipergunakan oleh LCD adalah data bus, RS, R/W dan E. Sinyal E dihubungkan ke P1.1 untuk mengaktifkan LCD. LCD akan aktif jika mikrokontroler memberikan instruksi tulis pada alamat LCD. Sedangkan P1.0 dipergunakan untuk memberikan sinyal RS yang membedakan data yang diberikan pada LCD. Sinyal RS diberikan ke LCD untuk membedakan sinyal antara instruksi program atau instruksi penulisan data. Untuk pin R/W akan

berlogika *low* (0) apabila dihubungkan dengan *ground* maka LCD difungsikan hanya untuk menuliskan program atau data ke display. Untuk mengambil data dari mikrokontroler maka pin data DB4-DB7 dari IC ini dihubungkan dengan P3.0 – P3.3 yang merupakan pin data dari mikrokontroler.



Gambar 3-3 Perancangan Rangkaian *Liquid Crystal Display* (LCD)

Keterangan gambar diatas yang merupakan LCD dengan kapasitas 16 x 2 karakter dengan pin – pin pada LCD sebagai berikut :

1. Pin 1 = Vss
2. Pin 2 = Vcc
3. Pin 3 = Vee
4. Pin 4 = RS
5. Pin 5 = R/W
6. Pin 6 = E
7. Pin 11 – Pin = DB4 – DB7

8. Pin 15 = +V
9. Pin 16 = -V

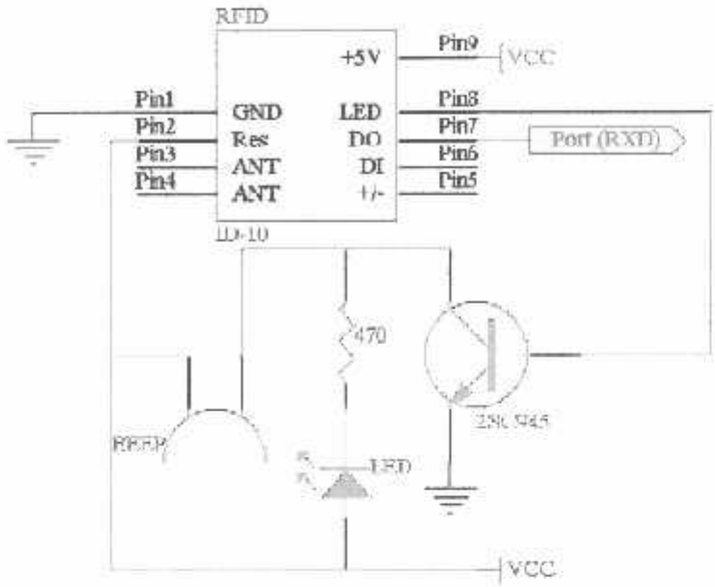
Keterangan :

1. R/W untuk menentukan data yang akan dibaca atau dihubungkan ke *Ground* untuk menulis saja.
2. E dihubungkan ke pin 13 sebagai *enable* yang merupakan sinyal.
3. R/S untuk mengaktifkan *register control* dihubungka ke pin 12.
4. Pada pin Vcc dan Vss digunakan variable resistor (trimpot) sebesar 10 KΩ sebagai pembagi tegangan yang berfungsi untuk mengatur intensitas cahaya pada tampilan LCD.

3.2.3. RFID *(Radio Freqency Identification)*

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

Pada perancangan alat ini juga digunakan LED sebagai indikator pendeteksian RFID. Dalam gambar 3-5 memperlihatkan rangakain *RFID* yang direncanakan.



Gambar 3-4. Rangkaian perencanaan RFID

3.2.3.1. Format Pembacaan ASCII

Salah satu tipe dari RFID *reader* ini yang digunakan pada alat ini adalah ID-10. Kartu RFID yang digunakan mempunyai nomor 0020248083. Kemudian angka tersebut dijadikan bilangan heksadesimal menjadi 134F613 H. Sebenarnya angka tersebut merupakan kode ASCII yang ada di kartu, yaitu 134F613.

Tabel 3-1. Format Pembacaan Kode ASCII

ASCII	1	3	4	F	6	1	3
Heksa	31	33	34	46	36	31	33

3.2.3.2. Frekuensi Kerja RFID

Faktor penting yang harus diperhatikan dalam RFID adalah frekuensi kerja dari sistem RFID. Ini adalah frekuensi yang digunakan untuk komunikasi wireless antara pembaca RFID dengan *tag* RFID.

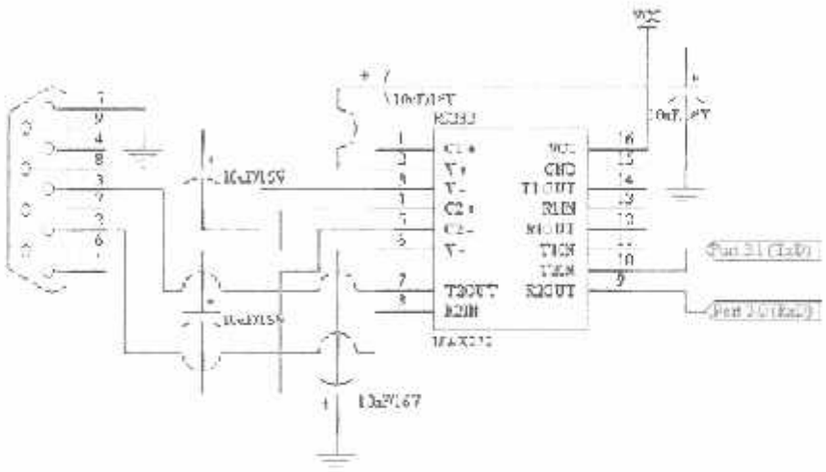
Ada beberapa band frekuensi yang digunakan untuk sistem RFID. Pemilihan dari frekuensi kerja sistem RFID akan mempengaruhi jarak komunikasi, interferensi dengan frekuensi sistem radio lain. Untuk frekuensi yang rendah umumnya digunakan *tag* pasif dan inilah yang dipakai dalam perancangan ini, dan untuk frekuensi tinggi digunakan *tag* aktif.

Pada frekuensi rendah *tag* pasif tidak dapat mentrasmisikan data dengan jarak yang jauh, karena keterbatasan daya yang diperoleh dari medan elektromagnetik. Akan tetapi komunikasi tetap dapat dilakukan tanpa kontak langsung. Pada kasus ini hal yang perlu mendapat perhatian adalah *tag* pasif harus terletak jauh dari objek logam karena logam secara signifikan mengurangi *fluks* dari medan magnet. Akibatnya tag RFID tidak bekerja dengan baik karena tag tidak menerima daya minimum untuk dapat bekerja.

3.2.4. Rangkaian Antarmuka saluran RS-232

Pada perencanaan hubungan antara MC dengan PC adalah dipergunakan komunikasi data secara serial yaitu port 1 atau yang sering dikenal dengan COM 1. Adapun kaki atau pin-pin yang dipakai adalah pada pin no. 2 yang berfungsi untuk sambungan *receive data*, pin no. 3 untuk sambungan *transmit data* yang berguna sebagai input data pada PC yang sebelumnya telah

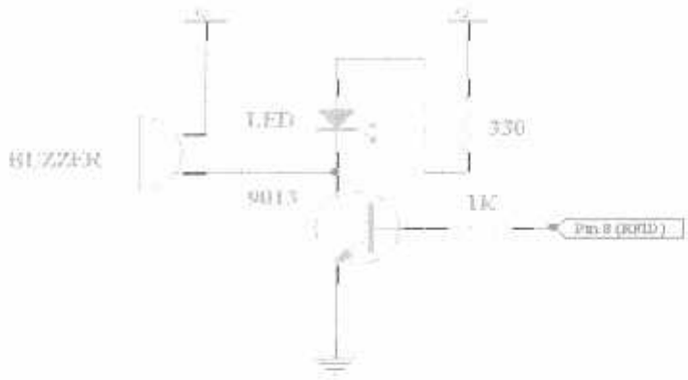
disesuaikan dulu level tegangannya dari RS-232 ke level tegangan TTL melalui sebuah IC MAX 232, dan pin no. 5 untuk *singal ground*. Dalam hal ini kecepatan transfer data per *bit* menggunakan 9600 bps. Rangkaian interface RS-232 diperlihatkan pada gambar 3-6.



Gambar 3-5. Rangkaian RS232

3.2.5. Rangkaian *driver buzzer* Dan Led

Pada umumnya buzzer adalah suatu peralatan yang dapat mengeluarkan bunyi apabila diberi tegangan masukan pada salah satu kakinya dan ked banyak digunakan scbagai indicator pada suatu rangkaian dimana kedua alat ini banyak digunakan sebagai tanda dini untuk mengetahui suatu system apakah telah bekerja dengan baik atau tidak berikut adalah gambar rangkaian *buzzer* dan *led* seperti pada gambar.3.6



Gambar 3.6. Rangakaian driver buzzer dan led

- 17 *driver* ini untuk mengaktifkan *buzzer* dan LED, dimana jika *buzzer* berbunyi dan lampu LED menyala menandakan bahwa Tag dan Reader telah berkomunikasi secara wireless.Untuk mengaktifkan *buzzer* dan LED, mikrokontroller masih membutuhkan sebuah rangkaian *driver*.

  Spesifikasi :

  Buzzer VBZ = 5 V IBZ = 8 mA

  LED VLED = 2,4 ILED = 10 mA

  SS9013 Hfe = 112 Ic = 5oo mA Vbe = 0,7

  RLED = (Vcc-VLED) / ILED = (5-2,4)/10mA = 260 Ω

- Karena dipasaran resistor 960 Ω tidak ada maka diganti dengan nilai resistor yang mendekati yaitu 330 Ω
- Agar transistor tidak terlalu jenuh saat terpicu oleh mikrokontroller, diperlukan sebuah resistansi pembatas arus pada basisnya (Rb).

  Ib = Ic / Hfe= 500mA/100 = 4,464 mA

- Rb = (Vbb-Vbe) / Ib= (5-0,7) / 4,464 mA = 963 Ω

- Karena dipasaran resistor 963 Ω tidak ada maka diganti dengan nilai resistor yang mendekati yaitu 1 KΩ

2.3 Perencanaan Perangkat Lunak/Software

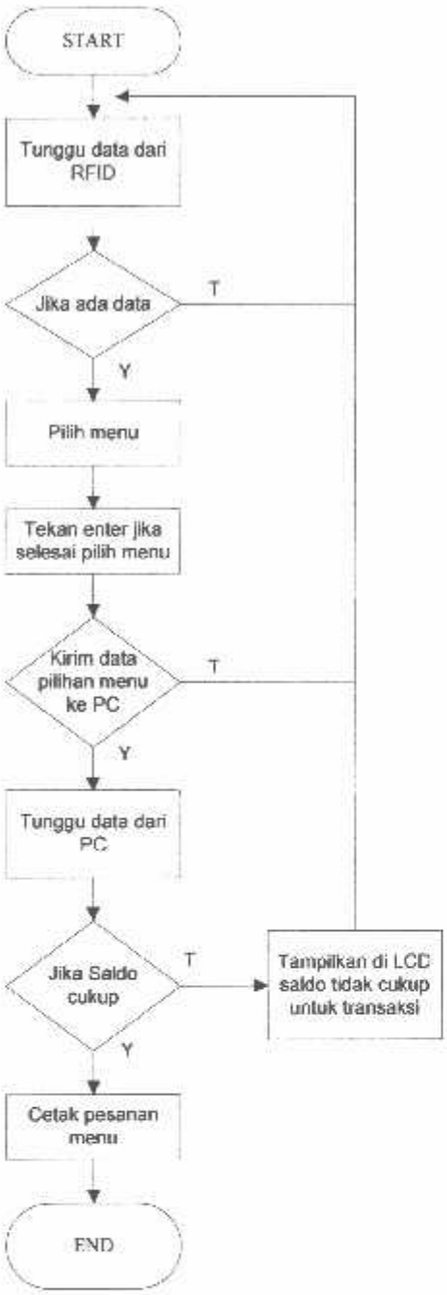
3.3.1. Software di Mikrokontroller renesas R8C /13 tiny

Untuk pemakaian Mikrokontroller renesas R8C /13 tiny di dalam suatu system, perlu direncanakan perangkat lunak Mikrokontroller renesas R8C /13 tiny yang dapat mengatur system tersebut. Perangkat lunak disini adalah susunan perintah-perintah (program) didalam memori yang hareus dilaksanakan adalah renesas. Di dalam suatu mikrokontroller memori merupakan suatu fasilitas utama karena disinilah disimpan perintah-perintah yang harus dijalankan. Memori disini dapat dibedakan menurut fungsinya menjadi memori program dan memori data.

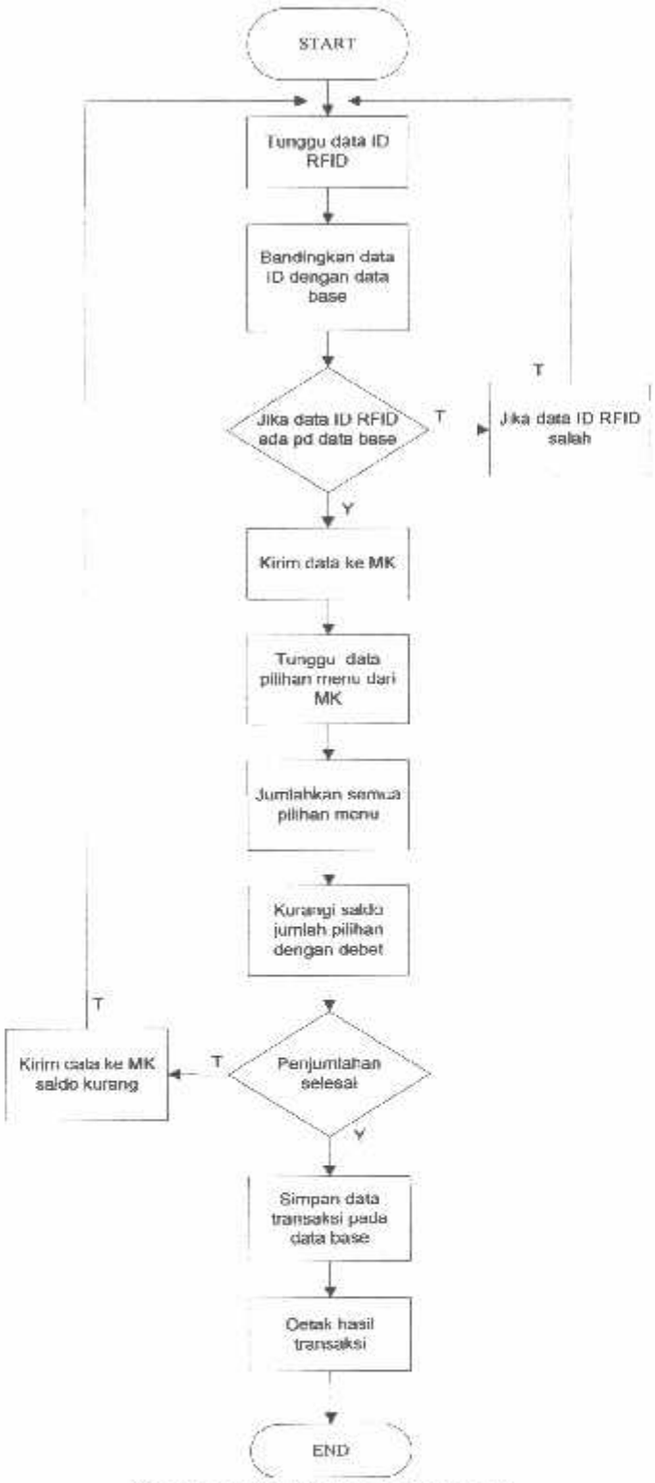
Untuk mendukung agar perangkat keras berfungsi sesuai dengan perencanaan, maka diperlukan perangkat lunak sebagai penunjangnya. Sistem aplikasi mikrokontroller R8C/13 ini dapat mengatur dan mengendalikan keseluruhan sistem apabila ada urutan instruksi yang mendefenisikan secara jelas urutan tugas yang harus dikerjakannya .

Urutan instruksi ini sangat penting untuk didefenisikan, karena mikrokontroller bekerja secara pasti berdasarkan urutan insruksi ini. Susunan logika perancangan yang salah tidak dapat diketahui oleh mikrokontroller. Selama instruksi yang diterima sesuai dengan aturannya, mikrokontroller tetap mengerjakan instruksi tersebut. Kesalahan seperti ini baru diketahui ketika kerja sistem aplikasi tidak sesuai dengan spesifikasi awal. Oleh karena itu, perancangan

perangkat lunak sangat menentukan dalam keberhasilan pembuatan perangkat lunak, sama pentingnya dengan perancangan perangkat keras. Perangkat lunak.



Gambar 3-7 Flowchart MCU



Gambar 3-8 Flowchart Delphi

BAB IV

PENGUJIAN ALAT

4.1. Pendahuluan

Dalam bab ini membahas tentang pengujian dan pengukuran dari peralatan yang dibuat. Secara umum pengujian ini bertujuan untuk mengetahui apakah piranti yang telah direalisasikan dapat bekerja sesuai dengan perencanaan yang telah direncanakan.

Pelanggan yang baru datang terlebih dahulu harus mempunyai kartu debet dengan cara membeli kartu debet tersebut ke kasir dengan nominal dari kartu yang dipilih. Sehingga apabila pelanggan tidak memiliki kartu debet tersebut maka pelanggan tidak bisa memesan menu-menu yang ada di restoran.

Setelah memiliki kartu debet, maka kartu tersebut ditempelkan ke card reader, dan mulai memesan menu. Program akan menunggu pilihan menu yang dipilih dan jumlah pesanannya dari pelanggan yang telah mengisi lengkap.

Setelah pesanan diterima oleh komputer, maka pesanan tersebut telah masuk ke data base komputer data order pelanggan akan tersimpan di komputer (PC) dan masuk ke data base laporan kas harian. Apabila pelanggan masih ingin menambah voucher karena akan menambah pesanan karna saldo sudah tidak mencukupi untuk melakukan transaksi maka pelanggan dapat mengisi voucher tambahan dengan memberitahukan kasir

untuk ditambahkan saldo voucernya dan setelah itu pelanggan memesan order makanan dan minuman.

4.2. Pengujian alat

4.2.1. Pengujian Sistem Mikrokontroller

- **Tujuan**

  Untuk mengetahui kondisi awal dari mikrokontroller apakah sudah sesuai dengan yang direncanakan.

- **Peralatan yang dibutuhkan**

  1. Komputer (PC).
  2. Led Display.

- **Prosedur Pengujian**

  Membuat program yang digunakan dalam pengujian mikrokontroller renesas R8C/13 tiny sebagai berikut :

```
void pilihan8(void)
{
  Tulis_LCD(0x80,"   MENU LAIN  ");
Tulis_LCD(0xC0,"BAKSO BAKAR 7500");
c = 0x38;

}
void main()
{
// Declare your local variables here
```

```
        unsigned char Debounce;
        unsigned long int Wait;

        asm("FCLR I");                          /* Interrupt disable */
        prcr = 1;                               /* Protect off */
        cm13 = 1;                               /* X-in X-out */
        cm15 = 1;                               /* XCIN-XCOUT drive
capacity select bit : HIGH */
        cm05 = 0;                               /* X-in on */
        cm16 = 0;                               /* Main clock = No
division mode */
        cm17 = 0;
        cm06 = 0;                               /* CM16 and CM17
enable */
        asm("nop");
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;                               /* Main clock change */
        prcr = 0;

                                                /* Protect on */
```

```
        sfr_init();
        delayy(50000);
    LCD_data(0,0x33); LCD_data(0,0x32); LCD_data(0,0x2F);
    LCD_data(0,0x0E); LCD_data(0,0x06); LCD_data(0,0x06);

    Tulis_LCD(0x80,"  Restoran DURO  ");
    Tulis_LCD(0xC0,"Jl.Karanglo Km 2 ");
    delayy(500000);
    delayy(500000);
    Tulis_LCD(0xC0,"ITN KAMPUS II MLG");
    delayy(500000);
    delayy(500000);
    delayy(500000);
    Tulis_LCD(0x80," DEKATKAN KARTU  ");
    Tulis_LCD(0xC0," ANDA PD SISTEM  ");
    delayy(500000);
```

4.3. Pengujian RFID

4.3.1. Tujuan

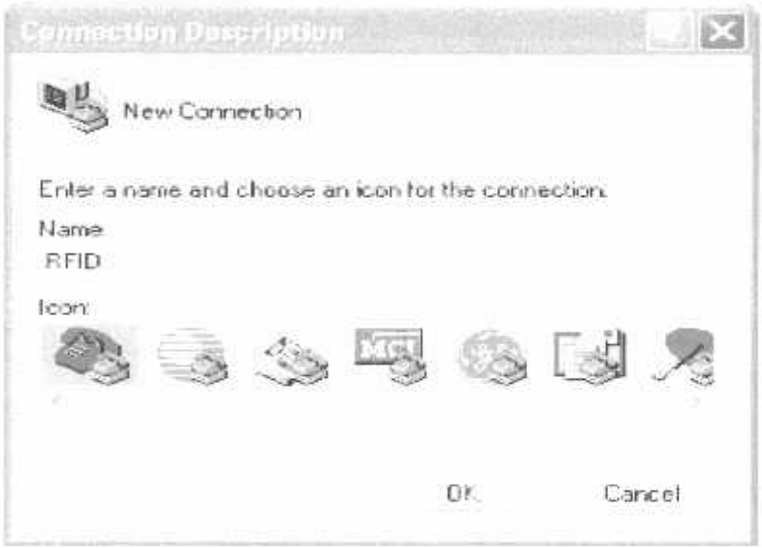
Tujuan dari pengujian ini adalah untuk mengetahui apakah *tag* RFID bisa dibaca oleh *reader* RFID. Adapun cara pengujiannya adalah dengan merangkai rangkaian RFID dan kemudian menghubungkan ke COM1 PC. Untuk menguji *reader* bisa membaca kartu RFID dilakukan melalui *Hyper Terminal.*

4.3.2. Prosedur pengujian

a. Menghubungkan rangkaian RFID ke COM1 PC.

b. Membuka *HyperTerminal (start→ allprogram→ cessories→ communication → hyper terminal)*

c. Memberi nama dan memilih *icon* pada *ConnectioDescription*



Gambar 4-1. Kotak Dialog Connection Description

d. Memilih COM1 pada kotak dialog *connect to*

Gbr 4-2. Kotak Dialog connect.

e. Pada COM1 *properties* mengubah *bits rate per second* menjadi 9600 dan *flow control* menjadi *none*.

Gambar 4-3. Kotak Dialog COM1 Properties

f. Menempatkan kartu pada jarak yang dijangkau *reader* sehingga menampilkan angka dari kartu tersebut.

Gambar 4-4. Identifikasi Reader Terhadap Kartu

4.3.3. Hasil Pengujian Pembacaan RFID

Tabel 4-1. Hasil Pengujian Pembacaan RFID

Jarak jangkuan	Kepekaan reader
1 Cm	Sangat peka
2 Cm	Sangat peka
3 Cm	Sangat peka
4 Cm	Sangat peka
5 Cm	Peka
6 Cm	Kurang peka
7 Cm	Kurang peka
8 Cm	Kurang peka
9 Cm	Kurang peka
10 Cm	Kurang peka

Tabel di atas merupakan hasil pengujian dimana kartu yang menghadap *reader* adalah bagian depan. Jarak yang baik untuk bisa teridentifikasi adalah 5 cm. Untuk bagian belakang menghasilkan data yang sama, tetapi untuk pengujian dimana kartu tegak lurus dengan *reader* hanya bisa saat kartu berjarak sangat dekat dengan *reader* (menempel).

4.4. Pengujian Rangkaian Tampilan LCD

4.4.1. Tujuan

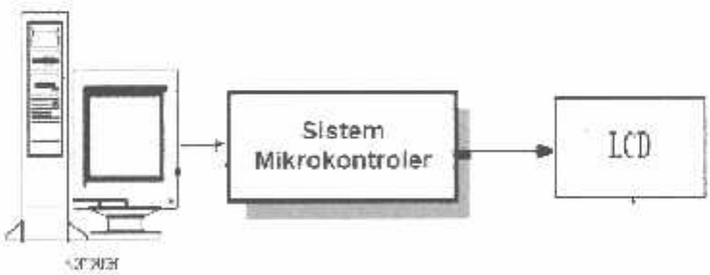
Untuk mengetahui kemampuan rangkaian tampilan yang sudah dibuat apakah dapat mendukung sistem yang direncanakan dan untuk menampilkan data pada LCD.

4.4.2. Peralatan yang Dibutuhkan

1. Komputer (PC)
2. Sistem Mikrokontroler dan LCD

4.4.3. Prosedur pengujian

1. Menyusun rangkaian seperti pada gambar 4-6
2. Menjalankan program untuk menampilkan tulisan ke LCD
   Programnya terlampir
3. Mengamati keluaran pada LCD
4. Isi memori program seperti dibawah ini,yang bertujuan untuk menampilkan tulisan



Gbr. 4-5. Diagram Blok Pengujian Rangkaian LCD

4.5. Pengujian Sistem Alat kasir menu cepat saji auto debet dengan menggunakan Teknologi RFID

4.5.1. Tujuan

Untuk mengetahui apakah semua sistem berjalan dengan normal dan untuk mengetahui error yang terjadi

4.5.2. Prosedur Pengujian

1. Menghubungkan keseluruan rangkaian sesuai dengan diagram blok
2. Menjalankan program Delphi
3. Melakukan proses identifikasi
4. Melewatkan tag RFID

4.5.3. Hasil Pengujian

Tampilan informasi pada program Delphi

a. Menampilkan No ID dari Tag RFID pelanggan

b. Menampilkan nama dan alamat

c. Menampilkan jumlah debet, jumlah kredit, jumlah saldo

d. Menampilkan menu PAHE 1s/d PAHE 3 dan pilihan menu lain

4.6. Analisis Hasil Pengujian

Dari informasi diatas bisa diketahui bahwa dengan memanfaatkan teknologi RFID untuk pembayaran menu makanan di restoran dengan auto debet, sangatlah efesien dan efektif karena proses yang lebih cepat dan mudah penggunannya.

4.6.1. Analisis Hasil Pengujian Sistem

Analisis hasil pengujian sistem merupakan analisis dari seluruh proses yang terjadi dalam perancangan alat ini.

4.6.2. Proses Identifikasi Dan Sistem Transaksi

a. Pada saat kartu tag berada dalam jarak jangkuan reader maka akan terjadi proses identifikasi yang bisa memberikan informasi mengenai sipemilik kartu tersebut.hal tersebut bisa dilihat dari gambar berikut:

GBr. 4-8. Kotak tampilan transaksi

Setelah dilakukan percobaan maka untuk mendapatkan hasil maksimal,maka kartu harus berada dalam jarak ± 5cm

b. Setelah proses identifikasi selesai langkah selanjutnya proses pembayaran Dan proses print out bukti pembayaran menu yang telah di pesan.

RESTORAN
'DURO'
Jln. KARANGLO KM 2
Tlp. 08563571239

3/2/2007 1:45:39 PM

NAMA : ferdy

ALAMAT : perum graha blok NN.6

JUMLAH DEBET : 11500
JUMLAH KREDIT : 5500

JUMLAH SALDO : 6000

PILIHAN MENU :		PILIHAN MENU LAIN :	
PAHE 1 :	5500	KOPI SUSU	: 0
PAHE 2 :	0	KOPI	: 0
PAHE 3 :	0	SOFT DRINK	: 0
		BAKSO KIKIL	: 0
		BAKSO BAKAR	: 0

Gbr. 4-9. Print Out Pembayaran transaksi

PENUTUP

5.1. Kesimpulan

Berdasarkan dari perancangan dan pembuatan kasir menu cepat saji auto debet dengan menggunakan kartu RFID berbasis mikrokontroller renesas R8C/13 tiny yang memanfaatkan teknologi RFID, maka dapat diambil kesimpulan sebagai berikut:

- **Spesifikasi alat :**

  1. Spesifikasi alat yang berupa LCD sebagai tampilan pemilihan menu dimana pelanggan memilih menu apa yang akan dipilih cukup dengan melihat di LCD.
  2. kartu RFID dimana sebagai alat pemesanan / pembelian makanan dan minuman.
  3. PC sebagai data base pelanggan atau data-data pesanan pelanggan yang memesan menu.
  4. RENESAS R8C/13 Tiny sebagai pengolahan penyimpan data data untuk tampilan menu di LCD dan sebagai pengontrol dan mengendalikan rangkaian-rangkaian yang dihubungkan dengan renesas tersebut.
  5. RS-232 sebagai alat pengkoneksi ke PC.
  6. printer sebagai pencetak hasil dari pesanan.

- **Hasil pengujian**

pada RFID yang digunakan berbentuk kartu dimana mempunyai jarak jangkau yang bisa dideteksi *reader* dengan baik sejauh 5 cm apabila kartu

tersebut ditempelkan ke card reader maka data dari card reader tersebut dikirim ke RS-232 oleh PC data tersebut di cocokkan dengan data base yang sudah terdaftar setelah itu PC mengirim data tersebut ke MK sebagai tanda bahwa ada pelanggan yang akan mau memesan makanan dan minuman oleh MK data tersebut difungsikan untuk mengaktifkan data pilihan menu lewat tombol yang ditampilkan di LCD.

- **Kelebihan alat**

  Dimana kelebihan alat ini dapat bisa mengurangi antrian pada saat membayar makanan dan minuman di restoran, Dengan alat ini mempermudah proses pembayaran restoran yang lebih efisien dan efektif karena prosessnya lebih cepat dan mudah penggunaannnya.

5.2 Saran

Beberapa saran yang penulis dapat sampaikan untuk pengembangan alat ini adalah sebagai berikut:

- Untuk pengembangan lebih lanjut diharapkan agar alat ini dapat ditingkatkan atau dikembangkan penggunaannya, guna mendapat hasil yang optimal dikemudian hari, tidak hanya sebatas sebagai alat sistem pembayaran di restoran.

DAFTAR PUSTAKA

1. Data sheet Renesas R8C/13 TINY
2. Data sheet RFID
3. Data sheet Maxim RS232
4. Totok budioko , ST (2003), Belajar Mikrokontroler bahasa C dgn sdcc (small device C compiler) Yogyakarta : Gava Media
5. Sudjadi (2005), Teori Dan Aplikasi Mikrokontroler, Yogyakarta : Graha Ilmu
6. http://www.maxim-ic.com.
7. http://www.digi-ware.com
8. M. Agus J. Alam, Belajar Sendiri Mengolah Database Dengan Borland Delphi 7, PT. Elex Media Komputindo,Jakarta 2003

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNIK INDUSTRI
JURUSAN TEKNIK ELEKTRO

LEMBAR PERBAIKAN SKRIPSI

Nama Mahasiswa	: Ferdy Ridwan Dinata
NIM	: 01.17.030
Jurusan	: Teknik Elektro S1
Konsentrasi	: Teknik Elektronika
Hari / Tanggal	: Kamis / 29 Maret 2007

No	Materi Perbaikan	Paraf
1.	Gambar Rangkaian Lengkap Rangkaian Mikrokontroller	
2.	Kesimpulan	
3.	Penjelasan Mekanisme RS-232	

Telah Diperiksa /Disetujui.

PENGUJI II

DR. Cahyo Crysdian, Msc
NIP. 1030400412

Mengetahui,

Dosen pembimbing I

Joseph Dedy Irawan, ST, MT.
NIP. 132 315 178

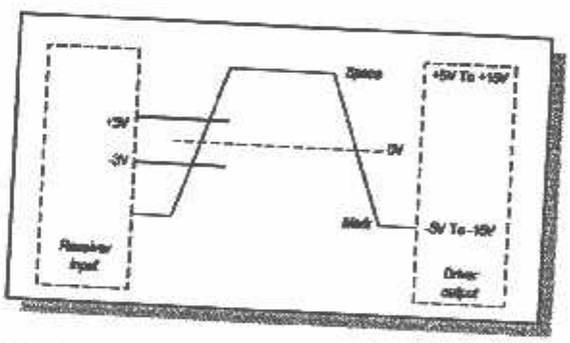
Dosen Pembimbing II

I Komang Somawirata, ST.MT
NIP.P. 103 01000 361

Mekanisme RS-232

RS-232 merupakan seperangkat alat yang berfungsi sebagai *interface* dalam proses transfer data secara serial. Metode pengiriman secara serial RS-232 adalah asinkron. Pengiriman asinkron berarti tidak membutuhkan pewaktu sebagai sinkronisasi. Dalam pengiriman serial *asinkron, clock* tidak dikirimkan, tetapi dikondisikan oleh *timing start bit* yang merupakan isyarat dari sumber ke tujuan untuk mengkodekan adanya pengiriman karakter sudah selesai dikirim.

Karakteristik elektris dari sistem RS-232 adalah mempunyai tegangan keluaran antara –15 Volt sampai dengan +15 Volt. Teganan +5 sampai +15 volt untuk mewakili level rendah (logika '0'/ *spacing)* dan tegangan -5 sampai -15 volt untuk mewakili level tinggi (logika '1'/ *marking).* Hal tersebut seperti ditunjukkan dalam Gambar 1



Gambar 1. Level Logika Standar RS-232

CCITT telah merekomendasikan karakteristik sinyal listrik pada rangkaian *Interface* V-24. Karakteristik sinyal listrik yang dimaksudkan adalah batas-batas tegangan yang digunakan. Dalam rekomendasi tersebut dibedakan antara sinyal data

dan sinyal kontrol, sinyal data sebagai logika '0' dan '1', sedangkan sinyal kontrol dinyatakan dengan ON dan OFF. Sinyal-sinyal tersebut mempunyai batasbatas (daerah) tegangan yang ditunjukkan pada tabel 2-9.

Tabel 1. Tabel Karakteristik Sinyal Listrik Protokol RS 232

Sinyal	Nilai	Daerah Tegangan
Sinyal data	1	-3V t Vp t -15V
	0	+3V d Vp d +15V
Sinyal kontrol	ON	+3V d Vp d +15V
	OFF	-3V t Vp t -15V

Dalam standar RS232, tegangan antara +3 sampai +15 Volt pada input *Line Receiver* dianggap sebagai *level* tegangan '0', dan tegangan antara –3 sampai –15 Volt dianggap sebagai *level* tegangan '1'.

Untuk mengurangi kemungkinan terjadinya gangguan *'cross talk'* antara kabel saluran sinyal RS 232, kecuraman perubahan tegangan sinyal dibatasi tidak boleh lebih dari 30 Volt/mikro-detik, makin besar kecuraman sinyal, makin besar pula kemungkinan terjadi *'cross talk'*. Di samping itu ditentukan pula kecepatan transmisi data seri tidak boleh lebih besar dari 20 KiloBit/Detik. Impedansi saluran dibatasi antara 3 Kilo-Ohm sampai 7 Kilo-Ohm, dalam standar RS232 yang pertama ditentukan pula panjang kabel tidak boleh lebih dari 15 Meter (50 feet), tapi ketentuan ini sudah di-revisi pada standar RS232 versi 'D'. Dalam ketentuan baru tidak lagi ditentukan panjang kabel maksimum, tapi ditentukan nilai kapasitan dari kabel tidak boleh lebih besar dari 2500 pF, sehingga dengan menggunakan kabel kualitas baik bisa dicapai jarak yang lebih dari 50 *feet*. a). Protokol Komunikasi pada

RS 232

Beberapa protokol pada *Interface* RS 232 adalah :

- *Start Bits*

  Merupakan sebuah bit dengan logic '0', bit ini yang menandakan bahwa akan ada karakter atau data yang mengikutinya. Bit ini langsung diberikan oleh sinyal *device* tanpa harus mengeset terlebih dahulu.

- *Data Bits*

  Merupakan bit yang mewakili dari karakter yang diikutinya, data bit ini dapat diset sepanjang antara 5 sampai 8 bit.

- *Parity Bits*

  Merupakan bit yang digunakan sebagai *error checking* pada *receiver*. *Parity bit* akan menghitung jumlah data yang berlogika '1' pada data bit. Perhitungan jumlah data bit tersebut tergantung dari jenis *parity* yang diset. Untuk *parity 'even'*, jumlah data bit yang berlogika '1' ditambah dengan *parity bit* akan menghasilkan jumlah yang ganjil. Sedangkan untuk *parity 'mark '*, merupakan *parity bit* yang selalu berlogika '1' begitu pula pada space, *parity* bit selalu berlogika ' 0' dan *parity 'none'* merupakan *parity bit* yang diabaikan.

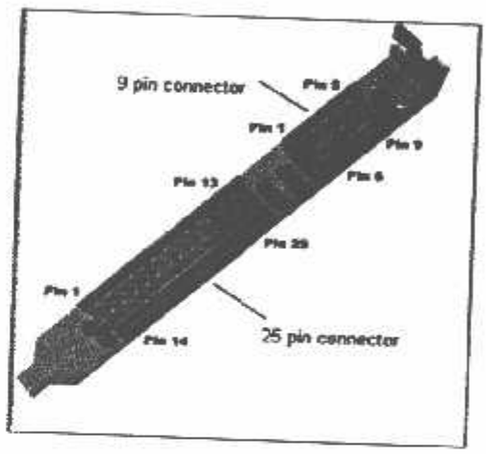
- *Stop Bits*

  Merupakan bit yang menandakan akhir dari suatu paket data (biasanya 1 *byte* data). Seperti pada data bit, bit ini langsung diberikan dari serial *device*. *Stop bit* ini dapat diset menjadi satu bit, satu setengah dan dua bit.

- *Boud Rate*

Baud Rate digunakan untuk menjukkan kecepatan dari transmisi (bits per second).

Di dalam komputer terdapat fasilitas komunikasi serial yang menggunakan standar RS-232, yaitu terletak pada COM1 dan COM2. Kedua fasilitas ini menggunakan konektor DB9 atau DB25 sebagai penghubung dengan piranti luar. Gambar konektor DB9 seperti terdapat dalam Gambar 2-11



Gambar 2. Konektor DB-9 dan DB-25

Fungsi masing-masing pin pada DB-9 seperti terdapat dalam Tabel 1

Tabel 2. Fungsi Pin RS-232 dalam DB-9

Pin	Nama	Fungsi
1	DCD *(Data Carrier Detect)*	Mendeteksi sinyal *carrier* dari modem lain
2	RD *(Receive Data Line)*/ (RXD)	Pengiriman data serial dari DCE ke DTE
3	TD *(Transmit Data Line) (TxD)*	Pengiriman data serial dari DTE ke DCE
4	DTR *(Data Terminal Relay)*	Memberitahu DCE bahwa DTE telah aktif dan siap
5	*Ground*	Referensi semua tegangan antar muka
6	DSR *(Data Set Ready)*	Memberitahu DTE bahwa DCE telah aktif dan siap
7	RTS *(Request To Send)*	Memberitahu DCE bahwa DTE akan mengirim data
8	CTS *(Clear To Send)*	Memberitahu DTE bahwa DCE siap menerima data
9	RI *(Ring Indikator)*	Aktif jika modem menerima

Jalur data (TxD dan RxD) untuk transport data, TxD adalah jalur output pada komputer, data dikirim dari pin ini. Sedangkan RxD adalah penerima untuk komputer, data yang datang akan diterima oleh pin ini. Pin ke empat adalah output

(RTS) di mana sebuah sinyal akan diberikan pada alat yang dihubungkan dengan maksud meminta kiriman data. CTS adalah sinyal masukan yang menunggu sinyal dari alat yang terhubung. Ketika alat tersebut menerima sinyal RTS dan bisa menerima data maka ia akan mengirimkan sinyal balik yang merupakan CTS. DTR adalah sinyal keluaran yang memberi tanda bahwa ada alat yang terhubung dan akan mengirimkan data. DSR merupakan sinyal input yangmana jika alat yang terhubung manerima sinyal DTR ia akan memberi sinyal balikkemudian diterima sebagai sinyal DSR. Spesifikasi RS-232 dapat dilihat dalam Tabel 3

Tabel 3. Spesifikasi RS-232

Keistimewaan	**Karakteristik**
Jenis operasi	*Single ended* (tak seimbang)
Jenis penggerak dan Penerima per jalur	1 *driver* 1 *receiver*
Data *rate* maksimum Panjang saluran maksimum	20 kbps 50 ft (15 m)
Tegangan keluaran penggerak Sensitivitas penerima	±5 - ±15 volt ±3 volt

Format data MK ke PC

sejauh ini kita sudah memperkenalkan Rs-232 Communiction dalam hubungan dengan PC [itu]. RS-232 komunikasi adalah synchronous. itu adalah suatu isyarat jam tidaklah dikirim dengan data [itu]. masing-masing kata[an] yang disamakan penggunaan adalah start bit, dan suatu jam internal pada [atas] sisi masing-masing, [menyimpan/pelihara] rekening pada [atas] pemilihan waktu.

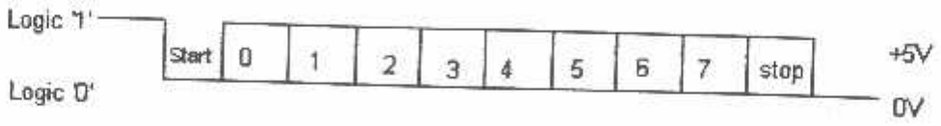


diagram di atas, menunjukkan bentuk gelombang yang xpected dari UART ketika menggunakan 8NI yang umum formt. 8NI menandakan 8 data bit, tidak parity kesamaan dan 1 perhentian bit. RS-232 garis, ketika kosong adalah di dalam tanda itu state (logika 1). suatu transmisi mulai dengan bit start yang mana adalah (logika 0). kemudian masing-masing bit diturunkan ia garis, satu demi satu. LSB (least Significant Bit) dikirim dulu. suatu Bit perhentian (Logika 1) kemudian menambahkan catatan kepada isyarat untuk menyusun transmisi itu.

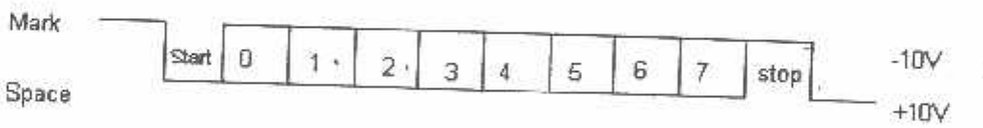
diagram, menunjukkan bit yang berikutnya setelah perhentian menggigit untuk;menjadi logika 0. sebanyak ini; sekian berarti kata[an] lain sedang mengikuti, dan ini adalah adalah start bit. jika tidak ada lebih data datang kemudian te menerima garis akan tinggal status di dalamnya kosong (logika 1). kita sudah temu sesuatu yang disebut pecahkan isyarat. ini adalah ketika garis data [disimpan/laksanakan di] suatu logika 0 status untuk waktu yang cukup panjang untuk mengirimkan suatu keseluruhan kata[an]. oleh karena itu jika yuo

tidak berusaha sangat keras - baris suatu status kosong, kemudian menerima akhir akan menginterpretasikan isyarat retakan sebagai ini.

Format data PC ke MK

data yang dikirim penggunaan metoda ini, disebut dibingkai antar suatu start dan bit perhentian. perlukah bit perhentian diterima sebagai logika 0, kemudian suatu penyusunan kesalahan akan terjadi. ini adalah umum, kapan kedua sisi melakukan kecepatan komunikasi secara berurutan.

di atas diagram hanya relevan untuk isyarat yang dengan seketika di URT. RS-232 logika mengukur penggunaan + 3 untuk + 25 volt untuk menandakan "Spasi" (Logika 0) dan - 3 untuk - 25 volt untuk a " Mark" (Logika 1). Manapun voltge di tengahnya daerah ini (dengan kata lain antara + 3 dan - 3 Volt) adalah tak tergambarkan. Oleh karena itu isyarat ini dihubungkan " RS-232 mengukur konvertor". Ini adalah isyarat menyajikan pada RS-232 koneksi komputer mu, menunjukkan di bawah



Format data RFID ke PC

Salah satu tipe dari RFID reader ini yang digunakan pada alat ini adalah ID-10. RFID *reader* ini memiliki dua bentuk output serial yaitu: ASCII dan Wiegand 26-bit. Pada perancangan alat ini digunakan output dengan

format ASCII, karena output ini sangat mudah untuk dihubungkan pada mikrokontroler.

Tabel 1. Fungsi Pin dan Format Data

Pin No.	Deskripsi	ASCII	Wiegand26
Pin 1	Zero Volts dan Tuning	GND 0V	GND 0V
Pin 2	Strap ke +5V	Reset Bar	Reset Bar
Pin 3	Ke External Antena dan	Antena	Antena
Pin 4	Ke External Antena	Antena	Antena
Pin 5	Format Selector (+/-)	Strap ke GND	Strap ke +5V
Pin 6	Data 1	CMOS	1 Output
Pin 7	Data 0	TTLData (inverted)	Zero Output
Pin 8	3.1 kHz Logic	Beeper / LED	Beeper / LED
Pin 9	DC Voltage Supply	5V (+)	5V (+)

Output yang memiliki format ASCII memiliki struktur sebagai berikut:

0 2	10 data karakter ASCII	Checksum	CR	LF	0 3

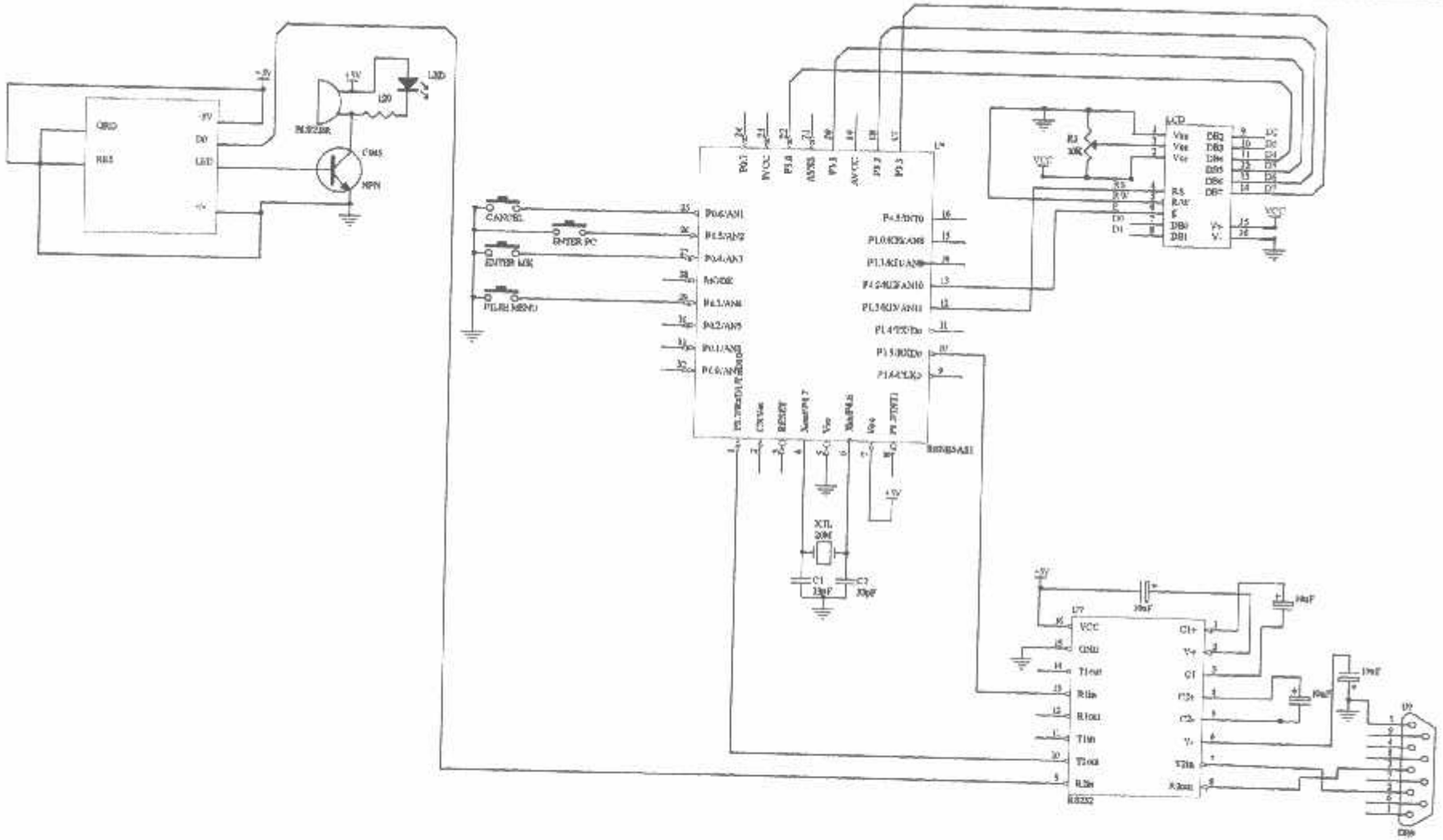
Checksum merupakan hasil EXOR *(Exclusive* OR) dari 5 biner data byte. Misalnya data output serial (dalam hexadesimal) yang kita tangkap adalah sebagai berikut:

O2	30	34	36	32	30	31	44	37	36	43	44	43	OD	OA	O3

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

	34	36	32	30	31	44	37	36	43
		6	2	0	1	D	7	6	C

Untuk data dengan angka 30 dan 34 merupakan data untuk jenis -jenis kartu dan tidak digunakan dalam proses konversi, yang akan dipakai disini adalah data yang ke 3 s/d 10. Hasil konversi dari data heksa ke dalam data ASCII adalah "6201D76C". Gabungkan karakter data ASCII menjadi bilangan Hexadesimal, kemudian konversikan bilangan hexadesimal tersebut ke dalam desimal. Hasilnya sebagai berikut: 6201D76C H menjadi 1644287852. angka-angka ini merupakan nomor kartu sebenarnya yang tertera pada badan kartu yang biasa disebut *tag* RFID.


LCD
CARDSL
ENTER PC
ENTER MSK
PUSH MENU
XTL
20M
C1
33pF
C2
30pF
MAX232
NPN

```
#include    <stdio.h>
#include        "sfr_r813.h"                    /* Definition o
#include        "serial.h"
#define SW_ON   0
#define SW_OFF  1
#define LOW     0
#define HIGH    1
#define Tombol1         p0_1
#define Tombol2         p0_2
#define Tombol3         p0_3
#define Tombol4         p0_4
#define Tombol5         p0_5
#define Tombol6         p0_6
#define Tombol7         p0_7
char Ratusan1 = 1 ;
char Ratusan2 = 2 ;
char Ratusan3 = 3 ;
char Ratusan4 = 4 ;
unsigned char Sem;
unsigned char c;
unsigned char data1;
unsigned char data2;
unsigned char data3;
unsigned char data4;
unsigned char data5;
char hit = 0;
/*****************************************************************
*   Tempat Subrutin /deklarasi prototype subrutin
*****************************************************************/
// contoh: void Delay(vcid) {..source code Subrutin Delay..}
// Declare your global variables here

/*****************************************************************
*****************************************************************/


/*****************************************************************
*************
*       Function : main()
*       program section
*****************************************************************
*************/
void delayy(long tunggu)
{
        while(tunggu--);
}

void LCD_data(char c,char dat)
{
        p1_2 = c;
                p1_2 = c;
```

```
        if ((dat & 0x80)==0x80) p3_3=1; else p3_3=0;
        if ((dat & 0x40)==0x40) p3_2=1; else p3_2=0;
        if ((dat & 0x20)==0x20) p3_1=1; else p3_1=0;
        if ((dat & 0x10)==0x10) p3_0=1; else p3_0=0;
        p1_3 = 1;                       p1_3 = 0;
        if ((dat & 0x08)==0x08) p3_3=1; else p3_3=0;
        if ((dat & 0x04)==0x04) p3_2=1; else p3_2=0;
        if ((dat & 0x02)==0x02) p3_1=1; else p3_1=0;
        if ((dat & 0x01)==0x01) p3_0=1; else p3_0=0;
        p1_3 = 1;                       p1_3 = 0;
        delayy(100);

}


void Tulis_LCD(char a, char* dat)
{
        char i = 0;
        LCD_data(0,a);
        while(dat[i] != 0)
        {
                LCD_data(1,dat[i]); i++;
        }
}
void Tulis_LCD_Hex(char posisi,char dat)
{
        char temp;
        LCD_data(0,posisi);
        temp=((dat>>4)&0x0f)|0x30;
        if (temp> 0x39)temp=temp+7;
        LCD_data(1,temp);
        temp=dat&0x0f|0x30;
        if (temp>0x39)temp=temp+7;
        LCD_data(1,temp);
}


void pilihan0(void)
{
}
void pilihan1(void)
{
    Tulis_LCD(0x80,"PAHE 1    Rp.5s00");
        Tulis_LCD(0xC0,"--========--=====");
        delayy(500000);
        delayy(500000);
        Tulis_LCD(0x80,"NASGOR MAWUT    ");
        Tulis_LCD(0xC0,"ES CAMPUR       ");
        c = 0x31;
}
void pilihan2(void)
{
```

```
    Tulis_LCD(0x80,"PAHE2    Rp.6000");
        Tulis_LCD(0xC0,"================");
        delayy(500000);
        delayy(500000);
        Tulis_LCD(0x80,"Mie Goreng      ");
        Tulis_LCD(0xC0,"Teh Botol SOSRO ");
        c = 0x32;
}
void pilihan3(void)
{
    Tulis_LCD(0x80,"PAHE3    Rp.6500");
        Tulis_LCD(0xC0,"----------------");
        delayy(500000);
        delayy(500000);
        Tulis_LCD(0x80,"NASGOR ISTIMEWA ");
        Tulis_LCD(0xC0,"KOPI SUSU       ");
    c = 0x33;
}
void pilihan4(void)
{
    Tulis_LCD(0x80,"    MENU LAIN   ");
        Tulis_LCD(0xC0,"KOPI SUSU Rp1500");
        c = 0x34;
}
void pilihan5(void)
{
    Tulis_LCD(0x80,"    MENU LAIN   ");
        Tulis_LCD(0xC0,"KOPI      Rp1000");
        c = 0x35;
}
void pilihan6(void)
{
    Tulis_LCD(0x80,"    MENU LAIN   ");
        Tulis_LCD(0xC0,"SNOP DRK Rp.2500");
        c = 0x36;
}
void pilihan7(void)
{
    Tulis_LCD(0x80,"    MENU LAIN   ");
        Tulis_LCD(0xC0,"BAKSO KIKIL 6000");
        c = 0x37;

}
void pilihan8(void)
{
    Tulis_LCD(0x80,"    MENU LAIN   ");
        Tulis_LCD(0xC0,"BAKSO BAKAR 7500");
        c = 0x38;

}
void main()
{
```

```
        // Declare your local variables here
        unsigned char Debounce;
        unsigned long int Wait;

        asm("FCLR I");                                  /* Interrupt di
        prcr = 1;                                               /* Prot
        cm13 = 1;                                               /* X-in
        cm15 = 1;                                               /* XCIN
*/
        cm05 = 0;                                               /* X-in
        cm16 = 0;                                               /* Main
        cm17 = 0;
        cm06 = 0;                                               /* CM16
        asm("nop");
        asm("nop");
        asm("nop");
        asm("nop");
        ocd2 = 0;                                               /* Main
        prcr = 0;


                                                        /* Protect on *
        sfr_init();
        delayy(50000);
        LCD_data(0,0x33); LCD_data(0,0x32); LCD_data(0,0x2F);
        LCD_data(0,0x0E); LCD_data(0,0x06); LCD_data(0,0x06);

        Tulis_LCD(0x80,"  Restoran DURO  ");
        Tulis_LCD(0xC0,"Jl.Karanglo Km 2 ");
        delayy(500000);
        delayy(500000);
        Tulis_LCD(0xC0,"ITN KAMPUS II MLG");
        delayy(500000);
        delayy(500000);
        delayy(500000);
        Tulis_LCD(0x80," DEKATKAN KARTU  ");
        Tulis_LCD(0xC0," ANDA PD SISTEM  ");
        delayy(500000);

        asm("FSET I"); /* Interrupt enable */

//interupsi receive


        while (1)

    {
        if (  p1_7 == SW_OFF )

        {
        if (Tombol7 == SW_ON)
        {
```

```
        delayy(50000);
        if (Tombol7 == SW_ON)
                hit--;
                }
        if (hit==12)
                hit = 0;
                switch(hit)
                {
                default:
                case 0: pilihan0();break;
                case 1: pilihan1();hit++;break;
                case 2: pilihan0();break;
                case 3: pilihan2();hit++;break;
                case 4: pilihan0();break;
                case 5: pilihan3();hit++;break;
                case 6: pilihan0();break;
                case 7: pilihan4();break;
                case 8: pilihan5();break;
                case 9: pilihan6();break;
                case 10: pilihan7();break;
                case 11: pilihan8();break;

                }

                if (p0_4 == 0)
        {
        ucon = 0x20;
  delayy(500000);

  te_ulcl = 1;                          /* Transmision
enabled */

  ultb = 0x43;
  while ( ti_ulcl == 0 )                                // tung
semua
    ;

  te_ulcl = 0;                          /* Transmision
disabled */

  delayy(500000);
  ucon = 0x00;
 }

                if (p0_6 == 0)
        {
  ucon = 0x20;
  delayy(500000);

  te_ulcl = 1;                          /* Transmision
enabled */
```

```
  ultb = c ;
  while ( ti_ulcl == 0 )                                    // tung
semua
    ;

  te_ulcl = 0;                              /* Transmision
disabled */

  delayy(500000);
  ucon = 0x00;
 }

        if (p0_5 == 0)
        {
        ucon = 0x20;
  delayy(500000);

  te_ulcl = 1;                              /* Transmision
enabled */

  ultb = 0x41;
  while ( ti_ulcl == 0 )                                    // tung
semua
    ;

  te_ulcl = 0;                              /* Transmision
disabled */

  delayy(500000);
  ucon = 0x00;
 }
        }else
                if (  p1_6 == SW_OFF )
                {
                p1_6 == 0;
            p1_5 == 1;
                Tulis_LCD(0x80,"MAAF SALDO ANDA ");
            Tulis_LCD(0xC0,"  TIDAK CUKUP   ");
        delayy(500000);
        delayy(500000);
        delayy(500000);
        delayy(500000);
        delayy(500000);
        delayy(500000);
        p1_6 == 0;
        p1_5 == 1;
                        }else
                        if (  p1_5 == SW_OFF )
                {
        Tulis_LCD(0x80," DEKATKAN KARTU ");
        Tulis_LCD(0xC0," ANDA PD SISTEM ");
        delayy(500000);
```

```
}else
                        if (  pl_4== SW_OFF )
                    ;
        Tulis_LCD(0x80,"TERIMAKASIH ATAS");
        Tulis_LCD(0xC0,"KUNJUNGAN ANDA  ");
        delayy(500000);
        }
}
}
```

```
nit cobaserial;

nterface

ses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, VaClasses, VaComm, Mask, DBCtrls, ExtCtrls, Buttons,
  DB, DBTables, Grids, DBGrids,

  Menus,Registry,
  MMSystem,
   QuickRpt, QRCtrls;

  type
  TForm1 = class(TForm)
    VaComm1: TVaComm;
    Edit1: TEdit;
    Memo1: TMemo;
    DBEdit1: TDBEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Edit10: TEdit;
    Edit11: TEdit;
    Edit12: TEdit;
    Edit13: TEdit;
    Edit14: TEdit;
    Edit15: TEdit;
    Edit16: TEdit;
    Edit17: TEdit;
    Edit18: TEdit;
    Edit19: TEdit;
    Edit20: TEdit;
    Edit21: TEdit;
    Edit22: TEdit;
    Edit23: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit24: TEdit;
    Panel1: TPanel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Panel3: TPanel;
    BitBtn1: TBitBtn;
    Panel2: TPanel;
    GroupBox1: TGroupBox;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Timer1: TTimer;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    Label15: TLabel;
    Label16: TLabel;
    DBEdit7: TDBEdit;
    DBEdit8: TDBEdit;
    Label13: TLabel;
    Label17: TLabel;
    GroupBox2: TGroupBox;
    Label14: TLabel;
```

```
    Label18: TLabel;
    Label19: TLabel;
    Edit25: TEdit;
    GroupBox3: TGroupBox;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Edit26: TEdit;
    GroupBox4: TGroupBox;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Edit27: TEdit;
    GroupBox5: TGroupBox;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Edit28: TEdit;
    Edit29: TEdit;
    Edit30: TEdit;
    Edit31: TEdit;
    Edit32: TEdit;
    Label31: TLabel;
    Button5: TButton;
    Button7: TButton;
    DataSource1: TDataSource;
    Table1: TTable;
    Table1NOMOR: TStringField;
    Table1NOID: TStringField;
    Table1NAMA: TStringField;
    Table1ALAMAT: TStringField;
    Table1TANGGAL: TDateField;
    Table1JAM: TTimeField;
    Table1DEBET: TStringField;
    Table1KREDIT: TStringField;
    Table1SALDO: TStringField;
    Table1PAHE1: TStringField;
    Table1PAHE2: TStringField;
    Table1PAHE3: TStringField;
    Table1KOPISUSU: TStringField;
    Table1KOPI: TStringField;
    Table1SOFTDRINK: TStringField;
    Table1BAKSOKIKIL: TStringField;
    Table1BAKSOBAKAR: TStringField;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Button3: TButton;
    Button4: TButton;
    Timer2: TTimer;
    Timer3: TTimer;
    Timer4: TTimer;
    Label32: TLabel;
    DBEdit9: TDBEdit;
    DBEdit10: TDBEdit;
    DBEdit11: TDBEdit;
    DBEdit12: TDBEdit;
    DBEdit13: TDBEdit;
    DBEdit14: TDBEdit;
    DBEdit15: TDBEdit;
    DBEdit16: TDBEdit;
    DBEdit17: TDBEdit;
    Edit33: TEdit;
    Edit34: TEdit;
    Edit35: TEdit;
    Edit36: TEdit;
    Edit37: TEdit;
    Edit38: TEdit;
    Edit39: TEdit;
    Edit40: TEdit;
    Edit41: TEdit;
    Edit42: TEdit;
```

```
    Edit43: TEdit;
    Edit44: TEdit;
    Edit45: TEdit;
    Edit46: TEdit;
    Edit47: TEdit;
    Edit48: TEdit;
    procedure VaComm1RxChar(Sender: TObject; Count: Integer);
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
    procedure Timer4Timer(Sender: TObject);
  private
    { Private declarations }
    Pesan : String;
  public
  procedure Delay(lama:longint);
    { Public declarations }
  end;

ar
 Form1: TForm1;

mplementation

ses DURO1;
rocedure TForm1.Delay(lama:longint);
ar ref:longint;
egin
ef:=GetTickCount;
epeat
pplication.ProcessMessages ;
ntil ((getTickCount-ref)>=lama);
nd;


$R *.dfm}

rocedure TForm1.VaComm1RxChar(Sender: TObject; Count: Integer);
ar

    Hasil : array [1..16] of integer;
    Data,data1,data2 : string;
    k,a,b,JUMKARAK,JUMKARAK1,JUMKARAK2 : integer;
     lembaran,lembaran1,lembaran2 : string;
 I: Integer;
 Tmp: string;
 dataseri : byte;
egin
 //baca data yang diterima oleh comport
 Tmp := VaComm1.ReadText;
  Form1.Memo1.Text := '';
  edit2.Text :='';
  Pesan := '';
   for I := 1 to Length(Tmp) do
   case Tmp[I] of
     #10:; //lewatkan
     #13:  //Tunggu Enter
       begin
           form1.Memo1.Lines.Add(Pesan);;
         Pesan := '';  //reset Pesan
       end;
     else //bukan #10 atau #13
       Pesan := Pesan + Tmp[I];
       form1.Edit2.Text := Pesan;
       form1.Edit3.Text := form1.Memo1.Text;
```

```
            edit4.Text :=' ';
            edit5.Text :=' ';
            edit6.Text :=' ';
            edit7.Text :=' ';
            edit8.Text :=' ';
            edit9.Text :=' ';
            edit10.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 :=' ';

           data2 := edit3.Text;
          JUMKARAK2 := LENGTH (EDIT3.Text);
          form1.Edit23.Text := floattostr(jumkarak2);
          form1.Timer2.Enabled := true;
        begin
          for b := 1 to jumkarak2 do

        Begin
           lembaran2 := Data2[b];
           case b of
           1: edit4.Text :=lembaran2;
           2: edit5.Text :=lembaran2;
           3: edit6.Text :=lembaran2;
           4: edit7.Text :=lembaran2;
           5: edit8.Text :=lembaran2;
           6: edit9.Text :=lembaran2;
           7: edit10.Text :=lembaran2;
           8: edit11.Text :=lembaran2;
           9: edit12.Text :=lembaran2;
           10: edit13.Text :=lembaran2;
           11: edit14.Text :=lembaran2;
           12: edit15.Text :=lembaran2;
           13: edit16.Text :=lembaran2;
           14: edit17.Text :=lembaran2;
           15: edit18.Text :=lembaran2;
           16: edit19.Text :=lembaran2;
           17: edit20.Text :=lembaran2;
           18: edit21.Text :=lembaran2;
           19: edit22.Text :=lembaran2;
           20: edit23.Text :=lembaran2;

        end;

    end;

     end;

nd;
nd;

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;
          end;
    end; }
procedure TForm1.FormCreate(Sender: TObject);
begin
form1.Timer2.Enabled := false;
form1.Timer3.Enabled := false;
form1.Timer4.Enabled := false;
FORM1.GroupBox2.Hide;
FORM1.GroupBox3.Hide;
FORM1.GroupBox4.Hide;
FORM1.Edit48.Hide;
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.Edit10.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.Edit28.Hide;
FORM1.Edit29.Hide;
FORM1.Edit30.Hide;
FORM1.Edit31.Hide;
FORM1.Edit32.Hide;
FORM1.Edit33.Hide;
FORM1.Edit34.Hide;
FORM1.Edit35.Hide;
FORM1.Edit36.Hide;
FORM1.Edit37.Hide;
FORM1.Edit38.Hide;
FORM1.Edit39.Hide;
FORM1.Edit40.Hide;
FORM1.Edit41.Hide;
FORM1.Edit42.Hide;
FORM1.Edit43.Hide;
FORM1.Edit44.Hide;
FORM1.Edit45.Hide;
FORM1.Edit46.Hide;
FORM1.Edit47.Hide;
FORM1.DBEdit1.Hide;
FORM1.DBEdit9.Hide;
FORM1.DBEdit10.Hide;
FORM1.DBEdit11.Hide;
FORM1.DBEdit12.Hide;
FORM1.DBEdit13.Hide;
FORM1.DBEdit14.Hide;
FORM1.DBEdit15.Hide;
FORM1.DBEdit16.Hide;
FORM1.DBEdit17.Hide;

..Form1.Open;
nd;

rocedure TForm1.Button1Click(Sender: TObject);
```

```
begin
VaComm1.WriteText(Edit1.Text);
 VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
end;

procedure TForm1.Button2Click(Sender: TObject);
var
       dataseri : STRING; tipe data kode komponen /variabel
begin
dataseri:= form1.Edit2.Text;

     begin
     label6.Caption :='Conect';
     if dataseri = '1' then
      Begin
          form1.Label1.Show;
          form1.Label2.Hide;
          form1.Label3.Hide;
          form1.Label4.Hide;
          form1.Label5.Hide;

      end;
      if dataseri = '2' then
      Begin

          form1.Label2.Show;
          form1.Label1.Hide;
          form1.Label3.Hide;
          form1.Label4.Hide;
          form1.Label5.Hide;
      end;
     if dataseri = '3' then
      begin
          form1.Label3.Show;
         form1.Label2.Hide;
         form1.Label1.Hide;
         form1.Label4.Hide;
         form1.Label5.Hide;

      end;

       if dataseri = '4' then
      begin
          form1.Label4.Show;
         form1.Label2.Hide;
         form1.Label3.Hide;
         form1.Label1.Hide;
         form1.Label5.Hide;

      end;
      if dataseri = '5' then
      begin
          form1.Label5.Show;
         form1.Label2.Hide;
         form1.Label3.Hide;
         form1.Label4.Hide;
         form1.Label1.Hide;
      end;


end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
 Jam,Menit,Detik,MiliDetik : Word;
 Jam1,menit1,detik1,msec1:word;
   Totalwaktu           :TDateTime;

begin
  DecodeTime(Time, Jam , Menit, Detik, MiliDetik);
  Panel3.Caption :=  IntToStr(Jam) +':';
```

```
                    IntToStr(Menit)+':'+
                    IntToStr(Detik);

nd;

rocedure TForm1.Button5Click(Sender: TObject);
ar
  kodeBintang1 :integer;
  StrCode12,StrCode2, StrCode1, StrCode3, StrCode4 : string;
  trCode5, trCode6, trCode7, trCode8 : string;
  trCode9, trCode10, trCode11, trCode12 : string;
egin
     FORM1.Table1.Insert;
     {kodeBintang1 := 0;}
     StrCode12 := InputBox ('MASUKKAN NAMA PELANGGAN ',
                    ' NAMA : ',
                    '');

    FORM1.Table1NAMA.Text := StrCode12;

    StrCode1 := InputBox ('MASUKKAN ALAMAT PELANGGAN  ',
                    ' ALAMAT : ',
                    '');

     FORM1.Table1ALAMAT.Text := StrCode1;


     StrCode3 := InputBox ('MASUKKAN JUMLAH UANG ',
                    ' DEBET : ',
                    '');

     FORM1.Table1DEBET.Text := StrCode3;

      StrCode4 := InputBox ('MASUKKAN NOMOR SERI KARTU',
                    ' NOMOR ID : ',
                    '');

     FORM1.Table1NOID.Text := StrCode4;


     Form1.Table1TANGGAL.AsDateTime:=date;
     Form1.Table1JAM.AsDateTime:=TIME;
     FORM1.Table1.Post;
nd;

rocedure TForm1.Button7Click(Sender: TObject);
egin
orm2.showmodal;
nd;

rocedure TForm1.BitBtn1Click(Sender: TObject);
egin
pplication.Terminate ;
nd;

rocedure TForm1.Button3Click(Sender: TObject);
ar x:integer;
  caridata:String;
  INAMAKOMPONEN : string;

begin

 INAMAKOMPONEN := InputBox('PENCARIAN MENURUT NOMOR KARTU PELANGGAN ',
                    'NOMOR : ',
                    '');

{caridata:=form1.Edit8.Text;  }
 FORM1.Table1.IndexName := 'NOMORID' ;
 if (not table1.FindKey ([INAMAKOMPONEN])) then
 Application.MessageBox ('Data Tidak Ditemukan',
 'Informasi',MB OK or MB_ICONHAND);
table1.IndexName : '';
```

```
nd;

rocedure TForm1.Button4Click(Sender: TObject);
ar x:integer;
  caridata:String;
  INAMAKOMPONEN : string;
egin

 INAMAKOMPONEN := InputBox('PENCARIAN MENURUT NAMA PELANGGAN ',
                           'NAMA : ',
                           '');

{caridata:=form1.Edit8.Text;  }
 FORM1.Table1.IndexName := 'NAMAA' ;
 if (not table1.FindKey ([INAMAKOMPONEN])) then
 Application.MessageBox ('Data Tidak Ditemukan',
 'Informasi',MB_OK or MB_ICONHAND);
table1.IndexName :='';

nd;

rocedure TForm1.Timer2Timer(Sender: TObject);
ar
      dataseri : STRING;
egin
orm1.Timer2.Enabled := false;
ataseri:= form1.Edit23.Text;

    begin
    label32.Caption :='Conect';
    if dataseri = '1' then
     Begin
     form1.Timer2.Enabled := false;
     form1.Timer4.Enabled := false;
     form1.Timer3.Enabled := true;
      end;
     if dataseri = '8' then
     Begin
     form1.Timer2.Enabled := false;
     form1.Timer3.Enabled := false;
     form1.Timer4.Enabled := true;

      end;



nd;
nd;

rocedure TForm1.Timer3Timer(Sender: TObject);
var
 r,s,t,u,v,w,x,y,z : real;
      dataseri : STRING;
egin
form1.Timer3.Enabled := false;
ataseri:= form1.Edit2.Text;

    begin
    label6.Caption :='Conect';
    if dataseri = '1' then
     Begin
        form1.GroupBox2.Show;
        form1.Edit33.Text := form1.Edit25.Text;
     end;
     if dataseri = '2' then
     Begin
        form1.GroupBox3.Show;
        form1.Edit34.Text := form1.Edit26.Text;
     end;
    if dataseri = '3' then
     begin
         form1.GroupBox4.Show;
        form1.Edit35.Text := form1.Edit27.Text;
```

```
      end;

       if dataseri = '4' then
      begin
         form1.Edit28.Show;
         form1.Edit36.Text := form1.Edit28.Text;

      end;
      if dataseri = '5' then
      begin
          form1.Edit29.Show;
         form1.Edit37.Text := form1.Edit29.Text;

      end;
      if dataseri = '6' then
      begin
          form1.Edit30.Show;
         form1.Edit38.Text := form1.Edit30.Text;
      end;
      if dataseri = '7' then
      begin
          form1.Edit31.Show;
         form1.Edit39.Text := form1.Edit31.Text;

      end;
      if dataseri = '8' then
      begin
          form1.Edit32.Show;
         form1.Edit40.Text := form1.Edit32.Text;

      end;
        if dataseri = 'C' then
      begin

         form1.Edit33.Text := '0';
         form1.Edit34.Text := '0';
         form1.Edit35.Text := '0';
         form1.Edit36.Text := '0';
         form1.Edit37.Text := '0';
         form1.Edit38.Text := '0';
         form1.Edit39.Text := '0';
         form1.Edit40.Text := '0';
        FORM1.GroupBox2.Hide;
ORM1.GroupBox3.Hide;
ORM1.GroupBox4.Hide;
ORM1.Edit28.Hide;
ORM1.Edit28.Hide;
ORM1.Edit29.Hide;
ORM1.Edit30.Hide;
ORM1.Edit31.Hide;
ORM1.Edit32.Hide;
      end;
if dataseri = 'A' then
      begin
      Form1.Table1.Edit ;
      Form1.Table1JAM.AsDateTime:=time;
      r := StrToFloat(form1.Edit33.Text);
      s := StrToFloat(form1.Edit34.Text);
      t := StrToFloat(form1.Edit35.Text);
      u := StrToFloat(form1.Edit36.Text);
      v := StrToFloat(form1.Edit37.Text);
      w := StrToFloat(form1.Edit38.Text);
      x := StrToFloat(form1.Edit39.Text);
      y := StrToFloat(form1.Edit40.Text);
      form1.Table1PAHE1.Value := form1.Edit33.text;
      form1.Table1PAHE2.Value := form1.Edit34.text;
      form1.Table1PAHE3.Value := form1.Edit35.text;
      form1.Table1KOPISUSU.Value := form1.Edit36.text;
      form1.Table1KOPI.Value := form1.Edit37.text;
      form1.Table1SOFTDRINK.Value := form1.Edit38.text;
      form1.Table1BAKSOKIKIL.Value := form1.Edit39.text;
      form1.Table1BAKSOBAKAR.Value := form1.Edit40.text;
```

```
      form1.Table1DEBET.Text := form1.Table1SALDO.Text;
       z := z + s + t - u + v + w + x + y ;
      Form1.Table1KREDIT.Value := FloatToStr (z);
      form1.Edit42.Text := form1.Table1KREDIT.Text;
      form1.Edit43.Text := form1.Table1DEBET.Text;
       w := StrToFloat(form1.Edit42.Text);
       y := StrToFloat(form1.Edit43.Text);
       z := y-w;

       if z >= 0 then
       begin
       form1.Table1SALDO.Value := FloatToStr (z);
       Form1.Table1.Post ;
       form1.Edit33.Text := '0';
         form1.Edit34.Text := '0';
         form1.Edit35.Text := '0';
         form1.Edit36.Text := '0';
         form1.Edit37.Text := '0';
         form1.Edit38.Text := '0';
         form1.Edit39.Text := '0';
         form1.Edit40.Text := '0';
         FORM1.GroupBox2.Hide;
ORM1.GroupBox3.Hide;
ORM1.GroupBox4.Hide;
ORM1.Edit28.Hide;
ORM1.Edit28.Hide;
ORM1.Edit29.Hide;
ORM1.Edit30.Hide;
ORM1.Edit31.Hide;
ORM1.Edit32.Hide;
      VaComm1.WriteText(Edit48.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
Delay(100);
VaComm1.WriteText(Edit47.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
      end;
     if z < 0 then
      begin
      form1.Edit33.Text := '0';
        form1.Edit34.Text := '0';
        form1.Edit35.Text := '0';
        form1.Edit36.Text := '0';
        form1.Edit37.Text := '0';
        form1.Edit38.Text := '0';
        form1.Edit39.Text := '0';
        form1.Edit40.Text := '0';
        FORM1.GroupBox2.Hide;
ORM1.GroupBox3.Hide;
ORM1.GroupBox4.Hide;
ORM1.Edit28.Hide;
ORM1.Edit28.Hide;
ORM1.Edit29.Hide;
ORM1.Edit30.Hide;
ORM1.Edit31.Hide;
ORM1.Edit32.Hide;
orm1.Table1PAHE1.Value := form1.Edit33.text;
     form1.Table1PAHE2.Value := form1.Edit34.text;
     form1.Table1PAHE3.Value := form1.Edit35.text;
     form1.Table1KOPISUSU.Value := form1.Edit36.text;
     form1.Table1KOPI.Value := form1.Edit37.text;
     form1.Table1SOFTDRINK.Value := form1.Edit38.text;
     form1.Table1BAKSOKIKIL.Value := form1.Edit39.text;
     form1.Table1BAKSOBAKAR.Value := form1.Edit40.text;
     form1.Table1DEBET.Text := form1.Table1SALDO.Text;
      form1.Table1SALDO.Value := form1.Table1DEBET.Text;
      Form1.Table1KREDIT.Value := '0';
      Form1.Table1.Post ;
VaComm1.WriteText(Edit46.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
 Delay(100);
VaComm1.WriteText(Edit47.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
      end;
```

```
    end;

nd;
nd;


rocedure TForm1.Timer4Timer(Sender: TObject);
ar x:integer;
  caridata:String;
  INAMAKOMPONEN : string;
egin
orm1.Timer4.Enabled := false;
        INAMAKOMPONEN := form1.Edit10.Text;

{caridata:=form1.Edit8.Text;  }
 FORM1.Table1.IndexName := 'NOMORI' ;
 if (not table1.FindKey ([INAMAKOMPONEN])) then
 Application.MessageBox ('Data Tidak Ditemukan',
 'Informasi',MB_OK or MB_ICONHAND);
table1.IndexName :='';
VaComm1.WriteText(Edit45.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
nd;

nd.
```

1. Overview

This MCU is built using the high-performance silicon gate CMOS process using a R8C/Tiny Series CPU core and is packaged in a 32-pin plastic molded LQFP. This MCU operates using sophisticated instructions featuring a high level of instruction efficiency. With 1M bytes of address space, it is capable of executing instructions at high speed.
The data flash ROM (2 KB X 2 blocks) is embedded.

1.1 Applications

Electric household appliance, office equipment, housing equipment (sensor, security), general industrial equipment, audio, etc.

1.2 Performance Outline

Table 1.1. lists the performance outline of this MCU.

Table 1.1 Performance outline

Item		Performance
CPU	Number of basic instructions	89 instructions
	Shortest instruction execution time	50 ns (f(XIN) = 20 MHz, Vcc = 3.0 to 5.5 V) 100 ns (f(XIN) = 10 MHz, Vcc = 2.7 to 5.5 V)
	Operating mode	Single-chip
	Address space	1M bytes
	Memory capacity	See Table 1.2.
Peripheral function	Interrupt	Internal: 11 factors, External: 5 factors, Software: 4 factors, Priority level: 7 levels
	Watchdog timer	15 bits x 1 (with prescaler) Reset start function selectable
	Timer	Timer X: 8 bits x 1 channel, Timer Y: 8 bits x 1 channel, Timer Z: 8 bits x 1 channel (Each timer equipped with 8-bit prescaler) Timer C: 16 bits x 1 channel Circuits of input capture and output compare.
	Serial interface	•1 channel Clock synchronous, UART •1 channel UART
	A/D converter	10-bit A/D converter: 1 circuit, 12 channels
	Clock generation circuit	2 circuits •Main clock generation circuit (Equipped with a built-in feedback resistor) •On-chip oscillator (high-speed, low-speed) On high-speed on-chip oscillator the frequency adjustment function is usable.
	Oscillation stop detection function	Stop detection of main clock oscillation
	Voltage detection circuit	Included
	Power on reset circuit	Included
	Port	Input/Output: 22 (including LED drive port), Input: 2 (LED drive I/O port: 8)
Electrical characteristics	Power supply voltage	Vcc = 3.0 to 5.5V (f(XIN) = 20MHz) Vcc = 2.7 to 5.5V (f(XIN) = 10MHz)
	Power consumption	Typ.9 mA (Vcc = 5.0V, (f(XIN) = 20MHz,High-speed mode) Typ.5 mA (Vcc = 3.0V, (f(XIN) = 10MHz,High-speed mode) Typ.35 µA (Vcc = 3.0V, Wait mode, Peripheral clock stops) Typ.0.7 µA (Vcc = 3.0V, Stop mode)
Flash memory	Program/erase voltage	Vcc = 2.7 to 5.5 V
	Number of program/erase	10,000 times (Data area)
		1,000 times (Program area)
Operating ambient temperature		-20 to 85°C -40 to 85°C (D-version)
Package		32-pin plastic mold LQFP

1.3 Block Diagram

Figure 1.1 shows this MCU block diagram.

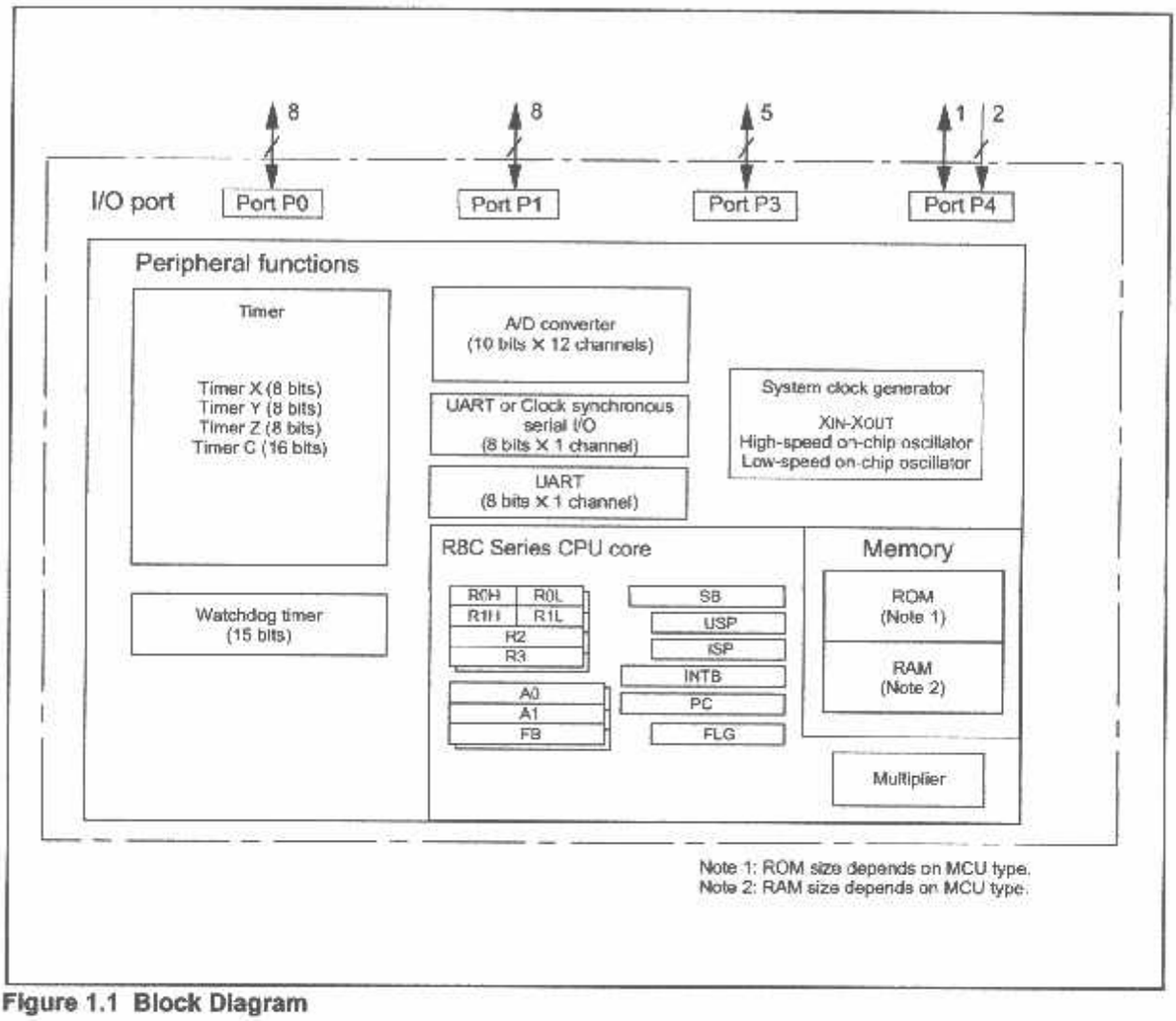


Figure 1.1 Block Diagram

1.4 Product Information

Table 1.2 lists the products.

Table 1.2 Product List

As of April 2005

Type No.	ROM capacity		RAM capacity	Package type	Remarks
	Program area	Data area			
R5F21132FP	8K bytes	2K bytes x 2	512 bytes	PLQP0032GB-A	Flash memory version
R5F21133FP	12K bytes	2K bytes x 2	768 bytes	PLQP0032GB-A	
R5F21134FP	16K bytes	2K bytes x 2	1K bytes	PLQP0032GB-A	
R5F21132DFP	8K bytes	2K bytes x 2	512 bytes	PLQP0032GB-A	D version
R5F21133DFP	12K bytes	2K bytes x 2	768 bytes	PLQP0032GB-A	
R5F21134DFP	16K bytes	2K bytes x 2	1K bytes	PLQP0032GB-A	

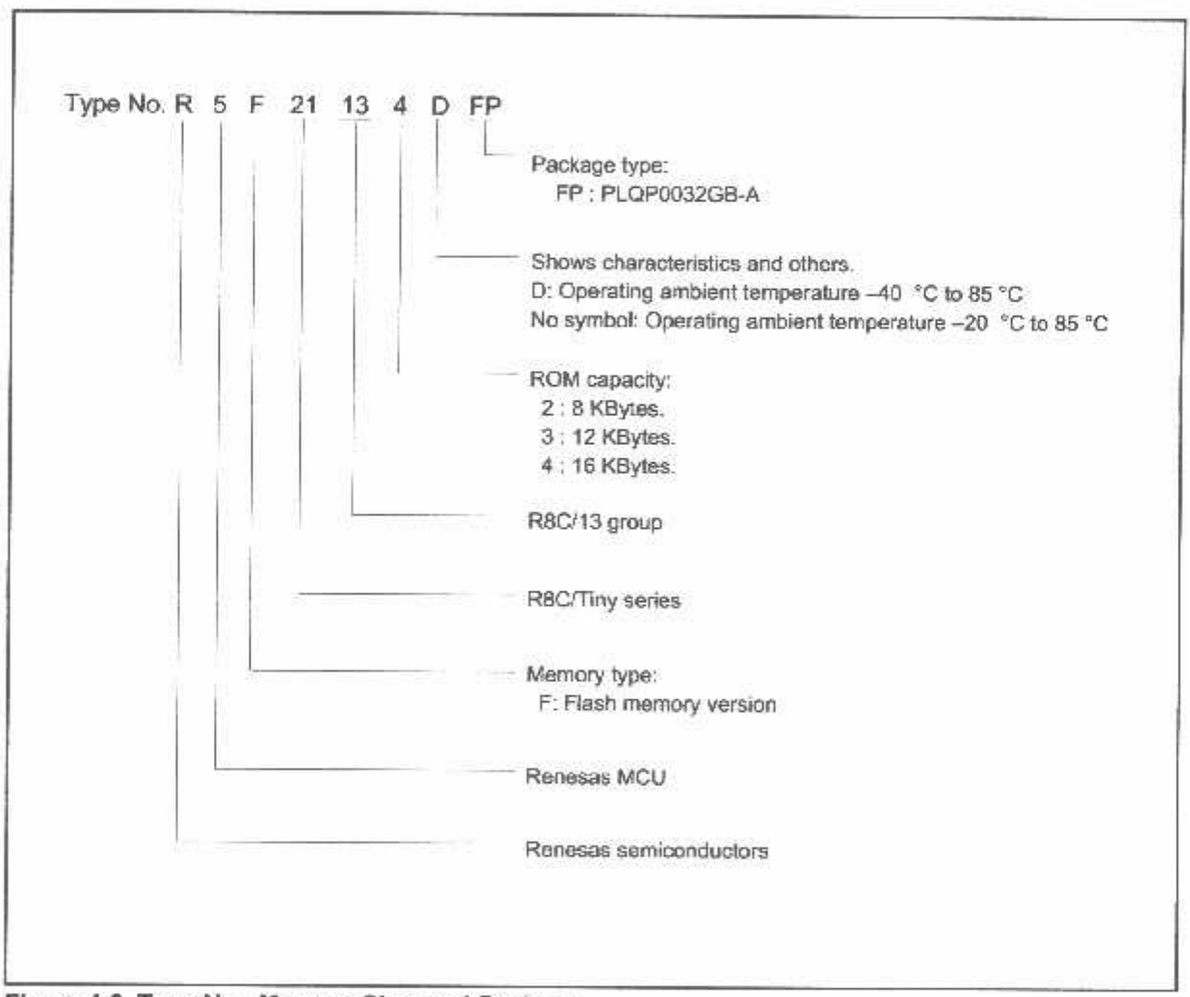


Figure 1.2 Type No., Memory Size, and Package

I.5 Pin Assignments

Figure 1.3 shows the pin configuration (top view).

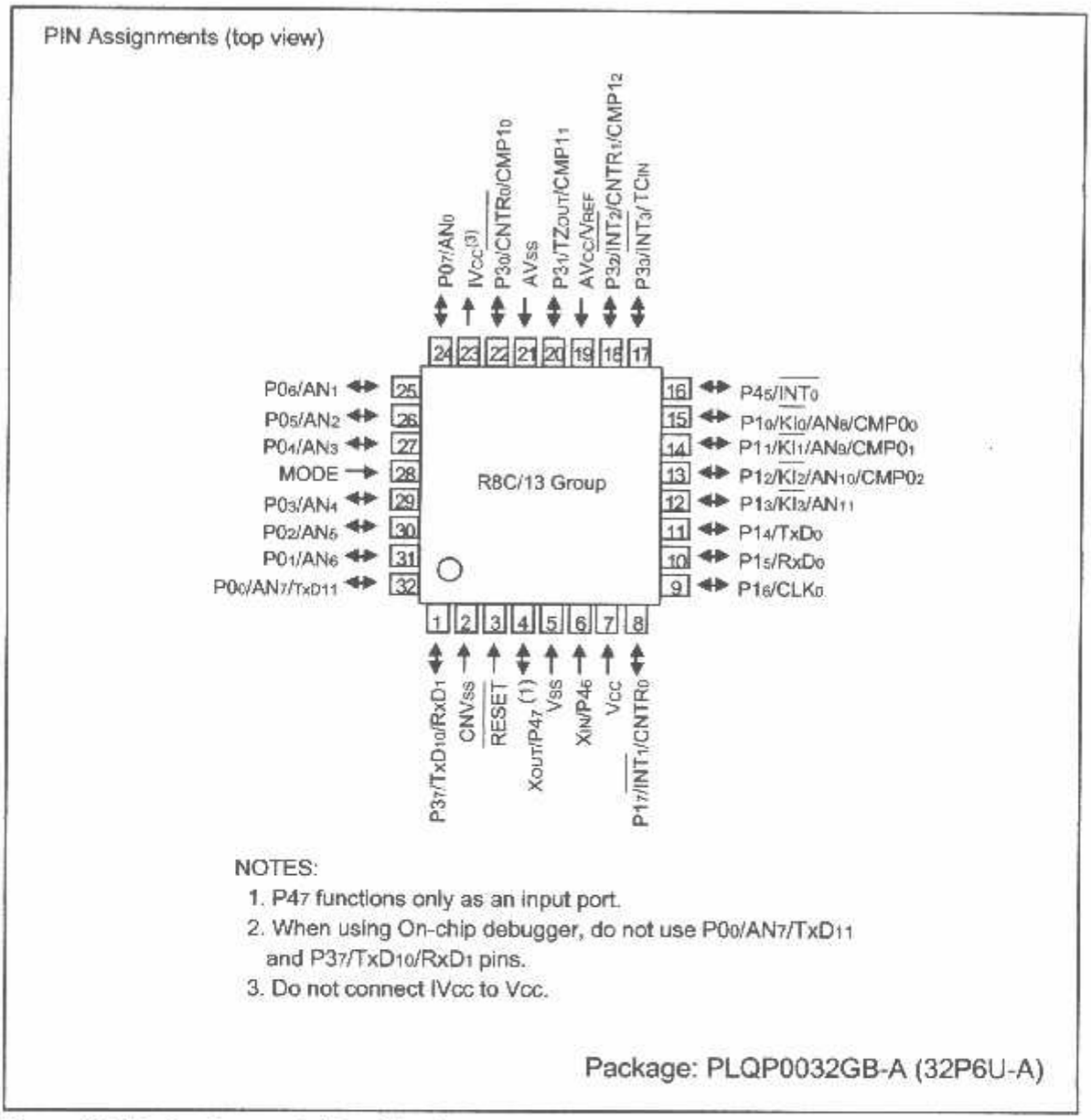


Figure 1.3 Pin Assignments (Top View)

1.6 Pin Description

Table 1.3 shows the pin description

Table 1.3 Pin description

Signal name	Pin name	I/O type	Function
Power supply input	Vcc, Vss	I	Apply 2.7 V to 5.5 V to the Vcc pin. Apply 0 V to the Vss pin.
IVcc	IVcc	O	This pin is to stabilize internal power supply Connect this pin to Vss via a capacitor (0.1 µF) Do not connect to Vcc
Analog power supply input	AVcc, AVss	I	These are power supply input pins for A/D converter. Connect the AVcc pin to Vcc. Connect the AVss pin to Vss. Connect a capacitor between pins AVcc and AVss.
Reset input	$\overline{\text{RESET}}$	I	"L" on this input resets the MCU.
CNVss	CNVss	I	Connect this pin to Vss via a resistor(1)
MODE	MODE	I	Connect this pin to Vcc via a resistor
Main clock input	XIN	I	These pins are provided for the main clock generating circuit I/O. Connect a ceramic resonator or a crystal oscillator between the XIN and XOUT pins. To use an externally derived clock, input it to the XIN pin and leave the XOUT pin open.
Main clock output	XOUT	O	
INT interrupt input	$\overline{\text{INT0}}$ to $\overline{\text{INT3}}$	I	These are $\overline{\text{INT}}$ interrupt input pins.
Key input interrupt input	$\overline{\text{KI0}}$ to $\overline{\text{KI3}}$	I	These are key input interrupt pins.
Timer X	CNTR0	I/O	This is the timer X I/O pin.
	$\overline{\text{CNTR0}}$	O	This is the timer X output pin.
Timer Y	CNTR1	I/O	This is the timer Y I/O pin.
Timer Z	TZOUT	O	This is the timer Z output pin.
Timer C	TCIN	I	This is the timer C input pin.
	CMP00 to CMP03, CMP10 to CMP13	O	These are the timer C output pins.
Serial interface	CLK0	I/O	This is a transfer clock I/O pin.
	RxD0, RxD1	I	These are serial data input pins.
	TxD0, TxD10, TxD11	O	These are serial data output pins.
Reference voltage input	VREF	I	This is a reference voltage input pin for A/D converter. Connect the VREF pin to Vcc.
A/D converter	AN0 to AN11	I	These are analog input pins for A/D converter.
I/O port	P00 to P07, P10 to P17, P30 to P33, P37, P45	I/O	These are 8-bit CMOS I/O ports. Each port has an I/O select direction register, allowing each pin in that port to be directed for input or output individually. Any port set to input can select whether to use a pull-up resistor or not by program. P10 to P17 also function as LED drive ports.
Input port	P46, P47	I	These are input only pins.

2. Central Processing Unit (CPU)

Figure 2.1 shows the CPU registers. The CPU has 13 registers. Of these, R0, R1, R2, R3, A0, A1 and FB comprise a register bank. There are two register banks.

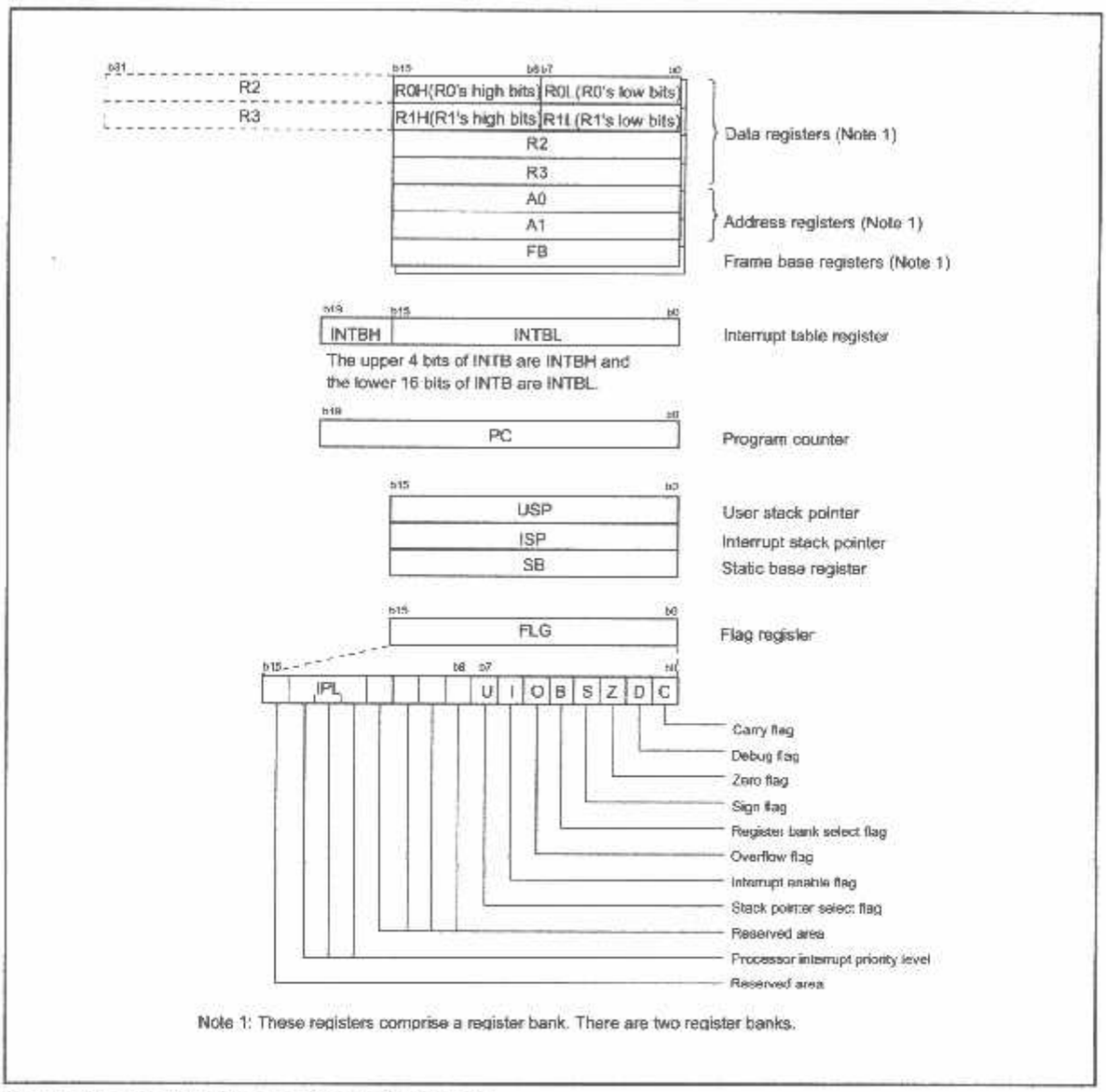


Figure 2.1 Central Processing Unit Register

2.1 Data Registers (R0, R1, R2 and R3)

The R0 register consists of 16 bits, and is used mainly for transfers and arithmetic/logic operations. R1 to R3 are the same as R0.

The R0 register can be separated between high (R0H) and low (R0L) for use as two 8-bit data registers. R1H and R1L are the same as R0H and R0L. Conversely, R2 and R0 can be combined for use as a 32-bit data register (R2R0). R3R1 is the same as R2R0.

2.2 Address Registers (A0 and A1)

The register A0 consists of 16 bits, and is used for address register indirect addressing and address register relative addressing. They also are used for transfers and logic/logic operations. A1 is the same as A0.
In some instructions, registers A1 and A0 can be combined for use as a 32-bit address register (A1A0).

2.3 Frame Base Register (FB)

FB is configured with 16 bits, and is used for FB relative addressing.

2.4 Interrupt Table Register (INTB)

INTB is configured with 20 bits, indicating the start address of an interrupt vector table.

2.5 Program Counter (PC)

PC is configured with 20 bits, indicating the address of an instruction to be executed.

2.6 User Stack Pointer (USP) and Interrupt Stack Pointer (ISP)

Stack pointer (SP) comes in two types: USP and ISP, each configured with 16 bits.
Your desired type of stack pointer (USP or ISP) can be selected by the U flag of FLG.

2.7 Static Base Register (SB)

SB is configured with 16 bits, and is used for SB relative addressing.

2.8 Flag Register (FLG)

FLG consists of 11 bits, indicating the CPU status.

2.8.1 Carry Flag (C Flag)

This flag retains a carry, borrow, or shift-out bit that has occurred in the arithmetic/logic unit.

2.8.2 Debug Flag (D Flag)

The D flag is used exclusively for debugging purpose. During normal use, it must be set to "0".

2.8.3 Zero Flag (Z Flag)

This flag is set to "1" when an arithmetic operation resulted in 0; otherwise, it is "0".

2.8.4 Sign Flag (S Flag)

This flag is set to "1" when an arithmetic operation resulted in a negative value; otherwise, it is "0".

2.8.5 Register Bank Select Flag (B Flag)

Register bank 0 is selected when this flag is "0" ; register bank 1 is selected when this flag is "1".

2.8.6 Overflow Flag (O Flag)

This flag is set to "1" when the operation resulted in an overflow; otherwise, it is "0".

2.8.7 Interrupt Enable Flag (I Flag)

This flag enables a maskable interrupt.
Maskable interrupts are disabled when the I flag is "0", and are enabled when the I flag is "1". The I flag is cleared to "0" when the interrupt request is accepted.

2.8.8 Stack Pointer Select Flag (U Flag)

ISP is selected when the U flag is "0"; USP is selected when the U flag is "1".
The U flag is cleared to "0" when a hardware interrupt request is accepted or an INT instruction for software interrupt Nos. 0 to 31 is executed.

2.8.9 Processor Interrupt Priority Level (IPL)

IPL is configured with three bits, for specification of up to eight processor interrupt priority levels from level 0 to level 7.
If a requested interrupt has priority greater than IPL, the interrupt is enabled.

2.8.10 Reserved Area

When write to this bit, write "0". When read, its content is indeterminate.

3. Memory

Figure 3.1 is a memory map of this MCU. The address space extends the 1M bytes from address 0000016 to FFFFF16.

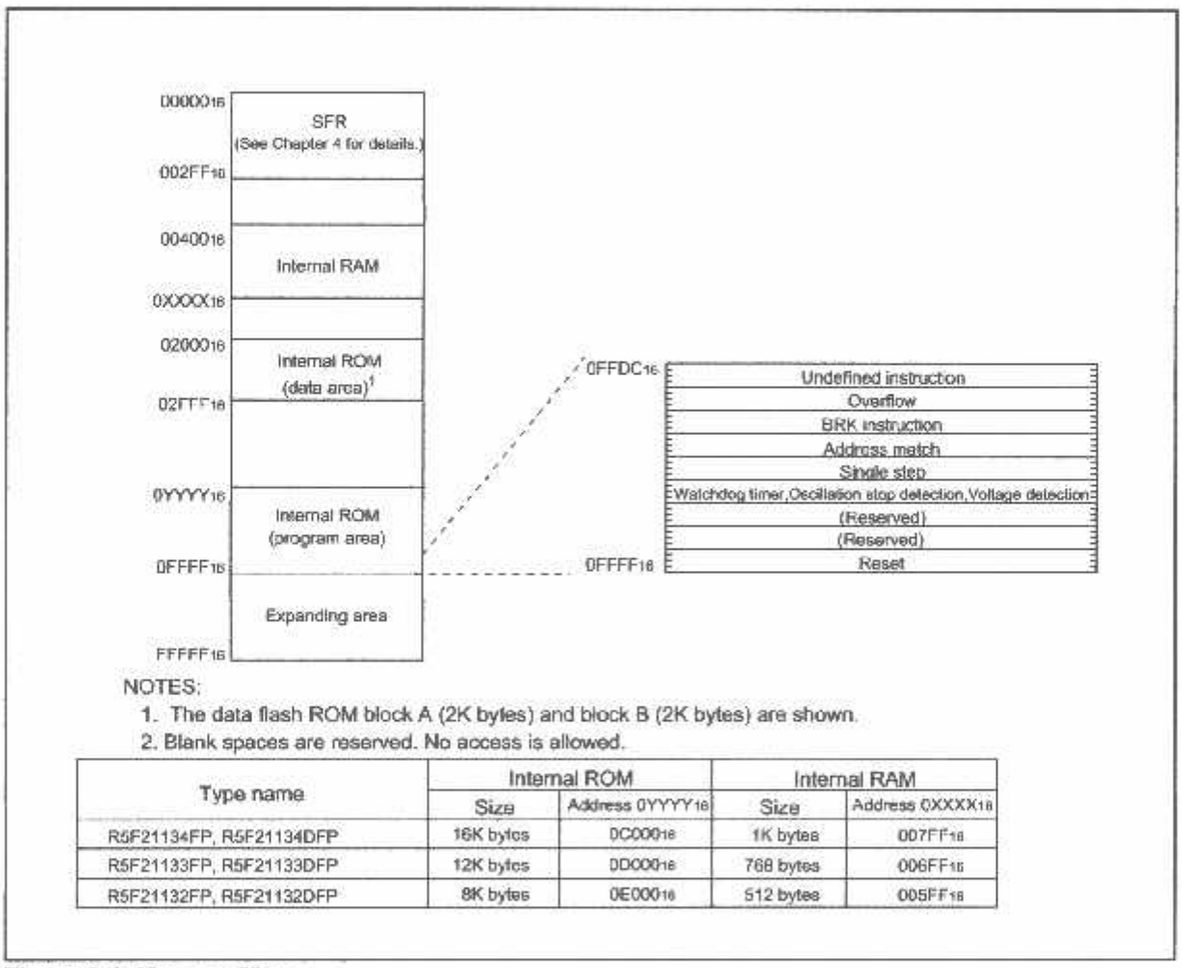
The internal ROM (program area) is allocated in a lower address direction beginning with address 0FFFF16. For example, a 16-Kbyte internal ROM is allocated to the addresses from 0C00016 to 0FFFF16.

The fixed interrupt vector table is allocated to the addresses from 0FFDC16 to 0FFFF16. Therefore, store the start address of each interrupt routine here.

The internal ROM (data area) is allocated to the addresses from 0200016 to 02FFF16.

The internal RAM is allocated in an upper address direction beginning with address 0040016. For example, a 1-Kbyte internal RAM is allocated to the addresses from 0040016 to 007FF16. In addition to storing data, the internal RAM also stores the stack used when calling subroutines and when interrupts are generated.

Special function registers (SFR) are allocated to the addresses from 0000016 to 002FF16. Peripheral function control registers are located here. Of the SFR, any space which has no functions allocated is reserved for future use and cannot be used by users.



NOTES:
1. The data flash ROM block A (2K bytes) and block B (2K bytes) are shown.
2. Blank spaces are reserved. No access is allowed.

Type name	Internal ROM		Internal RAM	
	Size	Address 0YYYY16	Size	Address 0XXXX16
R5F21134FP, R5F21134DFP	16K bytes	0C00016	1K bytes	007FF16
R5F21133FP, R5F21133DFP	12K bytes	0D00016	768 bytes	006FF16
R5F21132FP, R5F21132DFP	8K bytes	0E00016	512 bytes	005FF16

Figure 3.1 Memory Map

4. Special Function Register (SFR)

SFR(Special Function Register) is the control register of peripheral functions. Tables 4.1 to 4.4 list the SFR information

Table 4.1 SFR Information(1)[1]

Address	Register	Symbol	After reset
0000₁₆			
0001₁₆			
0002₁₆			
0003₁₆			
0004₁₆	Processor mode register 0 [1]	PM0	00₂6
0005₁₆	Processor mode register 1	PM1	00₂6
0006₁₆	System clock control register 0	CM0	01101000₂
0007₁₆	System clock control register 1	CM1	00₂000002
0008₁₆	High-speed on-chip oscillator control register 0	HR0	00₁6
0009₁₆	Address match interrupt enable register	AIER	XXXXXX002
000A₁₆	Protect register	PRCR	00XXX0002
000B₁₆	High-speed on-chip oscillator control register 1	HR1	40₁6
000C₁₆	Oscillation stop detection register	OCD	000001002
000D₁₆	Watchdog timer reset register	WDTR	XX16
000E₁₆	Watchdog timer start register	WDTS	XX16
000F₁₆	Watchdog timer control register	WDC	00011112
0010₁₆	Address match interrupt register 0	RMAD0	00₁6
0011₁₆			00₁6
0012₁₆			X0₁6
0013₁₆			
0014₁₆	Address match interrupt register 1	RMAD1	00₁6
0015₁₆			00₁6
0016₁₆			X0₁6
0017₁₆			
0018₁₆			
0019₁₆	Voltage detection register 1 [2]	VCR1	000010002
001A₁₆	Voltage detection register 2 [2]	VCR2	00₁6 [3]
			10000000₂ [4]
001B₁₆			
001C₁₆			
001D₁₆			
001E₁₆	INT0 input filter select register	INT0F	XXXXX0002
001F₁₆	Voltage detection interrupt register [2]	D4INT	00₁6 [3]
			01000012 [4]
0020₁₆			
0021₁₆			
0022₁₆			
0023₁₆			
0024₁₆			
0025₁₆			
0026₁₆			
0027₁₆			
0028₁₆			
0029₁₆			
002A₁₆			
002B₁₆			
002C₁₆			
002D₁₆			
002E₁₆			
002F₁₆			
0030₁₆			
0031₁₆			
0032₁₆			
0033₁₆			
0034₁₆			
0035₁₆			
0036₁₆			
0037₁₆			
0038₁₆			
0039₁₆			
003A₁₆			
003B₁₆			
003C₁₆			
003D₁₆			
003E₁₆			
003F₁₆			

X : Undefined
NOTES:
1. Blank columns are all reserved space. No access is allowed.
2. Software reset or the watchdog timer reset does not affect this register.
3. Owing to Reset input.
4. In the case of RESET pin = H retaining.

Table 4.2 SFR Information(2)(1)

Address	Register	Symbol	After reset
0040₁₆			
0041₁₆			
0042₁₆			
0043₁₆			
0044₁₆			
0045₁₆			
0046₁₆			
0047₁₆			
0048₁₆			
0049₁₆			
004A₁₆			
004B₁₆			
004C₁₆			
004D₁₆	Key input interrupt control register	KUPIC	XXXXX000₂
004E₁₆	AD conversion interrupt control register	ADIC	XXXXX000₂
004F₁₆			
0050₁₆	Compare 1 interrupt control register	CMP1IC	XXXXX000₂
0051₁₆	UART0 transmit interrupt control register	S0TIC	XXXXX000₂
0052₁₆	UART0 receive interrupt control register	S0RIC	XXXXX000₂
0053₁₆	UART1 transmit interrupt control register	S1TIC	XXXXX000₂
0054₁₆	UART1 receive interrupt control register	S1RIC	XXXXX000₂
0055₁₆	INT2 interrupt control register	INT2IC	XXXXX000₂
0056₁₆	Timer X interrupt control register	TXIC	XXXXX000₂
0057₁₆	Timer Y interrupt control register	TYIC	XXXXX000₂
0058₁₆	Timer Z interrupt control register	TZIC	XXXXX000₂
0059₁₆	INT1 interrupt control register	INT1IC	XXXXX000₂
005A₁₆	INT3 interrupt control register	INT3IC	XXXXX000₂
005B₁₆	Timer C interrupt control register	TCIC	XXXXX000₂
005C₁₆	Compare 0 interrupt control register	CMP0IC	XXXXX000₂
005D₁₆	INT0 interrupt control register	INT0IC	XX00X000₂
005E₁₆			
005F₁₆			
0060₁₆			
0061₁₆			
0062₁₆			
0063₁₆			
0064₁₆			
0065₁₆			
0066₁₆			
0067₁₆			
0068₁₆			
0069₁₆			
006A₁₆			
006B₁₆			
006C₁₆			
006D₁₆			
006E₁₆			
006F₁₆			
0070₁₆			
0071₁₆			
0072₁₆			
0073₁₆			
0074₁₆			
0075₁₆			
0076₁₆			
0077₁₆			
0078₁₆			
0079₁₆			
007A₁₆			
007B₁₆			
007C₁₆			
007D₁₆			
007E₁₆			
007F₁₆			

X : Undefined

NOTES:

1. Blank columns are all reserved space. No access is allowed.

Table 4.3 SFR Information(3)[1]

Address	Register	Symbol	After reset
0080₁₆	Timer Y, Z mode register	TYZMR	00₁₆
0081₁₆	Prescaler Y	PREY	FF₁₆
0082₁₆	Timer Y secondary	TYSC	FF₁₆
0083₁₆	Timer Y primary	TYPR	FF₁₆
0084₁₆	Timer Y, Z waveform output control register	PUM	00₁₆
0085₁₆	Prescaler Z	PREZ	FF₁₆
0086₁₆	Timer Z secondary	TZSC	FF₁₆
0087₁₆	Timer Z primary	TZPR	FF₁₆
0088₁₆			
0089₁₆			
008A₁₆	Timer Y, Z output control register	TYZOC	00₁₆
008B₁₆	Timer X mode register	TXMR	00₁₆
008C₁₆	Prescaler X	PREX	FF₁₆
008D₁₆	Timer X register	TX	FF₁₆
008E₁₆	Count source set register	TCSS	00₁₆
008F₁₆			
0090₁₆ 0091₁₆	Timer C register	TC	00₁₆ 00₁₆
0092₁₆			
0093₁₆			
0094₁₆			
0095₁₆			
0096₁₆	External input enable register	INTEN	00₁₆
0097₁₆			
0098₁₆	Key input enable register	KIEN	00₁₆
0099₁₆			
009A₁₆	Timer C control register 0	TCC0	00₁₆
009B₁₆	Timer C control register 1	TCC1	00₁₆
009C₁₆ 009D₁₆	Capture, compare 0 register	TM0	00₁₆ 00₁₆[2]
009E₁₆ 009F₁₆	Compare 1 register	TM1	FF₁₆ FF₁₆
00A0₁₆	UART0 transmit/receive mode register	U0MR	00₁₆
00A1₁₆	UART0 bit rate register	U0BRG	XX₁₆
00A2₁₆ 00A3₁₆	UART0 transmit buffer register	U0TB	XX₁₆ XX₁₆
00A4₁₆	UART0 transmit/receive control register 0	U0C0	000010002
00A5₁₆	UART0 transmit/receive control register 1	U0C1	000000102
00A6₁₆ 00A7₁₆	UART0 receive buffer register	U0RB	XX₁₆ XX₁₆
00A8₁₆	UART1 transmit/receive mode register	U1MR	00₁₆
00A9₁₆	UART1 bit rate register	U1BRG	XX₁₆
00AA₁₆ 00AB₁₆	UART1 transmit buffer register	U1TB	XX₁₆ XX₁₆
00AC₁₆	UART1 transmit/receive control register 0	U1C0	000010002
00AD₁₆	UART1 transmit/receive control register 1	U1C1	000000102
00AE₁₆ 00AF₁₆	UART1 receive buffer register	U1RB	XX₁₆ XX₁₆
00B0₁₆	UART transmit/receive control register 2	UCON	00₁₆
00B1₁₆			
00B2₁₆			
00B3₁₆			
00B4₁₆			
00B5₁₆			
00B6₁₆			
00B7₁₆			
00B8₁₆			
00B9₁₆			
00BA₁₆			
00BB₁₆			
00BC₁₆			
00BD₁₆			
00BE₁₆			
00BF₁₆			

X : Undefined

NOTES:

1. Blank columns are all reserved space. No access is allowed.
2. When the output compare mode is selected (the TCC13 bit in the TCC1 register = 1), the value is set to FFFF₁₆.

Table 4.4 SFR Information(4)(1)

Address	Register	Symbol	After reset
00C0₁₆	AD register	AD	XX₁₆
00C1₁₆			XX₁₆
00C2₁₆			
00C3₁₆			
00C4₁₆			
00C5₁₆			
00C6₁₆			
00C7₁₆			
00C8₁₆			
00C9₁₆			
00CA₁₆			
00CB₁₆			
00CC₁₆			
00CD₁₆			
00CE₁₆			
00CF₁₆			
00D0₁₆			
00D1₁₆			
00D2₁₆			
00D3₁₆			
00D4₁₆	AD control register 2	ADCON2	00₁₆
00D5₁₆			
00D6₁₆	AD control register 0	ADCON0	00000XXX2
00D7₁₆	AD control register 1	ADCON1	00₁₆
00D8₁₆			
00D9₁₆			
00DA₁₆			
00DB₁₆			
00DC₁₆			
00DD₁₆			
00DE₁₆			
00DF₁₆			
00E0₁₆	Port P0 register	P0	XX₁₆
00E1₁₆	Port P1 register	P1	XX₁₆
00E2₁₆	Port P0 direction register	PD0	00₁₆
00E3₁₆	Port P1 direction register	PD1	00₁₆
00E4₁₆			
00E5₁₆	Port P3 register	P3	XX₁₆
00E6₁₆			
00E7₁₆	Port P3 direction register	PD3	00₁₆
00E8₁₆	Port P4 register	P4	XX₁₆
00E9₁₆			
00EA₁₆	Port P4 direction register	PD4	00₁₆
00EB₁₆			
00EC₁₆			
00ED₁₆			
00EE₁₆			
00EF₁₆			
00F0₁₆			
00F1₁₆			
00F2₁₆			
00F3₁₆			
00F4₁₆			
00F5₁₆			
00F6₁₆			
00F7₁₆			
00F8₁₆			
00F9₁₆			
03FA₁₆			
00FB₁₆			
00FC₁₆	Pull-up control register 0	PUR0	00XX00002
00FD₁₆	Pull-up control register 1	PUR1	XXXXXX0X2
00FE₁₆	Port P1 drive capacity control register	DRR	00₁₆
00FF₁₆	Timer C output control register	TCOUT	00₁₆
01B3₁₆	Flash memory control register 4	FMR4	01000000₂
01B4₁₆			
01B5₁₆	Flash memory control register 1	FMR1	1000000X2
01B6₁₆			
01B7₁₆	Flash memory control register 0	FMR0	00000001₂
0FFFF₁₆	Option function select register (2)	OFS	Note 2

X : Undefined

NOTES:

1. The blank areas, 0100₁₆ to 01B2₁₆ and 01B8₁₆ to 02FF₁₆ are reserved and cannot be used by users.
2. The watchdog timer control bit is assigned. Refer to "Figure11.2 OFS, WDC, WDTR and WDTS registers" of Hardware Manual for details

5. Electrical Characteristics

Table 5.1 Absolute Maximum Ratings

Symbol	Parameter	Condition	Rated value	Unit
V_{CC}	Supply voltage	Vcc=AVcc	-0.3 to 6.5	V
AVcc	Analog supply voltage	Vcc=AVcc	-0.3 to 6.5	V
V_I	Input voltage		-0.3 to Vcc+0.3	V
V_O	Output voltage		-0.3 to Vcc+0.3	V
P_d	Power dissipation	Topr=25 °C	300	mW
T_{opr}	Operating ambient temperature		-20 to 85 / -40 to 85 (D version)	°C
T_{stg}	Storage temperature		-65 to 150	°C

Table 5.2 Recommended Operating Conditions

Symbol	Parameter	Conditions	Standard Min.	Standard Typ.	Standard Max.	Unit
V_{CC}	Supply voltage		2.7		5.5	V
AVcc	Analog supply voltage			Vcc[3]		V
Vss	Supply voltage			0		V
AVss	Analog supply voltage			0		V
V_{IH}	"H" Input voltage		0.8Vcc		Vcc	V
V_{IL}	"L" input voltage		0		0.2Vcc	V
$I_{OH(sum)}$	"H" peak all output currents: Sum of all pins' IOH (peak)				-60.0	mA
$I_{OH(peak)}$	"H" peak output current:				-10.0	mA
$I_{OH(avg)}$	"H" average output current				-5.0	mA
$I_{OL(sum)}$	"L" peak all output currents: Sum of all pins' IOL (peak)				60	mA
$I_{OL(peak)}$	"L" peak output current: Except P1o to P17				10	mA
	P1o to P17	Drive ability HIGH			30	mA
		Drive ability LOW			10	mA
$I_{OL(avg)}$	"L" average output current: Except P1o to P17				5	mA
	P1o to P17	Drive ability HIGH			15	mA
		Drive ability LOW			5	mA
f(XIN)	Main clock input oscillation frequency	3.0V ≤ Vcc ≤ 5.5V	0		20	MHz
		2.7V ≤ Vcc < 3.0V	0		10	MHz

Note
1: Referenced to Vcc = AVcc = 2.7 to 5.5V at Topr = -20 to 85 °C / -40 to 85 °C unless otherwise specified.
2: The mean output current is the mean value within 100ms.
3: Set Vcc=AVcc

Table 5.3 A/D Conversion Characteristics

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
–	Resolution		V_{ref} =Vcc			10	Bit
–	Absolute accuracy	10 bit mode	øAD=10 MHz, Vref=Vcc=5.0V			±3	LSB
		8 bit mode	øAD=10 MHz, Vref=Vcc=5.0V			±2	LSB
		10 bit mode	øAD=10 MHz, Vref=Vcc=3.3V[3]			±5	LSB
		8 bit mode	øAD=10 MHz, Vref=Vcc=3.3V[3]			±2	LSB
R_{LADDER}	Ladder resistance		V_{REF}=Vcc	10		40	kΩ
t_{CONV}	Conversion time	10 bit mode	øAD=10 MHz, Vref=Vcc=5.0V	3.3			µs
		8 bit mode	øAD=10 MHz, Vref=Vcc=5.0V	2.8			µs
V_{REF}	Reference voltage				Vcc[4]		V
V_{IA}	Analog input voltage			0		V_{ref}	V
–	A/D operation clock frequency[2]	Without sample & hold		0.25		10	MHz
		With sample & hold		1.0		10	MHz

Note
1: Referenced to Vcc=AVcc=2.7 to 5.5V at Topr = -20 to 85 °C / -40 to 85 °C unless otherwise specified.
2: When f_{AD} is 10 MHz more, divide the f_{AD} and make A/D operation clock frequency ($Ø_{AD}$) lower than 10 MHz.
3: When the AVcc is less than 4.2V, divide the f_{AD} and make A/D operation clock frequency ($Ø_{AD}$) lower than $f_{AD}/2$.
4: Set Vcc=Vref

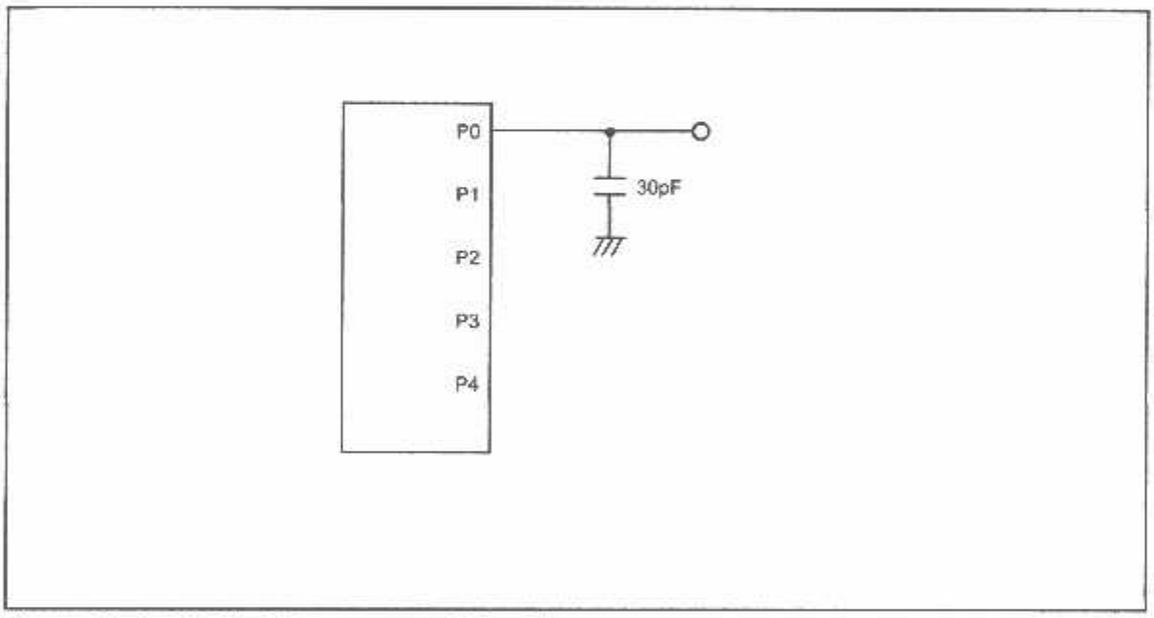


Figure 5.1 Port P0 to P4 measurement circuit

Table 5.4 Flash Memory (Program area) Electrical Characteristics

Symbol	Parameter	Measuring condition	Standard			Unit
			Min.	Typ.	Max	
–	Program/Erase cycle [2]		1000[3]	—	—	cycle
–	Byte program time	Vcc = 5.0 V at Topr = 25 °C	—	50	—	µs
–	Block erase time	Vcc = 5.0 V at Topr = 25 °C	—	0.4	—	s
td(SR-ES)	Time delay from Suspend Request until Erase Suspend		—	—	8	ms
–	Erase Suspend Request Interval		10	—	—	ms
–	Program, Erase Voltage		2.7	—	5.5	V
–	Read Voltage		2.7	—	5.5	V
–	Program, Erase Temperature		0	—	60	°C
–	Data-retention duration	Topr = 55 °C	20	—	—	year

Table 5.5 Flash Memory (Data area Block A, Block B) Electrical Characteristics [4]

Symbol	Parameter	Measuring condition	Standard			Unit
			Min.	Typ.	Max	
–	Program/Erase endurance[2]		10000 [3]	—	—	times
–	Byte program time(program/erase endurance ≤1000 times)	Vcc = 5.0 V at Topr = 25 °C	—	50	400	µs
–	Byte program time(program/erase endurance >1000 times)	Vcc = 5.0 V at Topr = 25 °C	—	65	—	µs
–	Block erase time(program/erase endurance ≤1000 times)	Vcc = 5.0 V at Topr = 25 °C	—	0.2	9	s
–	Block erase time(program/erase endurance >1000 times)	Vcc = 5.0 V at Topr = 25 °C	—	0.3	—	s
td(SR-ES)	Time delay from Suspend Request until Erase Suspend		—	—	8	ms
–	Erase Suspend Request Interval		10	—	—	ms
–	Program, Erase Voltage		2.7	—	5.5	V
–	Read Voltage		2.7	—	5.5	V
–	Program/Erase Temperature		-20(-40)[8]	—	85	°C
–	Data-retention duration	Topr = 55 °C	20	—	—	year

Note
1: Referenced to Vcc=AVcc=2.7 to 5.5V at Topr = 0°C to 60°C unless otherwise specified.
2: Definition of Program/Erase
The cycle of Program/Erase shows a cycle for each block.
If the program/erase number is "n" (n = 1000, 10000), "n" times erase can be performed for each block.
For example, if performing one-byte write to the distinct addresses on Block A of 2K-byte block 2048 times and then erasing that block, the number of Program/Erase cycles is one time.
However, performing multiple writes to the same address before an erase operation is prohibited (overwriting prohibited).
3: Maximum numbers of Program/Erase cycles for which all electrical characteristics is guaranteed.
4: Table 16.5 applies for Block A or B when the Program/Erase cycles are more than 1000. The byte program time up to 1000 cycles are the same as that of the program area (see Table 5.4).
5: To reduce the number of Program/Erase cycles, a block erase should ideally be performed after writing in series as many distinct addresses (only one time each) as possible. If programming a set of 16 bytes, write up to 128 sets and then erase them one time. This will result in ideally reducing the number of Program/Erase cycles. Additionally, averaging the number of Program/Erase cycles for Block A and B will be more effective. It is important to track the total number of block erases and restrict the number.
6: If error occurs during block erase, attempt to execute the clear status register command, then the block erase command at least three times until the erase error disappears.
7: Customers desiring Program/Erase failure rate information should contact their Renesas technical support representative.
8: -40 °C for D version.

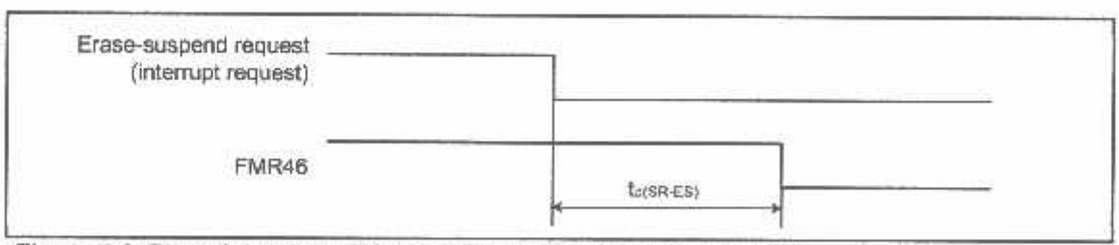


Figure 5.2 Time delay from Suspend Request until Erase Suspend

Table 5.6 Voltage Detection Circuit Electrical Characteristics

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
Vdet	Voltage detection level		3.3	3.8	4.3	V
	Voltage detection interrupt request generating time[2]			40		µs
	Voltage detection circuit self consumption current	VC27=1, VCC=5.0V		600		nA
td(E-A)	Waiting time until voltage detection circuit operation starts[3]				20	µs
Vccmin	Microcomputer operation voltage minimum value		2.7			V

NOTES:
1. The measuring condition is Vcc=AVcc=2.7V to 5.5V and Topr=-40°C to 85°C.
2. This shows the time until the voltage detection interrupt request is generated since the voltage passes Vdet.
3. This shows the required time until the voltage detection circuit operates when setting to "1" again after setting the VC27 bit in the VCR2 register to "0".

Table 5.7 Reset Circuit Electrical Characteristics (When Using Hardware Reset 2[1, 3])

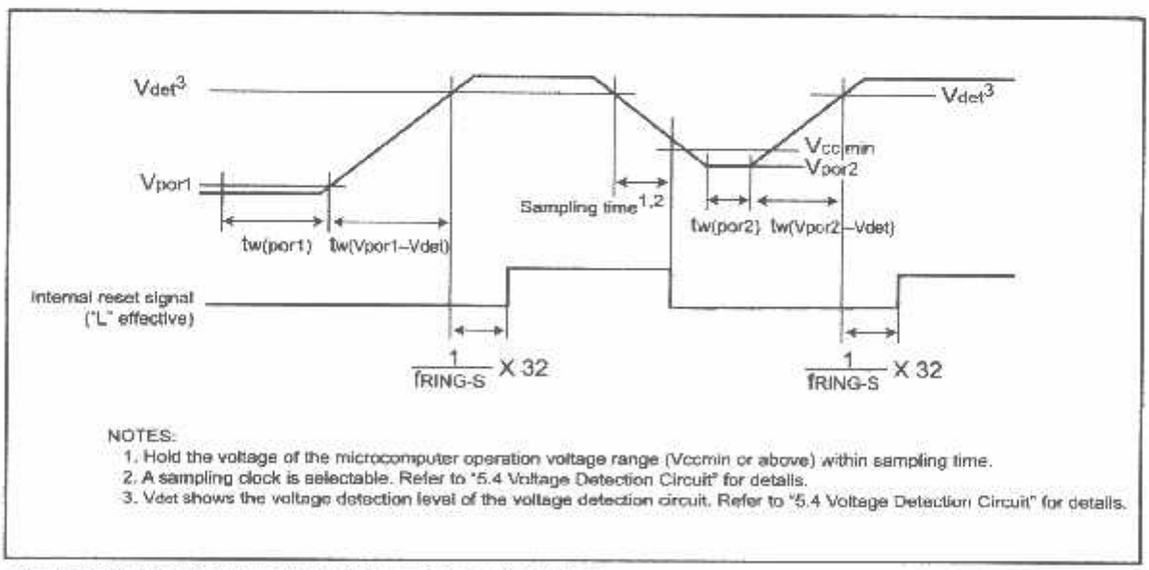
Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
Vpor2	Power-on reset valid voltage	-20°C ≤ Topr < 85°C	—		Vdet	V
tw(Vpor2-Vdet)	Supply voltage rising time when power-on reset is canceled[2]	-20°C ≤ Topr < 85°C, tw(por2) ≥ 0s[4]	—	—	100	ms

NOTES:
1. The voltage detection circuit which is embedded in a microcomputer is a factor to generate the hardware reset 2. Refer to 5.1.2 Hardware Reset 2.
2. This condition is not applicable when using Vcc ≥ 1.0V.
3. When turning power on after the external power has been held below the valid voltage for greater than 10 seconds, refer to Table 16.8 Reset Circuit Electrical Characteristics (When Not Using Hardware Reset 2).
4. tw(por2) is time to hold the external power below effective voltage (Vpor2).

Table 5.8 Reset Circuit Electrical Characteristics (When Not Using Hardware Reset 2)

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
Vpor1	Power-on reset valid voltage	-20°C ≤ Topr < 85°C	—		0.1	V
tw(Vpor1-Vdet)	Supply voltage rising time when power-on reset is canceled	0°C ≤ Topr ≤ 85°C, tw(por1) ≥ 10s[2]	—	—	100	ms
tw(Vpor1-Vdet)	Supply voltage rising time when power-on reset is canceled	-20°C ≤ Topr < 0°C, tw(por1) ≥ 30s[2]	—		100	ms
tw(Vpor1-Vdet)	Supply voltage rising time when power-on reset is canceled	-20°C ≤ Topr < 0°C, tw(por1) ≥ 10s[2]		—	1	ms
tw(Vpor1-Vdet)	Supply voltage rising time when power-on reset is canceled	0°C ≤ Topr < 85°C, tw(por1) ≥ 1s[2]	—		0.5	ms

NOTES:
1. When not using hardware reset 2, use with Vcc ≥ 2.7V.
2. tw(por1) is time to hold the external power below effective voltage (Vpor1).



NOTES:
1. Hold the voltage of the microcomputer operation voltage range (Vccmin or above) within sampling time.
2. A sampling clock is selectable. Refer to "5.4 Voltage Detection Circuit" for details.
3. Vdet shows the voltage detection level of the voltage detection circuit. Refer to "5.4 Voltage Detection Circuit" for details.

Figure 5.3 Reset Circuit Electrical Characteristics

Table 5.9 High-speed On-Chip Oscillator Circuit Electrical Characteristics

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
—	High-speed on-chip oscillator frequency 1 (fd(HRoffset)+fd(HR)) when the reset is released	VCC=5.0V, Topr=25 °C Set "40H" in the HR1 register		8	—	MHz
fd(HRoffset)	Settable high-speed on-chip oscillator minimum period	VCC=5.0V, Topr=25 °C Set "00H" in the HR1 register	—	61	—	ns
fd(HR)	High-speed on-chip oscillator period adjusted unit	Difference when setting "01H" and "00H" in the HR register	—	1	—	ns
—	High-speed on-chip oscillator temperature dependence(1)	Frequency fluctuation in temperature range of -10 °C to 50 °C	—	±5	—	%
—	High-speed on-chip oscillator temperature dependence(2)	Frequency fluctuation in temperature range of -40 °C to 85 °C	—	±10		%

NOTES:
1. The measuring condition is Vcc=AVcc=5.0 V and Topr=25 °C.

Table 5.10 Power Circuit Timing Characteristics

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
td(P-R)	Time for internal power supply stabilization during powering-on[2]		1		2000	µs
td(R-S)	STOP release time[3]				150	µs

Note
1: The measuring condition is Vcc=AVcc=2.7 to 5.5 V and Topr=25 °C.
2: This shows the wait time until the internal power supply generating circuit is stabilized during power-on.
3: This shows the time until BCLK starts from the interrupt acknowledgement to cancel stop mode.

Table 5.11 Electrical Characteristics (1) [Vcc=5V]

Symbol	Parameter		Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
VOH	"H" output voltage	Except XOUT	IOH=-5mA		Vcc-2.0		Vcc	V
			IOH=-200µA		Vcc-0.3	—	Vcc	V
		XOUT	Drive capacity HIGH	IOH=-1 mA	Vcc-2.0	—	Vcc	V
			Drive capacity LOW	IOH=-500µA	Vcc-2.0	—	Vcc	V
VOL	"L" output voltage	P10 to P17 Except XOUT	IOL= 5 mA				2.0	V
			IOL= 200 µA				0.45	V
		P10 to P17	Drive capacity HIGH	IOL= 15 mA	—	—	2.0	V
			Drive capacity LOW	IOL= 5 mA	—		2.0	V
			Drive capacity LOW	IOL= 200 µA	—	—	0.45	V
		XOUT	Drive capacity HIGH	IOL= 1 mA	—	—	2.0	V
			Drive capacity LOW	IOL=500 µA	—	—	2.0	V
VT+-VT-	Hysteresis	INT0, INT1, INT2, INT3, KI0, KI1 KI2, KI3, CNTR0, CNTR1, TCIN, RxD0, RxD1, P45			0.2		1.0	V
		RESET			0.2	—	2.2	V
IIH	"H" input current		VI=5V		—		5.0	µA
IIL	"L" input current		VI=0V		—	—	-5.0	µA
RPULLUP	Pull-up resistance		VI=0V		30	50	167	kΩ
RfXIN	Feedback resistance	XIN			—	1.0	—	MΩ
fRING-S	Low-speed on-chip oscillator frequency				40	125	250	kHz
VRAM	RAM retention voltage		At stop mode		2.0		—	V

Note
1 : Referenced to Vcc=AVcc=4.2 to 5.5V at Topr = -20 to 85 °C / -40 to 85 °C, f(XIN)=20MHz unless otherwise specified.

Table 5.12 Electrical Characteristics (2) [Vcc=5V]

Symbol	Parameter	Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
Icc	Power supply current (Vcc=3.3 to 5.5V) In single-chip mode, the output pins are open and other pins are VSS	High-speed mode	f(XIN)=20 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		9	15	mA
			f(XIN)=16 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		8	14	mA
			f(XIN)=10 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		5		mA
		Medium-speed mode	f(XIN)=20 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		4		mA
			f(XIN)=16 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		3		mA
			f(XIN)=10 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		2		mA
		High-speed on-chip oscillator mode	Main clock off High-speed on-chip oscillator on=8 MHz Low-speed on-chip oscillator on=125 kHz No division		4	8	mA
			Main clock off High-speed on-chip oscillator on=8 MHz Low-speed on-chip oscillator on=125 kHz Division by 8		1.5		mA
		Low-speed on-chip oscillator mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		470	900	µA
		Wait mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz When a WAIT instruction is executed[2] Peripheral clock operation VC27="0"		40	80	µA
		Wait mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz When a WAIT instruction is executed[2] Peripheral clock off VC27="0"		38	76	µA
		Stop mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator off CM10="1" Peripheral clock off VC27="0"		0.8	3.0	µA

NOTES

1: The power supply current measuring is executed using the measuring program on flash memory.

2: Timer Y is operated with timer mode.

Timing requirements (Unless otherwise noted: Vcc = 5V, Vss = 0V at Ta = 25 °C) [Vcc=5V]

Table 5.13 XIN input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(XIN)	XIN input cycle time	50		ns
tWH(XIN)	XIN input HIGH pulse width	25		ns
tWL(XIN)	XIN input LOW pulse width	25		ns

Table 5.14 CNTR0 input, CNTR1 input, $\overline{INT2}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(CNTR0)	CNTR0 input cycle time	100		ns
tWH(CNTR0)	CNTR0 input HIGH pulse width	40		ns
tWL(CNTR0)	CNTR0 input LOW pulse width	40		ns

Table 5.15 TCIN input, $\overline{INT3}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(TCIN)	TCIN input cycle time	400 [1]		ns
tWH(TCIN)	TCIN input HIGH pulse width	200 [2]		ns
tWL(TCIN)	TCIN input LOW pulse width	200 [2]		ns

NOTES

1 :When using the Timer C input capture mode, adjust the cycle time above (1/ Timer C count source frequency x 3).

2 : When using the Timer C input capture mode, adjust the pulse width above (1/ Timer C count source frequency x 1.5).

Table 5.16 Serial Interface

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(CK)	CLKi input cycle time	200		ns
tW(CKH)	CLKi input HIGH pulse width	100		ns
tW(CKL)	CLKi input LOW pulse width	100		ns
td(C-Q)	TxDi output delay time		80	ns
th(C-Q)	TxDi hold time	0		ns
tsu(D-C)	RxDi input setup time	35		ns
th(C-D)	RxDi input hold time	90		ns

Table 5.17 External interrupt $\overline{INT0}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tW(INH)	$\overline{INT0}$ input HIGH pulse width	250 [1]		ns
tW(INL)	$\overline{INT0}$ input LOW pulse width	250 [2]		ns

NOTES

1 : When selecting the digital filter by the $\overline{INT0}$ input filter select bit, use the $\overline{INT0}$ input HIGH pulse width to the greater value,either (1/ digital filter clock frequency x 3) or the minimum value of standard.

2 : When selecting the digital filter by the $\overline{INT0}$ input filter select bit, use the $\overline{INT0}$ input LOW pusle width to the greater value,either (1/ digital filter clock frequency x 3) or the minimum value of standard.

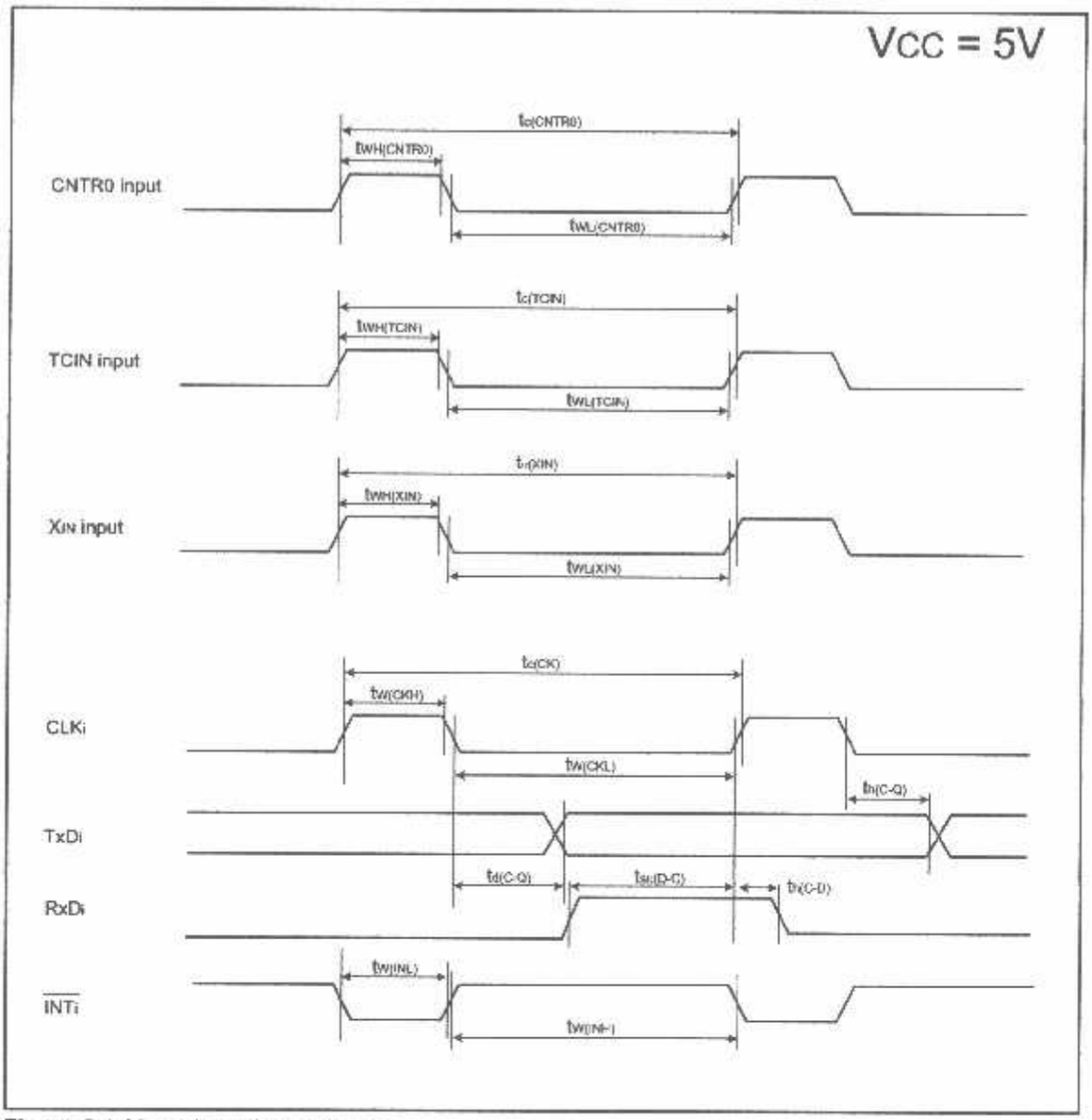


Figure 5.4 Vcc=5V timing diagram

Table 5.18 Electrical Characteristics (3) [Vcc=3V]

Symbol	Parameter		Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
V_{OH}	"H" output voltage	Except X_{OUT}	I_{OH}=-1mA		Vcc-0.5		Vcc	V
		X_{OUT}	Drive capacity HIGH	I_{OH}=-0.1 mA	Vcc-0.5		Vcc	V
			Drive capacity LOW	I_{OH}=-50 µA	Vcc-0.5		Vcc	V
V_{OL}	"L" output voltage	P10 to P17 Except X_{OUT}	I_{OL}= 1 mA				0.5	V
		P10 to P17	Drive capacity HIGH	I_{OL}= 2 mA	—		0.5	V
			Drive capacity LOW	I_{OL}= 1 mA	—		0.5	V
		X_{OUT}	Drive capacity HIGH	I_{OL}= 0.1 mA	—		0.5	V
			Drive capacity LOW	I_{OL}=50 µA	—	—	0.5	V
V_{T+}-V_{T-}	Hysteresis	INT0, INT1, INT2, INT3, KI0, KI1, KI2, KI3, CNTR0, CNTR1, T_{CIN}, RxD0, RxD1, P45			0.2	—	0.8	V
		RESET			0.2	—	1.8	V
I_{IH}	"H" input current		V_I=3V		—		4.0	µA
I_{IL}	"L" input current		V_I=0V		—	—	-4.0	µA
R_{PULLUP}	Pull-up resistance		V_I=0V		66	160	500	kΩ
R_{XIN}	Feedback resistance	X_{IN}				3.0		MΩ
f_{RING-S}	Low-speed on-chip oscillator frequency				40	125	250	kHz
V_{RAM}	RAM retention voltage		At stop mode		2.0			V

Note
1. Referenced to Vcc=AVcc=2.7 to 3.3V at Topr = -20 to 85 °C / -40 to 85 °C, f(X_{IN})=10MHz unless otherwise specified.

Table 5.19 Electrical Characteristics (4) [Vcc=3V]

Symbol	Parameter	Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
Icc	Power supply current (Vcc=2.7 to 3.3V) In single-chip mode, the output pins are open and other pins are Vss	High-speed mode	Xin=20 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		8	13	mA
			Xin=16 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		7	12	mA
			Xin=10 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz No division		5		mA
		Medium-speed mode	Xin=20 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		3		mA
			Xin=16 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		2.5		mA
			Xin=10 MHz (square wave) High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		1.6		mA
		High-speed on-chip oscillator mode	Main clock off High-speed on-chip oscillator on=8 MHz Low-speed on-chip oscillator on=125 kHz No division		3.5	7.5	mA
			Main clock off High-speed on-chip oscillator on=8 MHz Low-speed on-chip oscillator on=125 kHz Division by 8		1.5		mA
		Low-speed on-chip oscillator mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz Division by 8		420	800	µA
		Wait mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz When a WAIT instruction is executed[2] Peripheral clock operation VC27="0"		37	74	µA
		Wait mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator on=125 kHz When a WAIT instruction is executed[2] Peripheral clock off VC27="0"		35	70	µA
		Stop mode	Main clock off High-speed on-chip oscillator off Low-speed on-chip oscillator off CM10="1" Peripheral clock off VC27="0"		0.7	3.0	µA

NOTES
1: The power supply current measuring is executed using the measuring program on flash memory.
2: Timer Y is operated with timer mode.

Timing requirements (Unless otherwise noted: Vcc = 3V, Vss = 0V at Ta = 25 °C) [Vcc=3V]

Table 5.20 XIN input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(XIN)	XIN input cycle time	100		ns
tWH(XIN)	XIN input HIGH pulse width	40		ns
tWL(XIN)	XIN input LOW pulse width	40		ns

Table 5.21 CNTR0 input, CNTR1 input, $\overline{INT2}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(CNTR0)	CNTR0 input cycle time	300		ns
tWH(CNTR0)	CNTR0 input HIGH pulse width	120		ns
tWL(CNTR0)	CNTR0 input LOW pulse width	120		ns

Table 5.22 TCIN input, $\overline{INT3}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(TCIN)	TCIN input cycle time	1200 [1]		ns
tWH(TCIN)	TCIN input HIGH pulse width	600 [2]		ns
tWL(TCIN)	TCIN input LOW pulse width	600 [2]		ns

NOTES
1 :When using the Timer C input capture mode, adjust the cycle time above (1/ Timer C count source frequency x 3).
2 : When using the Timer C input capture mode, adjust the pulse width above (1/ Timer C count source frequency x 1.5).

Table 5.23 Serial Interface

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tC(CK)	CLKi input cycle time	300		ns
tW(CKH)	CLKi input HIGH pulse width	150		ns
tW(CKL)	CLKi input LOW pulse width	150		ns
td(C-Q)	TxDi output delay time		160	ns
th(C-Q)	TxDi hold time	0		ns
tsu(D-C)	RxDi input setup time	55		ns
th(C-D)	RxDi input hold time	90		ns

Table 5.24 External interrupt $\overline{INT0}$ input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
tW(INH)	$\overline{INT0}$ input HIGH pulse width	380 [1]		ns
tW(INL)	$\overline{INT0}$ input LOW pulse width	380 [2]		ns

NOTES
1 : When selecting the digital filter by the $\overline{INT0}$ input filter select bit, use the $\overline{INT0}$ input HIGH pulse width to the greater value,either (1/ digital filter clock frequency x 3) or the minimum value of standard.
2 : When selecting the digital filter by the $\overline{INT0}$ input filter select bit, use the $\overline{INT0}$ input LOW pusle width to the greater value,either (1/ digital filter clock frequency x 3) or the minimum value of standard.

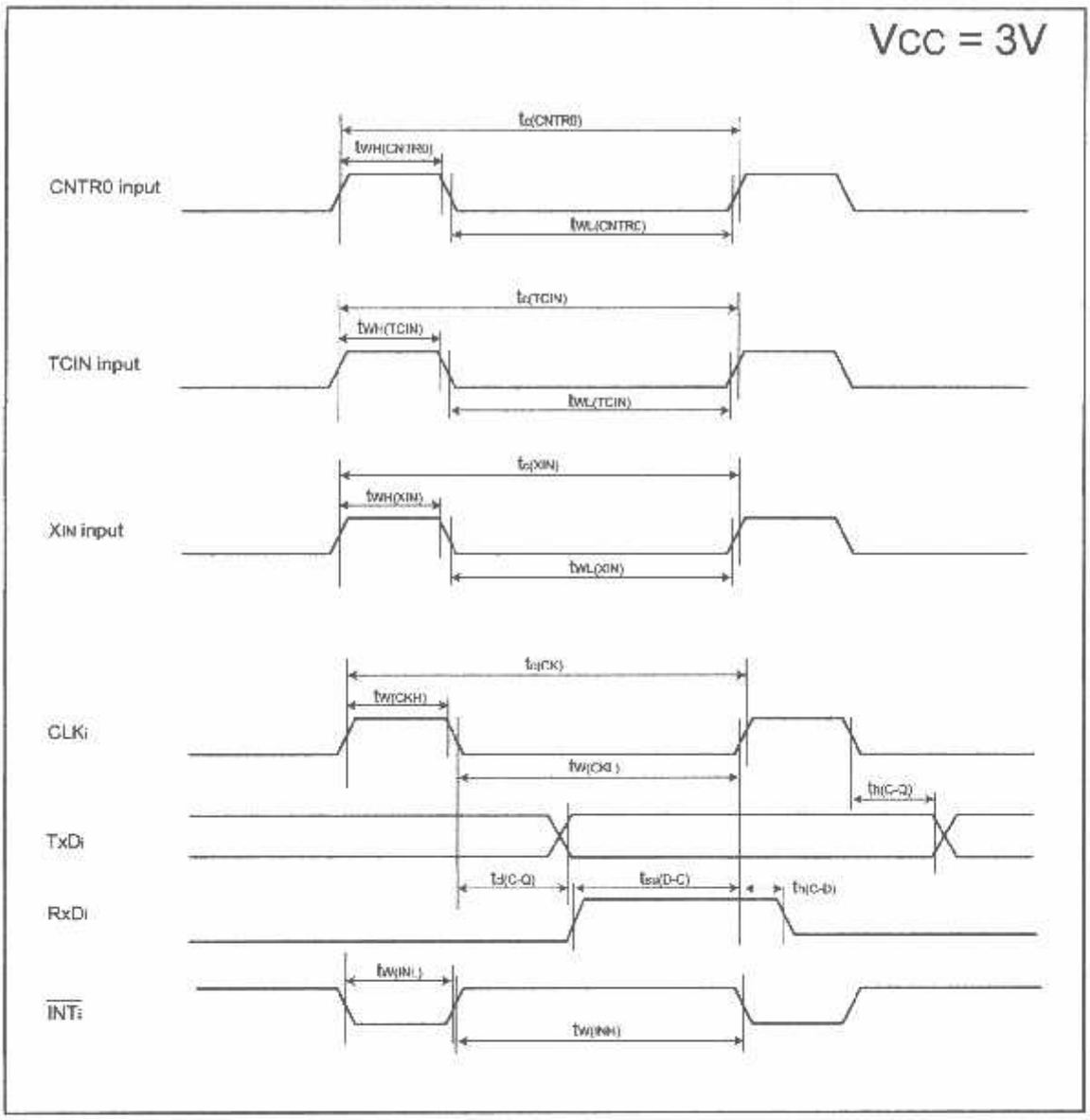
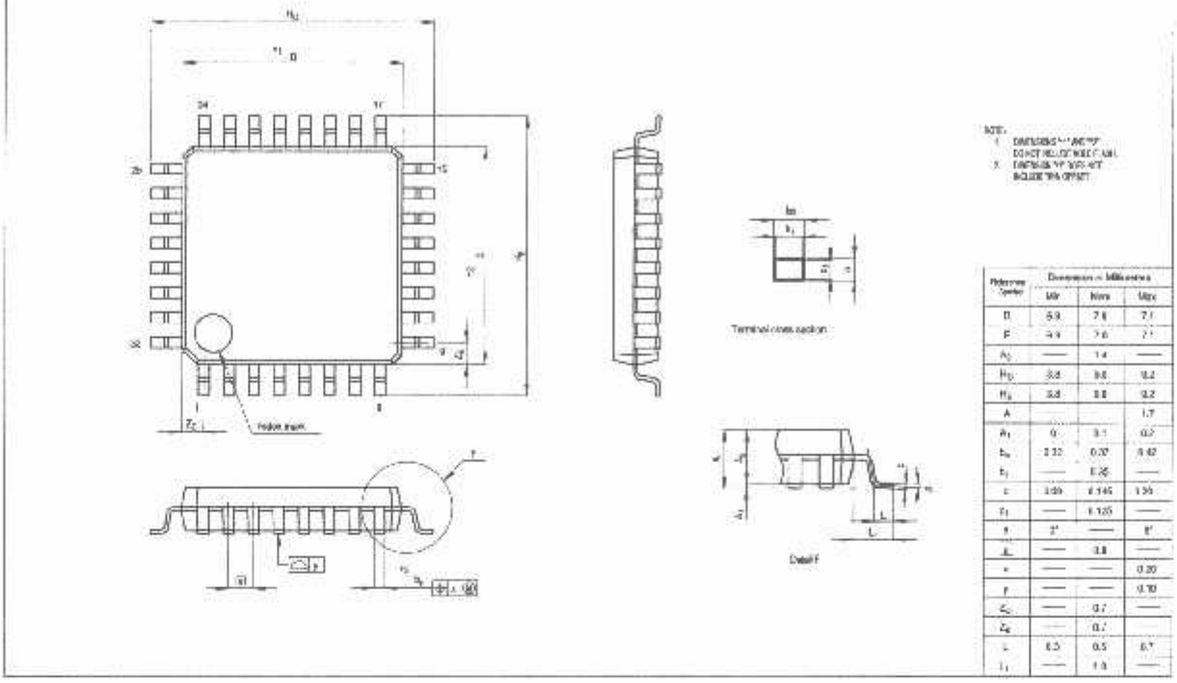


Figure 5.5 Vcc=3V timing diagram

Package Dimensions

JEITA Package Code	RENESAS Code	Previous Code	MASS(Typ.)
P-LQFP32-7x7-0.80	PLQP0032GB-A	32P6U-A	0.2g



Reference Symbol	Dimension in Millimeters		
	Min	Nom	Max
D	6.9	7.0	7.1
E	6.9	7.0	7.1
A_2	—	1.4	—
H_D	8.8	9.0	9.2
H_E	8.8	9.0	9.2
A			1.7
A_1	0	0.1	0.2
b_p	0.32	0.37	0.42
b_1	—	0.35	—
c	0.09	0.145	0.20
c_1	—	0.125	—
θ	0°	—	8°
R_1	—	0.3	—
x	—	—	0.20
y	—	—	0.10
Z_D	—	0.7	—
Z_E	—	0.7	—
L	0.5	0.5	0.7
L_1	—	1.0	—

Rev.	Date	Description	
		Page	Summary
1.10	Oct 28, 2003		First edition issued
1.20	Dec05, 2003	5	Figure 1.3 revised
		10	Chapter 4, NOTES revised
		16	Table 5.4 revised Table 5.5 revised
		17	Table 5.6 revised Figure 5.3 added
		18	Table 5.8 revised Table 5.10 revised
		21	Figure 5.3 revised to Figure 5.4
		22	Table 5.17 revised
		25	Figure 5.4 revised to Figure 5.5
2.00	Sep 30, 2004	All pages	Words standardized (on-chip oscillator, serial interface, A/D)
		2	Table 1.1 revised
		5	Figure 1.3, NOTES 3 added
		6	Table 1.3 revised
		9	Figure 3.1, NOTES added
		10-13	One body sentence in chapter 4 added ; Titles of Table 4.1 to 4.4 added
		12	Table 4.3 revised ; Table 4.4 revised
		14	Table 5.2 revised
		15	Table 5.3 revised
		16	Table 5.4 and Table 5.5 revised
		17	Table 5.6, 5.7 and 5.8 revised ; Figure 5.3 revised
		18	Table 5.9 and 5.11 revised
		19	Table 5.12 revised
		20	Table 5.13 revised
		22	Table 5.18 revised
		23	Table 5.19 revised
		24	Table 5.20 and Table 5.24 revised
2.10	Apr.27.2005	4	Table 1.2, Figure 1.2 package name revised
		5	Figure 1.3 package name revised
		10	Table 4.1 revised
		12	Table 4.3 revised
		15	Table 5.3 partly revised
		16	Table 5.4, Table 5.5 partly added

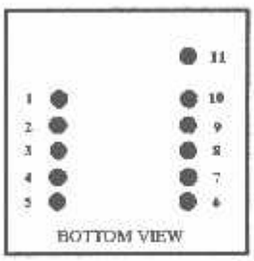
ev.	Date	Description	
		Page	Summary
.10	Apr.27.2005	17	Table 5.7, 5.8 revised
		18	Table 5.10, Table 5.11 partly revised
		22	Table 5.18 partly revised
		26	Package Dimensions revised

ID SERIES DATASHEET Feb 10 , 2004

ID-2 / ID-12 / ID-20

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

ID2 / ID12 / ID20 PIN-OUT



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

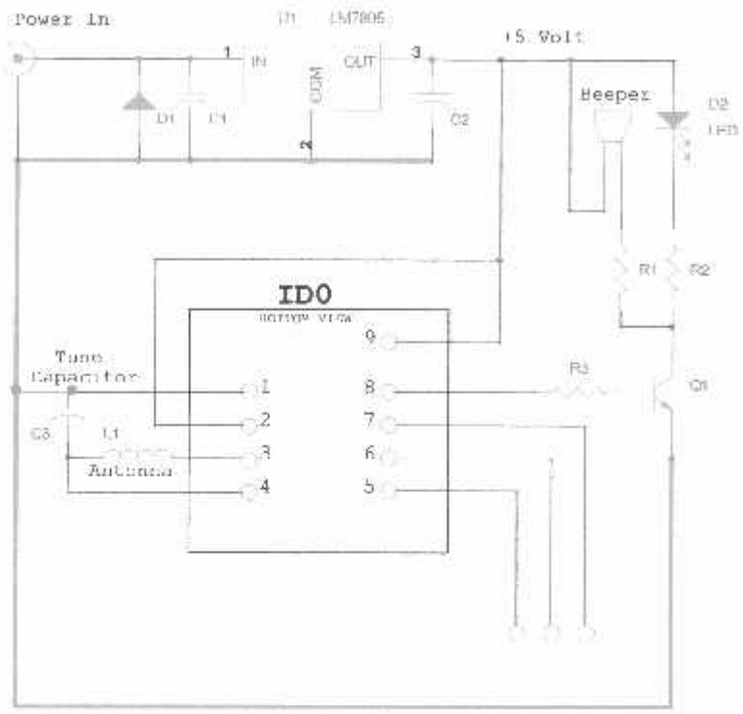
Operational and Physical Characteristics

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

Pin Description & Output Data Formats

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

Circuit Diagram for the ID0



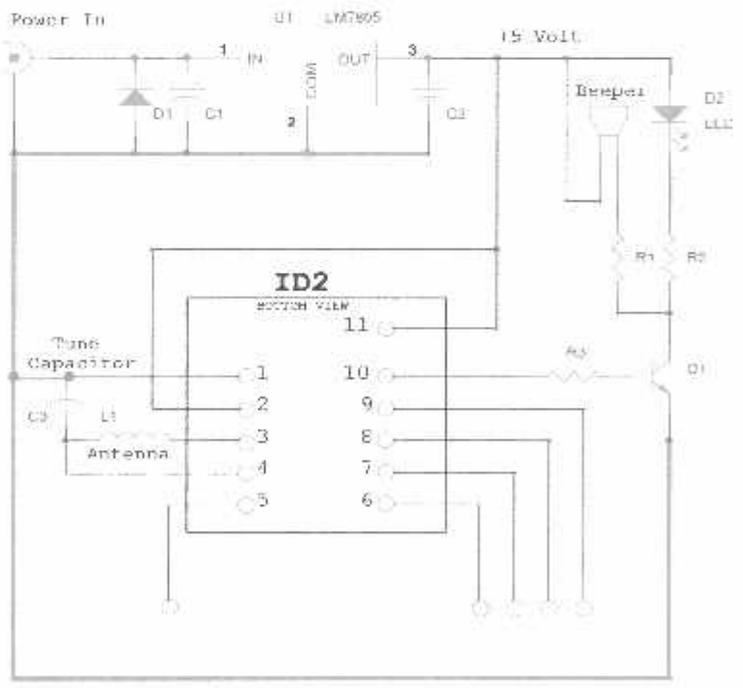
COMPONENT LIST
R1 = 100R
R2 = 1K
R3 = 1K
C1 = 100uF 16V
C2 = 100uF 10V
C3 = 1nF COG 100V *
Beeper = 2.7-3.5KHz 100R
D1 = 1N4001
D2 = GREEN LED
U1 = LM7805
Q1 = UTC8050 (NPN)
L1 = 640Uh

ID0 = ID Innovations ID0

* Please Note the ID0 has an internal tuning capacitor of 1.5nF and this makes the total tuning capacity = 2.5nF

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

Circuit Diagram for the ID2



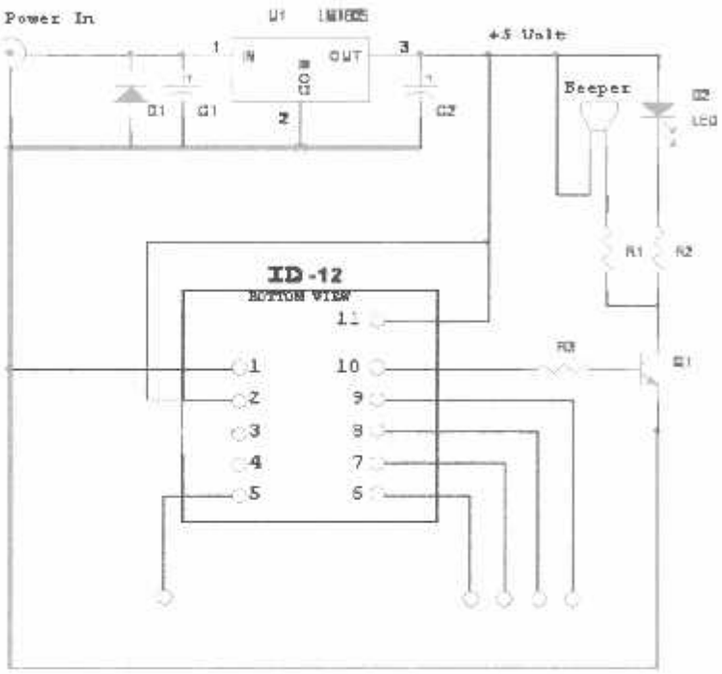
COMPONENT LIST
R1 = 100R
R2 = 1K
R3 = 1K
C1 = 100uF 16V
C2 = 100uF 10V
C3 = 1nF COG 100V *
Beeper = 2.7-3.5KHz 100R
D1 = 1N4001
D2 = GREEN LED
U1 = LM7805
Q1 = UTC8050 (NPN)
L1 = 640Uh

ID2 = ID Innovations ID2

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

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

Circuit Diagram for the ID-12



COMPONENT LIST
R1 = 100R
R2 = 1K
R3 = 1K
C1 = 100uF 16V
C2 = 100uF 10V
Beeper = 2.7-3.5KHz 100R
D1 = 1N4001
D2 = GREEN LED
U1 = LM7805
Q1 = UTC8050 (NPN)
ID2 = ID Innovations ID2

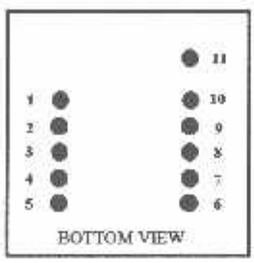
* Please Note the ID2 has an internal tuning capacitor of 1.5nF and this makes the total tuning capacity = 2.5nF

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

ID-2RW. ID-12RW Brief Data

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

ID2 / ID12 / ID20 PIN-OUT



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

Operational and Physical Characteristics

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

Description

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

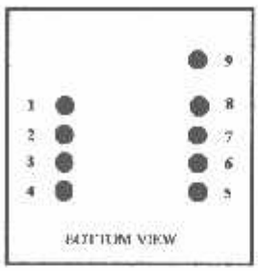
Low Cost Short-Range Proximity Readers

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

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

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

ID0 / ID10 / ID15 PIN-OUT



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

Operational and Physical Characteristics

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

Pin Description & Output Data Formats

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

DATA FORMATS

Output Data Structure – ASCII

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

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

Output Data Structure – Wiegand26

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

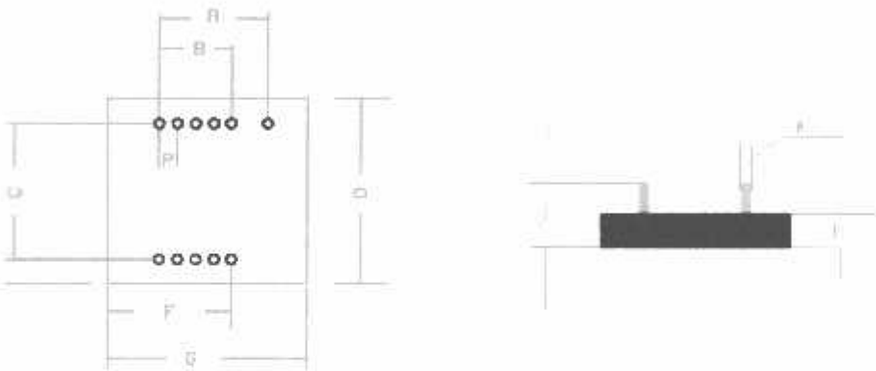
* = Parity start bit and stop bit

Output Data Magnetic ABA Track2

10 Leading Zeros	SS	Data	ES	LCR	10 Ending Zeros

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

Dimensions (Top View) (mm)



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

Note – measurements do not include any burring of edges.

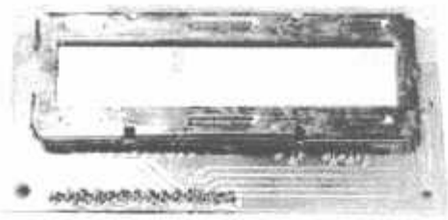
HOW TO USE
INTELLIGENT L.C.D.S

Part One

By Julyan Ilett

This paper was originally published as the first half of a two-part article in the February 1997 issue of *Everyday Practical Electronics* magazine (www.epemag.wimborne.co.uk), and is reproduced here with their kind permission.

How to use Intelligent L.C.D.s



By Julyan Ilett

An utterly "practical" guide to interfacing and programming intelligent liquid crystal display modules.

Part One

Recently, a number of projects using intelligent liquid crystal display (l.c.d.) modules have been featured in *EPE*. Their ability to display not just numbers, but also letters, words and all manner of symbols, makes them a good deal more versatile than the familiar 7-segment light emitting diode (l.e.d.) displays.

Although still quite expensive when purchased new, the large number of surplus modules finding their way into the hands of the "bargain" electronics suppliers, offers the hobbyist a low cost opportunity to carry out some fascinating experiments and realise some very sophisticated electronic display projects.

Basic Reading

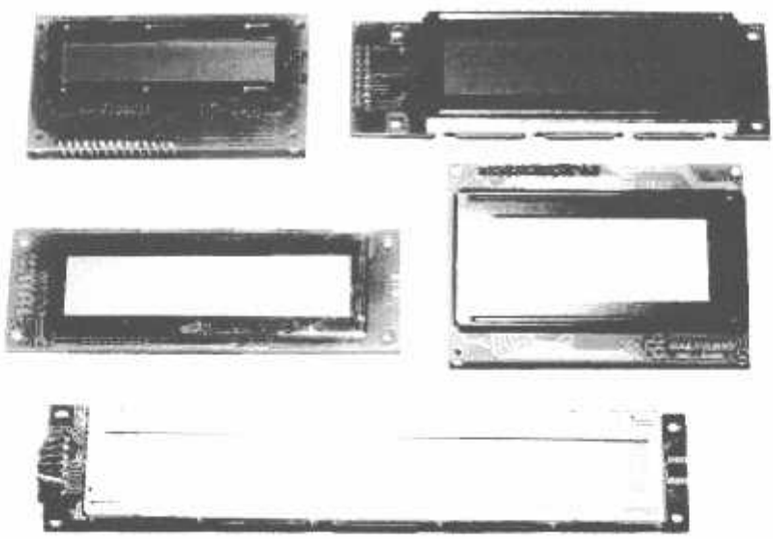
This article deals with the character-based l.c.d. modules which use the Hitachi HD44780 (or compatible) controller chip, as do most modules available to the hobbyist. Of course, these modules are not quite as advanced as the latest generation, full size, full colour, back-lit types used in today's laptop computers, but far from being "phased out," character-based l.c.d.s are still used extensively in commercial and industrial equipment, particularly where display requirements are reasonably simple.

The modules have a fairly basic interface, which mates well with traditional micro-processors such as the *Z*80 or the 6502. It is also ideally suited to the PIC microcontroller, which is probably the most popular microcontroller used by the electronics hobbyist.

However, even if, as yet, you know nothing of microcontrollers, and possess none of the PIC paraphernalia, don't despair, you can still enjoy all the fun of experimenting with l.c.d.s, using little more than a handful of switches!

Shapes and Sizes

Even limited to character-based modules, there is still a wide variety of shapes and sizes available. Line lengths of 8, 16, 20, 24, 32 and 40 characters are all standard, in one, two and four-line versions.



Several different liquid crystal technologies exist. "Supertwist" types, for example, offer improved contrast and viewing angle over the older "twisted nematic" types. Some modules are available with back-lighting, so that they can be viewed in dimly-lit conditions. The back-lighting may be either "electro-luminescent," requiring a high voltage inverter circuit, or simpler l.e.d. illumination.

Few of these features are important, however, for experimentation purposes. All types are capable of displaying the same basic information, so the cheaper types are probably the best bet initially.

Connections

Most l.c.d. modules conform to a standard interface specification. A 14-pin access is provided (14 holes for solder pin insertion or for an IDC connector) having eight data lines, three control lines and three power lines. The connections are laid out in one of two common configurations, either two rows of seven pins, or a single row of 14 pins. The two layout alternatives are displayed in Figure 1.

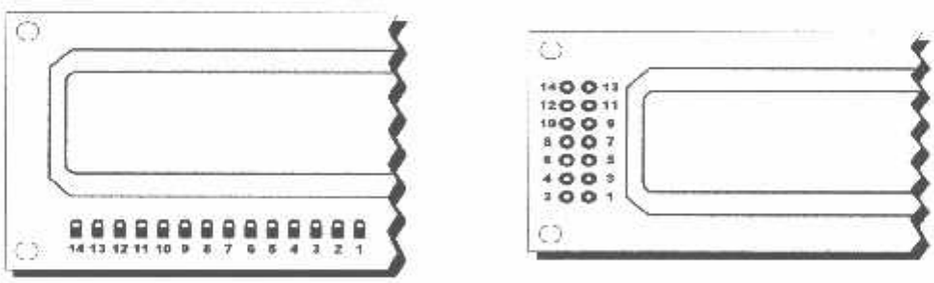


Figure 1: Pinouts of the two basic l.c.d formats.

On most displays, the pins are numbered on the l.c.d.'s printed circuit board, but if not, it is quite easy to locate pin 1. Since this pin is connected to ground, it often has a thicker p.c.b. track connected to it, and it is generally connected to the metalwork at some point.

The function of each of the connections is shown in Table 1. Pins 1 and 2 are the power supply lines, Vss and Vdd. The Vdd pin should be connected to the positive supply, and Vss to the 0V supply or ground.

Pin No.	Name	Function
1	Vss	Ground
2	Vdd	+ve supply
3	Vee	Contrast
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7

Table 1. Pinout functions for all the l.c.d. types.

Although the l.c.d. module data sheets specify a 5V d.c. supply (at only a few milliamps), supplies of 6V and 4.5V both work well, and even 3V is sufficient for some modules. Consequently, these modules can be effectively, and economically, powered by batteries.

Pin 3 is a control pin, Vee, which is used to alter the contrast of the display. Ideally, this pin should be connected to a variable voltage supply. A preset potentiometer connected between the power supply lines, with its wiper connected to the contrast pin is suitable in many cases, but be aware that some modules may require a negative potential; as low as 7V in some cases. For absolute simplicity, connecting this pin to 0V will often suffice.

Pin 4 is the Register Select (RS) line, the first of the three command control inputs. When this line is low, data bytes transferred to the display are treated as commands, and data bytes read from the display indicate its status. By setting the RS line high, character data can be transferred to and from the module.

Pin 5 is the Read/Write (R/W) line. This line is pulled low in order to write commands or character data to the module, or pulled high to read character data or status information from its registers.

Pin 6 is the Enable (E) line. This input is used to initiate the actual transfer of commands or character data between the module and the data lines. When writing to the display, data is transferred only on the high to low transition of this signal. However, when reading from the display, data will become available shortly after the low to high transition and remain available until the signal falls low again.

Pins 7 to 14 are the eight data bus lines (D0 to D7). Data can be transferred to and from the display, either as a single 8-bit byte or as two 4-bit "nibbles." In the latter case, only the upper four data lines (D4 to D7) are used. This 4-bit mode is beneficial when using a microcontroller, as fewer input/output lines are required.

Prototype Circuit

For an l.c.d. module to be used effectively in any piece of equipment, a microprocessor or microcontroller is usually required to drive it. However, before attempting to wire the two together, some initial (and very useful) experiments can be performed, by connecting up a series of switches to the pins of the module. This can be quite a beneficial step, even if you are thoroughly conversant with the workings of microprocessors.

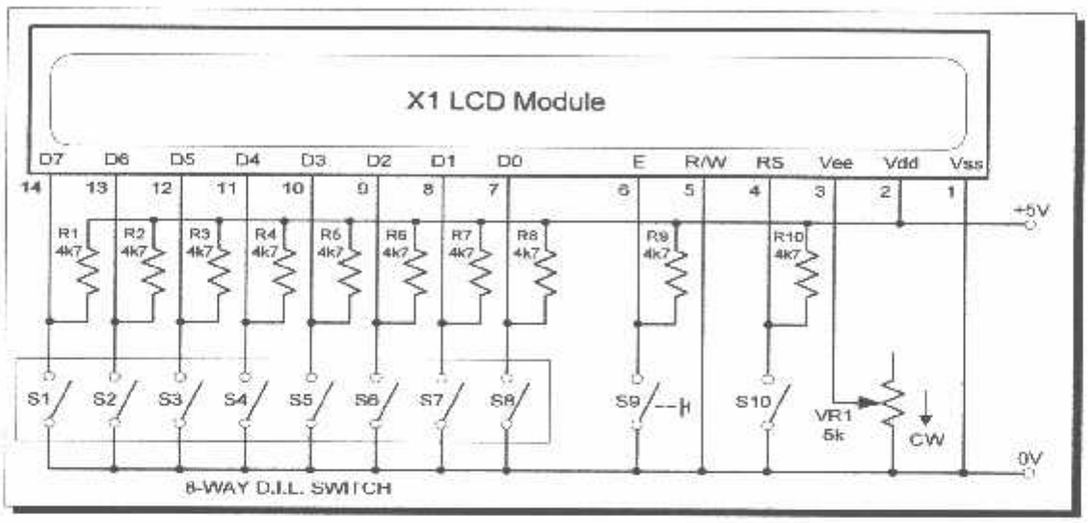


Figure 2: Circuit diagram for an l.c.d. experimental rig.

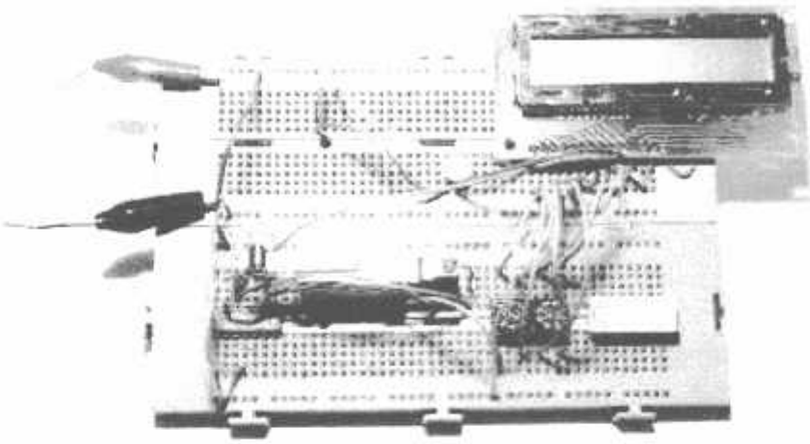
In Figure 2 is shown the circuit diagram of an l.c.d. experimentation rig. The circuit can be wired-up on a "plug-in" style prototyping board, using d.i.l. (dual-in-line) switches for the data lines (S1 to S8), a toggle switch for the RS input (S10), and a momentary action switch (or microswitch) for the E input (S9). The R/W line is connected to ground (0V), as the display is only going to be written to for the time being.

All of the resistors (R1 through R10) are 4K7 ohms. It is probably most convenient to use a s.i.l. (single-in-line) resistor pack for the eight pull-up resistors (R1 to R8) on the data lines. The other two resistors, R9 and R10, can be discrete types. Preset potentiometer VR1 (5K ohms) is used for the contrast control and is shown with one end left disconnected. If desired, this end can be connected to the positive line via a resistor of about 47K ohms (it should be connected to a negative supply, via a similar resistor, for those modules which require negative biasing).

All the switches should be connected so that they are "on" when in the "down" position, so that "down" generates a logic 0 (low) and "up" provides a logic 1 (high). The switches should also be arranged so that data bit D7 is on the left, and data bit D0 is on the right. In this way, binary numbers can be entered the right way round.

Initially, the contrast control should be adjusted fully clockwise, so that the contrast control input (Vee) is connected to ground. The initial settings of the switches are

unimportant, but it is suggested that the RS switch (S10) is "up" (set to logic 1), and the E switch (S9) is left unpressed. The data switches, S1 to S8, can be set to any value at this stage. All is now prepared to start sending commands and data to the l.c.d. module.



The experimental circuit can be built on plug-in prototyping boards.

Experiment 1: Basic Commands

When powered up, the display should show a series of dark squares, possibly only on part of the display. These character cells are actually in their off state, so the contrast control should be adjusted anti-clockwise (away from ground) until the squares are only just visible.

The display module resets itself to an initial state when power is applied, which curiously has the display blanked off, so that even if characters are entered, they cannot be seen. It is therefore necessary to issue a command at this point, to switch the display on.

A full list of the commands that can be sent is given in Table 2, together with their binary and hexadecimal values. The initial conditions of the l.c.d. after power-on are marked with an asterisk.

Throughout this article, emphasis will be placed on the binary value being sent since this illustrates which data bits are being set for each command. After each binary value, the equivalent hexadecimal value is quoted in brackets, the $ prefix indicating that it is hexadecimal.

The Display On/Off and Cursor command turns on the display, but also determines the cursor style at the same time. Initially, it is probably best to select a Blinking Cursor with Underline, so that its position can be seen clearly, i.e. code 00001111 ($0F).

Command	Binary D7	D6	D5	D4	D3	D2	D1	D0	Hex
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	1/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

1/D:	1=Increment*, 0=Decrement	R/L:	1=Right shift, 0=Left shift
S:	1=Display shift on, 0=Off*	8/4:	1=8-bit interface*, 0=4-bit interface
D:	1=Display on, 0=Off*	2/1:	1=2 line mode, 0=1 line mode*
U:	1=Cursor underline on, 0=Off*	10/7:	1=5x10 dot format, 0=5x7 dot format*
B:	1=Cursor blink on, 0=Off*		
D/C:	1=Display shift, 0=Cursor move	x = Don't care	* = Initialization settings

Table 2. The command control codes.

Set the data switches (S1 to S8) to 00001111 ($0F) and ensure that the RS switch (S10) is "down" (logic 0), so that the device is in Command mode. Now press the E switch (S9) momentarily, which "enables" the chip to accept the data, and Hey Presto, a flashing cursor with underline appears in the top left hand position!

If a two-line module is being used, the second line can be switched on by issuing the Function Set command. This command also determines whether an 8-bit or a 4-bit data transfer mode is selected, and whether a 5 x 10 or 5 x 7 pixel format will be used. So, for 8-bit data, two lines and a 5 x 7 format, set the data switches to binary value 00111000 ($38), leave RS (S10) set low and press the E switch, S9.

It will now be necessary to increase the contrast a little, as the two-line mode has a different drive requirement. Now set the RS switch to its "up" position (logic 1), switching the chip from Command mode to Character mode, and enter binary value 01000001 ($41) on the data switches. This is the ASCII code for a capital *A*.

Press the E switch, and marvel as the display fills up with capital *A*'s. Clearly, something is not quite right, and seeing your name in pixels is going to have to wait a while.

Bounce

The problem here is contact bounce. Practically every time the E switch is closed, its contacts will bounce, so that although occasionally only one character appears, most attempts will result in 10 or 20 characters coming up on the display. What is needed is a "debounce" circuit.

But what about the commands entered earlier, why didn't contact bounce interfere with them? In fact it did, but it doesn't matter whether a command is entered ("enabled") just once, or several times, it gets executed anyway. A solution to the bounce problem is shown in Figure 3.

Here, a couple of NAND gates are cross-coupled to form a set-reset latch (or flip-flop) which flips over and latches, so that the contact bounce is eliminated. Either a TTL 74LS00 or a CMOS 74HC00 can be used in this circuit. The switch must be an s.p.d.t. (single-pole, double-throw) type, a microswitch is ideal.

After modifying the circuit, the screen full of *A*'s can be cleared using the Clear Display command. Put binary value 00000001 ($01) on the data switches, set the RS switch to the "down" position and press the new modified E switch. The display is cleared.

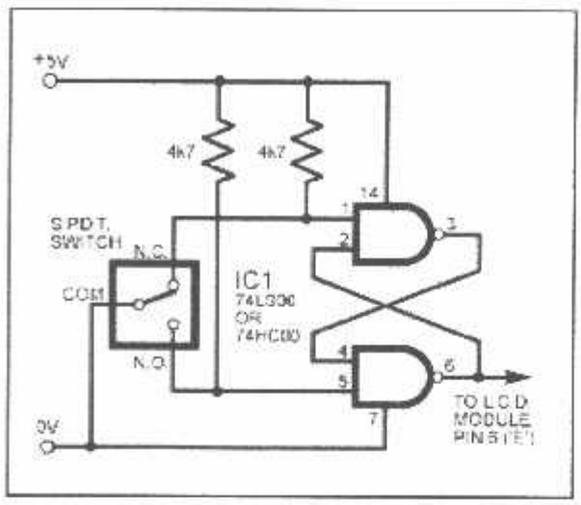


Figure 3. Switch debounce circuit.

Note that the output of the "de-bounce" circuit is high when the switch is pressed and low when the switch is released. Since it is the high to low transition that actually latches data into the l.c.d. module, it will be observed that characters appear on the display, not when the button is pressed, but when it is released.

Experiment 2: Entering Text

First, a little tip: it is manually a lot easier to enter characters and commands in hexadecimal rather than binary (although, of course, you will need to translate commands from binary into hex so that you know which bits you are setting). Replacing the d.i.l. switch pack with a couple of sub-miniature hexadecimal rotary switches is a simple matter, although a little bit of re-wiring is necessary.

The switches must be the type where On = 0, so that when they are turned to the zero position, all four outputs are shorted to the common pin, and in position "F", all four outputs are open circuit.

All the available characters that are built into the module are shown in Table 3. Studying the table, you will see that codes associated with the characters are quoted in binary and hexadecimal, most significant bits ("left-hand" four bits) across the top, and least significant bits ("right-hand" four bits) down the left.

Most of the characters conform to the ASCII standard, although the Japanese and Greek characters (and a few other things) are obvious exceptions. Since these intelligent modules were designed in the "Land of the Rising Sun," it seems only fair that their Katakana phonetic symbols should also be incorporated. The more extensive Kanji character set, which the Japanese share with the Chinese, consisting of several thousand different characters, is not included!

Table 3. Standard l.c.d character table.

Using the switches, of whatever type, and referring to Table 3, enter a few characters onto the display, both letters and numbers. The RS switch (S10) must be "up" (logic 1) when sending the characters, and switch E (S9) must be pressed for each of them. Thus

the operational order is: set RS high, enter character, trigger E, leave RS high, enter another character, trigger E, and so on.

The first 16 codes in Table 3, 00000000 to 00001111. ($00 to $0F) refer to the CGRAM. This is the Character Generator RAM (random access memory), which can be used to hold user-defined graphics characters. This is where these modules really start to show their potential, offering such capabilities as bargraphs, flashing symbols, even animated characters. Before the user-defined characters are set up, these codes will just bring up strange looking symbols.

Codes 00010000 to 00011111 ($10 to $1F) are not used and just display blank characters. ASCII codes "proper" start at 00100000 ($20) and end with 01111111 ($7F). Codes 10000000 to 10011111 ($80 to $9F) are not used, and 10100000 to 11011111 ($A0 to $DF) are the Japanese characters.

Codes 11100000 to 11111111 ($E0 to $FF) are interesting. Although this last block contains mainly Greek characters, it also includes the lower-case characters which have "descenders." These are the letters *g*, *j*, *p*, *q* and *y*, where the tail drops down below the base line of normal upper-case characters. They require the 5 x 10 dot matrix format, rather than the 5 x 7, as you will see if you try to display a lower-case *j*, for example, on a 5 x 7 module.

Some one-line displays have the 5 x 10 format facility, which allows these characters to be shown unbroken. With 5 x 7 two-line displays, the facility can be simulated by borrowing the top three pixel rows from the second line, so creating a 5 x 10 matrix.

For this simulation, set line RS low to put the chip into Command mode. On the data switches, enter the Function Set command using binary value 00110100 ($34). Press and release switch E. Return RS to high, and then send the character data for the last 32 codes in the normal way (remembering to trigger line E!).

Experiment 3: Addressing

When the module is powered up, the cursor is positioned at the beginning of the first line. This is address $00. Each time a character is entered, the cursor moves on to the next address, $01, $02 and so on. This auto-incrementing of the cursor address makes entering strings of characters very easy, as it is not necessary to specify a separate address for each character.

It may be necessary, however, to position a string of characters somewhere other than at the beginning of the first line. In this instance, a new starting address must be entered as a command. Any address between $00 and $7F can be entered, giving a total of 128 different addresses, although not all these addresses have their own display location. There are in fact only 80 display locations, laid out as 40 on each line in two-line mode, or all 80 on a single line in one-line mode. This situation is further complicated because not all display locations are necessarily visible at one time. Only a 40-character, two-line module can display all 80 locations simultaneously.

To experiment with addressing, first set the l.c.d. to two-line mode (if two lines are available), 8-bit data and 5 [P3] 7 format using the Function Set command, i.e. code 00111000 ($38). Note that the last two bits of this command are unimportant, as indicated by the X in the columns of Table 2, and either of them may be set to 0 or 1.

(From now on, we won't constantly remind you that RS must be set appropriately before Command or Character data is entered, or that E must be triggered after data has been entered you should know by now!)

Using the Display On/Off and Cursor command, set the display to On, with Underline and Blinking Cursor, code 00001111 ($0F). Now set the cursor to address 00001000 ($08). This is done by sending a Set Display Address command, binary value 10001000 ($88).

The cursor will jump to the ninth position on the display, at which point text can now be entered. The Set Display Address command is always 10000000 ($80) greater than the display address itself.

Experiment with different display addresses and note their display locations. Be aware that display addresses 00101000 to 00111111 ($28 to $3F) and 01101000 to 01111111 ($68 to $7F) cannot be used on any of the display types.

The relationship between addresses and display locations varies, depending on the type of module being used, but some typical examples are shown in Figure 4.

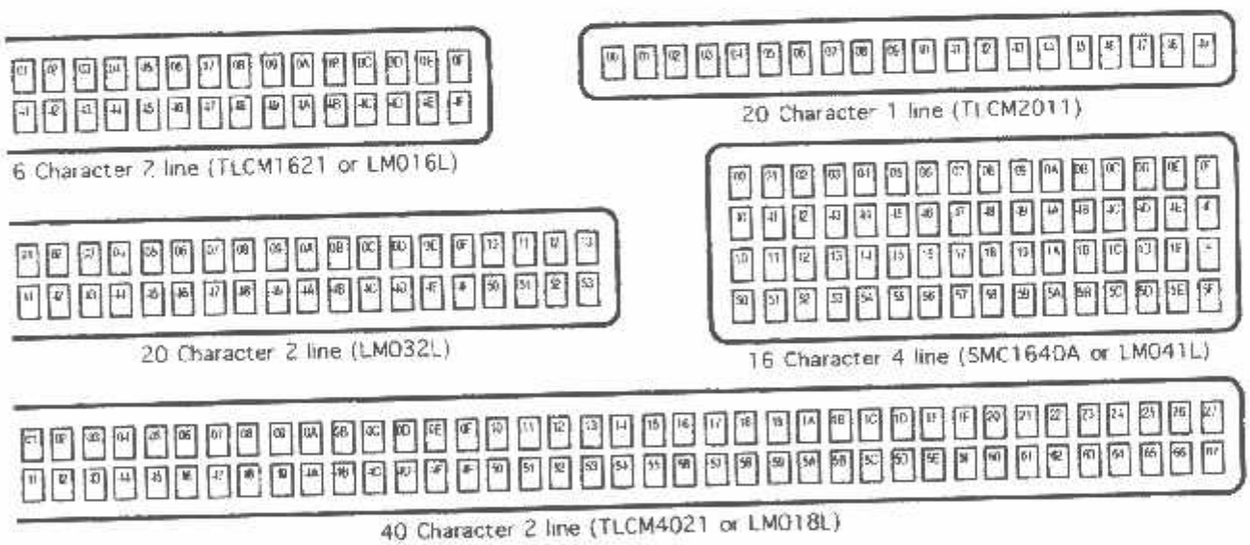


Figure 4: Examples of the relationship between addresses and display locations for typical module formats

Most are laid out conventionally, with two lines of characters, the first line starting at address 00000000 ($00) and the second line at address 01000000 ($40).
Two interesting exceptions were discovered during this article's research. The single-line module shown in Figure 4 is actually a two-line type, with the second line placed to the right of the first. In one-line mode, only the first 10 characters were visible.

The rather magnificent 4-line module is, actually, also a two-line type, with the two lines split and interlaced. This complicates the addressing a little, but can be sorted out with a bit of software.

Experiment 4: Shifting the Display

Regardless of which size l.c.d. module is being used, there are always 80 display locations that can be written to. On the smaller devices, not all 80 fit within the visible window of the module, but can be brought into view by shifting them all, either left or right, "beneath" the window area. This process must be carried out carefully, however, as it alters the relationship between addresses and their positions on the screen.

To experiment with shifting, first issue suitable Function Set, Display On/Off and Cursor commands, and, if necessary, the Clear Display command (you've met their codes above). Then enter all 26 letters of the alphabet as character data, e.g. 01000001 ($41) to 01011010 ($5A).

On a 16-character display, only *A* to *P* will be visible (the first 16 letters of the alphabet), and the cursor will have disappeared off the right-hand side of the display screen.

The Cursor/Display Shift command can now be used to scroll all the display locations to the left, "beneath" the l.c.d. window, so that letters *Q* to *Z* can be seen. The command is binary 00011000 ($18). Each time the command is entered (and using the E switch), the characters shift one place to the left. The cursor will re-appear from the right-hand side, immediately after the *Z* character.

Carry on shifting (*wasn't that a film title? Ed!*), and eventually the letters *A*, *B*, *C*, and so on, will also come back in from the right-hand side. Shifting eventually causes complete rotation of the display locations.

The binary command 00011100 ($1C) shifts the character locations to the right. It is important to note that this scrolling does not actually move characters into new addresses, it moves the whole address block left or right "underneath" the display window.

If the display locations are not shifted back to their original positions, then address $00 will no longer be at the left-hand side of the display. Try entering an Address Set command of value 10000000 ($80), after a bit of shifting, to see where it has moved to.

The Cursor Home command, binary 00000010 ($02), will both set the cursor back to address $00, and shift the address $00 itself back to the left-hand side of the display. This command can be used to get back to a known good starting position, if shifting and address setting gets a bit out of control.

The Clear Display command does the same as Cursor Home, but also clears all the display locations.

One final word about the Cursor/Display Shift command; it is also used to shift the cursor. Doing this simply increments or decrements the cursor address and actually has very little in common with shifting the display, even though both are achieved using the same command.

Experiment 5: Character Entry Mode

Another command listed in Table 2 is Character Entry Mode. So far, characters have been entered using auto-incrementing of the cursor address, but it is also possible to use auto-decrementing. Furthermore, it is possible to combine shifting of the display with both auto-incrementing and auto-decrementing.

Consider an electronic calculator. Initially, a single zero is located on the right-hand side of the display. As numbers are entered, they move to the left, leaving the cursor in a fixed position at the far right. This mode of character entry can be emulated on the l.c.d. module. Time for another experiment:

Send suitable Function Set, Display On/Off and Cursor commands as before. Next, and assuming a 16-character display, set the cursor address to 00010000 ($10). Then send the Character Entry Mode command, binary 00000111 ($07). This sets the entry mode to auto-increment/display shift left.

Finally, enter a few numbers from 0 to 9 decimal, i.e. from 00110000 to 00111001 ($30 to $39). Characters appear on the right-hand side and scroll left as more characters are entered, just like a normal calculator.

As seen in Table 2, there are four different Character Entry modes, 00000100 to 00000111 ($04 to $07), all of which have their different uses in real life situations.

Experiment 6: User-Defined Graphics

Commands 01000000 to 01111111 ($40 to $7F) are used to program the user-defined graphics. The best way to experiment with these is to program them "on screen." This is carried out as follows:

First, send suitable Function Set, Display On/Off and Cursor commands, then issue a Clear Display command. Next, send a Set Display Address command to position the cursor at address 00000000 ($00). Lastly, display the contents of the eight user character

locations by entering binary data 00000000 to 00000111 (S00 to $07) in turn. These characters will initially show up as garbage, or a series of stripes.

Now, send a Set CGRAM Address command, to start defining the user characters. Any value between 01000000 and 01111111 ($40 and $7F) is valid, but for now, use 01000000 ($40). The cursor will jump to the beginning of the second line, but ignore this, as it is not important.

Data entered from now on will build up the user-defined graphics, row by row. Try the following sequence of data: 00001110, 00010001, 00001110, 00000100, 00011111, 00000100, 00001010, 00010001 ($0E, $11, $0E, $04, $1F, $04, $0A, $11). A little "stick man" will appear on the display, with his feet in the gutter (the cursor line)!

By entering another set of eight bytes, the second user character can be defined, and so on.

How the CGRAM addresses correspond to the individual pixels of the user-defined graphics characters is illustrated in Figure 5. Up to eight graphics can be programmed, which then become part of the character set and can be called up using codes 00000000 to 00000111 ($00 to $07), or codes 00001000 to 00001111 ($08 to $0F), both of which produce the same result, i.e. 64 command codes available for user programming.

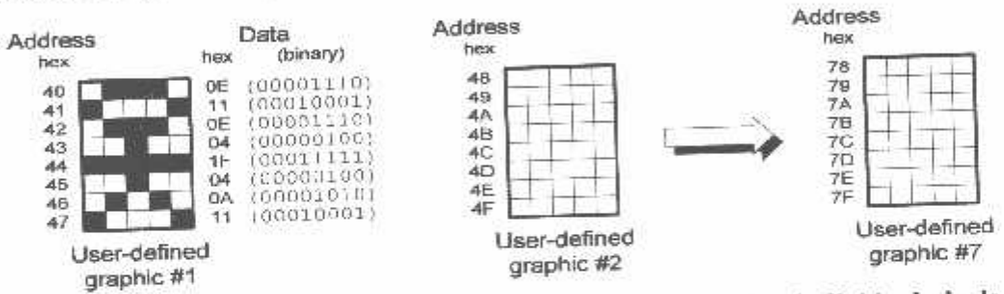


Figure 5: Showing how the CGRAM addresses correspond to individual pixels.

It can be seen that the basic character cell is actually eight pixels high by five pixels wide, but most characters just use the upper seven rows. The bottom row is generally used for the underline cursor. Since each character is only five pixels wide, only data bits 0 to 4 are used, bits 5 to 7 (the three "left-hand" bits) are ignored.

The CGRAM is volatile memory, which means that when the power supply is removed from the l.c.d. module, the user-defined characters will be lost. It is necessary for the microprocessor to load up the user-defined characters, by copying data from its own EPROM, early on in the program, certainly before it intends to display them.

Experiment 7: 4-Bit Data Transfer

The HD44780 l.c.d. control chip, found in most l.c.d. modules, was designed to be compatible with 4-bit microprocessors. The 4-bit mode is still very useful when interfacing to microcontrollers, including the PIC types.

Microcontroller input/output (I/O) pins are often at a premium and have to be rationed carefully between the various switches, displays and other input and output devices in a typical circuit. Bigger microcontrollers are available, which have more I/O pins, but miniaturisation is a key factor these days, along with cost, of course.

Once the display is put into 4-bit mode, using the Function Set command, it is a simple matter of sending two "nibbles" instead of one byte, for each subsequent command or character.

Nibble is a name devised by early computer enthusiasts in America, for half a byte, and is one of the more frivolous terms that has survived. By the time the 16-bit processors arrived, computing was getting serious, and the consumption analogies "gobble" and "munch" were never adopted!

When using 4-bit mode, only data lines D4 to D7 are used. On the prototype test rig, set the switches on the other lines, D0 to D3, to logic 0, and leave them there. Another experiment is now imminent.

In normal use, the unused data I/O lines D0 to D3 should either be left floating, or tied to one of the two power rails via a resistor of somewhere between 4k7[C24] and 47k[C24]. It is undesirable to tie them directly to ground unless the R/W line is also tied to ground, preventing them from being set into output mode. Otherwise the device could be programmed erroneously for 8-bit output, which could be unkind to lines D0 to D3, even though current limiting exists.

After power on, the l.c.d. module will be in 8-bit mode. The Function Set command must first be sent to put the display into 4-bit mode, but there is a difficulty. With no access to the lower four data lines, D0 to D3, only half the command can be applied.

Fortunately, or rather, by clever design, the 8-bit/4-bit selection is on data bit D4, which, even on the modified test rig, remains accessible. By sending a command with binary value 00100000 ($20), the 4-bit mode is invoked.

Now, another Function Set command can be sent, to set the display to two-line mode. Binary value 00101000 ($28) will do the trick. The value 00111000 ($38) may be a more familiar number, but it cannot be used now, or the display would be put straight back into 8-bit mode! Also, from now on, all commands and data must be sent in two halves, the upper four bits first, then the lower four bits.

Start by setting data lines D7, D6, D5 and D4 to 0010 ($2), the left-hand four bits of the 8-bit code, and press the E switch. Then we set the same four data lines to 1000 ($8), the right-hand four bits of the 8-bit code, and press the E switch again. It's a lot more laborious for a human being, but to a microcontroller, it's no problem!

Finish off by experimenting with other commands in 4-bit mode, and then try putting a few characters on the display.

A Final Note

The data sheets warn that under certain conditions, the l.c.d. module may fail to initialise properly when power is first applied. This is particularly likely if the Vdd supply does not rise to its correct operating voltage quickly enough.

It is recommended that after power is applied, a command sequence of three bytes of value 0011XXXX ($3X) is sent to the module. The value $30 is probably most convenient. This will guarantee that the module is in 8-bit mode, and properly initialised. Following this, switching to 4-bit mode (and indeed all other commands) will work reliably.

That's it – For Now!

Well, that's about it, really. You've made it this far, so now you know everything there is to know about l.c.d. modules. Well, almost everything!

The next step, of course, is to connect the display up to a controller of some sort, such as a PIC microcontroller, as will be seen next month. Then we shall also consider such things as signal timing and instruction delays.

Acknowledgement

The author expresses his gratitude to Bull Electrical in Hove and Greenweld Electronics in Southampton for their help in connection with this article.

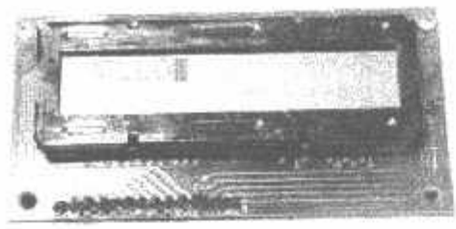
HOW TO USE INTELLIGENT L.C.D.S

Part Two

By Julyan Ilett

This paper was originally published as the second half of a two-part article in the March 1997 issue of *Everyday Practical Electronics* magazine (www.epemag.wimborne.co.uk), and is reproduced here with their kind permission.

How to use Intelligent L.C.D.s



By Julyan Ilett

An utterly "practical" guide to interfacing and programming intelligent liquid crystal display modules.

Part Two

In the first part of this article, the capabilities of character-based liquid crystal display (l.c.d.) modules were examined, using a few simple, practical experiments. A series of switches was all that was needed to evaluate the command set in its most fundamental form, in binary (or hexadecimal).

However, in almost all instances where an l.c.d. is to be used in a design, a microprocessor, or more probably a microcontroller, will be needed to drive it. This is the subject we examine now.

Good Times

The timing requirements of the HD44780 chip, the controlling device used in most character-based l.c.d. modules, are illustrated in Figure 6. The diagram provides the information for both read and write cycles, although some data sheets may show the two separately. Table 4 details the timing parameters referred to in Figure 6.

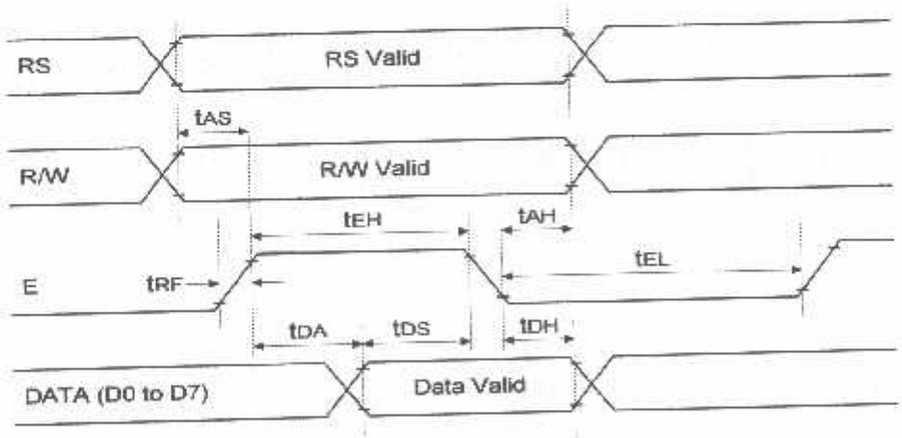


Figure 6: HD44780 timing diagram

In the experiments last month, commands were sent to the display by pressing switches on an experimental test rig. Nothing much went wrong there, so why is it necessary to have such a complex timing diagram?

Well, we human beings leave plenty of time between pressing one switch and the next, so the l.c.d. controller can easily keep up with us. Microcontrollers are faster than we are, though; they can toggle a control line several million times a second, and at such speeds the l.c.d. controller might not keep pace with the commands.

The timing diagram and its tabulated figures simply tell us how quickly the l.c.d. chip can respond so that we can program the microcontroller accordingly.

Let's take a typical microcontroller, one of the PIC devices which have become so popular, and see how we program it to control an l.c.d. from the quoted timing details.

Ve have published several PIC-based rojects in recent month's which are well vorth studying, along with their software istings. See the Back Issues *and* EPE 'CB Service *pages. Ed.)*

First, though, it must be pointed out that the discussions from now assume that you have a rudimentary understanding of programming PIC microcontrollers, and that you have suitable software and equipment for doing so. It is not the intention of this article to teach PIC programming.

The PIC microcontroller would be programmed to start by first setting the l.c.d.'s RS line to its correct logic level. This is the line that determines whether the l.c.d. should regard data as control instructions or character information. In cases where data needs to be read back from the l.c.d., the microcontroller must also have control over the R/W line (read/write), otherwise it should be connected to ground, as on the test rig.

The microcontroller can set up these two signals at the same time, or it may do one before the other, it doesn't really matter. What is important, is that they are both "valid" or "stable" for a minimum period of time before the level on the "E" (Enable) line is raised to a logic 1. On the diagram in Figure 6, this period is shown as "tAS" (time – address setup), and in the table this is specified as 140ns minimum. It can be more than 140ns, but it must not be any less.

Parameter	Description	Time
tAS	Address set up time	140ns min
tAH	Address hold time	10ns min
tDS	Data set up time	200ns min
tDH	Data hold time	20ns min
tDA	Data access time	320ns min
tEH	Enable high time time	450ns min
tEL	Enable low time	500ns min
tRF	Rise/Fall time	25ns max

Table 4: HD44780 Timing Parameters.

Once line E is high, it must not be brought low again until at least 450ns has elapsed, as is indicated by the "tEH" (time – enable high). Also, all eight data lines must be set to their

appropriate logic levels and allowed to stabilise for at least the "tDS" (time -- data setup) period of 200ns before bringing line E low again.

Note that the l.c.d. allows the data lines to be set up after line E is taken high. In the experiments last month, data was established well before the E switch was pressed, but either condition is allowed.

When line E is returned to a low level, there are also two hold times that must be taken into account. The "tAH" (time -- address hold) parameter indicates that the RS and R/W lines must not be altered for at least 10ns, and "tDH" (time -- data hold) shows that none of the data lines must change for at least 20ns.

One further restriction exists. The E line must not be taken high again (for the next command, that is) for another 500ns ("tEL": time -- enable low). This means that the total cycle time of the E line is 450ns plus 500ns. Allowing for the rise and fall times, indicated by "tRF", which should be no longer than 25ns each, an approximate value of 1µs can be calculated. This means that no more than one million commands (or one million characters) per second should be sent to the display, not a restriction that would normally present many problems!

Busy

The timing diagram doesn't tell the whole story, however. Much longer delays are required to enable the l.c.d. to process commands and data. Most instructions "tie up" the l.c.d. for 40µs, during which time it is said to be "busy." The Clear Display and Cursor Home commands, though, can take a lot longer.

Instruction	Time (Max)
Clear Display	82us to 1.64ms
Display & Cursor Home	40us to 1.64ms
Character Entry Mode	40us
Display On/Off & Cursor	40us
Display/Cursor Shift	40us
Function Set	40us
Set CGRAM Address	40us
Set Display Address	40us
Write Data	40us
Read Data	40us
Read Status	1us

Execution times for all the instructions are shown in Table 5. This includes all the commands, writing data to the display, and reading both data and status. The two Read instructions have not yet been experimented with, but reading the status of the l.c.d. is the method used to determine whether or not it is busy.

The practical implication of these instruction times is just a case of having to insert a delay between one instruction and the next. The first two commands, Clear Display and Cursor Home, have variable execution times that depend upon several factors. Not much is said about this variation in the data sheets, but it does involve returning the cursor to address 10000000 (&80), unshifting the display and, in the case of Clear Display, putting a space character into each display address.

There is one other important situation when the l.c.d. will be busy. This is immediately after it has been powered up. It takes some 10 to 15 milliseconds for the full initialisation sequence to be completed, during which time no instructions can be executed.

This has important implications for a circuit using a microcontroller. A suitable delay must be added to the beginning of the program, otherwise the l.c.d. won't be ready when the first few instructions are sent to it and could become locked up in a non-correctable condition, requiring the power to be switched off again for a while.

New Circuit

Time now to re-wire last month's experimental test rig to incorporate the PIC microcontroller. The circuit diagram of the modified arrangement is shown in Figure 7. There's no longer any need for the [illegible] output signals. It is not essential to use the PIC16F84 type specified in the diagram, the 54, 56, 61 and 71 types can all be used, but some minor changes may need to be made to [illegible] of the pin connections.

However, it is best to experiment with the PIC16C84 since it is the EEPROM (Electrically Erasable Programmable Read Only Memory) version of the microcontroller.

The use of this version is desirable because several different versions of software will [illegible] during the course of experimentation. Other versions of the microcontroller cannot be erased so easily, indeed some cannot be erased at all [illegible] OTP, One Time Programmable devices, for example).

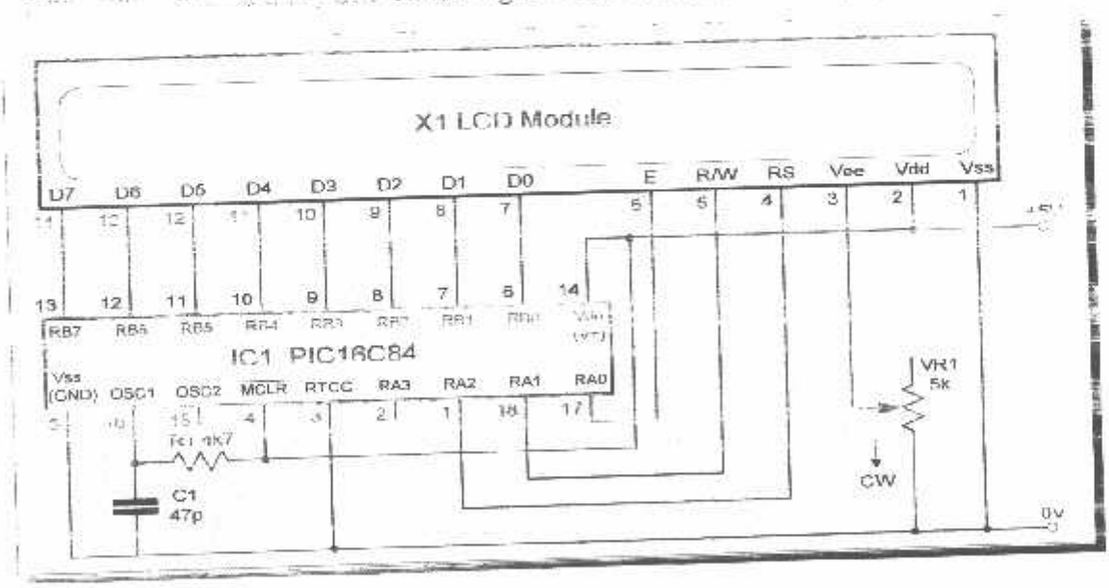
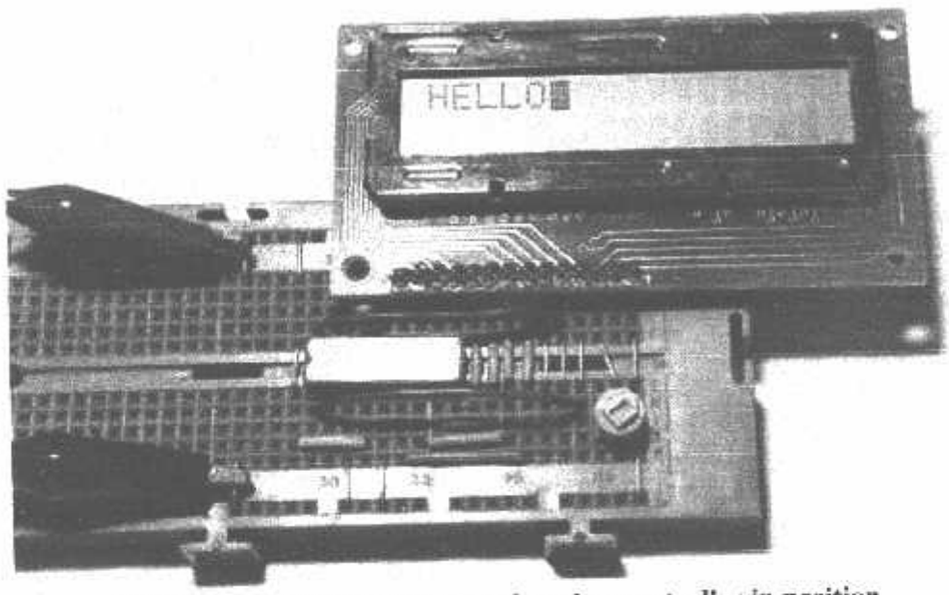


Figure 7: Circuit diagram for interfacing a PIC16C84 microcontroller to an l.c.d. module.

The microcontroller's Clock Option can be set for RC (resistor/capacitor) or any one of the XT (crystal) options, but the RC option is cheaper, and precise timing accuracy is not important in this instance. The values of the resistor R1 and capacitor C1 connected to the OSC1 input in Figure 7 will give a clock frequency of very approximately 2MHz. For the time-being, lower values of resistance or capacitance (for faster speeds) should be avoided, to ensure the software delays are sufficiently long.



The prototype test rig showing the microcontroller in position (it's actually a PIC16C54, although a PIC16C84 is recommended).

Experiment 8: PIC Program

Compile and program the contents of Listing 1 into the PIC microcontroller. It has been written for use with MPALC assembler software, although it can be readily translated to suit MPASM or TASM assembly.

Listing 1

```
list        p=16C84     shortdelay  ;tells assembler to generate code for this device
initialize  clrf        0D          ;clear register 0D, counter register
            clrf        0E          ;clear register 0E, short delay register
            clrf        0F          ;clear register 0F, long delay register
            clrf        05          ;Port A (register 05) outputs all set to logic 0
            clrf        06          ;Port B (register 06) outputs all set to logic 0
setports    movlw       0F8         ;Port A bits 0, 1, 2 as outputs (E, RS, R/W)
            tris        05
            movlw       00          ;Port B all bits as outputs (D0 to D7)
            tris        06
```

(...continued...)

Listing 1 (continued)

```
longdelay   call        shortdelay  ;long delay while lcd initialises
            decfsz      0F,f
            goto        longdelay
functionset bcf         05,02       ;RS line to 0 (Port A, bit 2)
            bcf         05,01       ;R/W line to 0 (Port A, bit 1)
            movlw       38          ;Function Set command
            movwf       06          ;put it on the data lines (Port B)
            call        pulse_e     ;pulse the E line high (Port A, bit 0)
            call        shortdelay
displayon   bcf         05,02       ;RS line to 0 (Port A, bit 2)
            bcf         05,01       ;R/W line to 0 (Port A, bit 1)
            movlw       0F          ;Display On/Off & Cursor command
            movwf       06          ;put it on the data lines (Port B)
            call        pulse_e     ;pulse the E line high (Port A, bit 0)
            call        shortdelay
            clrf        0D          ;set counter register to zero
message     movf        0D,w        ;put counter value in W
            call        text        ;get a character from the text table
            bsf         05,02       ;set RS line to 1 (Port A, bit 2)
            bcf         05,01       ;set R/W line to 0 (Port A, bit 1)
            movwf       06          ;put character on the data lines (Port B)
            call        pulse_e     ;pulse the E line high (Port A, bit 0)
            call        shortdelay  ;delay while l.c.d. is busy
            incf        0D,w        ;try incrementing the counter register
            xorlw       05          ;would that make it increase to 5?
            btfsc       03,02       ;set the zero flag in the status register
            goto        stop        ;stop if all characters displayed
            incf        0D,f        ;increment the counter register
            goto        message     ;go back and do the next character
stop        goto        stop        ;stop the program running
;Subroutines and text table
shortdelay  decfsz      0E,f        ;delay while l.c.d. is busy
            goto        shortdelay
            retlw       0
pulse_e     bsf         05,00       ;take E line high
            nop                     ;hold it high for one clock cycle
            bcf         05,00       ;take E line low again
            retlw       0
text        addwf       02,f        ;table of characters for message
            retlw       'H'
            retlw       'E'
            retlw       'L'
            retlw       'L'
            retlw       'O'
end
```

Once the PIC has been programmed, re-power up the circuit. The word HELLO will appear on the display. There may seem to be a lot of source code required to do such a simple job, but the program performs all the setting up that the display needs, and can form the basis of a more complex system.

Precisely what all these instructions do is important and will be described in some detail.

The first routine, "initialise," comprises five Clear File (clrf) instructions which set the contents of five registers to zero. Two of these registers, 05 and 06, relate to output Ports A and B.

When the microcontroller is powered up, all port pins are automatically set up as inputs, so that no damage is done to external circuitry. The "setports" routine uses "tris" instructions to redefine each bit of Ports A and B as either an input or an output.

(Be aware that Microchip, manufacturers of the PIC family, now discourage the use of "TRIS," a command becoming incompatable with their newer devices. There are alternative ways of achieving the same result, as discussed in the PIC data books. Ed.)

The "longdelay" routine keeps the microcontroller occupied while the l.c.d. is initialising. This delay must be no less than 15ms, but can be more, of course. The routines "functionset" and "displayon" are very similar and issue hexadecimal commands $38 and $0F (00111000 and 00001111) to the l.c.d. These numbers should be familiar from the experiments carried out in Part 1.

Both routines contain "call" instructions to two subroutines, "pulse_e" and "shortdelay," which can be seen towards the end of the listing. The "message" routine incorporates a program loop which is executed five times to output the five characters in the text table ("text") to the l.c.d. The PIC uses an unusual type of subroutine, comprising a list of "retlw" (return with literal) instructions which can be used to form tables of data.

Register $0D is used as a counter which is initially set to zero by the "clrf" instruction in the "initialise" routine. This value is then used as a pointer to the text table which contains the ASCII characters which spell HELLO.

The "stop" routine locks up the microcontroller to stop it doing anything else. Finally, the "end" directive is not a program command, but an instruction to tell the assembler to stop assembling.

A Good Read

The program in Listing 1 only writes to the display. In many applications this is quite satisfactory, and it has the advantage of allowing the R/W line on the l.c.d. to be connected to ground, which in turn saves an I/O (input/output) pin on the microcontroller.

It is possible (and sometimes necessary) to read data and status information from the l.c.d., but of course the R/W line must be actively connected in order to do this. Reading the

display differs from writing to it in some fundamental ways, so a re-examination of the timing diagram is now required, as the sequence of events is described.

Lines RS and R/W must be set up first, with R/W being set to a logic 1 this time. If RS is set high, data is returned indicating the character that is at the current cursor address. If RS is set low, a status byte is sent back, containing two separate items, bits 0 to 6 holding the current cursor address, and bit 7 containing the Busy flag.

The two Read instruction formats are shown in Table 6. After the necessary "address setup time" (tAS), the E line can be taken high. This is the point at which the read cycle differs from the write cycle, as the l.c.d.'s data lines will switch over to being outputs.

Instruction	RS	Binary D7	D6	D5	D4	D3	D2	D1	D0
Read Data	High	D	D	D	D	D	D	D	D
Read Status	Low	BF	A	A	A	A	A	A	A

D: Character data at current cursor address
A: Current cursor address ($00 to $7f)
BF: Busy Flag (0 = Ready, 1 = Busy)

Table 6: HD44780 Read Instructions.

Clearly, before the microcontroller starts this read cycle, it must change its data lines to inputs, otherwise outputs would be connected to outputs and a fight (known as *bus contention*) would ensue. In any case, if the microcontroller's data lines were not inputs at this time, it would not be able to read the data.

It takes a while for the l.c.d. to change its data lines to outputs, and stabilise the data on them, but it guarantees to do this within 320ns, the "data access time" (tDA). The microcontroller can then read this data in through its inputs, and as soon as it's happy that it's got it, the E line can go back down.

Most of the information that can be read back from the display must have been written there by the microcontroller in the first place, which explains why many designs can get away without having the R/W line connected up.

The Busy flag, though, can be useful to the microcontroller, to avoid using all those delay routines. For applications which need to put a lot of information on the display in a very short time, checking the Busy flag is the most efficient way of knowing when the display is ready.

Experiment 9: Status Reading

In this experiment, the program in Listing 1 will be altered to incorporate checking of the Busy flag. The plan here is to replace the subroutine "shortdelay," which has a fixed delay time, with another routine which will constantly check the Busy flag until it isn't busy any more.

Listing 2 shows the new subroutine, called "busywait." All occurrences of the "call shortdelay" instruction in Listing 1 should be replaced by "call busywait," including the three line section headed "longdelay." The program will put the message onto the display much more quickly than before, as unnecessary delays are eliminated.

Listing 2

```
busywait    movlw   0FF         ;Port Ball inputs (D0 to D7)
            tris    06
            bcf     05,02       ;RS line to 0 (Port A, bit 2)
            bsf     05,01       ;R/W line to 1 (Port A, bit 1)
            nop                 ;wait for tAS
busyread    bsf     0500        ;raise E line (Port A, bit 0)
            nop                 ;wait for tDA
            rlf     06,w        ;rotate BF into Carry flag
            bcf     05,00       ;lower E line (Port A, bit 0)
            nop                 ;wait for tEL
            nop                 ;wait for tEL
            btfsc   03,00       ;test Carry flag
            goto    busyread    ;if busy, go round again
            movlw   00          ;PortB all outputs (D0 to D7)
            tris    06
            retlw   0           ;return to main program
```

The first two lines of "busywait" change the assignment of Port B, so that all of its I/O lines become inputs. Following this, the RS and R/W lines are set up ready for the status read. For short delays, the "nop" (no operation) instruction can be used, it is ideal for the small delay times required by the l.c.d. interface.

The E line is then sent high and, after a short delay to allow for the data access time (tDA), the state of the Busy flag is read into the microcontroller. A "rotate left" (rlf) instruction is used here, to transfer the Busy flag on data line D7, into the PIC's Carry flag, where it can be stored prior to testing.

Line E is then taken low, after which a test is performed on the Carry flag using the "btfss" instruction. If the Carry flag is set, then the l.c.d. was busy at the moment the reading was taken, and the program branches back to perform another status read.

If the l.c.d. is found to be no longer busy, Port B is switched back for all bits to be outputs and the subroutine returns to the main program. The program uses more code, but saves time by avoiding unnecessary delays.

Experiment 10: Nibble Mode

The final experiment is to implement 4-bit data transfer mode between the l.c.d. and the microcontroller. This was examined in Experiment 7 in Part 1, so the technique should be reasonably well understood.

However, several changes need to be made, both to the circuit and to the program, details of which will be left to you to fully implement, but the principles involved are as follows:

Listing 3 shows some of the changes. Data lines D0 to D3 on the l.c.d. should be disconnected from the microcontroller (see Part 1 for how to deal with these unused l.c.d. lines). Data lines D0 to D3 on the microcontroller are now free to be used for other purposes, but for the time being can be left open circuit.

Listing 3

```
functionset  bcf       05,02      ;RS line to 0 (Port A, bit 2)
             bcf       05,01      ;R/W line to 0 (Port A, bit 1)
             movlw     20         ;1st Function Set command
             movwf     06         ;put it on the data lines (Port B)
             call      pulse_e    ;pulse the E line high (Port A, bit 0)
             call      busywait
functionset2 bcf       05,02      ;RS line to 0 (Port A, bit 2)
             bcf       05,01      ;R/W line to 0 (Port A, bit 1)
             movlw     28         ;2nd Function Set command
             movwf     0C         ;store command temporarily in 0C
             call      portnibble
             call      pulse_e    ;pulse the E line high (Port A, bit 0)
             swapf     0C,w       ;swap nibbles of 0C, put result in W
             call      portnibble
             call      pulse_e    ;pulse the E line high (Port A, bit 0)
             call      busywait
;Additional subroutine for nibble mode
portnibble   andlw     0F0        ;clear lower 4 bits of W
             iorwf     06,f       ;OR this with Port B
             iorlw     0F         ;set lower 4 bits of W
             andwf     06,f       ;AND this with Port B
             retlw     0
```

As we saw in Part 1, two separate Function Set commands are needed to set up the l.c.d. First, binary code 00100000 (hexadecimal $20) is sent while the l.c.d. is still in 8-bit mode, the mode which it automatically adopts when first switched on. This first code is followed by 00101000 ($28) sent as two separate nibbles, i.e. 0010 and 1000, both sent on lines D4 to D7. (Don't forget that lines RS and E must be dealt with appropriately when sending data.)

In Listing 3, the "functionset" routine of Listing 1 has been modified to send $20 instead of $38, and then a new routine, "functionset2," has been added, between "functionset" and "displayon," to send $2, and then $8. In the new routine, splitting a command byte into two nibbles is achieved by using the PIC's "swapf" instruction, which exchanges the upper and lower halves of any register.

The purpose of using 4-bit mode is that the other four bits of Port B (bits 0 to 3) can be used for something else, so writing data out on the upper half of Port B, must be done in such a way that it does not affect the lower half. In practice, *any* of the microcontroller's data lines can be used to send control the l.c.d., programming the software accordingly.

Individual "bit set" (bsf) or "bit clear" (bcf) instructions could be used to alter each bit in turn, but there is a simpler, more logical way, literally! A sequence of AND and OR instructions can be used to handle all eight bits of Port B, masking out those which must not be changed.

Listing 3 also shows a subroutine called "portnibble" which contains a sequence of four instructions that do the job. The upper four bits of the W register are transferred to the upper four bits of Port B, without affecting the lower four bits. A separate "pulse_e" call must be made for each of the two nibbles transferred, after which a single "busywait" call is added.

The "portnibble" routine is added to Listing 1 between the end of the "text" table and the "end" statement.

It is also necessary to alter the "displayon" routine of Listing 1 to operate in 4-bit mode, in the same way as is done in the "functionset2" routine. You can do the conversion for yourself to prove that you have understood so far!

More challenging, perhaps, are the modifications that have to be made to the "message" routine of the program. The procedure is the same, however, two 4-bit transfers being required instead of one 8-bit transfer. The use of 4-bit data transfer mode does add to the complexity of the software, but is well worth the effort as four extra I/O pins are released.

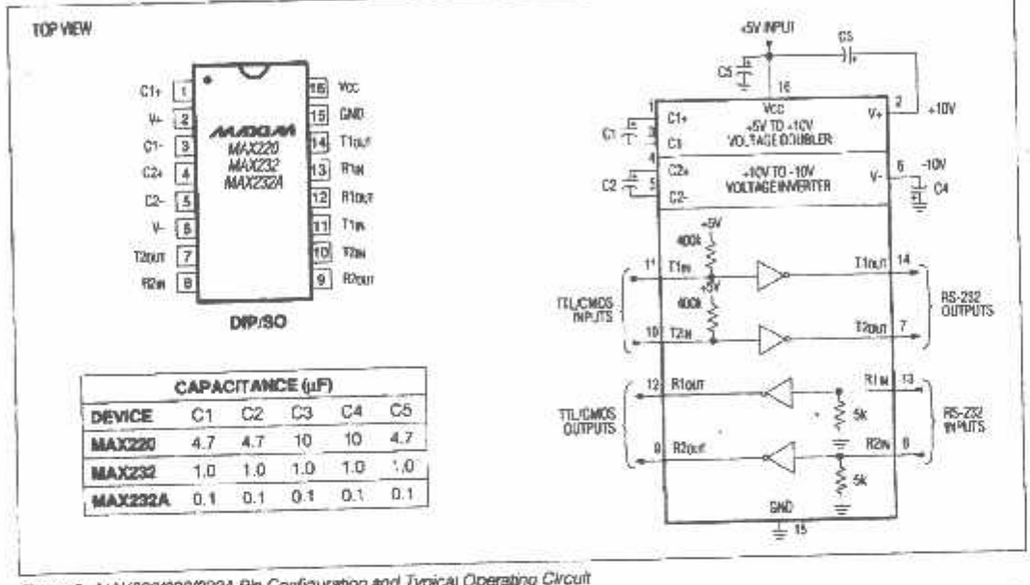
Digital Alternatives

So many electronic devices, these days, have a small keyboard and a liquid crystal display. For example, many of the better portable radio systems have dispensed with the potentiometer as a volume control, and the variable capacitor as a tuning control, and opted for a digital data entry and display alternative.

The advantages that such digital systems offer are undeniable, and even for the amateur constructor are readily achievable using low-cost but powerful microcontrollers, and inexpensive but versatile displays and keyboards, as the experiments in this two-part series have hopefully suggested to you.

(We have more PIC-controlled l.c.d. orientated projects in the pipeline. Ed.)

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/232/232A Pin Configuration and Typical Operating Circuit

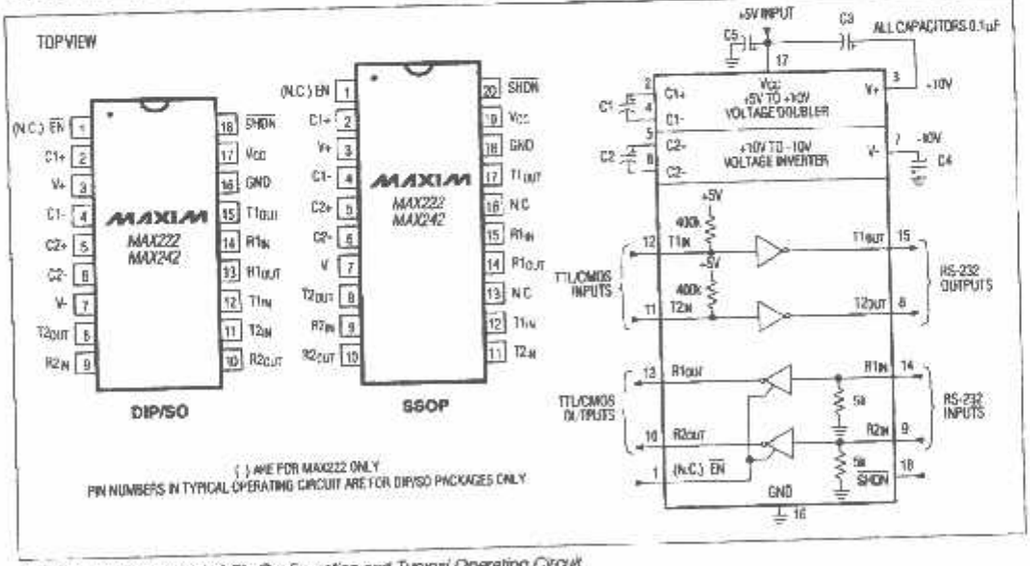


Figure 6. MAX222/MAX242 Pin Configuration and Typical Operating Circuit

t 4 : Interfacing Devices to RS-232 Ports

S-232 Waveforms

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

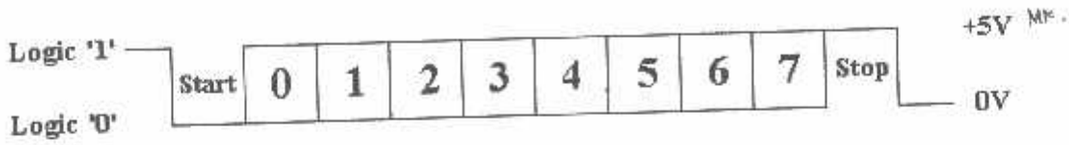


Figure 4 : TTL/CMOS Serial Logic Waveform

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

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

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

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

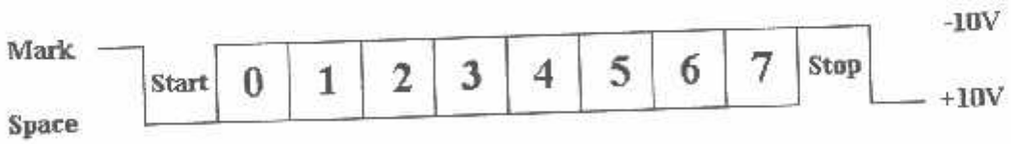


Figure 5 : RS-232 Logic Waveform

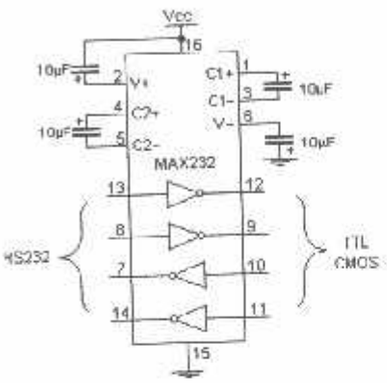
The above waveform applies to the Transmit and Receive lines on the RS-232 port. These lines carry serial data, hence the name Serial Port. There are other lines on the RS-232 port which, in essence are *Parallel* lines. These lines (RTS, CTS, DCD, DSR, DTR, RTS and RI) are also at RS-232 Logic Levels.

S-232 Level Converters

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

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

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



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

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

There are also many variations of these devices. The large value of capacitors are not only bulky, but also expensive. Therefore other devices [illegible] *1 micro farad Capacitors)*. However the MAX-232 is the most common, and thus we will use this RS-232 Level Converter in our examples.

Making use of the Serial Port

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

50 and Compatible UARTs

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

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

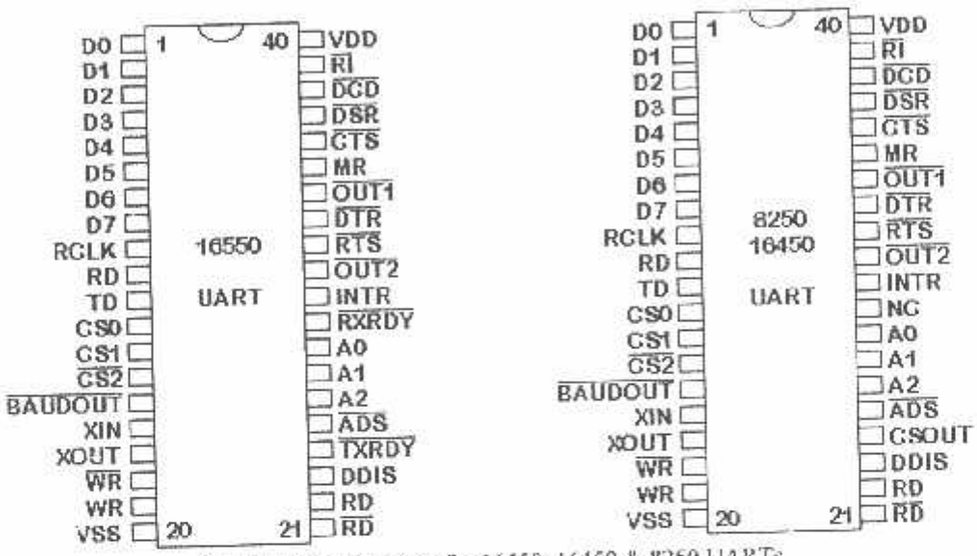


Figure 8 : Pin Diagrams for 16550, 16450 & 8250 UARTs

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

)P6402, AY-5-1015 / D36402R-9 etc UARTs

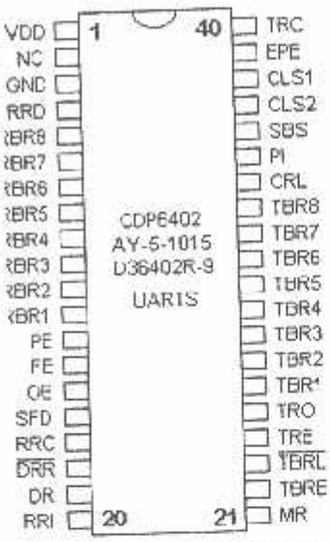


Figure 9 : Pinout of CDP6402

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

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

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

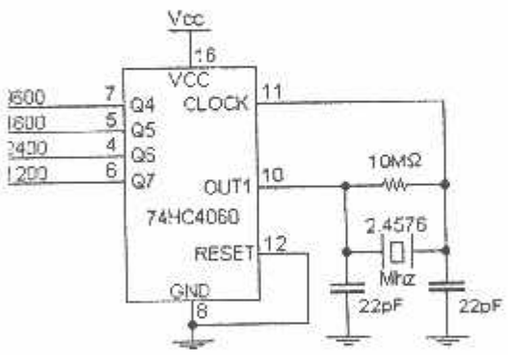
Pin 16	SFD	Status Flag Disable	When High, Status Flag Outputs (PE, FE, OE, DR and TBRE) are High Impedance
Pin 17	RRC	Receiver Register Clock	x16 Clock input for the Receiver Register.
Pin 18	nDRR	Data Received Reset	Active Low. When low, sets Data received Output Low (i.e. Clears DR)
Pin 19	DR	Data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

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

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

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

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



ire 10 : Baud Rate Generator using a 74HC4060

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

Table 19 : Possible Baud Rates using a 74HC4060

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

Microcontrollers

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

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

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

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

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

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

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

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Ferdy Ridwan Dinata
NIM : 01.17.030
Masa Bimbingan : 20 Februari 2007s/d 20 Agustus 2007
Judul Skripsi : Perencanaan dan Pembuatan kasir menu cepat saji direstoran auto debet dengan menggunakan kartu RFID berbasis Mikrokontroller Renesas R8C/13 Tiny.

No	Tanggal	Uraian	Paraf Pembimbing
1		Sumber gambar bab 2	
2		Flowchart	
3		Format penulisan Sub bab	
4		Kesimpulan	
5		Diagram blok.	
6		Acc Makalah Seminar hasil	
7		Acc bab I s/d 5, daftar isi	
8	19-3-2007	Acc complete	
9			
10			

Malang, 19 - 3 - 2007
Dosen Pembimbing

(Joseph Deddy Irawan.ST.MT)
NIP.P. 103.9800.324

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Ferdy Ridwan Dinata
NIM : 01.17.030
Masa Bimbingan : 20 Februari 2007s/d 20 Agustus 2007
Judul Skripsi : Perencanaan dan Pembuatan kasir menu cepat saji direstoran auto debet dengan menggunakan kartu RFID berbasis Mikrokontroller Renesas R8C/13 Tiny

No	Tanggal	Uraian	Paraf Pembimbing
1	26/07/07	Konsultasi Bab 4, Bab 5, Bab 3	
2		Konsultasi Bab 4 & Bab 5	
3		Melulas Seminar	
4	14/03/07	Acc Melulas Skripsi	
5			
6			
7			
8			
9			
10			

Malang, 14-03-2007
Dosen Pembimbing

(Komang Somawirata,ST,MT)
NIP.P. 103.0100.361