SKRIPSI

PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLER AT89S51

Disusun Oleh:

OCVAN TEJA PRATAMA

01.17.056

KONSENTRASI TEKNIK ELEKTRONIKA

JURUSAN TEKNIK ELEKTRO S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2009

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLER AT89S51

SKRIPSI

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

Disusun oleh:

OCVAN TEJA PRATAMA

NIM: 01.17.056

Diperiksa dan Disetujui Dosen Pembimbing	Mengetahui Ketua Jurusan T. Elektro S-1
(I Komang Somawirata,ST,MT) NIP. 1030100361	(Ir.F.Yudi Limpraptono,MT) NIP.Y.1039500274

KONSENTRASI TEKNIK ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2009

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	:	OCVAN TEJA PRATAMA
NIM	:	01.17.056
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika
Judul Skripsi	:	

"PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLLER AT89S51"

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

Hari	:	Senin
Tanggal	:	23 Maret 2009
Dengan nilai	:	71,2 (B+)

Panitia Ujian Skripsi:

Ketua Majelis Penguji

(Ir. H.Sidik Noertjahjono, MT)
NIP.Y.1028700163

Sekretaris Majelis Penguji

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

Anggota Penguji:

Penguji I

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

Penguji II

(Irmalia Suryani Faradisa, ST, MT)
NIP.P.1030000356

PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLER AT89S51

OCVAN TEJA PRATAMA
Jurusan Teknik Elektronika, Fakultas Teknologi Industri, Institut Teknologi Nasional
Jl. Raya Karangploso KM 2 Malang
vans_pratama@ymail.com

Abstrak

Makalah ini membahas mengenai high pass fiter digital. Filter digital adalah suatu piranti yang dibutuhkan oleh system-sistem elektronika. Pengimplementasian suatu rancangan filter digital kedalam perangkat keras yang mampu memilki kinerja secara waktu nyata (real time) sampai sekarang masih menjadi permasalahan tersendiri. Dalam perancangan ini akan dimplementasikan sebuah filter digital lowpass dengan komponen-komponen yang mana terdiri dari filter yang dapat bersifat aktif dan pasif yaitu Switched Capasitor MF10CCN kedalam perangkat keras Mikrokontroller AT89S51. Untuk mengetahui respon dari Low Pass Filter tersebut dibutuhkan program bantu bernama RMAA (Righ Mark Audio Analyzer), untuk pengujian secara manual dilakukan dengan Oscilloscope dan Function generator. Hasil dari pengujian high Pass Filter bekerja dengan baik, semua frekuensi di atas cut off diloloskan.
Kata kunci *: AT 89S51, MF10CCN, High Pass Filter Digital, RMAA.*

Abstrack

This paper is research to High Pass Filter Digital Analyze. Digital Filter is the component which us electronical system. Digital Filter are he planning application in the hardware can with processing an real time was to get now problem complementary. In application which implemantation to High Pass Filter Digital, where the component – component is to standing from filter to each activity filter and passive is your Switched Capasitor Filter (MF 10) and in the hardware is Microcontroller AT89S51. For the detected respons from Low Pass Filter Digital can helping Program assembly RMAA (Right Mark Audio Analyzer)for the tes any all to manual which oscilloscope an function generator. The nominally from test Low Pass Filter Digital are hardest working can be free high cut off frequency
***Key Word** : AT89S51, MF 10CCN, High Pass Filter Digital, RMAA.*

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah memberikan Rahmat kepada kita semua dan khususnya penulis sehingga dapat menyelesaikan laporan Skripsi ini dengan judul :

PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLER AT89S51

Pembuatan skripsi ini guna memenuhi syarat akhir kelulusan pendidikan jenjang strata-1 di Program Studi Elektonika Jurusan Teknik Elektro Institut Teknologi Nasional Malang.

Dengan segala kerendahan hati atas keberhasilan penyelesaian laporan Skripsi ini, Penyusun mengucapan terima kasih kepada :

1. Kedua Orang Tuaku dan adik2-ku terima kasih atas segala doa dan dukungannya.
2. Bapak Dr. Ir. Abraham Lomi, MSEE, selaku Rektor Institut Teknologi Nasional Malang
3. Bapak Ir. F. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro Institut Teknologi Malang.
4. Bapak I Komang Somawirata, ST, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, saran dan pemikiran dalam menyelesaikan laporan skripsi ini.
5. Teman–teman teknik elektronika S-1 yang telah membantu dalam penyelesaian skripsi ini saya ucapkan terima kasih.

6. Teman–teman di kontrakan Perum Pondok Alam Sigura-gura A4/7, terimakasih atas dukungannya.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang bersifat membangun sangat diharapkan dari pembaca khususnya mahasiswa Teknik Jurusan Elekto ITN Malang, agar di masa datang akan lebih baik lagi.

Harapan dari penulis semoga skripsi ini dapat bermanfaat.

Malang, Maret 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLER AT89S51

A. Latar Belakang

Berdasarkan perkembangan teknologi sekarang ini menggunakan system digital. Perkembangan yang pesat ini merupahkan hasil kemajuan teknologi komputer digital dan industri rangkaian terintegrasi, dimulai dengan integrasi skala medium (MSI) dan kemudian integrasi skala besar (LSO) dan sekarang integrasi skala yang sangat besar (VLSI). Rangkaian digital yang murah dan relatif cepat ini memungkinkan untuk mengkonstruksi system digital cangih yang dapat melakukan fungsi dan tugas pemrosesan sinyal digital komplek, karena itu pemrosessan siyal dengan piranti analog sekarang dilakukan dengan perangkat keras digital yang lebih murah.

Sebagian besar sinyal – sinyal yang ditemukan dalam sians dan teknologi adalah analog, karena sinyal – sinyal seperti itu dapat diproses secara langsung dengan system analog yang tepat (seperti tapis atau penganalisis frekuensi). Pemrosesan sinyal digital meyediakan suatu metode alternatif untuk pemrosesan sinyal analog. Ada beberapa alasan mengapa pemrosesan sinyal digital untuk analog lebih baik untuk memproses sinyal itu secara langsung menjadi analog, karena suatu sistem digital yang dapat diprogram memiliki keluwesan untuk mengkonfigurasi ulang operasi – operasi pemrosesan sinyal digital secara sederhana dengan cara merubah programnya.

B. Tujuan

Tujuan dari tugas akhir ini adalah bahwa kita dapat mengetahui bagaimana cara membuat high pass filter digital menggunakan MF10 yang dikontrol oleh mikrokontroller AT89S51.

C. Rumusan Masalah

Dalam pembuatan High Pass Filter Digital ini pertimbangan keakuratan juga memainkan peranan penting dalam penentuan bentuk sinyal. Maka dalam perumusan masalah ini akan membahas :

1. Bagaimana cara merancang sebuah high pass filter digital?
2. Bagaimana cara mengimplementasikan high pass filter digital ke dalam mikrokontroler AT89S51?

D. Batasan Masalah

Dalam perencanaan alat high pass filter digital berbasiskan mikrokontroler AT89S51 ini dapat merumuskan beberapa masalah yang akan dibahas sebagai berikut :

- Hanya membahas high pass filter digital saja.
- Tidak membahas masalah catu daya.
- Frekuensi yang dipakai 20Hz – 20KHz.

E. Metodologi

Metodologi yang digunakan dalam pembuatan aplikasi ini sebagi berikut :

- Studi Literatur

  Dengan mempelajari berbagai literatur yang berhubungan dengan sistem kerja alat sebelum dilakukan perancangan dan pembuatan keseluruhan sistem.

- Perancangan Perangkat Keras

  Sebelum membuat sistem kerja alat tersebut yakni menggunakan mekanika alat berupa : Mikrokontroler AT89S51, LCD, Keypad, MF 10.

- Pembuatan dan Pengujian Perangkat Keras

  Perancangan yang dilakukan realisasi pengujian masing – masing bagian

  (sub sistem) dari perangkat tersebut.

- Pengujian Keseluruhan Sistem

  Sub sistem yang sudah ada diitegrasikan menjadi sebuah sistem kerja alat dan dilakukan pengujian sistem tersebut apakah telah berjalan sesuai dengan yang direncanakan.

F. Sistematika Penulisan

Sistematika dalam penulisan skripsi ini adalah sebagai berikut :

BAB I : PENDAHULUAN

Membuat Latar Belakang, Tujuan, Rumusan Masalah, Batasan Masalah dan Sistematika Penulisan.

BAB II : DASAR TEORI

Membahas semua teori dasar yang berkaitan langsung dengan perencanaaan sistem alat.

BAB III : PERANCANGAN DAN PEMBUATAN ALAT

Membahas perancangan dan pembuatan sistem dari alat yang meliputi hardware dan software.

BAB IV : PENGUJIAN ALAT

Membahas tentang pengujian dan pengukuran karakteristik dari hasil perancangan software dan hardware setelah pengambilan sampel dari hasil pengujian alat, setelah itu dilakukan analisa hasil pengujian alat tersebut.

BAB V : PENUTUP

Berisi kesimpulan dari keseluruhan dari perancangan dan pembuatan alat, serta berisi saran untuk pengembangan alat.

BAB II

LANDASAN TEORI

Dalam merencanakan dan merealisasikan sistem ini dibutuhkan pemahaman mengenai pengetahuan yang berhubungan dengan aplikasi tersebut. Pemahaman tersebut akan sangat bermanfaat dalam perancangan perangkat keras maupun perangkat lunak. Adapun pengetahuan yang mendukung perencanaan dan realisasi alat antara lain pengetahuan tentang high pass filter, mikrokontroller AT89S51, LCD, Keypad, MF10.

2.1. Filter Aktif.

Filter adalah sebuah rangkaian yang dirancang agar melewatkan suatu pita frekuensi tertentu untuk memperlemah semua isyarat di luar pita ini. Jaringan filter bisa bersifat aktif maupun pasif, jaringan filter pasif hanya berisi tahanan, inductor dan kapasitor. Induktor jarang digunakan dalam filter aktif, sebab ukurannya besar, mahal dan memiliki komponen – komponen bertahanan dalam yang besar.

Filter aktif mempunyai keuntungan dibandingkan dengan filter pasif yaitu :

1. Penguatan dan frekuensinya mudah diatur, selama op-amp masih memberikan penguatan dari sinyal input tidak sekaku seperti pada filter pasif. Pada dasarnya filter aktif lebih gampang diatur.
2. Tidak ada masalah beban, karena tahanan input tinggi dan tahanan output rendah. Filter aktif tidak membebani sumber input.

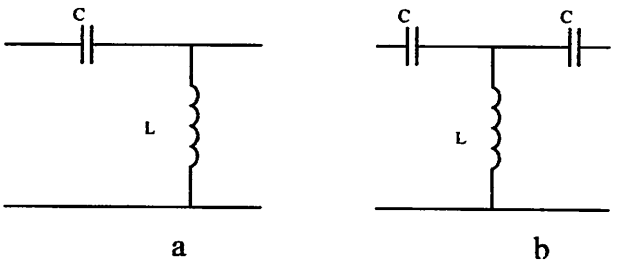
3. Umumnya filter aktif lebih ekonomis dari pada filter pasif, karena pemilihan variasi dari op-ampyang murah dan tanpa inductor yang biasanya harganya mahal.

Dalam perencanaan dan pembuatan alat pengontrol alat-alat elektronik rumah tangga menggunakan frekuensi ultra tinggi, jenis filter yang digunakan adalah filter pasif yaitu high pass filter dan band pass filter. Pemilihan jenis filter ini dikarenakan pengontrol untuk frekuensi tinggi bekerja pada frekuensi tinggi yaitu diatas cut off.

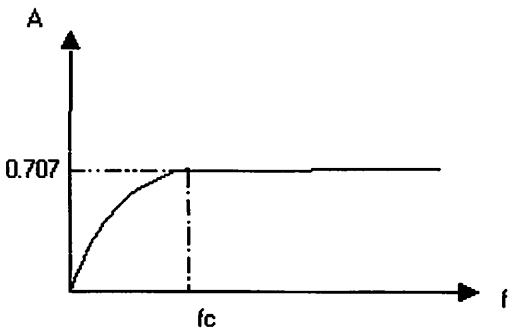
2.1.1. High Pass Filter.

Filter high pass adalah sebuah rangkaian yang memperlemah semua isyarat di bawah suatu frekuensi cutoff tertentu dan melewatkan semua isyarat yang frekuensinya di atas frekuensi cutoff itu.

Konfigurasi filter tersebut dapat dilihat dalam gambar 2-1, beserta kurva karakteristiknya.



Gambar 2.1. Konfigurasi Filter



Gambar 2.2. Konfigurasi High Pass Filter.

Gambar 2.1. fc adalah frekuensi cut off yang memiliki nialai sebesar $\frac{1}{\sqrt{2}}$ atau 0.707 nilai maksimumnya. Titik ini merupakan titik daya setengah, karena pada titik ini daya masuk kerangakain menjadi daya setengah maksimal. Besarnya L dan C dalam gambar 2-8 dapat dihitung dengan persamaan sebagai berikut :

$$C = \frac{1}{4\pi fcR} \quad \text{.....(2-1)}$$

$$L = \frac{R}{4\pi fc} \quad \text{.....(2-2)}$$

$$fc = \frac{1}{4\pi\sqrt{LC}} \quad \text{.....(2-3)}$$

Dengan :

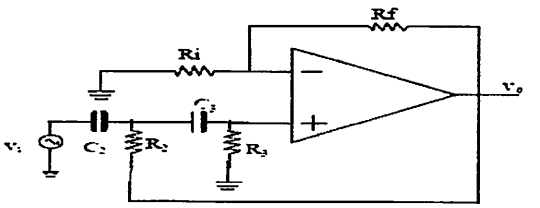
L = Induktansi (H)

C = Kapasitansi (F)

R = Resistansi beban (Ω)

fc = Frekuensi cut off (Hz)

Filter dalam gambar 2-1a disebut sebagai rangkaian high paas filter dasar. Gambar 2-1b merupakan rangkaian serupa yang dikenal sebgai filter jenis T. Nilai L dan C pada kedua rangkaian dapat diperoleh malalui persamaan (2-2) dan 2-3). Gambar rangkaiannya sebagai berikut :



Gambar 2.3. Rangkaian high pass filter.

Dengan demikian dapat dirancang frekuensi cutoff dari high pass filter tersebut dengan ketentuan :

- Meyamakan nilai C1 = C2 = C3
- Mencari nilai R2 = ½ R1

$$R1 = \frac{1{,}414}{\omega C}$$

- $$R1 = \frac{1{,}414}{2\pi f o C}$$
- $$f o = \frac{1{,}414}{2\pi R1 C}$$

2.2. Mikrokontroller AT89S51

Perbedaan mendasar antara mikrokontroller dan mikroprosesor adalah mikrokontroler selain memiliki CPU juga dilengkapi memori dan *input output* yang merupakan kelengkapan sebagai sistem minimum mikrokomputer sehingga

sebuah mikrokontroller dapat dikatakan sebagai mikrokomputer dalam keping tunggal *(Single Chip Microcomputer)* yang dapat berdiri sendiri.

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

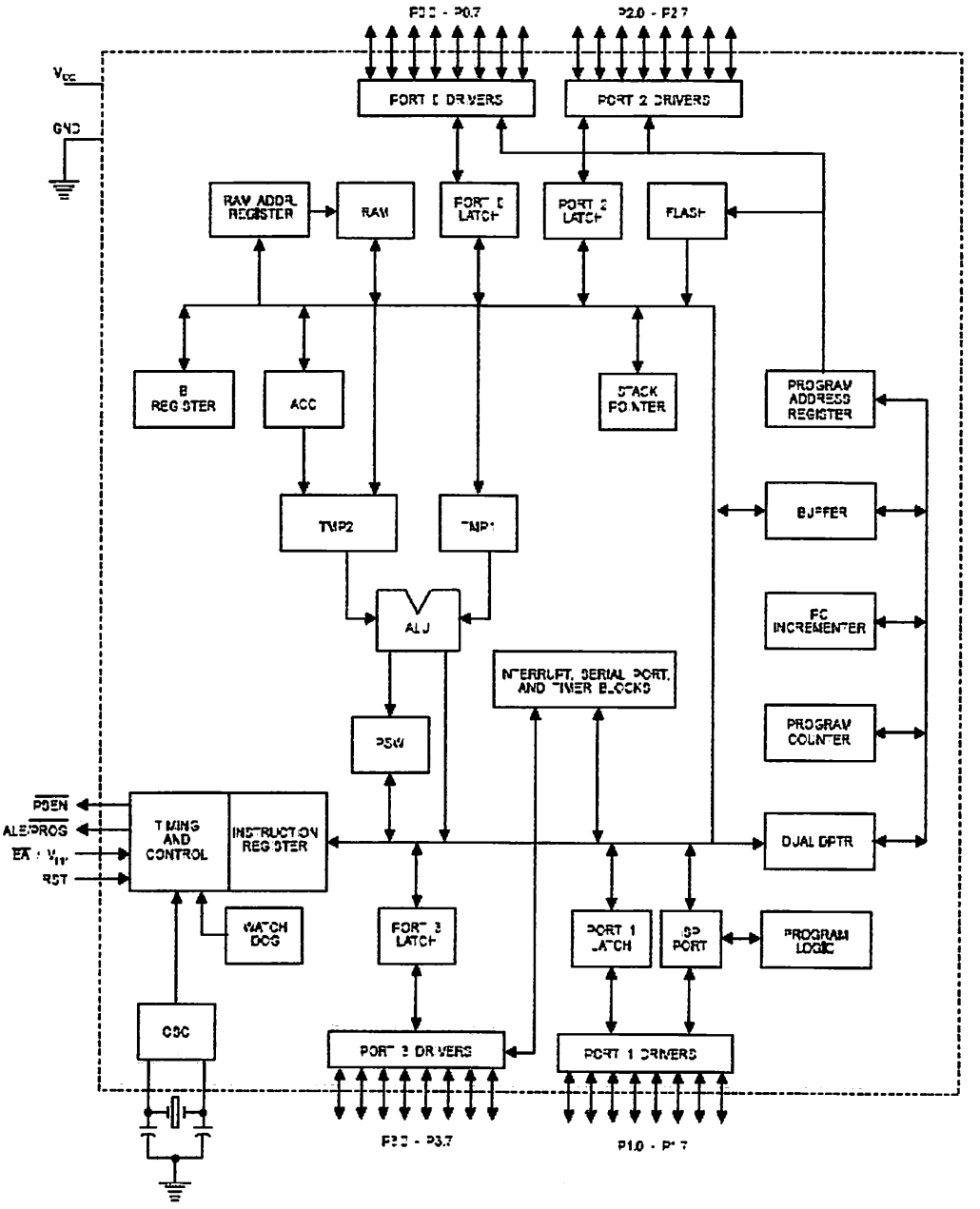
2.2.1. Perangkat keras mikrokontroler AT89S51

Secara umum Mikrokontroller AT89S51 memiliki :

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

- Kecepatan pelaksanaan intruksi per siklus 1μs pada frekuensi clock 12 Mhz
- 2 DPTR (*Data Pointer*)
- *Watchdog Timer*
- Fleksibel ISP Programming

Dengan keistimewaan Di atas pembuatan alat menggunakan AT89S51 menjadi lebih sederhana dan tidak memerlukan IC pendukung yang banyak. Adapun blok diagram dari Mikrokontroller AT89S51 adalah sebagai berikut :

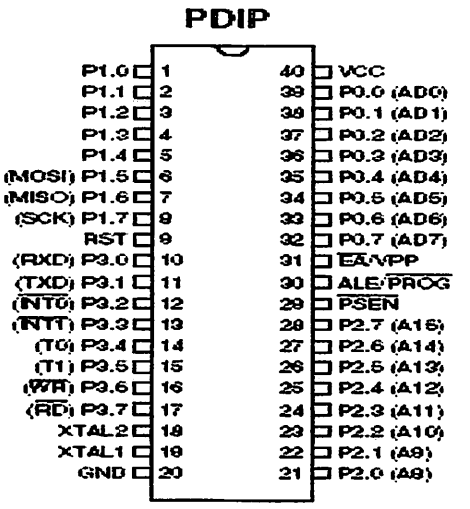


Gambar 2.4. Diagram Blok Mikrokontroller AT89S51

(Sumber : Data Sheet Atmel AT89S51, halaman 3)

2.2.2. Konfigurasi Pin Mikrokontroller AT89S51

Mikrokontroller AT89S51 terdiri dari 40 pin dengan konfigurasi sebagai berikut :



Gambar 2.5. IC AT89S51

(Sumber : Data Sheet Atmel AT89S51)

Fungsi tiap pin-nya adalah sebagai berikut :

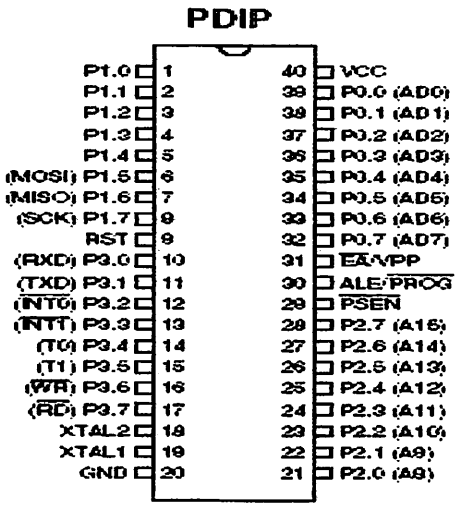
1. Pin 1 sampai 8, **Port 1**

   Merupakan 8 bit I/O Bi-directional yang dilengkapi dengan internal Pull - Up. Ketika diberikan logika '1' pin ini akan di *Pull-Up* secara *internal* sehingga dapat digunakan sebagai *input*. Sebagai masukan jika pin – pin ini dihubungkan ke ground maka masing – masing pin ini dapat menghantarkan arus karena di *Pull-High* secara internal. Port 1 juga menerima *Low Order Address Bytes* selama melakukan verifikasi program.

Pada *port* 1 di AT89S51 pin ini mempunyai alternatif seperti pada tabel berikut ini:

2.2.2. Konfigurasi Pin Mikrokontroller AT89S51

Mikrokontroller AT89S51 terdiri dari 40 pin dengan konfigurasi sebagai berikut :



Gambar 2.5. IC AT89S51

(Sumber : Data Sheet Atmel AT89S51)

Fungsi tiap pin-nya adalah sebagai berikut :

1. Pin 1 sampai 8, **Port 1**

   Merupakan 8 bit I/O Bi-directional yang dilengkapi dengan internal Pull - Up. Ketika diberikan logika '1' pin ini akan di *Pull-Up* secara *internal* sehingga dapat digunakan sebagai *input*. Sebagai masukan jika pin – pin ini dihubungkan ke ground maka masing – masing pin ini dapat menghantarkan arus karena di *Pull-High* secara internal. Port 1 juga menerima *Low Order Address Bytes* selama melakukan verifikasi program.

Pada *port* 1 di AT89S51 pin ini mempunyai alternatif seperti pada tabel berikut ini:

Port Pin	Alternative Functions
P1.5	MOSI (Master Output Sleve Input)
P1.6	MISO ((Master Input Sleve Output)
P1.7	SCK (Serial Clock)

Tabel 2.1. Fungsi – Fungsi Alternative Port 1

(Sumber : Data Sheet Atmel AT89S51)

2. Pin 9, **RST (*Reset*)**

   Merupakan pin yang aktif tinggi (*high*), pin ini aktif tinggi selama dua siklus mesin yang akan membuat mikrokontroler AT 89S51 menjalankan rutin *reset.*

3. Pin 10 sampai 17, **Port 3**

   Port 3 sebagai 8 bit I/O Bi-directional yang dilengkapi dengan *Pull-Up Internal.* Penyangga keluaran port 3 dapat memberikan atau menyerap arus empat masukan TTL (sekitar 1,6 mA). Jika diberikan logika '1' pada pin - pin port 3, maka masing – masing pin akan di *Pull High* oleh *Pull-Up internal* sehingga dapat digunakan sebagai *input*-an. Sebagai inputan, jika pin – pin port 3 dihubungkan ke *ground*, maka masing – masing kaki akan memberikan arus karena di *Pull High* secara internal, dimana Port 3 juga mempunyai fungsi-fungsi khusus yang dimiliki oleh keluarga MCS-51. Fungsi tersebut dapat dilihat dalam berikut ini :

Nama Penyemat	Fungsi Khusus
Port 3.0	RxD (port masukan serial)
Port 3.1	TxD (port keluaran serial)
Port 3.2	/INTO (masukan interupsi eksternal 0)

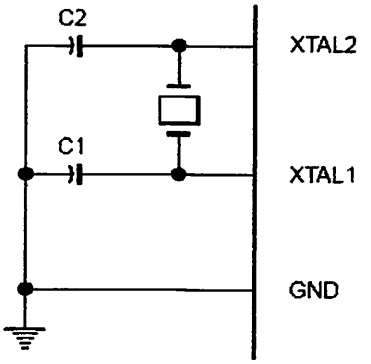
Port 3.3	/INT1 (masukan interupsi eksternal 1)
Port 3.4	T0 (masukan pewaktu eksternal 0)
Port 3.5	T1 (masukan pewaktu eksternal 1)
Port 3.6	/WR (sinyal tulis memori data eksternal)
Port 3.7	/RD (sinyal baca memori data eksternal)

Tabel 2.2. Fungsi Khusus Pada Port 3

(Sumber : Data Sheet Atmel AT89S51)

4. Pin 18 sampai 19, ***X-TAL 1 dan X-TAL 2***

X-TAL 1 merupakan masukan ke rangkaian osilator *internal* sedangkan X-TAL 2 keluaran dari rangkaian osilator *internal*. Untuk keperluan ini diperlukan kapasitor penstabil sebesar 30μF. Dan nilai dari X-TAL tersebut antara 3 – 33 Mhz. Untuk lebih jelasnya dapat dilihat gambar pemasangan X-TAL serta kapasitor yang digunakannya



Gambar 2.6. Osilator Eksternal AT89S51

(Sumber : Data Sheet Atmel AT89S51)

5. Pin 20, **GND (*Ground*)**

Dihubungkan dangan *ground* rangkaian.

6. Pin 21 sampai 28, **Port 2**

Port 2 berfungsi sebagai 8 bit I/O Bi-directional yang dilengkapi dengan internal *Pull-Up* Penyangga keluaran port 2 dapat memberikan atau

menyerap arus empat masukan TTL (sekitar 1,6 mA) Jika diberikan logika '1' pada pin – pin port 2, maka masing – masing pin akan di *Pull High* secara *internal* sehingga dapat digunakan sebagai *input*-an. Sebagai *input*-an jika pin – pin port 2 dihubungkan ke *ground* (di *Pull-Low*), maka, masing – masing pin dapat menghantarkan arus karena di *Pull High* secara *internal*. Port 2 mengeluarkan alamat bagian tinggi (A8-A15), selama pengambilan instruksi dari memori program eksternal dan selama pengaksesan memori data eksternal yang menggunakan perintah dengan alamat 16-bit (dengan perintah "MOVX @DPTR").

7. Pin 29, $\overline{\text{PSEN}}$ (***Program Store Enable***)

   Pin ini aktif rendah yang merupakan *strobe* pembacaan ke program memori eksternal.

8. Pin 30, **ALE** (***Address Latch Enable***) / $\overline{\textit{PROG}}$

   Keluaran ALE menghasilkan pulsa – pulsa untuk menahan alamat rendah (A0-A7) pada port 0, selama dilakukan proses baca atau tulis memori *external*. Pin ini juga berfungsi sebagai masukan pulsa program ($\overline{\textit{PROG}}$) selama pemrograman EEPROM *external*. Pada operasi normal, ALE akan berpulsa dengan laju 1/6 dari frekuensi kristal dan dapat digunakan sebagai pewaktuan atau pendetakan (*clocking*).

9. Pin 31, ***EA*** / $\overline{\text{VPP}}$ (***External Access***)

   Dapat diberikan logika rendah (*ground*) atau logika tinggi (+5V). Jika diberikan logika tinggi maka mikrokontroller akan mengakses program dari ROM *internal* (EEPROM/*Flash Memori*), dan jika diberikan logika

rendah maka mikrokontroler akan mengakses program dari memori *external* yang berlokasi 0000h sampai FFFFh.

10. Pin 32 sampai 32, **Port 0**

Port 0 terdiri dari 8 saluran *input* atau *output* dua arah, tanpa internal *pull-up*. Port 0 merupakan bus alamat rendah (A_0 – A_7), yang dimultipleks dengan saluran bus data (D0-D7), yang digunakan pada saat mengakses memori data *external* dan memori program *external*.

11. Pin 40,**VCC**

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

2.2.3. Organisasi Memori

Organisasi yang dimiliki oleh AT89S51 yang terdiri atas :

1. RAM Internal

Memori sebesar 128 *byte* yang biasanya digunakan untuk menyimpan variabel atau data yang bersifat sementara. RAM *internal* terdiri atas :

- ***Register Banks***

  AT89S51 mempunyai delapan buah register yang terdiri atas R0 hingga R7. Kedelapan register ini selalu terletak pada alamat 00H hingga 07H pada setiap kali sistem direset. Namun posisi R0 hingga R7 dapat dipindah ke bank 1 (08 H hingga 0FH), bank 2 (10H hingga 17H) dan bank 3 (18H hingga 1FH), dengan mengatur bit RS0 dan RS1.

- ***Bit Addressable*** RAM

  RAM pada alamat 20H hingga 2FH dapat diakses secara pengalamatan *bit* (*bit addressable*) sehingga hanya dengan sebuah

instruksi saja setiap *bit* dalam area ini dapat *diset, clear, AND, OR.*

- **RAM keperluan umum**

  RAM keperluan umum dimulai dari alamat 30H hingga 7FH dan dapat diakses dengan pengalamatan langsung maupun tak langsung. Pengalamatan langsung dilakukan ketika salah satu *operand* merupakan bilangan yang menunjukkan lokasi yang dialamati.

2. Special Function Register

Register fungsi khusus (*Special Function Register*) terletak pada 128 *byte* bagian atas memori data internal dan berisi *register-register* untuk pelayanan *latch port, timer, program status words, control peripheral* dan sebagainya. Alamat register fungsi khusus ditunjukkan pada Tabel 2.3.

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

Simbol	Nama Register	Nilai Pada Saat Reset	Alamat
ACC	Accumulator	00_H	$E0_H$
B	Register B	00_H	$F0_H$
PSW	Program Status Word	00_H	$D0_H$
SP	Stack Pointer	07_H	81_H
DPTR	Data Pointer 2 Byte		
DPL	Bit rendah	0000_H	82_H
DPH	Bit Tinggi	0000_H	83_H
P0	Port 0	$0FF_H$	80_H

P1	Port 1	$0FF_H$	90_H
P2	Port 2	$0FF_H$	$A0_H$
P3	Port 3	$0FF_H$	$B0_H$
IP	Interupt Periority Control	$XXX00000_B$	$D8_H$
IE	Interupt Enable Control	$0XX00000_B$	$A8_H$
TMOD	Timer/Counter Mode Control	00_H	89_H
TCON	Timer/Counter Control	00_H	88_H
TH0	Timer/Counter 0 High Control	00_H	$8C_H$
TL0	Timer/Counter 0 Low Control	00_H	$8A_H$
TH1	Timer/Counter 1 High Control	00_H	$8D_H$
TL1	Timer/Counter 1 Low Control	00_H	$8B_H$
SCON	Serial Control	00_H	98_H
SBUF	Serial Data Buffer	Independen	99_H
PCON	Power Control		87_H

Tabel 2.3. Special Function Register

(Sumber : Hafindo *Elektronic & Education*, Malang, 2001)

Beberapa macam register fungsi khusus yang sering digunakan adalah sebagai berikut ini :

- *Accumulator* (ACC) merupakan register untuk penambahan dan pengurangan. Perintah *mnemonic* untuk mengakses akumulator disederhanakan sebagai A.
- Register B merupakan register khusus yang berfungsi melayani operasi perkalian dan pembagian.
- *Program Status Word* (PSW) yang terletak pada alamat D0H terdiri dari beberapa *bit* status yang menggambarkan kejadian di akumulator sebelumnya. Yaitu *carry bit*, *auxiliary carry*, dua *bit* pemilih bank, bendera *overflow*, *parity*

bit, dan dua bendera yang dapat didefinisikan sendiri oleh pemakai. Keterangannya sebagai berikut :

- Flag Carry

  Flag Carry (terletak pada alamat D7H) mempunyai fungsi sebagai pendeteksi terjadinya kelebihan pada operasi penjumlahan atau terjadi pinjam (*borrow*) pada operasi pengurangan. Misalnya jika data pada accumulator adalah FFH dan dijumlahkan dengan bilangan satu atau lebih, akan terjadi kelebihan dan membuat carry menjadi Set, sedangkan jika data pada accumulator adalah 00H dan dikurangkan dengan bilanagan satu atau lebih, akan terjadi pemimjaman dan membuat carry juga menjadi set.

- Flag Auxilary Carry

  Flag Auxilary Carry akan selalu Set pada saat proses penjumlahan terjadi carry dari bit ketiga hingga bit keempat.

- Flag 0

  Flag 0 digunakan untuk tujuan umum bergantung pada kebutuhan pemakai.

- Bit Pemilih Register Bank

  Register Bank Select Bits (RS0 dan RS1) atau Bit Pemilih Register Bank digunakan untuk menentukan lokasi dari Register Bank (R0 hingga R7) pada memori. RS0 dan RS1 selalu bernilai nol setiap kali system direset sehingga lokasi dari R0 hingga R7 akan berada di alamat 00H hingga 07H.

- Flag Overflow

  Flag Overflow akan diset jika pada operasi aritmatik menghasilkan bilangan yang lebih besar dari pada 128 atau lebih kecil dari – 128.

- Bit Pariti

  Bit Pariti akan diset jika jumlah bit 1 dalam accumulator adalah ganjil dan akan clear jika jumlah bit 1 dalam accumulator genap. Jika data dalam accumulator adalah 10101110b atau AEH pariti akan diset. Data AEH mempunyai lima bit yang berkondisi 1 atau dapat disebut mempunyai bit 1 dalam jumlah yang ganjil. Bit pariti ini digunakan untuk proses yang berhubungan dengan serial port yaitu sebagai *Check sum*.

- *Stack Pointer* (SP) merupakan register 8 bit yang dapat diletakkan di alamat manapun pada RAM *internal*. Isi *register* ini ditambah sebelum data disimpan, selama instruksi PUSH dan CALL. Pada saat *reset*, *register* SP diinisialisasi pada alamat 07_H, sehingga *stack* akan dimulai pada lokasi 08_H.
- *Data Pointer* (DPTR) terdiri dari dua register, yaitu untuk *byte* tinggi (*Data Pointer High*, DPH) dan byte rendah (*Data Pointer Low*, DPL) yang berfungsi untuk pengalamatan alamat 16 bit.
- Port 0 sampai Port 3 merupakan register yang berfungsi untuk membaca dan mengeluarkan data pada port 0, 1, 2, 3. Masing-masing register ini dapat dialamati per-*byte* maupun per-*bit*.
- *Serial data buffer* (SBUF) merupakan dua *register* yang terpisah, *register buffer* pengirim dan sebuah *register buffer* penerima. Meletakkan data pada SBUF berarti meletakkan pada *buffer* pengirim yang akan mengirimkan data melalui transmisi serial. Membaca data SBUF berarti menerima data dari *buffer* penerima.

- *Control Register* terdiri dari register yang mempunyai fungsi kontrol. Untuk mengontrol sistem interupsi, terdapat dua register khusus, yaitu *register* IP (*Interupt Priority*) dan *register* IE (*Interupt Enable*). Untuk mengontrol pelayanan *timer/counter* terdapat *register* khusus, yaitu register TCON (*timer/counter control*) serta pelayanan port serial menggunakan register SCON (*Serial Port Control*).
- Register Timer

  AT89S51 mempunyai dua buah 16 bit Timer/Counter, yaitu Timer 0 dan Timer 1. Timer 0 terletak pada alamat 8AH untuk TL0 dan 8CH untuk TH0 dan Timer 1 terletak pada alamat 8BH untuk TL1 dan 8DH untuk TH1.
- Register Interupt

  89S51 mempunyai lima buah interupsi dengan sua level prioritas interupsi. Interupsi akan selalu nonaktif setiap kali system di – reset. Register – register yang berhubungan dengan interrupt adalah *Interrupt Enable Register* (IE) atau Register Pengaktif Interupsi pada alamat A8H untuk mengatur keaktifan tiap – tiap interrupt dan Interrupt Priority Register (IP) atau Register Prioritas Interupsi pada alamat B8H.
- Register Port Serial

  AT89S51 mempunyai sebuah *on chip serial port* (serial port dalam keping) yang dapat digunakan untuk berkomunikasi dengan peralatan lain yang menggunakan serial port juga seperti modem, shift register dan lain – lain.

  Buffer (Penyangga) untuk proses pengiriman maupun pengambilan data terletak pada register SBUF, yaitu pada alamat 99H. Sedangkan untuk

mengatur mode serial dapat dilakukan dengan mengubah isi dari SCON yang terletak pada alamat 98H.

3. Flash PEROM

AT89S51 memiliki 4Kb *Flash* PEROM (*Programmable and Erassable Read Only Memori*), yaitu ROM yang dapat ditulis ulang atau dihapus menggunakan sebuah perangkat programmer hingga 1000 kali. Program yang ada pada *Flash* PEROM akan dijalankan jika pada saat sistem di-*reset*, pin EA/VP berlogika satu sehingga mikrokontroler aktif berdasarkan program yang ada pada *flash* PEROMnya. Namun, jika EA/VP berlogika nol, mikrokontroller aktif berdasarkan program yang berada pada memori *external*.

2.2.4. Mode Pengalamatan.

Mode pengalamatan yang digunakan pada AT89S51 adalah sebagai berikut:

a) Mode pengalamatan segera (*immediate addressing mode).*

Cara ini menggunakan konstanta, misalnya: **MOV A, #20H.** Data konstanta merupakan data yang menyatu dengan instruksi, contoh instruksi tersebut diatas mempunyai arti bahwa data konstantanya yaitu 20H, (sebagai data konstanta harus diawali dengan '#') disalin ke akumulator A.

b) Mode pengalamatan langsung (*direct addressing mode*).

Cara ini dipakai untuk menunjuk data yang berada di suatu lokasi memori dengan cara menyebut lokasi (alamat) memori tempat data tersebut berada, misalnya: **MOV A, 30H.** Instruksi ini mempunyai arti bahwa data yang berada di dalam memori dengan lokasi 30h disalin ke

akumulator. Bedanya dengan pengalamatan segera yaitu jika pada pengalamatan segera menggunakan tanda '#' yang menandai 20H sebagai data konstan, sedangkan pada instruksi ini tidak menggunakan '#' sehingga 30H diartikan sebagai suatu lokasi memori.

c) Mode pengalamatan tidak langsung (*indirect addressing mode*).

Cara ini dipakai untuk mengakses data yang berada di dalam memori, tetapi lokasi memori tidak disebut secara langsung tapi di-'titip'-kan ke register lain, misalnya: **MOV A, @R0.** R0 adalah register serba guna yang dipakai untuk menyimpan lokasi memori, sehingga instruksi ini mempunyai arti memori yang alamat lokasinya tersimpan dalam R0 isinya disalin ke akumulator A. Tanda '@' dipakai untuk menandai lokasi memori yang tersimpan di dalam R0. *Register* serba guna R0 berfungsi sebagai register penyimpan alamat (*indirect address*), selain R0 register serba guna lainnya, R1 juga bisa dipakai sebagai register penampung alamat.

d) Mode pengalamatan register (*register addressing mode*).

Misalnya: **MOV A, R5** ,instruksi ini mempunyai arti bahwa data dalam register serba guna R5 disalin ke akumulator A. Instruksi ini menjadikan register serba guna R0 sampai R7 sebagai tempat penyimpanan data yang praktis dan kerjanya sangat cepat.

e) Mode pengalamatan kode tidak langsung (*code indirect addressing mode*).

MCS51 mempunyai cara penyebutan data dalam memori program yang dilakukan secara tak langsung, misalnya: **MOVC A, @A+DPTR.** Instruksi MOV diganti dengan MOVC, tambahan huruf C tersebut dimaksud untuk membedakan bahwa instruksi ini digunakan untuk memori program (MOV tanpa huruf C artinya digunakan untuk memori data). Tanda '@' digunakan untuk menandai A+DPTR yang berfungsi untuk menyatakan lokasi memori yang isinya disalin ke Akumulator A, dalam hal ini nilai yang tersimpan dalam DPTR (*Data Pointer Register* – 2 byte) ditambah dengan nilai yang tersimpan dalam akumulator A (1 byte) sama dengan lokasi memori program yang diakses.

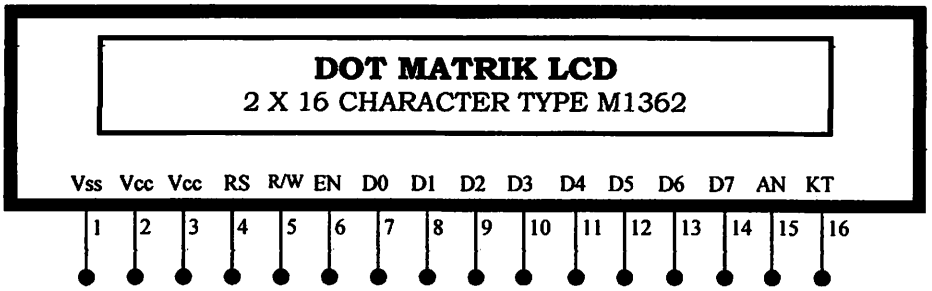
2.3 LCD *(Liquid Crystal Display)*

2.3.1 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 resolusinya jika dibandingkan dengan 7 segment.



Gambar 2.7. Deskripsi pin pada LCD Tipe M1632

(Sumber : LCD Manual Book)

LCD tipe M1632 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

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 dengan arus max 112 mA dan V = +4,1V
16	Light LCD	-	*Ground*

Tabel 2.4. Konfigurasi Pin-pin LCD

(Sumber : LCD Manual Book)

2.3.2 Interuksi Operasi Dasar

2.3.2.1 *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.

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*

Tabel 2.5. Tabel Register Seleksi

(Sumber : LCD Manual Book)

2.3.2.2 *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.

2.3.2.3 *Address Counter*

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

2.3.2.4 *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.

2.3.2.5 *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.

2.3.2.6 *Character Generator RAM (CG RAM)*

CG RAM digunakan untuk pembuatan karakter tersendiri melalui program.

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.* DB7 sebagai *busy flag*
E	1	I	MPU	Sinyal Start *(read/write)*
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

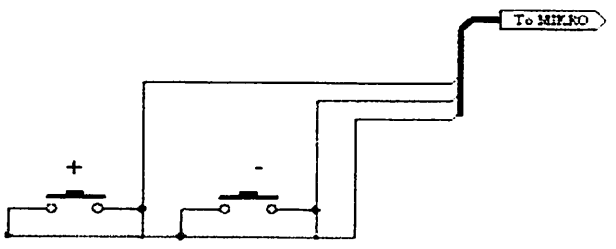
Tabel 2.6. Fungsi Terminal Pada LCD

(Sumber : LCD Manual Data Book)

2.4. Keypad

Untuk mempermudah penggunaan mikrokontroler sebagai alat pemroses, maka diperlukan sarana yang dapat menjadi penghubung antara pengguna dan alat kontrol, yaitu sebagai sarana input data yang nantinya akan diolah oleh mikrokontroler.

Peralatan input data yang dapat menunjang mikrokontroler adalah beberapa saklar tekan yang menyatakan angka dan karakter yang disusun berbentuk matrik 2 kolom. Keypad ini berfungsi untuk memberikan masukan setting yang diinginkan. Rangkaian keypad menggunakan keypad 3 kolom. Rangkaian keypad ini dihubungkan langsung ke mikrokontroler. Adapun skema rangkaian keypad ditunjukkan pada gambar berikut ini :



Gambar 2.8. Rangkaian Keypad

2.5.MF10

2.5.1. Pengenalan Dasar.

Switched Kapasitor Filter (MF 10) merupakan komponen elektronika yang berfungsi sebagai saklar elektrik dimana terdiri dari filter yang dapat bersifat aktif dan pasif. Biasanya pada masing – masing blok terdapat 3 – 4 resistor yang dibagi menjadi 3 outputan pin pada external clock, 1 dari outputannya dapat dikonfigurasikan pada allpass dan highpass filter. Pada ferkuensi yang di tengah digunakan untuk lowpass dan banpass filter pada ordo ke 2 secara langsung tergantung pada frekuensi clocknya yang terdapat pada resistor external. Pada highpass filter frekuensi yang digunakan bisa sampai pada ordo ke 4, dan apabila yang digunakan lebih dari ordo ke 4 maka harus menggunakan MF10 sebagai ic filter.

Keunggulan dari Switched Capasitor Filter (MF 10)

- Mudah dalam Penggunaan.
- Pulsa Akurasi Rasio Frekuensi pusat ± 0,6 %.
- Dalam keadaan diam kualitas pulsa mempertahankan stabilitas frekuensi.
- Sensitifitas rendah dalam penggunaan komponen variatif.
- Mudah diaplikasikan dengan Low Pass Filter, High Pass Filter, dan Band Pass Filter.
- Rentan Frekuensi operasional sampai 30 Khz.
- Fo x Q range maksimum mencapai 200 KHz.

- 20 Pin dengan lebar jalur dalam 0.3 mm.
- Bentuk kemasan yang minimum tapi terdiri 20 Pin.

2.5.2. Spesifikasi Switched Capasitor Filter (MF 10).

- Tegangan yang digunakan ($V^+ - V^-$) : 14 Volt.
- Tegangan dalam tiap Pin : $V^+ + 0{,}3V$

  $V^- - 0.3$ V
- Arus masukan dalam tiap Pin : 5 mA.
- Arus Masukan : 20 mA.
- Dissipation Power : 500mW.
- Suhu Normal : 150 ° C.
- ESD Susceptabiliti : 2000 V.
- Informasi Soldering

  N Kemasan 10 Detik : 260°C.

  Vapor Phase 60 Detik : 215°C.

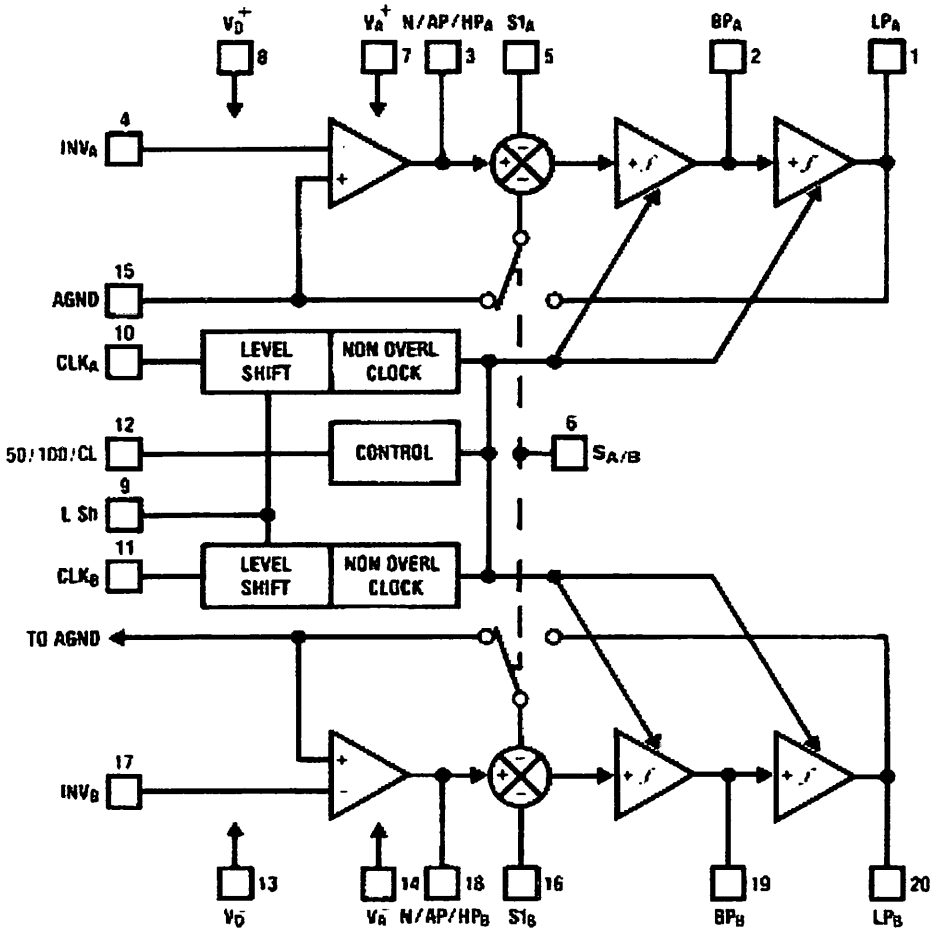
  Infrared 15 Detik : 220°C.
- Rating Operasi.

  Suhu Rata – Rata : $T_{Min} \leq T_A \leq T_{Max}$

  MF10ACN, MF10CCN : $0°C \leq T_A \leq 70°c$.

  MF 10CCWM : $0°C \leq T_A \leq 70°c$.

Untuk blok diagram sistem MF10 dapat dilihat pada gambar 2.9 di bawah ini :



Gambar 2.9. Blok Diagram IC MF 10

2.5.3. Karakteristik IC MF 10 (Switched Kapasitor Filter).

Electrical Characteristics

V^+ = +5.00V and V^- = -5.00V unless otherwise specified. Boldface limits apply for T_{MIN} to T_{MAX}; all other limits $T_A = T_J$ = 25°C.

Symbol	Parameter		Conditions		MF10ACN, MF10CCN, MF10CCWM			Units
					Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
$V^+ - V^-$	Supply Voltage	Min					9	V
		Max					14	V
I_S	Maximum Supply Current		Clock Applied to Pins 10 & 11 No Input Signa		6	12	12	mA
f_O	Center Frequency Range	Min	$f_O \times Q < 200$ kHz		0.1		0.2	Hz
		Max			30		20	kHz
f_{CLK}	Clock Frequency Range	Min			5.0		10	Hz
		Max			1.5		1.0	MHz
f_{CLK}/f_O	50:1 Clock to Center Frequency Ratio Deviation	MF10A	Q = 10 Mode 1	$V_{pin\,2}$ = 5V f_{CLK} = 250 KHz	±0.2	±0.6	±0.6	%
		MF10C			±0.2	±1.5	±1.5	%
f_{CLK}/f_O	100:1 Clock to Center Frequency Ratio Deviation	MF10A	Q = 10 Mode 1	$V_{pin\,2}$ = 0V f_{CLK} = 500 kHz	±0.2	±0.6	±0.6	%
		MF10C			±0.2	±1.5	±1.5	%
	Clock Feedthrough		Q = 10 Mode 1		10			mV
	Q Error (MAX) (Note 4)		Q = 10 Mode 1	$V_{pin\,2}$ = 5V f_{CLK} = 250 kHz	±2	±6	±6	%
				$V_{pin\,2}$ = 0V f_{CLK} = 500 kHz	±2	±6	±6	%
H_{DLP}	DC Lowpass Gain		Mode 1 R1 = R2 = 10k		0	±0.2	±0.2	dB
V_{OS1}	DC Offset Voltage (Note 5)				±5.0	±20	±20	mV
V_{OS2}	DC Offset Voltage	Min	$V_{ph\,12}$ = +5V	S_{A-B} = V^-	-150	-185	-185	mV

Tabel 2.7. Karakteristik IC MF10

Logic Input Characteristics

Boldface limits apply for T_{MIN} to T_{MAX}; all other limits $T_A = T_J = 25°C$

Parameter		Conditions	MF10ACN, MF10CCN, MF10CCWM			Units
			Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
CMOS Clock Input Voltage	Min Logical "1"	$V^+ = +5V$, $V^- = -5V$, $V_{LSH} = 0V$		+3.0	+3.0	V
	Max Logical "0"			-3.0	-3.0	V
	Min Logical "1"	$V^+ = +10V$, $V^- = 0V$, $V_{LSH} = +5V$		+8.0	+8.0	V
	Max Logical "0"			+2.0	+2.0	V
TTL Clock Input Voltage	Min Logical "1"	$V^+ = +5V$, $V^- = -5V$, $V_{LSH} = 0V$		+2.0	+2.0	V
	Max Logical "0"			+0.8	+0.8	V

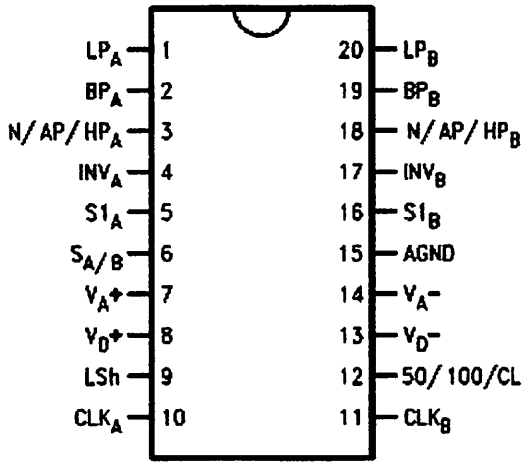
Logic Input Characteristics (Continued)

Boldface limits apply for T_{MIN} to T_{MAX}; all other limits $T_A = T_J = 25°C$

Parameter		Conditions	MF10ACN, MF10CCN, MF10CCWM			Units
			Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
	Min Logical "1"	$V^+ = +10V$, $V^- = 0V$, $V_{LSH} = 0V$		+2.0	+2.0	V
	Max Logical "0"			+0.8	+0.8	V

Tabel 2.8. Tabel Karakteristik Logic Input.

2.5.4. Konfigurasi Pin – Pin IC MF 10 (Switched Kapasitor Filter).



Gambar 2.10. Konfigurasi Pin – Pin IC MF 10 (Switched Kapasitor Filter).

$LP_A(1,20), BP_B(2,19), N/AP/HP(3,18)$

Fungsi ganda dari keluaran Low Pass, Band Pass, dimana arus keluar berkisar 1,5 mA sampai 3 mA dan lebar swing 1 Volt.

$INV_A(4,17)$

Masukan inverting dari Op-Amp akan menghasilkan impedansi tinggi, tetapi apabila mendapat masukan dari non inverting maka AGND akan menjadikan INV_A dan INV_B menjadi impedansi rendah.

$S1_A(5,16)$

Sebagai pin dari sinyal masukan yang dapat digunakan untuk segala jenis konfigurasi filter . S1 dapat bekerja apabila mendapat impedansi bias mencapai 1 Ω. Bila S1 tidak bekerja maka sinyal masukan dilangsungkan ke AGND.

$S_{A/B}$ (6)

Saklar pin akan aktif bila dihubungkan dengan satu masukan AGND ($S_{A/B}$ kondisi V^-) atau untuk keluaran Low Pass Filter ($S_{A/B}$ kondisi V^+)m Memiliki fleksibilitas dalam konfigurasi filter dan operasi variasi mode.

V_A^- (7), V_D^+ (8)

Sumber analog positif dan sumber digital positif

CLK_A(10), CLKB(11)

Mengaktifkan saklar Kapasitor Filter maksimum pada saat op-amp bekerja maksimal pada system.

50/100/CL(12)

Jika pada saat akif detak pulsa frekuensi utama 50 : 1, jika pada saat pasif detak pulsa frekuensi utama 100 : 1.

2.5.5. Persamaan Nilai.

f_{CLK} adalah detak sinyal frekuensi luar yang diaplikasi pada pin 10 atau 11.

f_o adalah frekuensi pusat pada saat mendapat frekuensi maksimum pada saat Band Pass Filter mendapatkan frekuensi maksimum.

f_{notch} adalah kondisi pada saat frekuensi minimum dan pada idealnya f = 0.

F_z akan aktif apabila f_o dan Q_z aktif.

Q adalah faktor kualitas dari semua filter.

Q_z adalah factor kualiatas kedua dari semua pass filter dan nilainya relatife.

Maka persamaan nilai yang diperoleh :

$$H_{AP}(s) = \frac{H_{OAP}\left(s^2 - \frac{s\omega_O}{Q_Z} + \omega_O^2\right)}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

Q_z adalah kualitas faktor relative untuk semua karakteristik.

Dimana : Q_z = Respons untuk semua Pass.

H_{OBP} = Gain (dalam V/V) pada saat keluaran Band Pass, dimana f = f_o.

H_{OLP} = Gain (dalam V/V) pada saat keluaran Low Pass, dimana f = 0 Hz.

H_{OHP} = Gain (dalam V/V) pada saat keluaran High Pass, dimana f = $f_{CLK}/2$.

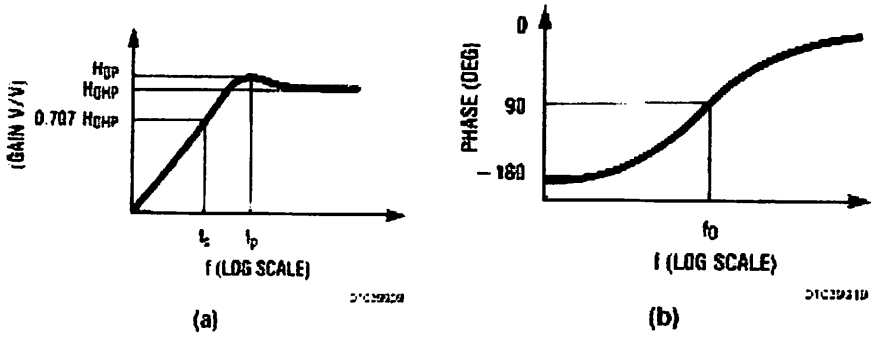
H_{ON} = Gain (dalam V/V) pada saat tidak ada keluaran, dimana f = 0.

$H_{ON\,1}$ = Gain (dalam V/V) pada saat f keluaran 0 Hz.

$H_{ON\,2}$ = Gain (dalam V/V) pada saat f keluaran sama dengan = $f_{CLK}/2$.

f_{notch} adalah frekuensi minimum (Idealnya 0).

f_Z adalah frekuensi pusat yang stabil bila f_o dan Q_Z kondisi *High.*



Gambar 2.11. Grafik 2^{nD} Order High Pass Response

BAB III

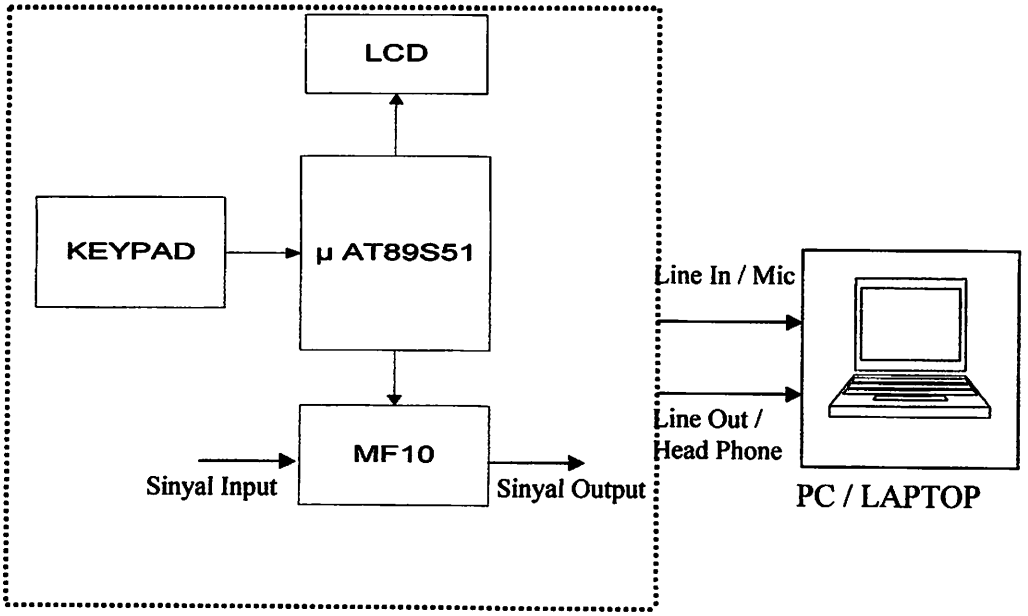
PERENCANAAN DAN PEMBUATAN ALAT

3.1. Pendahuluan

Bab ini membahas tentang perencanaan dan pembuatan high pass filter yang berbasiskan mikrokontroller AT89S51. Perencanaan ini meliputi spesikasi alat, perencanaan blok-blok rangkaian penyusun sistem dan perencanaan perangkat lunak yang mengintegrasikan kerja seluruh sistem. Perencanaan ini dibuat bertahap sesuai dengan diagram blok di bawah ini.

3.2. Diagram Blok Rangkaian

Dalam pembuatan dan perancangan *High Pass Filter Digital* dengan menggunakan *mikrokontroler AT89S51* ini, dalam rancangan suatu rangkaiannya ada beberapa hal yang perlu diperhatikan yaitu tentang perencanaan sistem kerja rangkaian dari masing – masing bagian yang mempunyai fungsi yang diinginkan :



Gambar 3.1. Diagram Blok Rangkaian

Keterangan Blok Diagram

- **MikrokontrolerAT89S51**

  Rangkaian Mikrokontroler AT89S51 berfungsi sebagai pengolah data.

- **LCD**

  Sebagai Unit keluaran dari Mikrocontroller (Media penampil dalam bentuk matrik titik).

- **KEYPAD**

  Keypad tersebut difungsikan sebagai masukan dari external interupt.

- **MF 10**

  Switched Kapasitor Filter (MF 10) merupakan komponen elektronika yang berfungsi sebagai saklar elektrik dimana terdiri dari filter yang dapat bersifat aktif dan pasif.

3.3. Prinsip Kerja

Mikrokontroller berfungsi untuk membangkitkan pulsa gelombang kotak (*Square Wave*), dimana frekuensinya diatur oleh *software* yang sesuai dengan menu awal untuk memilih *frekuensi cut off*. Frekuensi cut off ini dikalikan dengan 50, dimana angka ini berasal dari rangkaian skema yaitu pada *IC MF10* dikaki no.12 atau kaki dengan nama (50/100/CL) yang disambungkan ke *VCC*, yang artinya bahwa bila kaki ini disambungkan ke *VCC* maka frekuensi clock yang diberikan harus sebesar 50x dari *frekuensi center sinyal* yang dikehendaki (frekuensi cut off bila difungsikan sebagai *LPF/HPF*), dengan demikian mikrokontroller akan mengolahnya agar dihasilkan *oscilator* gelombang kotak sebesar 50x yang dikehendaki.

Oscilator gelombang kotak ini diberikan ke *IC MF10* pada kaki 10&11 atau kaki dengan nama *CLKA&CLKB*, sehingga *IC MF10* akan bekerja sebagai filter dengan frekuensi kerja cut off sebesar frekuensi clock dari *mikrokontroller*. Jadi pada intinya *mikrokontroller* hanya bekerja untuk menghasilkan frekuensi clock yang dibutuhkan oleh *IC MF10*, dimana frekuensi clock tersebut bisa variabel yang bisa diatur oleh *software*.

3.4. Perencanaan Hardware

Dalam perencanaan ini rancangan *hardware* yang dibuat tujuannya adalah untuk mendukung dan memberikan kemudahan proses kerja pada rancangan *software* agar nantinya dapat sesuai dengan kondisi yang diinginkan. Sedangkan perancangan *hardware* ini dibagi menjadi empat bagian :

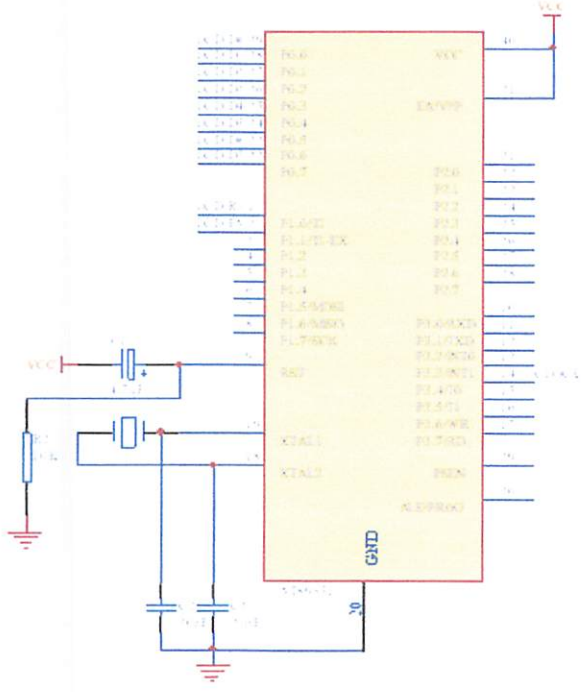
- .Mikrokontroller AT89S51
- LCD
- Keypad
- IC Filter MF10

3.3.1. Mikrokontroller AT89S51

AT89S51 adalah mikrokontroller keluaran Atmel dengan 4K byte *Flash PEROM (Programmable and Erasable Read Only Memory)*, AT89S51 merupakan memori dengan teknologi *nonvolatile* memori, artinya isi memori tersebut dapat diisi ulang ataupun dihapus berulang kali.

Memori ini biasa digunakan untuk menyimpan instruksi (perintah) berstandar *MCS – 51 code* sehingga memungkinkan mikrokontroler ini untuk bekerja dalam mode *Single Chip Operation* (Mode Operasi Keping Tunggal) yang

tidak memerlukan *Eksternal Memori* (Memori luar) untuk menyimpan *source code* tersebut. Ini berarti hanya dengan sebuah mikrokontroler AT89S51 sudah dapat mengontrol keseluruhan sistem kerja tanpa perlu dibantu dengan peralatan eksternal lainnya. Port – port yang digunakan dalam skripsi ini adalah :



Gambar 3.2. Rangkaian Mikrokontroller AT89S51

Pada perancangan alat ini mikrokontroler menggunakan *clock* sebesar 12Mhz. Mikrokontroler juga memiliki *internal clock generator* yang berfungsi sebagai sumber *clock*, tetapi masih memerlukan rangkaian tambahan untuk membangkitkan *clock* yang dibutuhkan oleh mikrokontroler.

Rangkaian *external* ini terdiri dari dua buah kapasitor dan sebuah kristal dengan ketentuan yang disesuaikan dalam *data sheet*, yaitu :

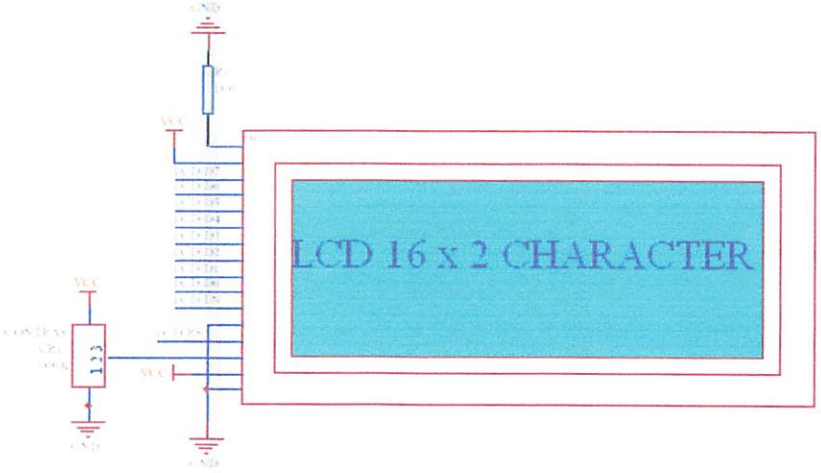
C2 dan C3 = 30pF

X_*crystal* = 18Mhz.

3.3.2. LCD

Dalam aplikasi ini menggunakan sebuah layar LCD (*Liquid Crystal Display*) yaitu jenis M1632 yang merupakan LCD dua baris dengan setiap barisnya terdiri atas 16 karakter. Masukan yang diperlukan untuk mengendalikan modul ini berupa bus data yang masih ter-*multiplex* dengan bus alamat. Sementara pengendalian dot matrik LCD dilakukan secara internal oleh kontroler yang sudah terpasang pada modul LCD.

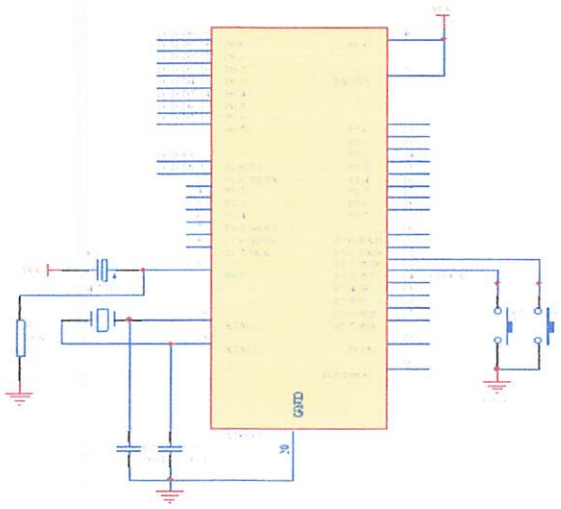
Rangkaian *display* ditunjukkan dalam Gambar 3-3. Saluran data D_0–D_7 dihubungkan pada pin *shift register*. Sedangkan RS dan EN dihubungkan pada port 1.0 dan port 1.1 mikrokontroler AT89S51.



Gambar 3.3. Rangkaian LCD 16 X 2

3.3.3. KEYPAD MATRIK

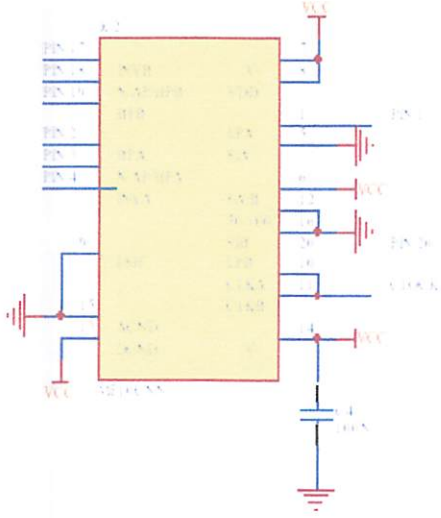
Rangkaian *keypad* dalam perancangan menggunakan jenis *keypad matrik*. Dari kombinasi tersebut menghasilkan 2 tombol yaitu tombol untuk mengatur frekuensi up dan down sebagai masukkan pada external interupt seperti ditunjukkan pada gambar 3.3 di bawah ini :



Gambar 3.4. Rangkaian Keypad Matrik

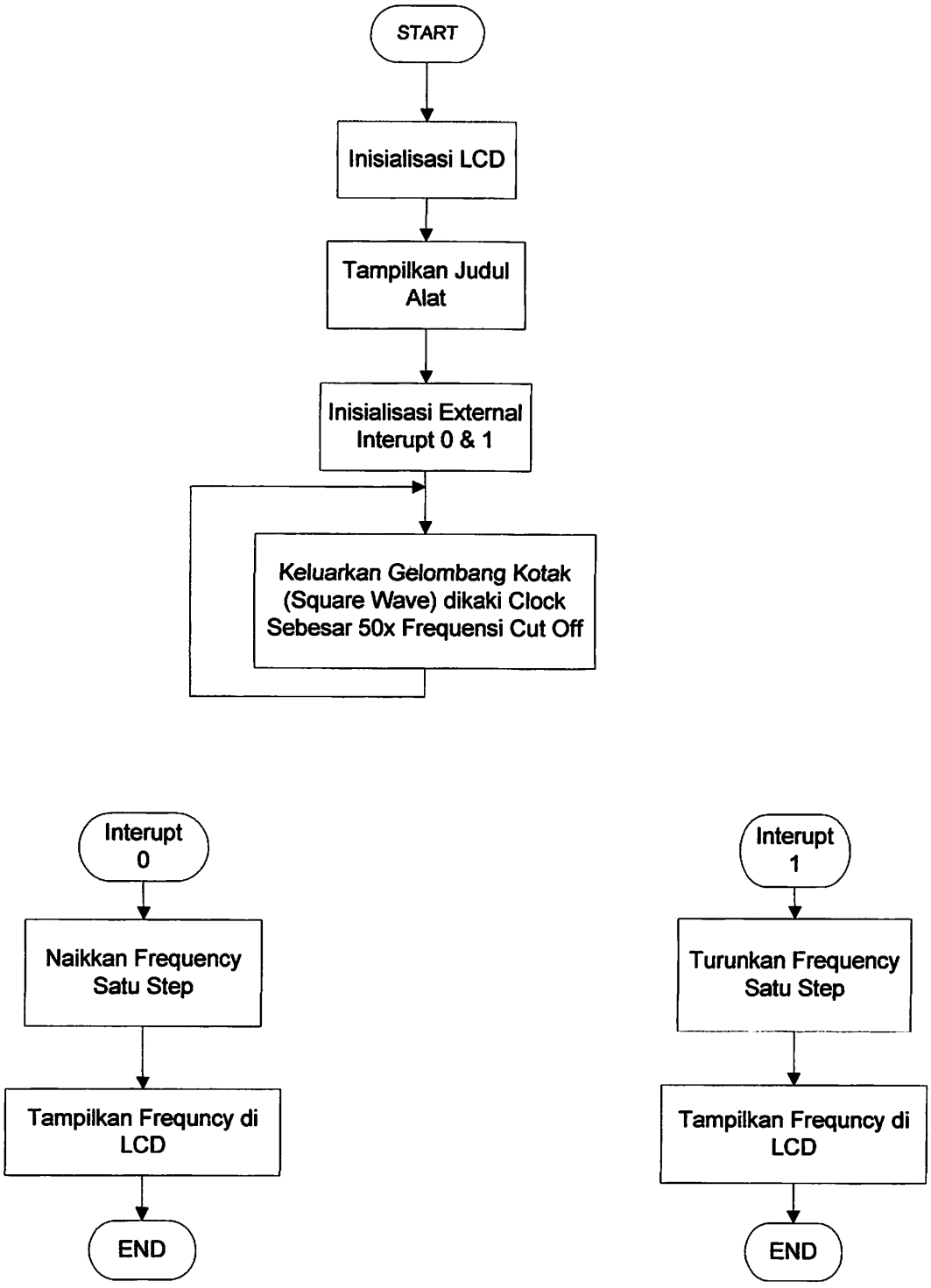
3.3.4. IC Filter MF10

Switched Kapasitor Filter (MF 10) merupakan komponen elektronika yang berfungsi sebagai saklar elektrik dimana terdiri dari filter yang dapat bersifat aktif dan pasif seperti yang ditunjukkan pada gambar 3.5 di bawah ini :



Gambar 3.5. IC Filter MF10

3.5. Flowchart



Gambar 3.6. Flowchart

BAB IV

PENGUJIAN ALAT

4.1. Pendahuluan

Pada bab ini membahas tentang pengujian alat yang telah dibuat. Secara umum pengujian ini bertujuan untuk mengetahui apakah alat dapat bekerja sesuai dengan spesifikasi perencanaan yang telah ditetapkan. Pengujian dilakukan dengan cara menguji sistem yang dibuat secara per blok. Dengan demikian dapat diketahui kepresisian kerja dari alat yang direncanakan dan dibuat. Secara umum tujuan dari pengujian alat tersebut adalah sebagai berikut :

1. Mengetahui proses kerja dari masing-masing rangkaian.
2. Memudahkan pendataan spesifikasi alat.
3. Mengetahui hasil dari suatu perencanaan yang telah dibuat.
4. Memudahkan perawatan dan perbaikan apabila sewaktu waktu terjadi kerusakan.

4.2. Pengujian LCD

4.2.1. Tujuan Pengujian LCD

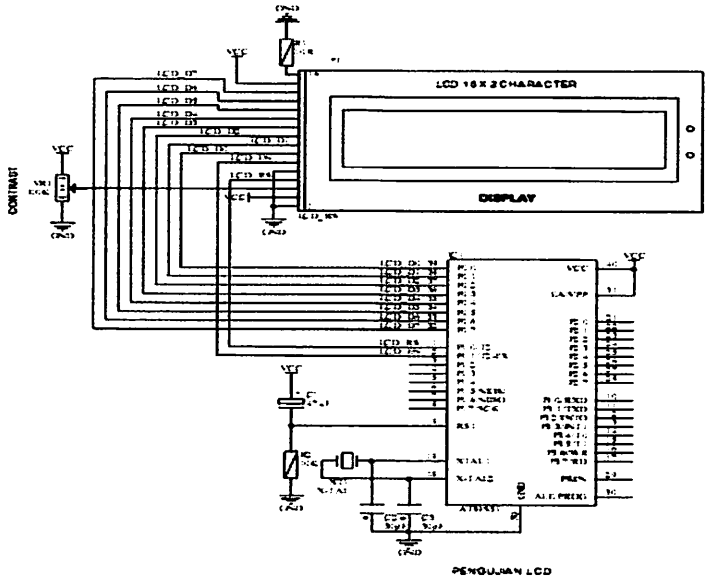
Untuk mengetahui kemampuan rangkaian tampilan yang sudah dibuat apakah dapat mendukung sistem yang direncanakan untuk menampilkan data pada *LCD.*

4.2.2. Peralatan yang digunakan

1. *Personal Computer.*
2. Minimum sistem mikrokontroler AT89S51.
3. Catu daya 5V DC.

4.2.3. Prosedur Pengujian

1. Merangkai peralatan seperti dalam Gambar 4.1.



Gambar 4.1. Rangkaian Pengujian LCD

2. Membuat program *assembler* yang digunakan dalam pengujian LCD ini merupakan program yang menampilkan Nama dan NIM pada LCD. Program yang dibuat adalah sebagai berikut :

4.2.4. Listing Program Pengujian LCD

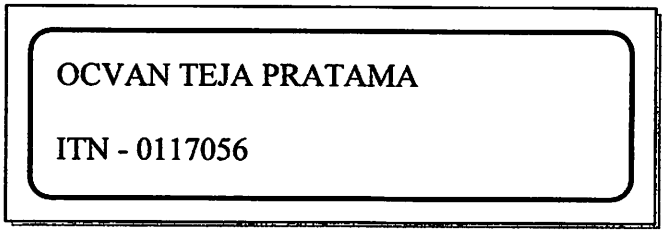
```
//=============================================================
//   PROGRAM PENGUJIAN LCD 16X2 CHARACTER
// UNTUK MENAMPILKAN NAMA DAN NIM PADA SATU LAYAR
// OLEH : OCVAN TEJA PRATAMA
// NIM  : 0117056
```

```
// TEKNIK ELEKTRO
// ITN - MALANG
//=========================================================

#include <AT89X51.H>
#define      LCDData        P0
sbit LCD_RS  = P1_0;
sbit LCD_EN        = P1_1;


void delay(long lama) { while(lama--); }
void LCD_data(char c,char dat)
{
    LCD_RS = c;
    LCDData=dat;
    LCD_EN = 1;
    LCD_EN = 0;
    delay(1000);
}
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 main()
{
```

```
// ========== initialisasi LCD ==========
delay(50000);
LCD_data(0,0x3F); LCD_data(0,0x0D); LCD_data(0,0x06);
LCD_data(0,0x01); LCD_data(0,0x0C);
// ========== Tampilkan judul alat ==========
while(1)
{
        Tulis_LCD(0x80,"OCVAN T.PRATAMA");
        Tulis_LCD(0xC0," ITN - 0117056 ");
        delay(100000);
}
}
```

4.2.5. Hasil Pengujian

Dari hasil pengujian didapatkan bahwa rangkaian LCD dapat menampilkan karakter-karakter, sesuai dengan data yang dikirimkan oleh *EPROM Emulator*. Tampilan penampil kristal cair terdiri atas 2 baris yang masing-masing mempunyai 16 karakter.



Gambar 4.2. Hasil Pengujian LCD

4.3. Pengujian *Switch Capacitor Filter MF10CNN*

4.3.1. Peralatan yang digunakan :

-. Oscilloscope.

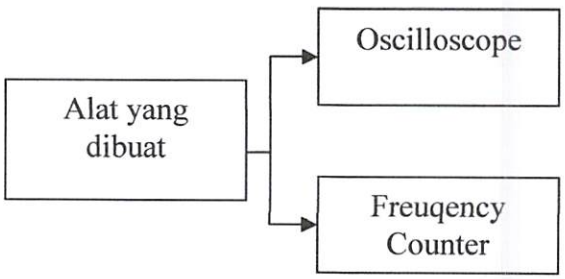
-. Function Generator.

-. PC / Laptop dengan Audio Card.

-. Frequency Counter.

-. Modul yang dibuat.

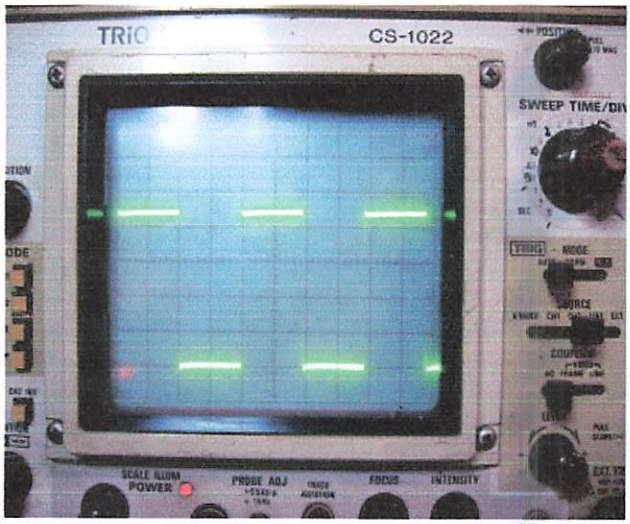
4.3.2. Blok Diagram Pengukuran Frequncy Clock yang dihasilkan oleh Mikrokontroller



Gambar 4.3. Blok Diagram Pengukuran Frequency

Blok Diagram Pengukuran Frequncy Clock yang dihasilkan oleh Mikrokontroller. Utuk oscilloscope dan frequency counter tersambung pada kaki 10 atau 11 pada IC *MF10CNN* terhadap ground.

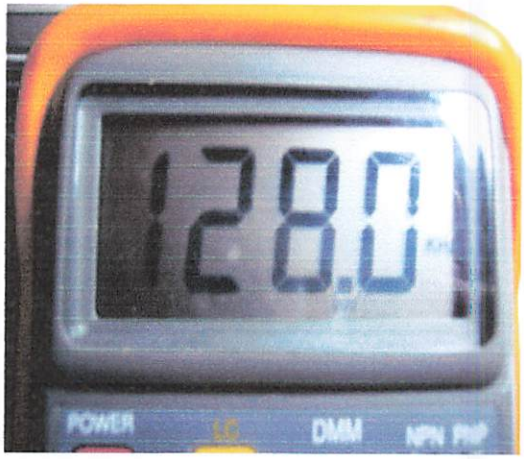
Gambar 4.4. Modul yang dibuat



Gambar 4.5. Hasil Pembacaan pada Oscilloscope

T/div = 2μS/div

V/div = 1 Volt/div



Gambar 4.6. Hasil Pembacaan Frequency Counter

Dari percobaan di atas maka didapat frequency clock = 3,9 kotak atau T = 3,9 x 2μS = 7,8μS atau F = 1 / T = 128,2 kHz.

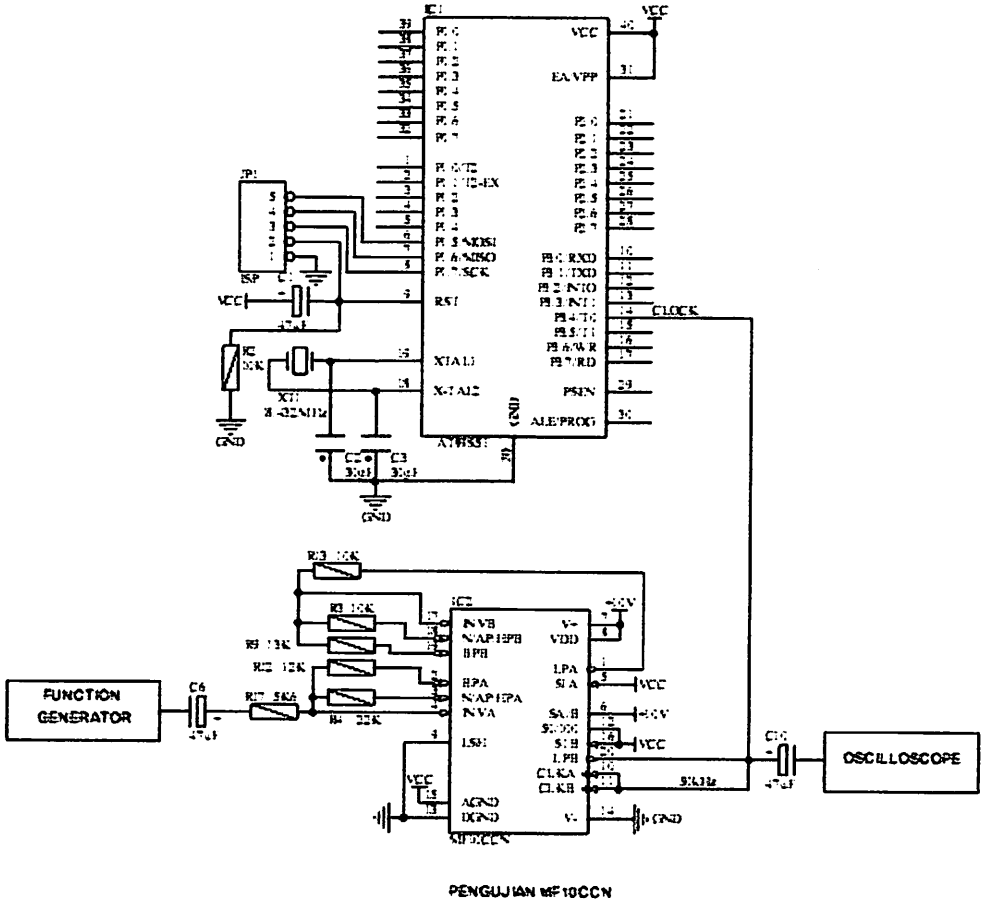
4.3.3. Listing Program yang digunakan pada Pengujian *Switch Capacitor Filter MF10CNN.*

Listing program yang dibuat adalah sebagai berikut :

```
;=============================================================
; PENGUJIAN SWITCH CAPACITOR FILTER MF10CCN
; FREKUENSI SIGNAL INPUT 1kHz
; FREKUENSI CLOCK MF10 = 50kHz
; OLEH : OCVAN TEJA PRATAMA
; NIM  : 0117056
; TEKNIK ELEKTRO
; ITN - MALANG
;=============================================================
$include(Reg51.inc)
clock       bit     P3.4
temp                equ     50h

    org     0h
    jmp     mulai
    org     03h
    mov     A,temp
    cjne    A,#0FFh,terus
    jmp     batal
terus:
    inc     temp
    reti
    org     13h
    mov     A,temp
```

```
        jz      batal
        dec     temp
batal:
        reti
mulai:
        setb EA ; Enable Interupt
        setb EX0 ; aktifkan Int 0
        setb EX1 ; aktifkan Int 1
awal:
;       mov     R2,#1
ulang1:
        mov     R0,temp
ulang2:
        djnz    R0,ulang2
;       djnz    R2,ulang1
        cpl     clock
        jmp     awal
;
        End
```

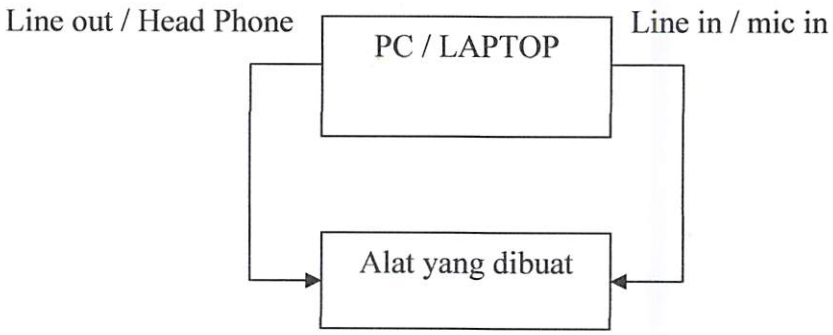
4.3.4. RANGKAIAN *SWITCH CAPACITOR FILTER MF10CCN*



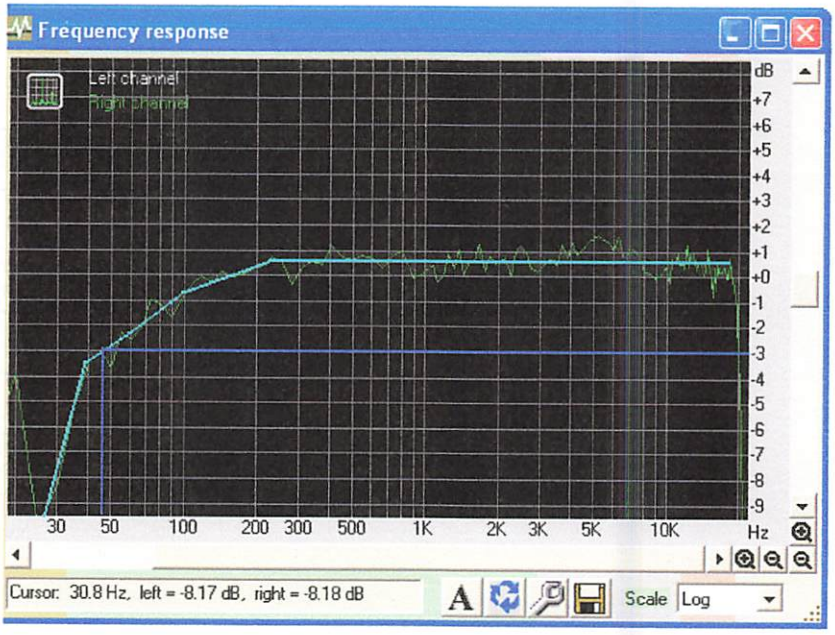
Gambar 4.7. Rangkaian Switch Capacitor Filter MF10CCN

4.4. Pengujian *Respon Frequncy High Pass Filter*

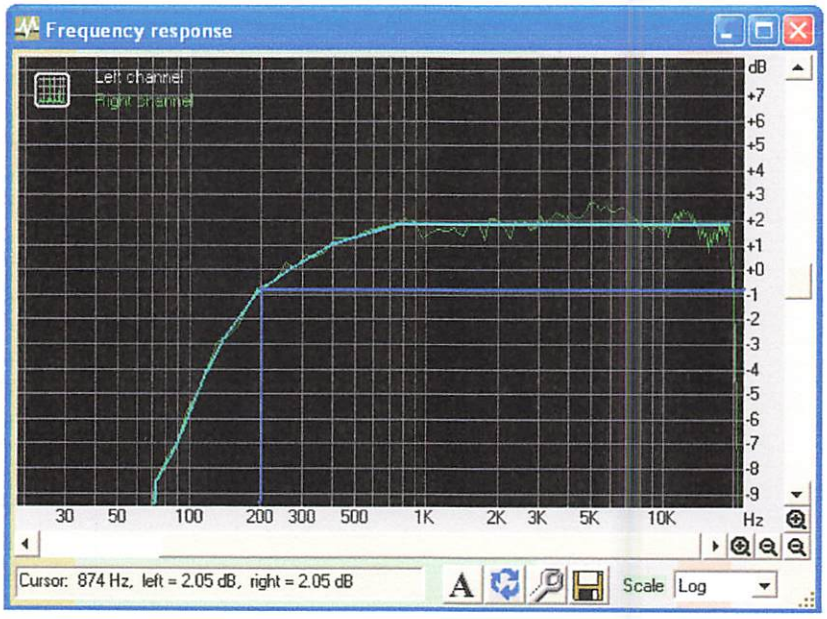
Pengujian ini membutuhkan program bantu yang bernama *RMAA (Righ Mark Audio Anayizer Versi 6.0.6)* yang diinstal pada sebuah Laptop atau PC dengan *I/O sound card* yang tidak mempunyai *line in* atau *mic in* dan *line out* atau *head phone out.*



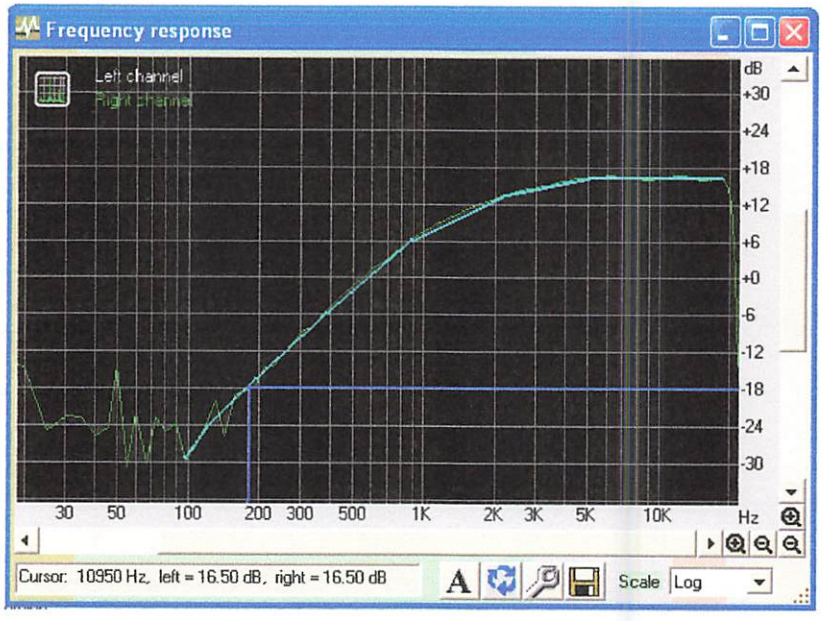
Gambar 4.8. Block Diagram Pengujian



Gambar 4.9. Pada Frekunsi 2380Hz, Menghasilkan Frekuensi Cut Off Sebesar 47Hz

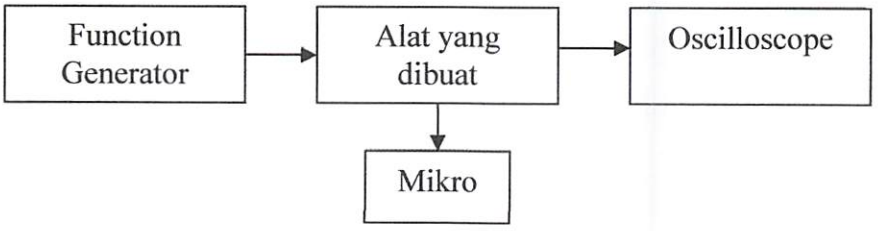


Gambar 4.10. Pada Frekunsi 1000Hz, Menghasilkan Frekuensi Cut Off Sebesar 200Hz



Gambar 4.11. Pada Frekunsi 2,5kHz, Menghasilkan Frekuensi Cut Off Sebesar 128Hz

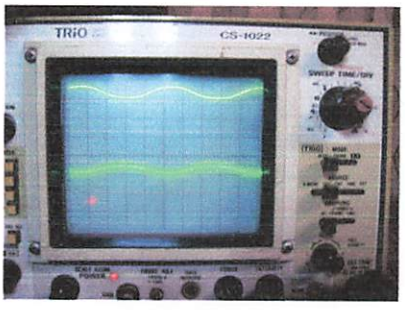
4.5. Pengujian Secara Manual Menggunakan Oscilloscope dan Function Generator.



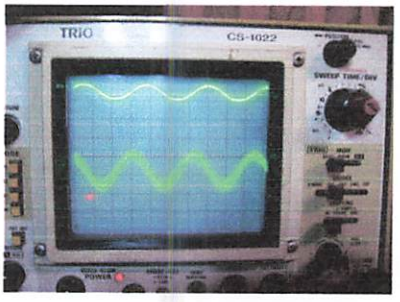
Gambar 4.12. Blok Diagram Pengujian

Gambar 4.13. Gambar Pengujian Alat

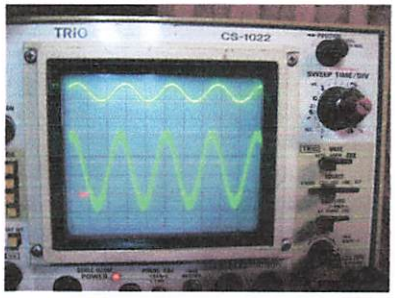
- Untuk V/div = 1 V/div
  T/div = 5µS/div
  Fclock = 51kHz



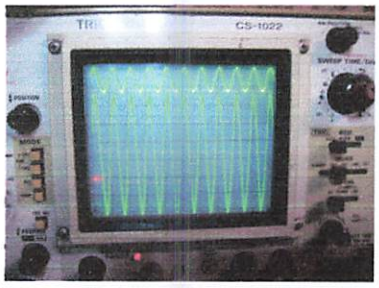
Gambar 4.14. Pada Frequency = 550Hz



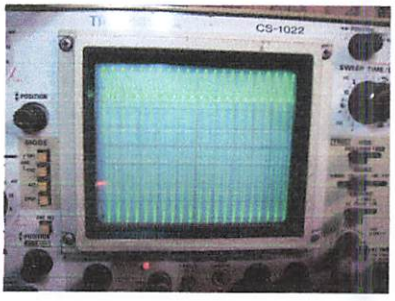
Gambar 4.15. Pada Frequency = 800Hz



Gambar 4.16. Pada Frequency = 1kHz

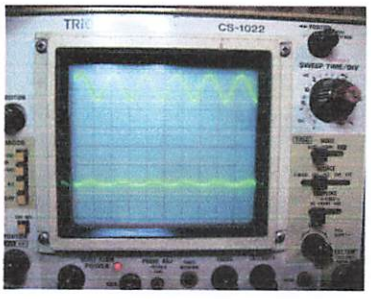


Gambar 4.17. Pada Frequency = 5kHz

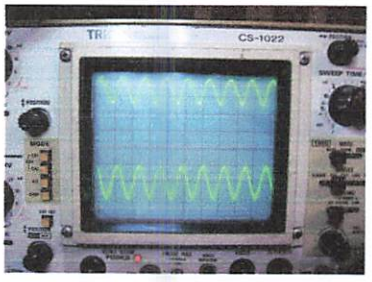


Gambar 4.18. Pada Frequency = 10kHz

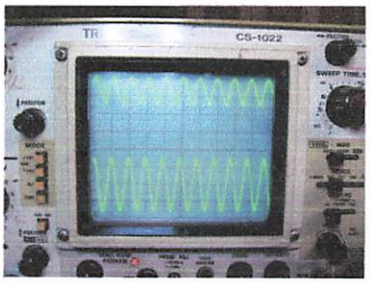
- Untuk V/div = 1V/div
  T/div = 2μS/div
  Fclock = 51kHz



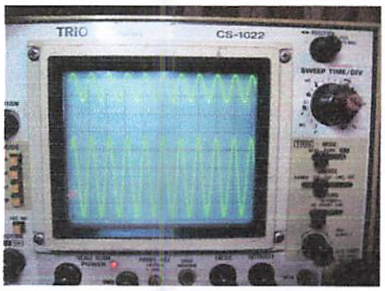
Gambar 4.19. Pada Frequency = 1kHz



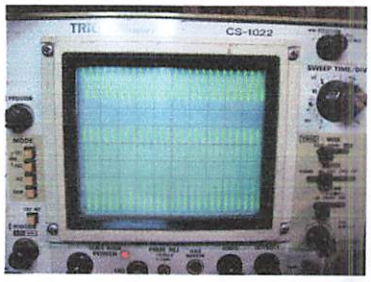
Gambar 4.20. Pada Frequency = 1,5kHz



Gambar 4.21. Pada Frequency = 2kHz



Gambar 4.22. Pada Frequency = 5kHz



Gambar 4.23. Pada Frequency = 10kHz

Gambar dari semua percobaan di atas menunjukkan bahwa fungsi HPF sudah bekerja dengan baik yaitu dengan meloloskan frekuensi atas di atas ferkuensi cut off.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil pengujian dan analisa dari alat yang telah dibuat maka dapat disimpulkan :

1. *MF10* adalah *IC* yang bisa dipakai sebagai filter serbaguna dengan fo / *frekuensi cut off* yang bisa diubah-ubah dengan cara memberikan f clock sesuai yang dikehendaki.
2. Memungkinkan untuk dikontrol melalui *mikrokontroller* sebagai *variabel filter.*
3. *Bandwidth* bisa diatur melalui *external* komponen yang dipasangkan.
4. Mampu dibuat cascade untuk memperoleh faktor kualitas yang tinggi atau dibuat *2 channel.*

5.2. Saran

1. Bisa diaplikasikan untuk filter multi fungsi yaitu : *HPF, LPF, BPF, Notch Filter* yang dikontrol oleh *mikrokontroller.*
2. Diperlukan selektifitas untuk meminimalis noise yang ditimbulkan karena umpan balik.
3. Untuk perancangan ke depan diharapkan alat ini dapat disempurnakan untuk memperoleh hasil yang lebih baik.
4. Diperlukan ketelitian untuk mendapatkan hasil yang memuaskan.

DAFTAR PUSTAKA

1. PutraE, Agfrianto, " Belajar mikrokontroler AT89S51/52/55 Teori dan Aplikasi " Gava Media, Yogyakarta 2002.
2. Nalwan, Paulus A, 2003, Panduan Praktis Teknik Antarmuka dan Pemograman mikrokontroler AT89S51/52. Jakarta : PT. Gramedia.
3. Data Sheet uln 2003, www.Allegromicro.com.
4. www.electronics-tutorial.com
5. Data Sheet MF10CCN.

LAMPIRAN

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

FORMULIR PERBAIKAN SKRIPSI

Nama : **OCVAN TEJA PRATAMA**
NIM : **01.17.056**
Masa Bimbingan : **04 Februari 2009 s/d 04 Juli 2009**
Judul : **"PERANCANGAN DAN PEMBUATAN HIGH PASS FILTER MENGGUNAKAN MF10 YANG DIKONTROL OLEH MIKROKONTROLLER AT89S51 "**

No.	MATERI PERBAIKAN	PARAF
1.	Blok Diagram Diperbaiki Disesuaikan Dengan Prinsip Kerja Alat	
2.	Prinsip Kerjanya Dijelaskan	
3.	Tata Tulis Laporannya Diperbaiki	
4.	Setting Frequency Cut Off	
5.	Pengujian Mestinya Mengarah Pada Respon Fekuensi, dan Seberapa Tepat Proses Filtering Sesuai Dengan Frekuensi Cut Off yang Dirancang	
6.	Belum Bisa Menjawab Cara Setting Frekuensi Cut Off Filter Melalui Mikrokontroller	

Disetujui:

Penguji I

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

Penguji II

(Irmalia Suryani Faradisa, ST, MT)
NIP.P.1030000356

Mengetahui
Dosen Pembimbing

(I Komang Somawirata, ST, MT)
NIP.1030100361

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

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

NAMA : Ovan Teja P
NIM : 0117056
Perbaikan meliputi :

1) Pengujian unjuknya mengarah pd respon frekuensi, dan seberapa tepat proses filtering sesuai dg frek. cutt off yg dirancang.
2) Belum bisa menjawab cara setting frek. cut off filter melalui mikrokontroler.

Malang, 200

(Ir. F. Yudi L, MT)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

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

NAMA : Ovan Teja
N I M : 0117056
Perbaikan meliputi :

- Block diagram diperbaiki disesuaikan dgn prinsip kerja alat
- prinsip kerjanya dijelaskan.
- Tata tulis laporan nya diperbaiki
- setting fa = ✓

Malang, 23 - 03 2003

( )

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika/Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: Ocvan Teja Pratama			Nim: 0117056
2.	Waktu Pengajuan	Tanggal: 13	Bulan: 01	Tahun: 2009
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen*) I Komang S, ST, MT		Ketua Jurusan Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274	
5.	Judul yang diajukan mahasiswa:	Perancangan dan Pembuatan High Pass Filter Digital Berbasis Mikrokontroler AT89S51		
6.	Perubahan judul yang disetujui Dosen sesuai materi bidang ilmu	....		
7.	Catatan:			
	Persetujuan Judul skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu		Disetujui 13 - 1 - 2009 Dosen I Komang S. Somawirata	

Perhatian:

1. Formulir pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-2

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

Lampiran : 1 (satu) berkas
Pembimbing Skripsi

Kepada : Yth. Bapak I Komang Somawirata,ST,MT
Dosen Institut Teknologi Nasional
MALANG

Yang bertanda tangan di bawah ini :

Nama	: Ocvan Teja Pratama
Nim	: 01.17.056
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika

Dengan ini mengajukan permohonan, kiranya Bapak bersedia menjadi Dosen Pembimbing Utama, untuk penyusunan Skripsi dengan judul (proposal terlampir) :

Perencanaan dan Pembuatan High Pass Filter Digital Berbasis Mikrokontroler AT89S51

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Tugas Akhir Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan Bapak kami Ucapkan terima kasih.

Malang, Januari 2009

Mengetahui
Ketua Jurusan Teknik Elektro S-1

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

Hormat Kami,

Ocvan Teja Pratama
NIM. 01.17.056

Form S-3a

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

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari Mahasiswa :

Nama : Ocvan Teja Pratama

Nim : 01.17.056

Semester : 15

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Elektronika

Dengan ini menyatakan bersedia / tidak bersedia *) Membimbing skripsi dari Mahasiswa tersebut, dengan judul :

Perencanaan dan Pembuatan High Pass Filter Digital Berbasis Mikrokontroler AT89S51

Demikian Surat Pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang, Januari 2009

Kami Yang Membuat Pernyataan,

I Komang Somawirata,ST,MT
NIP. 1030100361

Catatan :

Setelah disetujui agar Formulir ini
Diserahkan mahasiswa/I yang bersangkutan
Kepada Jurusan untuk diproses lebih lanjut
*)Coret yang tidak perlu

Form S-3b

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

FORMULIR BIMBINGAN SKRIPSI

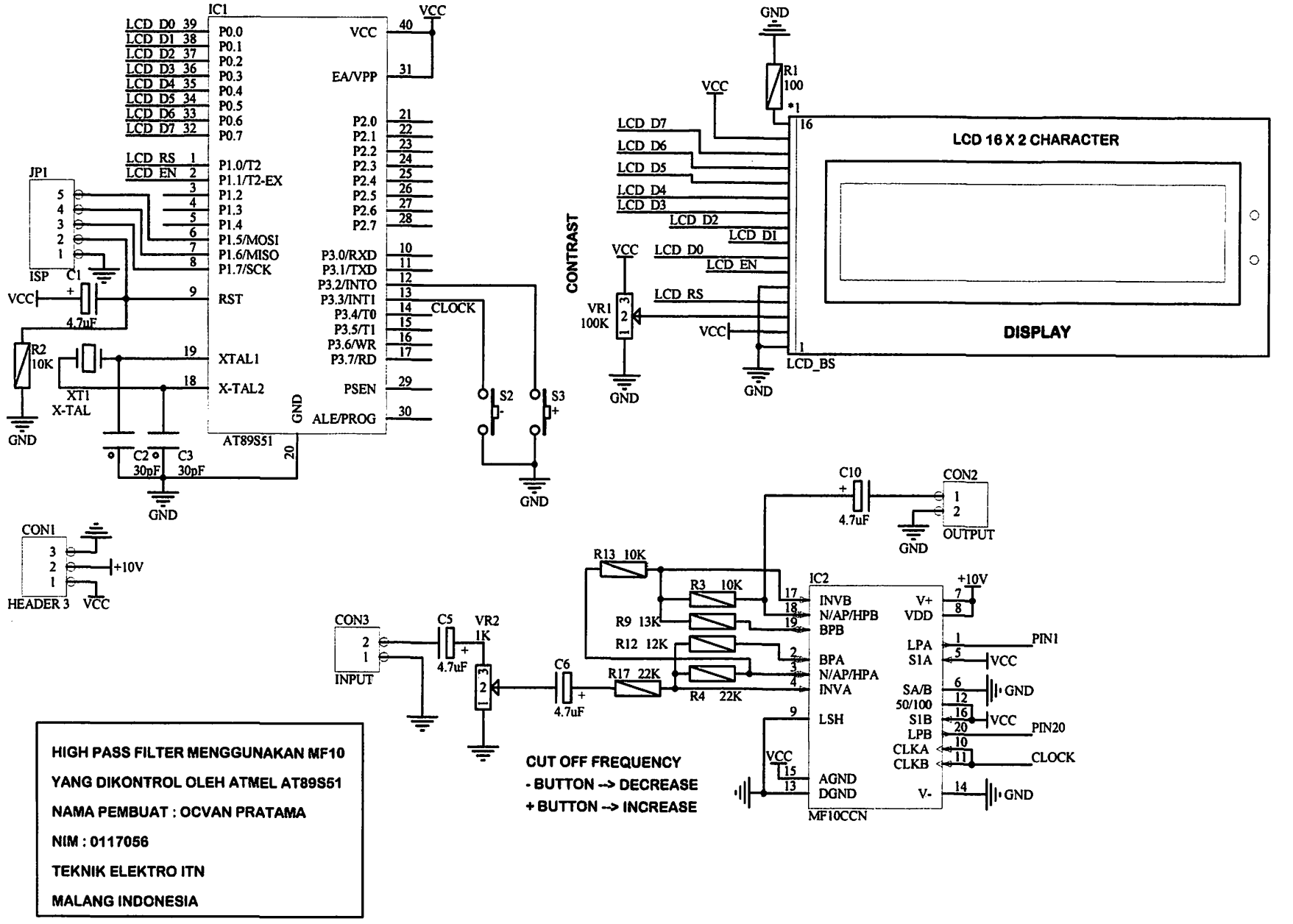
Nama : Ocvan Teja Pratama
Nim : 01.17.056
Masa Bimbingan : 04 Februari 2009 s/d 04 April 2009
Judul Skripsi : Perancangan Dan Pembuatan High Pass Filter Menggunakan MF10 Yang Dikontrol Oleh Mikrokontroller AT89S51

No	Tanggal	Uraian	Paraf Pembimbing
1	10-02-2009	Penambahan Judul skripsi	
2	14-02-2009	Perubahan pada Blok Diagram	
3	18-02-2009	Bab III. Perencanaan & Pembuatan Alat.	
4	19-02-2009	Bab IV. Pengujian Alat.	
5			
6			
7			
8			
9			
10			

Malang,
Dosen pembimbing

I Komang Somawirata ST, MT
NIP. 1030100361

Form S-4a


IC1
AT89S51
LCD D0 39
LCD D1 38
LCD D2 37
LCD D3 36
LCD D4 35
LCD D5 34
LCD D6 33
LCD D7 32
LCD RS 1
LCD EN 2
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P1.0/T2
P1.1/T2-EX
P1.2
P1.3
P1.4
P1.5/MOSI
P1.6/MISO
P1.7/SCK
RST
XTAL1
X-TAL2
GND
VCC
EA/VPP
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
P3.0/RXD
P3.1/TXD
P3.2/INTO
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
PSEN
ALE/PROG
JP1
ISP
C1
4.7uF
R2
10K
XT1
X-TAL
C2
30pF
C3
30pF
CLOCK
S2
S3
CONTRAST
VR1
100K
R1
100
LCD 16 X 2 CHARACTER
DISPLAY
LCD_BS
LCD D7
LCD D6
LCD D5
LCD D4
LCD D3
LCD D2
LCD D1
LCD D0
LCD EN
LCD RS
CON1
HEADER 3
+10V
CON2
OUTPUT
C10
4.7uF
CON3
INPUT
C5
4.7uF
VR2
1K
C6
4.7uF
R13 10K
R3 10K
R9 13K
R12 12K
R17 22K
R4 22K
IC2
MF10CCN
INVB
N/AP/HPB
BPB
BPA
N/AP/HPA
INVA
LSH
AGND
DGND
V+
VDD
LPA
S1A
SA/B
50/100
S1B
LPB
CLKA
CLKB
V-
PIN1
PIN20
CLOCK
HIGH PASS FILTER MENGGUNAKAN MF10
YANG DIKONTROL OLEH ATMEL AT89S51
NAMA PEMBUAT : OCVAN PRATAMA
NIM : 0117056
TEKNIK ELEKTRO ITN
MALANG INDONESIA
CUT OFF FREQUENCY
- BUTTON --> DECREASE
+ BUTTON --> INCREASE

```
;=============================================================================
;                          T U G A S   A K H I R
; HIGH PASS FILTER BERBASISKAN IC NATIONAL SEMICONDUCTOR MF10CCN YANG
; DIKONTROL OLEH IC MIKROKONTROLER ATMEL AT89S51
; OLEH : OCVAN TEJA PRATAMA
; NIM  : 0117056
; TEKNIK ELEKTRO
; ITN - MALANG
;=============================================================================

$include(Reg51.inc)

clock    bit     P3.4
temp             equ     50h
freq             equ     51h
decH             equ     52h
decL             equ     53h
HexVal   equ     55h
hitung1  equ     2Eh
hitung2  equ     2Fh
Start            bit     2DH.0

; Awal program pertama dihidupkan
         org     0h
         jmp     mulai
;=============================================
; Alamat interrupt 0
;=============================================
         org     03h
         clr     EA
         jmp     sambungan_int
;=============================================
; Alamat interrupt 1
;=============================================
         org     13h
         clr     EA
sambungan_int:
         mov     A,hitung2
         dec     A
         mov     hitung2,A
         jnz     terus0
         dec     hitung1
         mov     A,hitung1
         jz              hitungan_habis
         mov     hitung2,#100
         mov     R1,#0CDh
         call    write_inst
         mov     HexVal,hitung1
         call    hex2des
         call    tulis_angka
terus0:
         jb              P3.2,cek1
         mov     hitung1,#50
         mov     A,freq
```

```
        dec     A
        jz              sambungan_int
        jmp     terus2

cek1:
        jb              P3.3,sambungan_int
        mov     hitung1,#50
        mov     A,freq
        inc     A
        cjne    A,#06h,terus2
        jmp     sambungan_int
terus2:
        mov     freq,A
        mov     R1,#0C5h
        call    write_inst
        mov     HexVal,freq
        call    hex2des
        call    tulis_angka
        call    delay
        call    delay
        call    delay
        call    delay
        call    delay
        call    delay
        jmp     sambungan_int
hitungan_habis:
        mov     DPTR,#Run
        call    barisa
        call    barisb
        setb    Start
        mov     A,freq
        dec     A
        jz              _frekuensi_1
        dec     A
        jz              _frekuensi_2
        dec     A
        jz              _frekuensi_3
        dec     A
        jz              _frekuensi_4
        dec     A
        jmp     frekuensi_5

_frekuensi_1:
        jmp     frekuensi_1
_frekuensi_2:
        jmp     frekuensi_2
_frekuensi_3:
        jmp     frekuensi_3
_frekuensi_4:
        jmp     frekuensi_4
```

```
;=============================================
; ======= Routine delay ==========
;=============================================
delay:
        mov     R7,#50
        djnz    R7,$
        djnz    R6,delay
        ret
;
Ldelay:
        mov     R2,#030h
Ld1:
        call    delay
        djnz    R2,Ld1
        ret
; ==== Routine menulis instruksi ke LCD ====
write_inst:
        mov     P1,#0F0h
wrdt:
        mov     P0,R1   ;intruksi ke LCD
        setb    P1.1                ;module
        clr     P1.1
        call    delay
        ret
; === untuk menulis data ke LCD =====
write_data:
        mov     P1,#0F1h
        jmp     wrdt

;=== Menulis LCD baris atas ===
barisa:
        mov     R3,#16
        mov     R1,#80h
tulis0:
        call    write_inst
tulis1: clr A
        movc    A,@A+DPTR
        mov     R1,A
        Inc     DPTR
        call    write_data
        djnz    R3,Tulis1
        ret
;=== Menulis LCD baris bawah ===
barisb:
        mov     R3,#16
        mov     R1,#0C0h
        jmp     tulis0
;=============================================
; Untuk menulis angka di LCD
; angka pada decH decL
;=============================================
tulis_angka:
        mov     A,decH
        anl     A,#0Fh
```

```
        add     A,#30H
        mov     R1,A
        call    write_data
;
        mov     A,decl
        swap    A
        anl     A,#0Fh
        add     A,#30H
        mov     R1,A
        call    write_data
;
        mov     A,decl
        anl     A,#0Fh
        add     A,#30H
        mov     R1,A
        call    write_data
        ret
;=============================================
; Routine untuk konversi bilangan
; hexadesimal ke desimal
; input hex -> HexVal
; output desimal -> DecH + DecL
;=============================================
hex2des:
        mov     PSW,#0
        mov     dech,#0
        mov     A,Hexval
        swap    A
        anl     A,#0Fh
        mov     temp,A
        jz      dibawah_0Fh
        clr     A
tambah_16:
        Add     A,#16h
        da      A
        mov     R6,A
        clr     A
        addc    A,dech
        mov     dech,A
        mov     A,R6
        djnz    temp,tambah_16
sdh_ditambah:
        mov     decl,A
        mov     A,hexval
        anl     A,#0Fh
        da      A
        add     A,decl
        da      A
        mov     decl,A
        clr     A
        addc    A,dech
        mov     dech,A
        jmp     _exit
dibawah_0Fh:
```

```
        clr     C
        mov     A,hexval
        da      A
        mov     decl,A
_exit:  ret
;=============================================
; ==== untuk menulis character 2 baris ====
tulis:
        mov     R4,#3
        mov     DPTR,#Judul
tulis4:
        call    barisa
        call    barisb
        call    Ldelay
        djnz    R4,tulis4
        ret
;=============================================
; awal program ada disini
;=============================================

mulai:
        mov     freq,#1
        mov     hitung1,#10
        mov     hitung2,#100
; ==== initialisasi LCD ====
        mov     R1,#03Fh
        call    write_inst
        call    write_inst
        mov     R1,#0Dh
        call    write_inst
        mov     R1,#06h
        call    write_inst
        mov     R1,#01h
        call    write_inst
        mov     R1,#0Ch
        call    write_inst
        call    tulis
        setb    EA ; Enable Interupt
        setb    EX0 ; aktifkan Int 0
        setb    EX1 ; aktifkan Int 1
        mov     A,#9

; Pilihan Menu
        mov     DPTR,#Menu
        call    barisa
        call    barisb
        clr     Start
tunggu:
        jnb     Start,tunggu


; program looping disini selalu
frekuensi_1:
        mov     dptr,#F1
```

```
        call    barisb
_f1:
        cpl     clock
        sjmp    _f1
;
frekuensi_2:
        mov     dptr,#F2
        call    barisb
_f2:
        cpl     clock
        nop
        nop
        nop
        nop
        sjmp    _f2
;
frekuensi_3:
        mov     dptr,#F3
        call    barisb
_f3:
        cpl     clock
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        sjmp    _f3
;
frekuensi_4:
        mov     dptr,#F4
        call    barisb
_f4:
        cpl     clock
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        sjmp    _f4
;
frekuensi_5:
        mov     dptr,#F5
        call    barisb
_f5:
        cpl     clock
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
                nop
        nop
        nop
        nop
        nop
```

```
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
        nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
        nop
nop
nop
```

```
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
      nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
                nop
        nop
        nop
        jmp     _f5
;=============================================
; tabel data untuk tampil di LCD
;=============================================
Judul:
        DB      "  TUGAS AKHIR  "
        DB      "HIGH PASS FILTER"
        DB              "OCVAN T. PRATAMA"
        DB              "  ITN - 0117056 "
Menu:
        DB              " CUT OFF FREQ : "
        DB              "Data:000 Run:000"
Run:
        DB              "Running ........"
        DB              "Fcutoff= .....Hz"
F1:
        DB              "Fcutoff= 5000 Hz"
F2:
        DB              "Fcutoff= 2250 Hz"
F3:
        DB              "Fcutoff= 1000 Hz"
F4:
        DB              "Fcutoff=  500 Hz"
F5:
        DB              "Fcutoff=  100 Hz"

;
        end
```

LAMPIRAN DATA SHEET

eatures

- Compatible with MCS-51® Products
- K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- .0V to 5.5V Operating Range
- ully Static Operation: 0 Hz to 33 MHz
- hree-level Program Memory Lock
- 28 x 8-bit Internal RAM
- 2 Programmable I/O Lines
- wo 16-bit Timer/Counters
- ix Interrupt Sources
- ull Duplex UART Serial Channel
- ow-power Idle and Power-down Modes
- nterrupt Recovery from Power-down Mode
- Watchdog Timer
- ual Data Pointer
- ower-off Flag
- ast Programming Time
- lexible ISP Programming (Byte and Page Mode)

escription

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

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

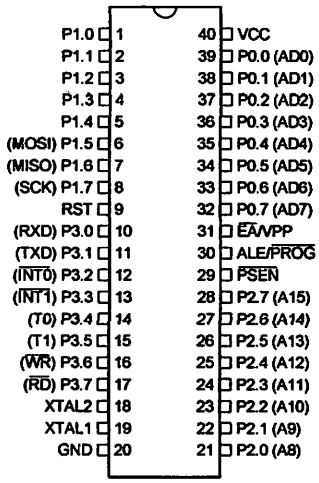
8-bit Microcontroller with 4K Bytes In-System Programmable Flash

AT89S51

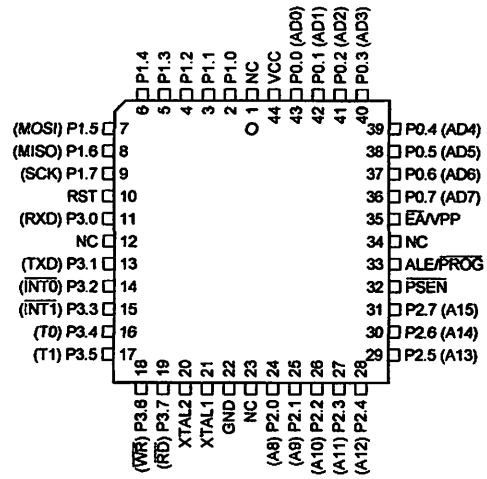
Rev. 2487A–10/01

n Configurations

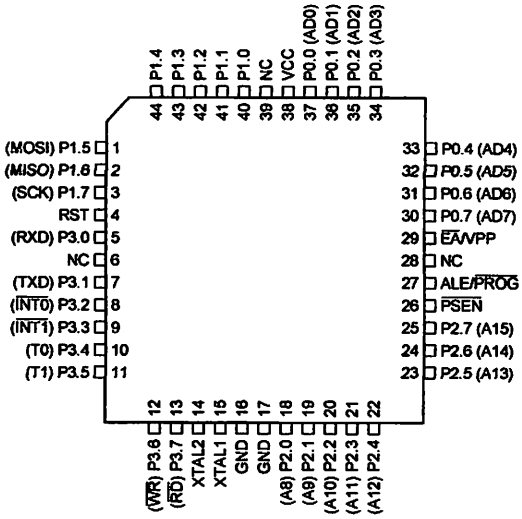
PDIP



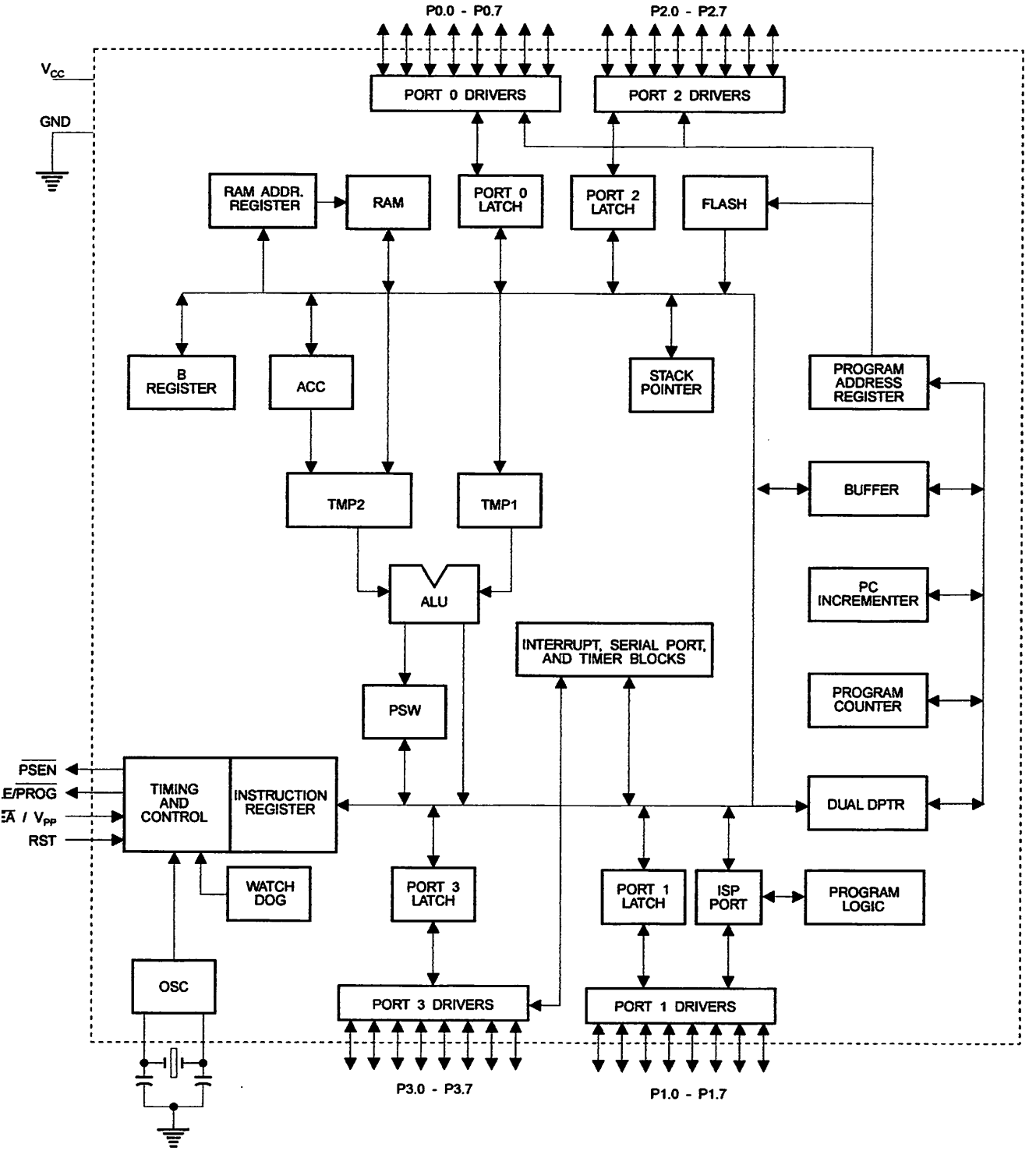
PLCC



TQFP



ock Diagram


P0.0 - P0.7
P2.0 - P2.7
V_{CC}
GND
PORT 0 DRIVERS
PORT 2 DRIVERS
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
FLASH
B REGISTER
ACC
STACK POINTER
PROGRAM ADDRESS REGISTER
TMP2
TMP1
BUFFER
ALU
PC INCREMENTER
INTERRUPT, SERIAL PORT, AND TIMER BLOCKS
PSW
PROGRAM COUNTER
PSEN
E/PROG
EA / V_{PP}
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DUAL DPTR
WATCH DOG
PORT 3 LATCH
PORT 1 LATCH
ISP PORT
PROGRAM LOGIC
OSC
PORT 3 DRIVERS
PORT 1 DRIVERS
P3.0 - P3.7
P1.0 - P1.7

n Description

CC Supply voltage.

ND Ground.

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

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

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

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

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

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

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

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

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

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

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

Port 3 also serves the functions of various special features of the AT89S51, as shown in the following table.

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

ST

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

LE/$\overline{PROG}$

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

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

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

$\overline{SEN}$

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

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

$\overline{A}$/VPP

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

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

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

TAL1

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

TAL2

Output from the inverting oscillator amplifier

ɔecial
ınction
ɛgisters

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

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

ɔle 1. AT89S51 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H
	80H	88H	90H	98H	0A0H	0A8H	0B0H	0B8H	

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

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

Table 2. AUXR: Auxiliary Register

AUXR Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

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

– Reserved for future expansion

DISALE Disable/Enable ALE

DISALE Operating Mode

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

1 ALE is active only during a MOVX or MOVC instruction

DISRTO Disable/Enable Reset out

DISRTO

0 Reset pin is driven High after WDT times out

1 Reset pin is input only

WDIDLE Disable/Enable WDT in IDLE mode

WDIDLE

0 WDT continues to count in IDLE mode

1 WDT halts counting in IDLE mode

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

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

Table 3. AUXR1: Auxiliary Register 1

AUXR1

Address = A2H

Reset Value = XXXXXXX0B

Not Bit Addressable

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

– Reserved for future expansion

DPS Data Pointer Register Select

DPS

0 Selects DPTR Registers DP0L, DP0H

1 Selects DPTR Registers DP1L, DP1H

emory rganization

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

ogram Memory

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

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

ta Memory

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

atchdog mer ne-time abled with eset-out)

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

sing the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC=1/FOSC. To make the best use of the WDT, it

should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

DT During ıwer-down ıd Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt, which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S51 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

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

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S51 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

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

ıRT

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

ner 0 and 1

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

errupts

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

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

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

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

Table 4. Interrupt Enable (IE) Register

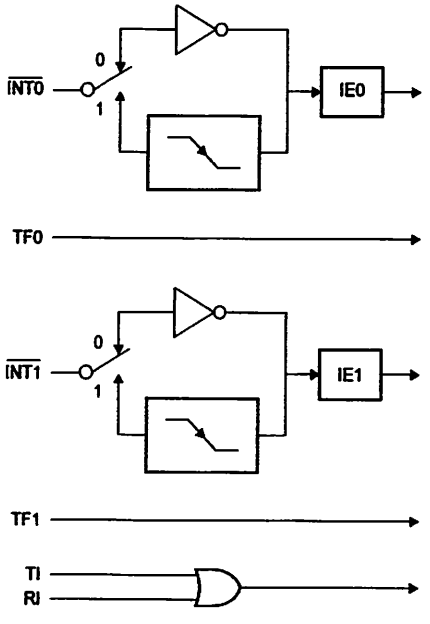
(MSB)							(LSB)
EA	–	–	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

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

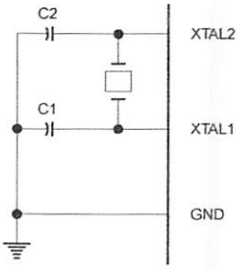
Figure 1. Interrupt Sources



scillator haracteristics

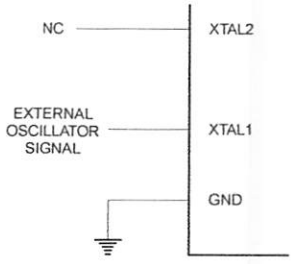
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected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

Figure 2. Oscillator Connections



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

Figure 3. External Clock Drive Configuration



e Mode

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

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

wer-down ode

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

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

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

ogram emory Lock ts

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

Table 6. Lock Bit Protection Modes

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

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

ogramming e Flash – rallel Mode

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

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

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

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

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

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

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

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

(000H) = 1EH indicates manufactured by Atmel
(100H) = 51H indicates 89S51
(200H) = 06H

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

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

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

ogramming e Flash – erial Mode

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

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

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

rial ogramming gorithm

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

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

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

Power-off sequence (if needed):

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

Set RST to "L".

Turn V_{CC} power off.

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

ırial ogramming ɜtruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 8 on page 18.

'ogramming terface – ırallel Mode

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

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

ɔle 7. Flash Programming Modes

ɔde	V_{CC}	RST	$\overline{\text{PSEN}}$	ALE/ $\overline{\text{PROG}}$	$\overline{\text{EA}}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.3-0 Address	P1.7-0 Address
rite Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A11-8	A7-0
ɜad Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A11-8	A7-0
rite Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
rite Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
rite Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
ɜad Lock Bits 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
ip Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
ɜad Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	0000	00H
ɜad Device ID	5V	H	L	H	H	L	L	L	L	L	51H	0001	00H
ɜad Device ID	5V	H	L	H	H	L	L	L	L	L	06H	0010	00H

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

Figure 4. Programming the Flash Memory (Parallel Mode)

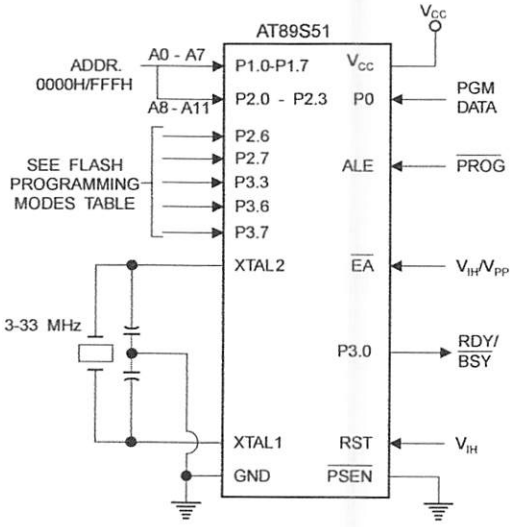
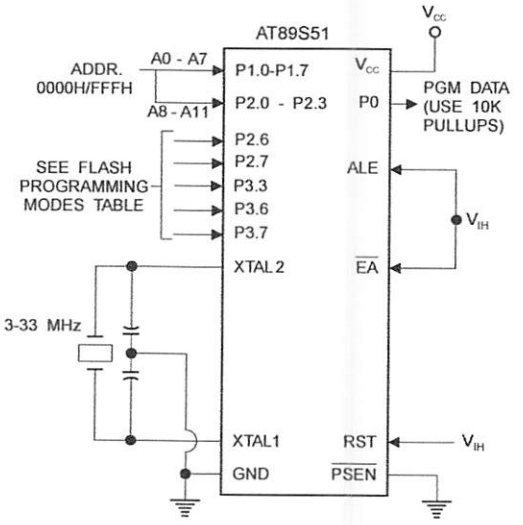


Figure 5. Verifying the Flash Memory (Parallel Mode)

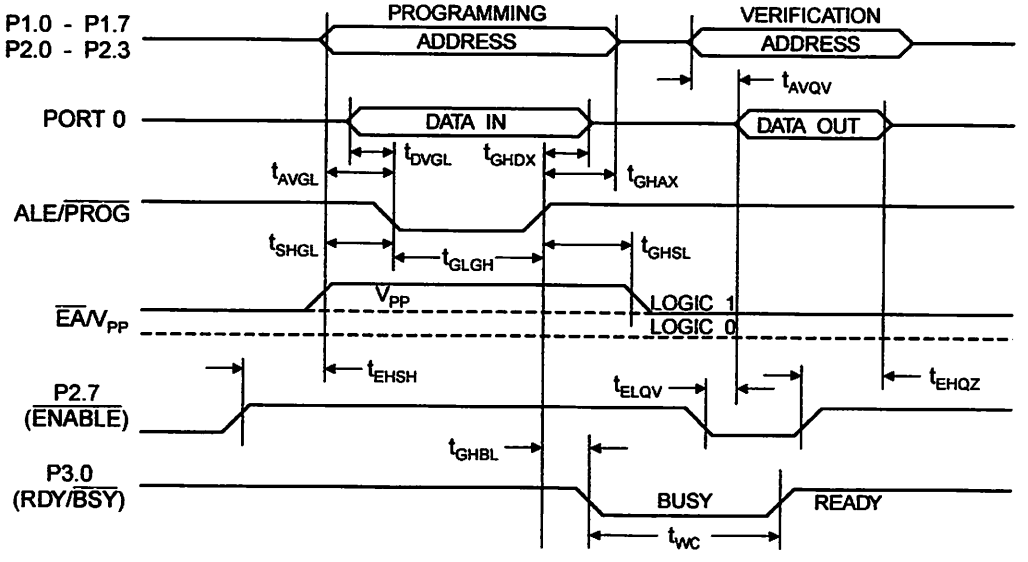


ash Programming and Verification Characteristics (Parallel Mode)

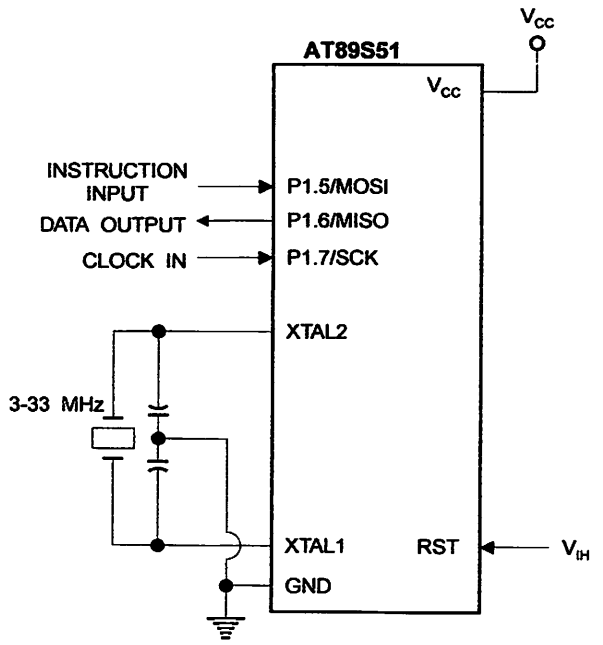
= 20°C to 30°C, V_{CC} = 4.5 to 5.5V

mbol	Parameter	Min	Max	Units
PP	Programming Supply Voltage	11.5	12.5	V
D	Programming Supply Current		10	mA
C	V_{CC} Supply Current		30	mA
CLCL	Oscillator Frequency	3	33	MHz
AVGL	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
GHAX	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
DVGL	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
GHDX	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
EHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
SHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
GHSL	V_{PP} Hold After $\overline{PROG}$	10		µs
GLGH	$\overline{PROG}$ Width	0.2	1	µs
AVQV	Address to Data Valid		$48t_{CLCL}$	
ELQV	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
EHQZ	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
GHBL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
WC	Byte Write Cycle Time		50	µs

ure 6. Flash Programming and Verification Waveforms – Parallel Mode

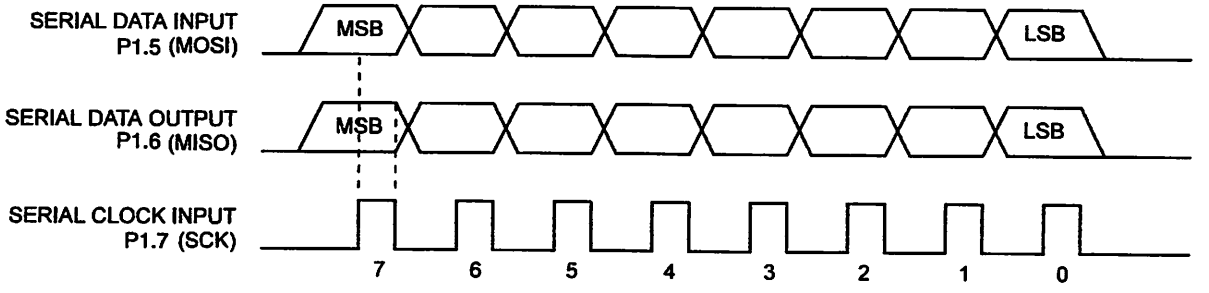


ure 7. Flash Memory Serial Downloading



ash Programming and Verification Waveforms – Serial Mode

ure 8. Serial Programming Waveforms



ble 8. Serial Programming Instruction Set

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

es:
1. The signature bytes are not readable in Lock Bit Modes 3 and 4.
2. B1 = 0, B2 = 0 →Mode 1, no lock protection
   B1 = 0, B2 = 1 →Mode 2, lock bit 1 activated
   B1 = 1, B2 = 0 →Mode 3, lock bit 2 activated
   B1 = 1, B1 = 1 →Mode 4, lock bit 3 activated

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

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

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

Serial Programming Characteristics

Figure 9. Serial Programming Timing

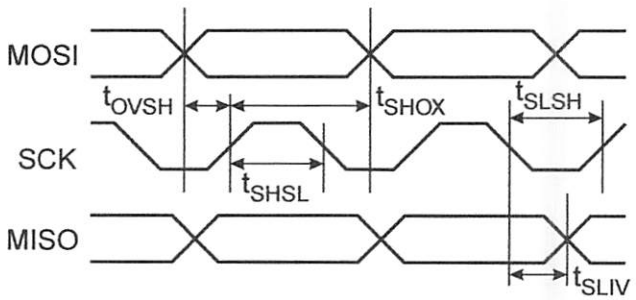


Table 9. Serial Programming Characteristics, T_A = -40° C to 85° C, V_{CC} = 4.0 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0		33	MHz
t_{CLCL}	Oscillator Period	30			ns
t_{SHSL}	SCK Pulse Width High	8 t_{CLCL}			ns
t_{SLSH}	SCK Pulse Width Low	8 t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns
t_{ERASE}	Chip Erase Instruction Cycle Time			500	ms
t_{SWC}	Serial Byte Write Cycle Time			64 t_{CLCL} + 400	µs

ɔsolute Maximum Ratings*

ɔerating Temperature	-55°C to +125°C
orage Temperature	-65°C to +150°C
ltage on Any Pin th Respect to Ground	-1.0V to +7.0V
aximum Operating Voltage	6.6V
Output Current	15.0 mA

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

C Characteristics

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

mbol	Parameter	Condition	Min	Max	Units
	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
L1	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
H	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
H1	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
OL	Output Low Voltage(1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
OL1	Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
OH	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
OH1	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
ST	Reset Pulldown Resistor		50	300	KΩ
	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode(2)	V_{CC} = 5.5V		50	µA

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

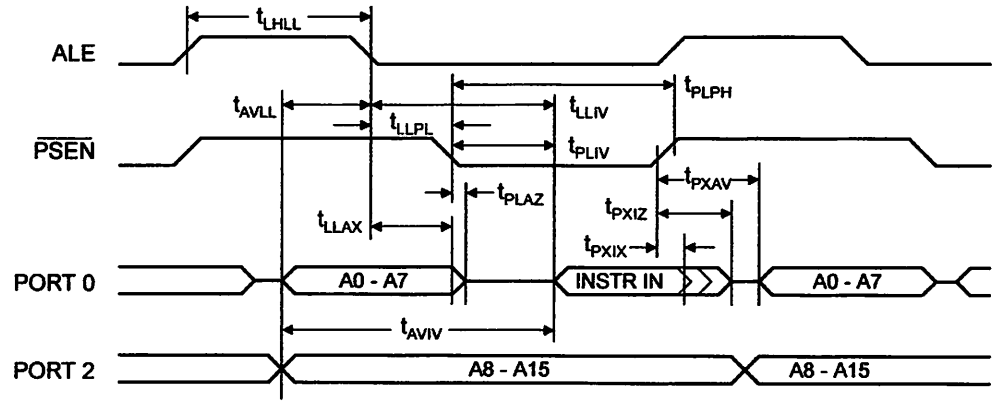
C Characteristics

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

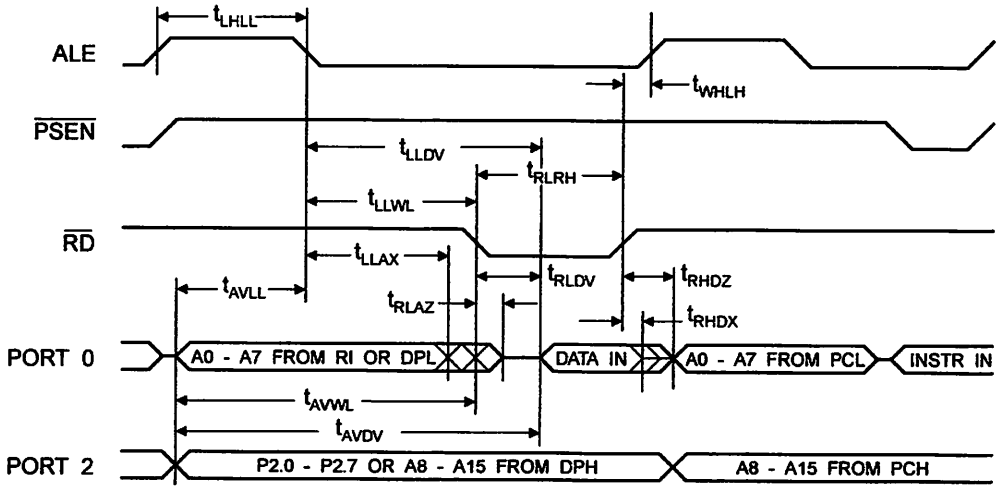
xternal Program and Data Memory Characteristics

ymbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{CLCL}	Oscillator Frequency			0	33	MHz
LLL	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
LL	Address Valid to ALE Low	43		t_{CLCL}-25		ns
AX	Address Hold After ALE Low	48		t_{CLCL}-25		ns
IV	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
PL	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-25		ns
PH	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-45		ns
IV	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-60	ns
IX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
IZ	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-25	ns
AV	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
IV	Address to Valid Instruction In		312		$5t_{CLCL}$-80	ns
AZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
RH	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
WH	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
DV	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
DX	Data Hold After $\overline{RD}$	0		0		ns
DZ	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
DV	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
DV	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
WL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
WL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
WX	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-30		ns
WH	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-130		ns
QX	Data Hold After $\overline{WR}$	33		t_{CLCL}-25		ns
AZ	$\overline{RD}$ Low to Address Float		0		0	ns
LH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-25	t_{CLCL}+25	ns

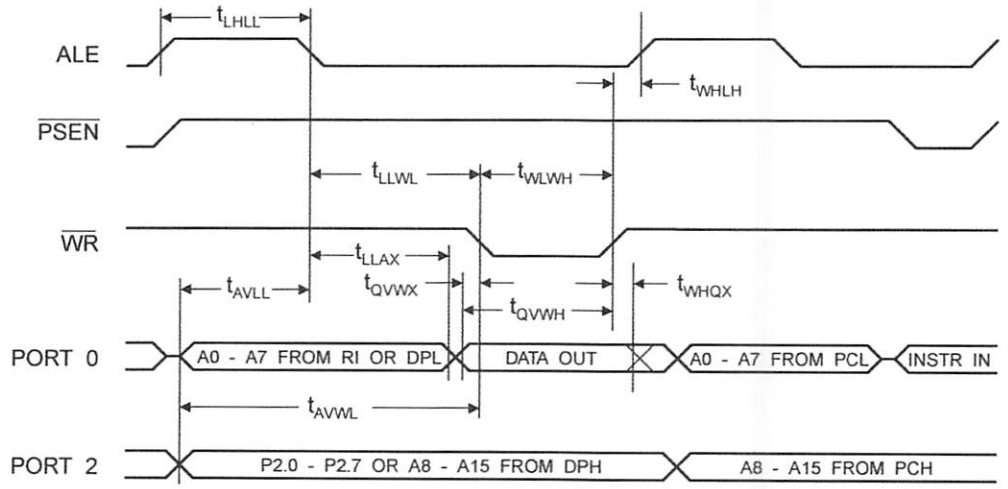
ternal Program Memory Read Cycle



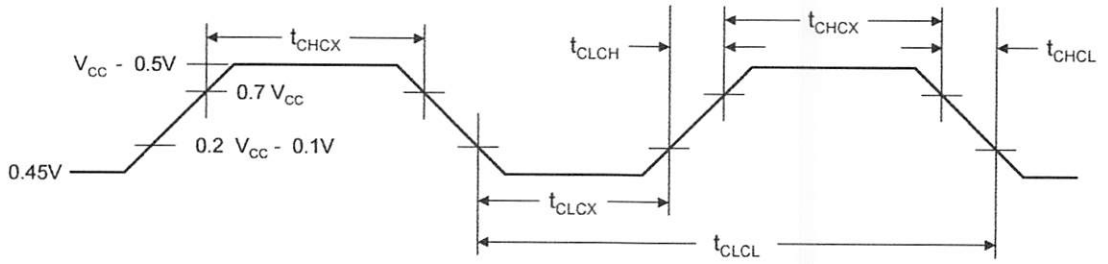
ternal Data Memory Read Cycle



xternal Data Memory Write Cycle



xternal Clock Drive Waveforms



xternal Clock Drive

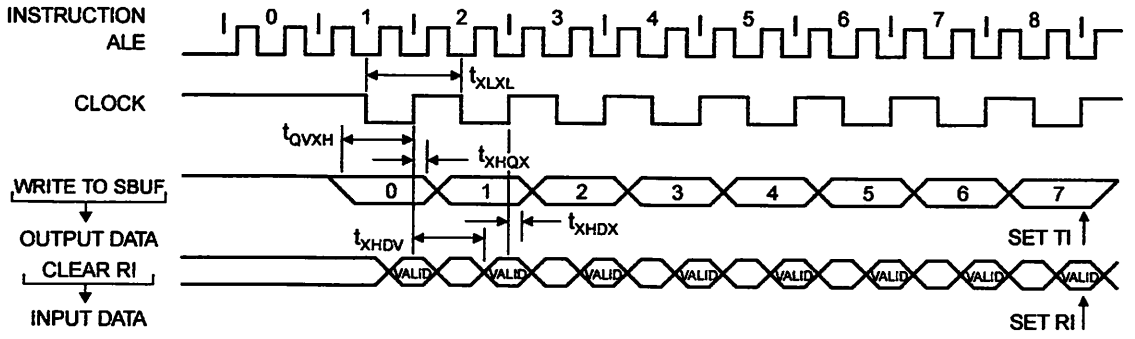
ymbol	Parameter	Min	Max	Units
CLCL	Oscillator Frequency	0	33	MHz
CL	Clock Period	30		ns
HCX	High Time	12		ns
CX	Low Time	12		ns
CH	Rise Time		5	ns
HCL	Fall Time		5	ns

ɪrial Port Timing: Shift Register Mode Test Conditions

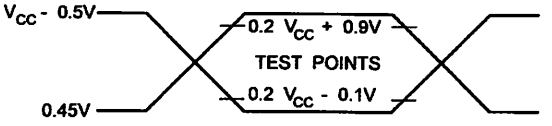
ɜ values in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

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

ıift Register Mode Timing Waveforms

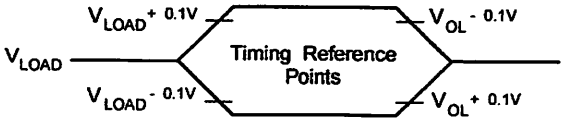


; Testing Input/Output Waveforms[1]



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

ɔat Waveforms[1]



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

dering Information

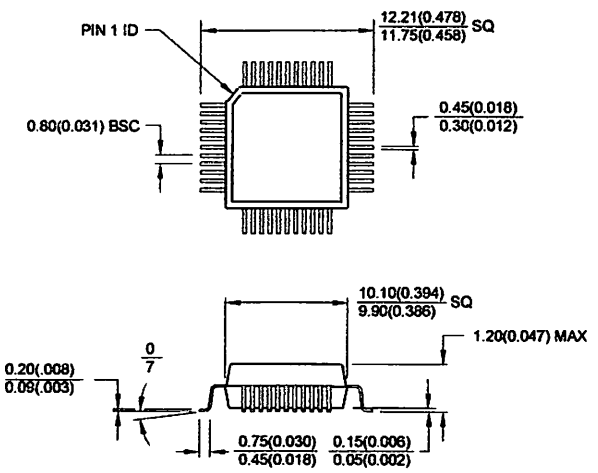
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S51-24AC AT89S51-24JC AT89S51-24PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT89S51-24AI AT89S51-24JI AT89S51-24PI	44A 44J 40P6	Industrial (-40°C to 85°C)
33	4.5V to 5.5V	AT89S51-33AC AT89S51-33JC AT89S51-33PC	44A 44J 40P6	Commercial (0°C to 70°C)

= Preliminary Availability

	Package Type
A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

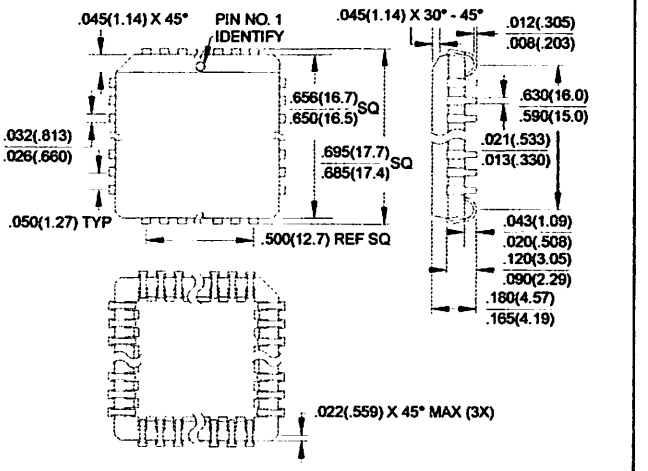
ckaging Information

44A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
Dimensions in Millimeters and (Inches)*

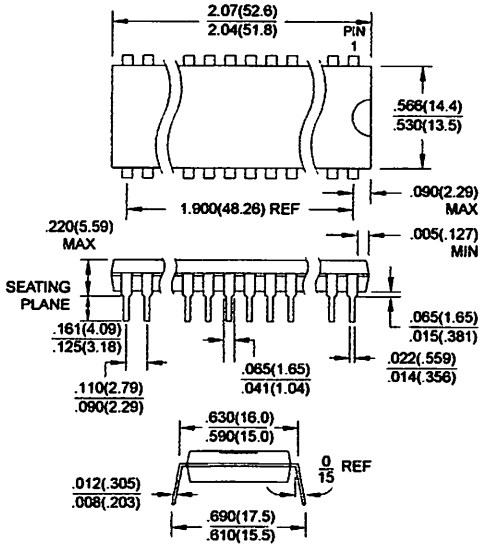


*Controlling dimension: millimeters

44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)



40P6, 40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AC



tmel Headquarters

rporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

rope
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

ia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

pan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Product Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Atmel Heilbronn
Theresienstrasse 2
POB 3535
D-74025 Heilbronn, Germany
TEL (49) 71 31 67 25 94
FAX (49) 71 31 67 24 23

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 0 2 40 18 18 18
FAX (33) 0 2 40 18 19 60

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

2487A–10/01/xM

crocontroller Instruction Set

interrupt response time information, refer to the hardware description chapter.

tructions that Affect Flag Settings(1)

struction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
DD	X	X	X	CLR C	O		
DDC	X	X	X	CPL C	X		
JBB	X	X	X	ANL C,bit	X		
JL	O	X		ANL C,/bit	X		
V	O	X		ORL C,bit	X		
	X			ORL C,/bit	X		
RC	X			MOV C,bit	X		
LC	X			CJNE	X		
ETB C	1						

e: 1. Operations on SFR byte address 208 or bit addresses 209-215 (that is, the PSW or bits in the PSW) also affect flag settings.

e Instruction Set and Addressing Modes

	Register R7-R0 of the currently selected Register Bank.
rect	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
R_i	8-bit internal data RAM location (0-255) addressed indirectly through register R1or R0.
ata	8-bit constant included in instruction.
ata 16	16-bit constant included in instruction.
dr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.
dr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of program memory as the first byte of the following instruction.
	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
	Direct Addressed bit in Internal Data RAM or Special Function Register.

struction Set Summary

	0	1	2	3	4	5	6	7
0	NOP	JBC bit,rel [3B, 2C]	JB bit, rel [3B, 2C]	JNB bit, rel [3B, 2C]	JC rel [2B, 2C]	JNC rel [2B, 2C]	JZ rel [2B, 2C]	JNZ rel [2B, 2C]
1	AJMP (P0) [2B, 2C]	ACALL (P0) [2B, 2C]	AJMP (P1) [2B, 2C]	ACALL (P1) [2B, 2C]	AJMP (P2) [2B, 2C]	ACALL (P2) [2B, 2C]	AJMP (P3) [2B, 2C]	ACALL (P3) [2B, 2C]
2	LJMP addr16 [3B, 2C]	LCALL addr16 [3B, 2C]	RET [2C]	RETI [2C]	ORL dir, A [2B]	ANL dir, A [2B]	XRL dir, a [2B]	ORL C, bit [2B, 2C]
3	RR A	RRC A	RL A	RLC A	ORL dir, #data [3B, 2C]	ANL dir, #data [3B, 2C]	XRL dir, #data [3B, 2C]	JMP @A + DPTR [2C]
4	INC A	DEC A	ADD A, #data [2B]	ADDC A, #data [2B]	ORL A, #data [2B]	ANL A, #data [2B]	XRL A, #data [2B]	MOV A, #data [2B]
5	INC dir [2B]	DEC dir [2B]	ADD A, dir [2B]	ADDC A, dir [2B]	ORL A, dir [2B]	ANL A, dir [2B]	XRL A, dir [2B]	MOV dir, #data [3B, 2C]
6	INC @R0	DEC @R0	ADD A, @R0	ADDC A, @R0	ORL A, @R0	ANL A, @R0	XRL A, @R0	MOV @R0, @data [2B]
7	INC @R1	DEC @R1	ADD A, @R1	ADDC A, @R1	ORL A, @R1	ANL A, @R1	XRL A, @R1	MOV @R1, #data [2B]
8	INC R0	DEC R0	ADD A, R0	ADDC A, R0	ORL A, R0	ANL A, R0	XRL A, R0	MOV R0, #data [2B]
9	INC R1	DEC R1	ADD A, R1	ADDC A, R1	ORL A, R1	ANL A, R1	XRL A, R1	MOV R1, #data [2B]
A	INC R2	DEC R2	ADD A, R2	ADDC A, R2	ORL A, R2	ANL A, R2	XRL A, R2	MOV R2, #data [2B]
B	INC R3	DEC R3	ADD A, R3	ADDC A, R3	ORL A, R3	ANL A, R3	XRL A, R3	MOV R3, #data [2B]
C	INC R4	DEC R4	ADD A, R4	ADDC A, R4	ORL A, R4	ANL A, R4	XRL A, R4	MOV R4, #data [2B]
D	INC R5	DEC R5	ADD A, R5	ADDC A, R5	ORL A, R5	ANL A, R5	XRL A, R5	MOV R5, #data [2B]
E	INC R6	DEC R6	ADD A, R6	ADDC A, R6	ORL A, R6	ANL A, R6	XRL A, R6	MOV R6, #data [2B]
F	INC R7	DEC R7	ADD A, R7	ADDC A, R7	ORL A, R7	ANL A, R7	XRL A, R7	MOV R7, #data [2B]

e: Key: [2B] = 2 Byte, [3B] = 3 Byte, [2C] = 2 Cycle, [4C] = 4 Cycle, Blank = 1 byte/1 cycle

struction Set Summary (Continued)

	8	9	A	B	C	D	E	F
0	SJMP REL [2B, 2C]	MOV DPTR,# data 16 [3B, 2C]	ORL C, /bit [2B, 2C]	ANL C, /bit [2B, 2C]	PUSH dir [2B, 2C]	POP dir [2B, 2C]	MOVX A, @DPTR [2C]	MOVX @DPTR, A [2C]
1	AJMP (P4) [2B, 2C]	ACALL (P4) [2B, 2C]	AJMP (P5) [2B, 2C]	ACALL (P5) [2B, 2C]	AJMP (P6) [2B, 2C]	ACALL (P6) [2B, 2C]	AJMP (P7) [2B, 2C]	ACALL (P7) [2B, 2C]
2	ANL C, bit [2B, 2C]	MOV bit, C [2B, 2C]	MOV C, bit [2B]	CPL bit [2B]	CLR bit [2B]	SETB bit [2B]	MOVX A, @R0 [2C]	MOVX wR0, A [2C]
3	MOVC A, @A + PC [2C]	MOVC A, @A + DPTR [2C]	INC DPTR [2C]	CPL C	CLR C	SETB C	MOVX A, @RI [2C]	MOVX @RI, A [2C]
4	DIV AB [2B, 4C]	SUBB A, #data [2B]	MUL AB [4C]	CJNE A, #data, rel [3B, 2C]	SWAP A	DA A	CLR A	CPL A
5	MOV dir, dir [3B, 2C]	SUBB A, dir [2B]		CJNE A, dir, rel [3B, 2C]	XCH A, dir [2B]	DJNZ dir, rel [3B, 2C]	MOV A, dir [2B]	MOV dir, A [2B]
6	MOV dir, @R0 [2B, 2C]	SUBB A, @R0	MOV @R0, dir [2B, 2C]	CJNE @R0, #data, rel [3B, 2C]	XCH A, @R0	XCHD A, @R0	MOV A, @R0	MOV @R0, A
7	MOV dir, @R1 [2B, 2C]	SUBB A, @R1	MOV @R1, dir [2B, 2C]	CJNE @R1, #data, rel [3B, 2C]	XCH A, @R1	XCHD A, @R1	MOV A, @R1	MOV @R1, A
8	MOV dir, R0 [2B, 2C]	SUBB A, R0	MOV R0, dir [2B, 2C]	CJNE R0, #data, rel [3B, 2C]	XCH A, R0	DJNZ R0, rel [2B, 2C]	MOV A, R0	MOV R0, A
9	MOV dir, R1 [2B, 2C]	SUBB A, R1	MOV R1, dir [2B, 2C]	CJNE R1, #data, rel [3B, 2C]	XCH A, R1	DJNZ R1, rel [2B, 2C]	MOV A, R1	MOV R1, A
A	MOV dir, R2 [2B, 2C]	SUBB A, R2	MOV R2, dir [2B, 2C]	CJNE R2, #data, rel [3B, 2C]	XCH A, R2	DJNZ R2, rel [2B, 2C]	MOV A, R2	MOV R2, A
B	MOV dir, R3 [2B, 2C]	SUBB A, R3	MOV R3, dir [2B, 2C]	CJNE R3, #data, rel [3B, 2C]	XCH A, R3	DJNZ R3, rel [2B, 2C]	MOV A, R3	MOV R3, A
C	MOV dir, R4 [2B, 2C]	SUBB A, R4	MOV R4, dir [2B, 2C]	CJNE R4, #data, rel [3B, 2C]	XCH A, R4	DJNZ R4, rel [2B, 2C]	MOV A, R4	MOV R4, A
D	MOV dir, R5 [2B, 2C]	SUBB A, R5	MOV R5, dir [2B, 2C]	CJNE R5, #data, rel [3B, 2C]	XCH A, R5	DJNZ R5, rel [2B, 2C]	MOV A, R5	MOV R5, A
E	MOV dir, R6 [2B, 2C]	SUBB A, R6	MOV R6, dir [2B, 2C]	CJNE R6, #data, rel [3B, 2C]	XCH A, R6	DJNZ R6, rel [2B, 2C]	MOV A, R6	MOV R6. A
F	MOV dir, R7 [2B, 2C]	SUBB A, R7	MOV R7, dir [2B, 2C]	CJNE R7, #data, rel [3B, 2C]	XCH A, R7	DJNZ R7, rel [2B, 2C]	MOV A, R7	MOV R7, A

e: Key: [2B] = 2 Byte, [3B] = 3 Byte, [2C] = 2 Cycle, [4C] = 4 Cycle, Blank = 1 byte/1 cycle

le 1. AT89 Instruction Set Summary[1]

ɜmonic		Description	Byte	Oscillator Period
THMETIC OPERATIONS				
)	A,R_n	Add register to Accumulator	1	12
)	A,direct	Add direct byte to Accumulator	2	12
)	A,@R_i	Add indirect RAM to Accumulator	1	12
)	A,#data	Add immediate data to Accumulator	2	12
)C	A,R_n	Add register to Accumulator with Carry	1	12
)C	A,direct	Add direct byte to Accumulator with Carry	2	12
)C	A,@R_i	Add indirect RAM to Accumulator with Carry	1	12
)C	A,#data	Add immediate data to Acc with Carry	2	12
)B	A,R_n	Subtract Register from Acc with borrow	1	12
)B	A,direct	Subtract direct byte from Acc with borrow	2	12
)B	A,@R_i	Subtract indirect RAM from ACC with borrow	1	12
)B	A,#data	Subtract immediate data from Acc with borrow	2	12
	A	Increment Accumulator	1	12
	R_n	Increment register	1	12
	direct	Increment direct byte	2	12
	@R_i	Increment direct RAM	1	12
;	A	Decrement Accumulator	1	12
;	R_n	Decrement Register	1	12
;	direct	Decrement direct byte	2	12
;	@R_i	Decrement indirect RAM	1	12
	DPTR	Increment Data Pointer	1	24
.	AB	Multiply A & B	1	48
	AB	Divide A by B	1	48
	A	Decimal Adjust Accumulator	1	12

: 1. All mnemonics copyrighted © Intel Corp., 1980.

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS				
ANL	A,R_n	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@R_i	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,R_n	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@R_i	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,R_n	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@R_i	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
LOGICAL OPERATIONS (continued)				

emonic		Description	Byte	Oscillator Period
	A	Rotate Accumulator Right	1	12
C	A	Rotate Accumulator Right through the Carry	1	12
AP	A	Swap nibbles within the Accumulator	1	12
TA TRANSFER				
V	A,R_n	Move register to Accumulator	1	12
V	A,direct	Move direct byte to Accumulator	2	12
V	A,@R_i	Move indirect RAM to Accumulator	1	12
V	A,#data	Move immediate data to Accumulator	2	12
V	R_n,A	Move Accumulator to register	1	12
V	R_n,direct	Move direct byte to register	2	24
V	R_n,#data	Move immediate data to register	2	12
V	direct,A	Move Accumulator to direct byte	2	12
V	direct,R_n	Move register to direct byte	2	24
V	direct,direct	Move direct byte to direct	3	24
V	direct,@R_i	Move indirect RAM to direct byte	2	24
V	direct,#data	Move immediate data to direct byte	3	24
V	@R_i,A	Move Accumulator to indirect RAM	1	12
V	@R_i,direct	Move direct byte to indirect RAM	2	24
V	@R_i,#data	Move immediate data to indirect RAM	2	12
V	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
VC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
VC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
VX	A,@R_i	Move External RAM (8-bit addr) to Acc	1	24
A TRANSFER (continued)				

Mnemonic		Description	Byte	Oscillator Period
MOVX	A,@DPTR	Move Exernal RAM (16-bit addr) to Acc	1	24
MOVX	@R_i,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R_n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R_i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				

emonic		Description	Byte	Oscillator Period
ALL	addr11	Absolute Subroutine Call	2	24
ALL	addr16	Long Subroutine Call	3	24
T		Return from Subroutine	1	24
TI		Return from interrupt	1	24
MP	addr11	Absolute Jump	2	24
MP	addr16	Long Jump	3	24
MP	rel	Short Jump (relative addr)	2	24
P	@A+DPTR	Jump indirect relative to the DPTR	1	24
	rel	Jump if Accumulator is Zero	2	24
Z	rel	Jump if Accumulator is Not Zero	2	24
NE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
NE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
NE	R_n,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
NE	@R_i,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
NZ	R_n,rel	Decrement register and Jump if Not Zero	2	24
NZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
P		No Operation	1	12

ble 2. Instruction Opcodes in Hexadecimal Order

ex ode	Number of Bytes	Mnemonic	Operands
	1	NOP	
	2	AJMP	code addr
	3	LJMP	code addr
	1	RR	A
	1	INC	A
	2	INC	data addr
	1	INC	@R0
	1	INC	@R1
	1	INC	R0
	1	INC	R1
	1	INC	R2
	1	INC	R3
	1	INC	R4
	1	INC	R5
	1	INC	R6
	1	INC	R7
	3	JBC	bit addr,code addr
	2	ACALL	code addr
	3	LCALL	code addr
	1	RRC	A
	1	DEC	A
	2	DEC	data addr
	1	DEC	@R0
	1	DEC	@R1
	1	DEC	R0
	1	DEC	R1
	1	DEC	R2
	1	DEC	R3
	1	DEC	R4
	1	DEC	R5
	1	DEC	R6
	1	DEC	R7
	3	JB	bit addr,code addr
	2	AJMP	code addr
	1	RET	
	1	RL	A
	2	ADD	A,#data
	2	ADD	A,data addr

Hex Code	Number of Bytes	Mnemonic	Operands
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2

ex ode	Number of Bytes	Mnemonic	Operands
8	1	ORL	A,R3
C	1	ORL	A,R4
D	1	ORL	A,R5
E	1	ORL	A,R6
F	1	ORL	A,R7
	2	JNC	code addr
	2	ACALL	code addr
	2	ANL	data addr,A
	3	ANL	data addr,#data
	2	ANL	A,#data
	2	ANL	A,data addr
	1	ANL	A,@R0
	1	ANL	A,@R1
	1	ANL	A,R0
	1	ANL	A,R1
	1	ANL	A,R2
	1	ANL	A,R3
	1	ANL	A,R4
	1	ANL	A,R5
	1	ANL	A,R6
	1	ANL	A,R7
	2	JZ	code addr
	2	AJMP	code addr
	2	XRL	data addr,A
	3	XRL	data addr,#data
	2	XRL	A,#data
	2	XRL	A,data addr
	1	XRL	A,@R0
	1	XRL	A,@R1
	1	XRL	A,R0
	1	XRL	A,R1
	1	XRL	A,R2
	1	XRL	A,R3
	1	XRL	A,R4
	1	XRL	A,R5
	1	XRL	A,R6
	1	XRL	A,R7
	2	JNZ	code addr

Hex Code	Number of Bytes	Mnemonic	Operands
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0

ex ode	Number of Bytes	Mnemonic	Operands
	1	SUBB	A,@R1
	1	SUBB	A,R0
	1	SUBB	A,R1
	1	SUBB	A,R2
	1	SUBB	A,R3
	1	SUBB	A,R4
	1	SUBB	A,R5
	1	SUBB	A,R6
	1	SUBB	A,R7
	2	ORL	C,/bit addr
	2	AJMP	code addr
	2	MOV	C,bit addr
	1	INC	DPTR
	1	MUL	AB
		reserved	
	2	MOV	@R0,data addr
	2	MOV	@R1,data addr
	2	MOV	R0,data addr
	2	MOV	R1,data addr
	2	MOV	R2,data addr
	2	MOV	R3,data addr
	2	MOV	R4,data addr
	2	MOV	R5,data addr
	2	MOV	R6,data addr
	2	MOV	R7,data addr
	2	ANL	C,/bit addr
	2	ACALL	code addr
	2	CPL	bit addr
	1	CPL	C
	3	CJNE	A,#data,code addr
	3	CJNE	A,data addr,code addr
	3	CJNE	@R0,#data,code addr
	3	CJNE	@R1,#data,code addr
	3	CJNE	R0,#data,code addr
	3	CJNE	R1,#data,code addr
	3	CJNE	R2,#data,code addr
	3	CJNE	R3,#data,code addr
	3	CJNE	R4,#data,code addr

Hex Code	Number of Bytes	Mnemonic	Operands
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0

ode	Number of Bytes	Mnemonic	Operands
	1	MOVX	A,@R1
	1	CLR	A
	2	MOV	A,data addr
	1	MOV	A,@R0
	1	MOV	A,@R1
	1	MOV	A,R0
	1	MOV	A,R1
	1	MOV	A,R2
	1	MOV	A,R3
	1	MOV	A,R4
	1	MOV	A,R5
	1	MOV	A,R6
	1	MOV	A,R7
	1	MOVX	@DPTR,A
	2	ACALL	code addr
	1	MOVX	@R0,A
	1	MOVX	@R1,A
	1	CPL	A
	2	MOV	data addr,A
	1	MOV	@R0,A
	1	MOV	@R1,A
	1	MOV	R0,A
	1	MOV	R1,A
	1	MOV	R2,A
	1	MOV	R3,A
	1	MOV	R4,A
	1	MOV	R5,A
	1	MOV	R6,A
	1	MOV	R7,A

struction Definitions

ALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7 through 5, and the second byte of the instruction. The subroutine called must therefore start within the same 2 K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label SUBRTN is at program memory location 0345 H. After executing the following instruction,

```
ACALL      SUBRTN
```

at location 0123H, SP contains 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC contains 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10	a9	a8	1	0	0	0	1

a7	a6	a5	a4	a3	a2	a1	a0

Operation: ACALL

$(PC) \leftarrow (PC) + 2$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{7-0})$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{15-8})$
$(PC_{10-0}) \leftarrow$ page address

D A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise, OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B), and register 0 holds 0AAH (10101010B). The following instruction,

```
ADD     A,R0
```

leaves 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

D A,R_n

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	1	r	r	r

Operation: ADD
(A) ← (A) + (R_n)

D A,direct

Bytes: 2

Cycles: 1

Encoding:

0	0	1	0	0	1	0	1	direct address

Operation: ADD
(A) ← (A) + (direct)

D A,@R_i

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	1	1	i

Operation: ADD
(A) ← (A) + ((R_i))

D A,#data

Bytes: 2

Cycles: 1

Encoding:

0	0	1	0	0	1	0	0	immediate data

Operation: ADD
(A) ← (A) + #data

DC A, <src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The following instruction,

```
ADDC    A,R0
```

leaves 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

DC A,R_n

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	1	r	r	r

Operation: ADDC
(A) ← (A) + (C) + (R_n)

DC A,direct

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	1	direct address

Operation: ADDC
(A) ← (A) + (C) + (direct)

DC A,@R_i

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	1	1	i

Operation: ADDC
(A) ← (A) + (C) + ((R_i))

DC A,#data

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	0	immediate data

Operation: ADDC
(A) ← (A) + (C) + #data

MP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7 through 5, and the second byte of the instruction. The destination must therfore be within the same 2 K block of program memory as the first byte of the instruction following AJMP.

Example: The label JMPADR is at program memory location 0123H. The following instruction,

```
AJMP        JMPADR
```

is at location 0345H and loads the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10	a9	a8	0	0	0	0	1

a7	a6	a5	a4	a3	a2	a1	a0

Operation: AJMP
(PC) ← (PC) + 2
($PC_{10\text{-}0}$) ← page address

L <dest-byte>,<src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: If the Accumulator holds 0C3H (11000011B), and register 0 holds 55H (01010101B), then the following instruction,

```
ANL     A,R0
```

leaves 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction clears combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The following instruction,

```
ANL     P1,#01110011B
```

clears bits 7, 3, and 2 of output port 1.

- A,R_n

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	1	r	r	r

Operation: ANL
(A) ← (A) ∧ (R_n)

L A,direct

Bytes: 2

Cycles: 1

Encoding:

0	1	0	1	0	1	0	1	direct address

Operation: ANL
(A) ← (A) ∧ (direct)

L A,@R_i

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	0	1	1	i

Operation: ANL
(A) ← (A) ∧ ((R_i))

L A,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	0	1	0	1	0	0	immediate data

Operation: ANL
(A) ← (A) ∧ #data

L direct,A

Bytes: 2

Cycles: 1

Encoding:

0	1	0	1	0	0	1	0	direct address

Operation: ANL
(direct) ← (direct) ∧ (A)

L direct,#data

Bytes: 3

Cycles: 2

Encoding:

0	1	0	1	0	0	1	1	direct address	immediate data

Operation: ANL
(direct) ← (direct) ∧ #data

C,<src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0, then ANL C clears the carry flag; otherwise, this instruction leaves the carry flag in its current state. A slash (/) preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC.7 = 1, and OV = 0:

```
MOV     C,P1.0      ;LOAD CARRY WITH INPUT PIN STATE
ANL     C,ACC.7     ;AND CARRY WITH ACCUM. BIT 7
ANL     C,/OV       ;AND WITH INVERSE OF OVERFLOW FLAG
```

C,bit

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	0	0	1	0	bit address

Operation: ANL
(C) ← (C) ∧ (bit)

C,/bit

Bytes: 2

Cycles: 2

Encoding:

1	0	1	1	0	0	0	0	bit address

Operation: ANL
(C) ← (C) ∧ ¬ (bit)

NE <dest-byte>,<src-byte>, rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
                CJNE    R7, # 60H, NOT_EQ
;               . . .   . . . . .       ;R7 = 60H.
NOT_EQ:         JC      REQ_LOW         ;IF R7 < 60H.
;               . . .   . . . . .       ;R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the following instruction,

```
WAIT:   CJNE    A, P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program loops at this point until the P1 data changes to 34H.)

NE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1	0	1	0	1	direct address	rel. address

Operation:

```
(PC) ← (PC) + 3
IF (A) < > (direct)
THEN
    (PC) ← (PC) + relative offset
IF (A) < (direct)
THEN
    (C) ← 1
ELSE
    (C) ← 0
```

NE A,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1	0	1	0	0	immediate data	rel. address

Operation:
(PC) ← (PC) + 3
IF (A) < > *data*
THEN
    (PC) ← (PC) + *relative offset*
IF (A) < *data*
THEN
    (C) ← 1
ELSE
    (C) ← 0

NE R_n,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1	1	r	r	r	immediate data	rel. address

Operation:
(PC) ← (PC) + 3
IF (R_n) < > *data*
THEN
    (PC) ← (PC) + *relative offset*
IF (R_n) < *data*
THEN
    (C) ← 1
ELSE
    (C) ← 0

NE @R_i,data,rel

Bytes: 3

Cycles: 2

Encoding:

1	0	1	1	0	1	1	i	immediate data	rel. address

Operation:
(PC) ← (PC) + 3
IF ((R_i)) < > *data*
THEN
    (PC) ← (PC) + *relative offset*
IF ((R_i)) < *data*
THEN
    (C) ← 1
ELSE
    (C) ← 0

R A

Function: Clear Accumulator

Description: CLR A clears the Accumulator (all bits set to 0). No flags are affected

Example: The Accumulator contains 5CH (01011100B). The following instruction,CLR Aleaves the Accumulator set to 00H (00000000B).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	0	1	0	0

Operation: CLR
(A) ← 0

R bit

Function: Clear bit

Description: CLR bit clears the indicated bit (reset to 0). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The following instruction,CLR P1.2 leaves the port set to 59H (01011001B).

R C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	0	1	1

Operation: CLR
(C) ← 0

R bit

Bytes: 2

Cycles: 1

Encoding:

1	1	0	0	0	0	1	0

bit address

Operation: CLR
(bit) ← 0

A

Function: Complement Accumulator

Description: CPLA logically complements each bit of the Accumulator (one's complement). Bits which previously contained a 1 are changed to a 0 and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The following instruction,

```
CPL     A
```

leaves the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	0	1	0	0

Operation: CPL
(A) ← ¬ (A)

bit

Function: Complement bit

Description: CPL bit complements the bit variable specified. A bit that had been a 1 is changed to 0 and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data is read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5BH (01011101B). The following instruction sequence,CPL P1.1CPL P1.2 leaves the port set to 5BH (01011011B).

C

Bytes: 1

Cycles: 1

Encoding:

1	0	1	1	0	0	1	1

Operation: CPL
(C) ← ¬ (C)

bit

Bytes: 2

Cycles: 1

Encoding:

1	0	1	1	0	0	1	0

bit address

Operation: CPL
(bit) ← ¬ (bit)

A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3 through 0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition sets the carry flag if a carry-out of the low-order four-bit field propagates through all high-order bits, but it does not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this sets the carry flag if there is a carry-out of the high-order bits, but does not clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DAA apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B), representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B), representing the packed BCD digits of the decimal number 67. The carry flag is set. The following instruction sequence

```
ADDC    A,R3
DA      A
```

first performs a standard two's-complement binary addition, resulting in the value 0BEH (10111110) in the Accumulator. The carry and auxiliary carry flags are cleared.

The Decimal Adjust instruction then alters the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag is set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum of 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the following instruction sequence,

```
ADD     A, # 99H
DA      A
```

leaves the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 - 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0

Operation: DA

```
-contents of Accumulator are BCD
IF      [[(A3-0) > 9] ∨ [(AC) = 1]]
        THEN (A3-0) ← (A3-0) + 6
             AND
IF      [[(A7-4) > 9] ∨ [(C) = 1]]
        THEN (A7-4) ← (A7-4) + 6
```

C byte

Function: Decrement

Description: DEC byte decrements the variable indicated by 1. An original value of 00H underflows to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The following instruction sequence,

```
DEC     @R0
DEC     R0
DEC     @R0
```

leaves register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

C A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	1	0	0

Operation: DEC
(A) ← (A) - 1

C R_n

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	1	r	r	r

Operation: DEC
(R_n) ← (R_n) - 1

C direct

Bytes: 2

Cycles: 1

Encoding:

0	0	0	1	0	1	0	1

direct address

Operation: DEC
(direct) ← (direct) - 1

C @R_i

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	1	1	i

Operation: DEC
((R_i)) ← ((R_i)) - 1

AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags are cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register are undefined and the overflow flag are set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The following instruction,

```
DIV     AB
```

leaves 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 x 18) + 17. Carry and OV are both cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	0	0	0	1	0	0

Operation: DIV

$(A)_{15-8} \leftarrow (A)/(B)$

$(B)_{7-0}$

NZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H underflows to 0FFH. No flags are affected. The branch destination is computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The following instruction sequence,

```
DJNZ    40H,LABEL_1
DJNZ    50H,LABEL_2
DJNZ    60H,LABEL_3
```

causes a jump to the instruction at label LABEL_2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way to execute a program loop a given number of times or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The following instruction sequence,

```
        MOV     R2, # 8
TOGGLE: CPL     P1.7
        DJNZ    R2,TOGGLE
```

toggles P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse lasts three machine cycles; two for DJNZ and one to alter the pin.

NZ R_n,rel

Bytes: 2

Cycles: 2

Encoding:

1	1	0	1	1	r	r	r	rel. address

Operation: DJNZ
(PC) ← (PC) + 2
(R_n) ← (R_n) - 1
IF (R_n) > 0 or (R_n) < 0
THEN
(PC) ← (PC) + rel

NZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

1	1	0	1	0	1	0	1	direct address	rel. address

Operation: DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) - 1
IF (direct) > 0 or (direct) < 0
THEN
(PC) ← (PC) + rel

<byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH overflows to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7EH (011111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The following instruction sequence,

```
INC     @R0
INC     R0
INC     @R0
```

leaves register 0 set to 7FH and internal RAM locations 7EH and 7FH holding 00H and 41H, respectively.

A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	1	0	0

Operation: INC
(A) ← (A) + 1

R_n

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	1	r	r	r

Operation: INC
(R_n) ← (R_n) + 1

direct

Bytes: 2

Cycles: 1

Encoding:

0	0	0	0	0	1	0	1

direct address

Operation: INC
(direct) ← (direct) + 1

@R_i

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	1	1	i

Operation: INC
((R_i)) ← ((R_i)) + 1

DPTR

Function: Increment Data Pointer

Description: INC DPTR increments the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed, and an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H increments the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. The following instruction sequence,

```
INC     DPTR
INC     DPTR
INC     DPTR
```

changes DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1	0	1	0	0	0	1	1

Operation: INC

(DPTR) ← (DPTR) + 1

bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, JB jump to the address indicated; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The following instruction sequence,

```
JB      P1.2,LABEL1
JB      ACC. 2,LABEL2
```

causes program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0	0	1	0	0	0	0	0	bit address	rel. address

Operation: JB

(PC) ← (PC) + 3

IF (bit) = 1

THEN

(PC) ← (PC) + rel

bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, JBC branches to the address indicated; otherwise, it proceeds with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: The Accumulator holds 56H (01010110B). The following instruction sequence,

```
JBC     ACC.3,LABEL1
JBC     ACC.2,LABEL2
```

causes program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:

0	0	0	1	0	0	0	0	bit address	rel. address

Operation: JBC

```
(PC) ← (PC) + 3
IF   (bit) = 1
     THEN
          (bit) ← 0
          (PC) ← (PC) +rel
```

rel

Function: Jump if Carry is set

Description: If the carry flag is set, JC branches to the address indicated; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The following instruction sequence,

```
JC      LABEL1
CPL     C
JC      LABEL 2
```

sets the carry and causes program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0	1	0	0	0	0	0	0	rel. address

Operation: JC

```
(PC) ← (PC) + 2
IF   (C) = 1
     THEN
          (PC) ← (PC) + rel
```

P @A+DPTR

Function: Jump indirect

Description: JMP @A+DPTR adds the eight-bit unsigned contents of the Accumulator with the 16-bit data pointer and loads the resulting sum to the program counter. This is the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions branches to one of four AJMP instructions in a jump table starting at JMP_TBL.

```
        MOV     DPTR, # JMP_TBL
        JMP     @A + DPTR
JMP_TBL: AJMP   LABEL0
        AJMP    LABEL1
        AJMP    LABEL2
        AJMP    LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution jumps to label LABEL2. Because AJMP is a 2-byte instruction, the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1	0	0	1	1

Operation: JMP
(PC) ← (A) + (DPTR)

bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a 0, JNB branches to the indicated address; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The following instruction sequence,

```
JNB     P1.3,LABEL1
JNB     ACC.3,LABEL2
```

causes program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0	0	1	1	0	0	0	0	bit address	rel. address

Operation: JNB
(PC) ← (PC) + 3
IF (bit) = 0
THEN (PC) ← (PC) + rel

rel

Function: Jump if Carry not set

Description: If the carry flag is a 0, JNC branches to the address indicated; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signal relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The following instruction sequence,

```
JNC     LABEL1
CPL     C
JNC     LABEL2
```

clears the carry and causes program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0	1	0	1	0	0	0	0	rel. address

Operation: JNC
(PC) ← (PC) + 2
IF (C) = 0
THEN (PC) ← (PC) + rel

rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, JNZ branches to the indicated address; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The following instruction sequence,

```
JNZ     LABEL1
INC     A
JNZ     LABEL2
```

sets the Accumulator to 01H and continues at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0	1	1	1	0	0	0	0	rel. address

Operation: JNZ
(PC) ← (PC) + 2
IF (A) ≠ 0
THEN (PC) ← (PC) + rel

rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are 0, JZ branches to the address indicated; otherwise, it proceeds with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01H. The following instruction sequence,

```
JZ      LABEL1
DEC     A
JZ      LABEL2
```

changes the Accumulator to 00H and causes program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	0	0	0	rel. address

Operation: JZ
(PC) ← (PC) + 2
IF (A) = 0
THEN (PC) ← (PC) + rel

ALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K byte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label SUBRTN is assigned to program memory location 1234H. After executing the instruction,

```
LCALL     SUBRTN
```

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1234H.

Bytes: 3

Cycles: 2

Encoding:

0	0	0	1	0	0	1	0	addr15-addr8	addr7-addr0

Operation: LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC) ← $addr_{15-0}$

MP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label JMPADR is assigned to the instruction at program memory location 1234H. The instruction,

```
LJMP      JMPADR
```

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0	0	0	0	0	0	1	0	addr15-addr8	addr7-addr0

Operation: LJMP
(PC) ← $addr_{15-0}$

V <dest-byte>,<src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV     R0,#30H     ;R0 < = 30H
MOV     A,@R0       ;A < = 40H
MOV     R1,A        ;R1 < = 40H
MOV     B,@R1       ;B < = 10H
MOV     @R1,P1      ;RAM (40H) < = 0CAH
MOV     P2,P1       ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

V A,R_n

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	1	r	r	r

Operation: MOV
(A) ← (R_n)

V A,direct

Bytes: 2

Cycles: 1

Encoding:

1	1	1	0	0	1	0	1

direct address

Operation: MOV
(A) ← (direct)

V A,ACC is not a valid Instruction.

V A,@R_i

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	0	1	1	i

Operation: MOV
(A) ← ((Ri))

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	1	1	0	1	0	0	immediate data

Operation: MOV
(A) ← #data

MOV R_n,A

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	1	r	r	r

Operation: MOV
(R_n) ← (A)

MOV R_n,direct

Bytes: 2

Cycles: 2

Encoding:

1	0	1	0	1	r	r	r	direct addr.

Operation: MOV
(R_n) ← (direct)

MOV R_n,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	1	1	1	r	r	r	immediate data

Operation: MOV
(R_n) ← #data

MOV direct,A

Bytes: 2

Cycles: 1

Encoding:

1	1	1	1	0	1	0	1	direct address

Operation: MOV
(direct) ← (A)

MOV direct,R_n

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	1	r	r	r	direct address

Operation: MOV
(direct) ← (R_n)

direct,direct

Bytes: 3

Cycles: 2

Encoding:

1	0	0	0	0	1	0	1	dir. addr. (scr)	dir. addr. (dest)

Operation: MOV
(direct) ← (direct)

direct,@R_i

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	0	1	1	i	direct addr.

Operation: MOV
(direct) ← ((R_i))

direct,#data

Bytes: 3

Cycles: 2

Encoding:

0	1	1	1	0	1	0	1	direct address	immediate data

Operation: MOV
(direct) ← #data

@R_i,A

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	0	1	1	i

Operation: MOV
((R_i)) ← (A)

@R_i,direct

Bytes: 2

Cycles: 2

Encoding:

1	0	1	0	0	1	1	i	direct addr.

Operation: MOV
((R_i)) ← (direct)

@R_i,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	1	1	0	1	1	i	immediate data

Operation: MOV
((R_i)) ← #data

/ <dest-bit>,<src-bit>

Function: Move bit data

Description: MOV <dest-bit>,<src-bit> copies the Boolean variable indicated by the second operand into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B).

```
MOV     P1.3,C
MOV     C,P3.3
MOV     P1.2,C
```

leaves the carry cleared and changes Port 1 to 39H (00111001B).

/ C,bit

Bytes: 2

Cycles: 1

Encoding:

1	0	1	0	0	0	1	0	bit address

Operation: MOV
(C) ← (bit)

/ bit,C

Bytes: 2

Cycles: 2

Encoding:

1	0	0	1	0	0	1	0	bit address

Operation: MOV
(bit) ← (C)

/ DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: *MOV DPTR,#data16 loads the Data Pointer with the 16-bit constant indicated. The 16-bit constant is loaded into* the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the lower-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

```
MOV     DPTR, # 1234H
```

loads the value 1234H into the Data Pointer: DPH holds 12H, and DPL holds 34H.

Bytes: 3

Cycles: 2

Encoding:

1	0	0	1	0	0	0	0	immed. data15-8	immed. data7-0

Operation: MOV
(DPTR) ← $\#data_{15-0}$
DPH ← DPL ← $\#data_{15-8}$ ← $\#data_{7-0}$

/C A,@A+ <base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte or constant from program memory. The address of the byte fetched is the sum of the original unsigned 8-bit Accumulator contents and the contents of a 16-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL_PC:  INC   A
         MOVC  A,@A+PC
         RET
         DB    66H
         DB    77H
         DB    88H
         DB    99H
```

If the subroutine is called with the Accumulator equal to 01H, it returns with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separate the MOVC from the table, the corresponding number is added to the Accumulator instead.

/C A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1	0	0	1	0	0	1	1

Operation: MOVC
(A) ← ((A) + (DPTR))

/C A,@A+PC

Bytes: 1

Cycles: 2

Encoding:

1	0	0	0	0	0	1	1

Operation: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

VX <dest-byte>,<src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, which is why "X" is appended to MOV. There are two types of instructions, differing in whether they provide an 8-bit or 16-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an 8-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins are controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a 16-bit address. P2 outputs the high-order eight address bits (the contents of DPH), while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents, while the P2 output buffers emit the contents of DPH. This form of MOVX is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.

It is possible to use both MOVX types in some situations. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2, followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX    A,@R1
MOVX    @R0,A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

VX A,@R_i

Bytes: 1

Cycles: 2

Encoding:

1	1	1	0	0	0	1	i

Operation: MOVX
(A) ← ((Ri))

VX A,@DPTR

Bytes: 1

Cycles: 2

Encoding:

1	1	1	0	0	0	0	0

Operation: MOVX
(A) ← ((DPTR))

VX @R_i,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	1	i

Operation: MOVX
$((R_i)) \leftarrow (A)$

VX @DPTR,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	0	0

Operation: MOVX
$(DPTR) \leftarrow (A)$

AB

Function: Multiply

Description: MUL AB multiplies the unsigned 8-bit integers in the Accumulator and register B. The low-order byte of the 16-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH), the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	1	0	0	1	0	0

Operation: MUL
$(A)_{7\text{-}0} \leftarrow (A) \times (B)$
$(B)_{15\text{-}8}$

P

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: A low-going output pulse on bit 7 of Port 2 must last exactly 5 cycles. A simple SETB/CLR sequence generates a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the following instruction sequence,

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETB    P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	0	0

Operation: NOP
(PC) ← (PC) + 1

L <dest-byte> <src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the following instruction,

```
ORL     A,R0
```

leaves the Accumulator holding the value 0D7H (1101011IB).When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL     P1,#00110010B
```

sets bits 5, 4, and 1 of output Port 1.

L A,R_n

Bytes: 1

Cycles: 1

Encoding:

0	1	0	0	1	r	r	r

Operation: ORL
(A) ← (A) ∨ (R_n)

A,direct

Bytes: 2

Cycles: 1

Encoding:

0	1	0	0	0	1	0	1	direct address

Operation: ORL
(A) ← (A) ∨ (direct)

A,@R_i

Bytes: 1

Cycles: 1

Encoding:

0	1	0	0	0	1	1	i

Operation: ORL
(A) ← (A) ∨((R$_i$))

A,#data

Bytes: 2

Cycles: 1

Encoding:

0	1	0	0	0	1	0	0	immediate data

Operation: ORL
(A) ← (A) ∨ #data

direct,A

Bytes: 2

Cycles: 1

Encoding:

0	1	0	0	0	0	1	0	direct address

Operation: ORL
(direct) ← (direct) ∨ (A)

direct,#data

Bytes: 3

Cycles: 2

Encoding:

0	1	0	0	0	0	1	1	direct addr.	immediate data

Operation: ORL
(direct) ← (direct) ∨ #data

L C,<src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash (/) preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV     C,P1.0      ;LOAD CARRY WITH INPUT PIN P10
ORL     C,ACC.7     ;OR CARRY WITH THE ACC. BIT 7
ORL     C,/OV       ;OR CARRY WITH THE INVERSE OF OV.
```

L C,bit

Bytes: 2

Cycles: 2

Encoding:

0	1	1	1	0	0	1	0	bit address

Operation: ORL
(C) ← (C) ∨ (bit)

L C,/bit

Bytes: 2

Cycles: 2

Encoding:

1	0	1	0	0	0	0	0	bit address

Operation: ORL
(C) ← (C) ∨ ($\overline{bit}$)

P direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The following instruction sequence,

```
POP     DPH
POP     DPL
```

leaves the Stack Pointer equal to the value 30H and sets the Data Pointer to 0123H. At this point, the following instruction,

```
POP     SP
```

leaves the Stack Pointer set to 20H. In this special case, the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:

1	1	0	1	0	0	0	0	direct address

Operation: POP
(direct) ← ((SP))
(SP) ← (SP) - 1

;H direct

Function: Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine, the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The following instruction sequence,

```
PUSH    DPL
PUSH    DPH
```

leaves the Stack Pointer set to 0BH and stores 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding:

1	1	0	0	0	0	0	0	direct address

Operation: PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The following instruction,

```
RET
```

leaves the Stack Pointer equal to the value 09H. Program execution continues at location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	0	0	0	1	0

Operation: RET
(PC_{15-8}) ← ((SP))
(SP) ← (SP) - 1
(PC_{7-0}) ← ((SP))
(SP) ← (SP) - 1

TI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt was pending when the RETI instruction is executed, that one instruction is executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The following instruction,

RETI

leaves the Stack Pointer equal to 09H and returns program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	1	0	0	1	0

Operation: RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RA

Function: Rotate Accumulator Left

Description: *The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are* affected.

Example: The Accumulator holds the value 0C5H (11000101B). The following instruction,

RL A

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1

Operation: RL
$(A_n + 1) \leftarrow (A_n)$ n = 0 - 6
$(A_0) \leftarrow (A_7)$

A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H(11000101B), and the carry is zero. The following instruction,

RLC A

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1

Operation: RLC

$(A_n + 1) \leftarrow (A_n)$ n = 0 - 6

$(A_0) \leftarrow (C)$

$(C) \leftarrow (A_7)$

A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The following instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1

Operation: RR

$(A_n) \leftarrow (A_n + 1)$ n = 0 - 6

$(A_7) \leftarrow (A_0)$

A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), the carry is zero. The following instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1

Operation: RRC

$(A_n) \leftarrow (A_n + 1)$ n = 0 - 6

$(A_7) \leftarrow (C)$

$(C) \leftarrow (A_0)$

'B <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The following instructions,

```
SETB    C
SETB    P1.0
```

sets the carry flag to 1 and changes the data output on Port 1 to 35H (00110101B).

'B C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	0	1	1

Operation: SETB
(C) ← 1

'B bit

Bytes: 2

Cycles: 1

Encoding:

1	1	0	1	0	0	1	0	bit address

Operation: SETB
(bit) ← 1

IP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction 127 bytes following it.

Example: The label RELADR is assigned to an instruction at program memory location 0123H. The following instruction,

```
SJMP    RELADR
```

assembles into location 0100H. After the instruction is executed, the PC contains the value 0123H.

Note: Under the above conditions the instruction following SJMP is at 102H. Therefore, the displacement byte of the instruction is the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH is a one-instruction infinite loop.

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0	0	0	0	0	rel. address

Operation: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

3B A,<src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7 and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple-precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3 and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers, OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

```
SUBB    A,R2
```

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by CLR C instruction.

3B A,R_n

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	1	r	r	r

Operation: SUBB
(A) ← (A) - (C) - (R_n)

3B A,direct

Bytes: 2

Cycles: 1

Encoding:

1	0	0	1	0	1	0	1	direct address

Operation: SUBB
(A) ← (A) - (C) - (direct)

3B A,@R_i

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	0	1	1	i

Operation: SUBB
(A) ← (A) - (C) - ((R_i))

3B A,#data

Bytes: 2

Cycles: 1

Encoding:

1	0	0	1	0	1	0	0	immediate data

Operation: SUBB
(A) ← (A) - (C) - #data

AP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3 through 0 and bits 7 through 4). The operation can also be thought of as a 4-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

```
SWAP    A
```

leaves the Accumulator holding the value 5CH (01011100B).

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	1	0	0

Operation: SWAP
(A_{3-0}) D (A_{7-4})

H A,<byte>

Function: Exchange Accumulator with byte variable

Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The Accumulator holds the value 3FH (0011111IB). Internal RAM location 20H holds the value 75H (01110101B). The following instruction,

```
XCH     A,@R0
```

leaves RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the accumulator.

H A,R_n

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	1	r	r	r

Operation: XCH
(A) D $((R_n)$

H A,direct

Bytes: 2

Cycles: 1

Encoding:

1	1	0	0	0	1	0	1	direct address

Operation: XCH
(A) D (direct)

H A,@R_i

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	1	1	i

Operation: XCH
(A) D $((R_i))$

HD A,@R_i

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3 through 0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The following instruction,

```
XCHD    A,@R0
```

leaves RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	1	i

Operation: XCHD

(A_{3-0}) D $((R_{i3-0}))$

<dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

```
XRL     A,R0
```

leaves the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The following instruction,

```
XRL     P1,#00110001B
```

complements bits 5, 4, and 0 of output Port 1.

A,R_n

Bytes: 1

Cycles: 1

Encoding:

0	1	1	0	1	r	r	r

Operation: XRL

(A) ← (A) ∀ (R_n)

- **A,direct**

  Bytes: 2

  Cycles: 1

  Encoding:

0	1	1	0	0	1	0	1	direct address

  Operation: XRL
  (A) ← (A) ∀ (direct)

- **A,@R_i**

  Bytes: 1

  Cycles: 1

  Encoding:

0	1	1	0	0	1	1	i

  Operation: XRL
  (A) ← (A) ∀ ((R_i))

- **A,#data**

  Bytes: 2

  Cycles: 1

  Encoding:

0	1	1	0	0	1	0	0	immediate data

  Operation: XRL
  (A) ← (A) ∀ #data

- **direct,A**

  Bytes: 2

  Cycles: 1

  Encoding:

0	1	1	0	0	0	1	0	direct address

  Operation: XRL
  (direct) ← (direct) ∀ (A)

- **direct,#data**

  Bytes: 3

  Cycles: 2

  Encoding:

0	1	1	0	0	0	1	1	direct address	immediate data

  Operation: XRL
  (direct) ← (direct) ∀ #data

Introducing the MF10: A Versatile Monolithic Active Filter Building Block

National Semiconductor
Application Note 307
Tim Regan
April 1998

A unique alternative for active filter designs is now available with the introduction of the MF10. This new CMOS device can be used to implement precise, high-order filtering functions with no reactive components required.

Filter design takes one of two approaches: passive or active. Passive designs combine resistors, capacitors and inductors to perform specific frequency filtering in applications where precision is less important than mass producibility. For very high frequency applications, a passive approach is quite often the only way to go. Active filters combine op amps and discrete transistors, primarily with resistors and capacitors, to provide impedance buffering and filter parameter tunability. In precision filters, it is most desirable to have an independent "handle" for each of three basic filter parameters: resonant frequency (f_o), Q or quality factor, and the passband gain (H_o). As a general rule, the degree of tunability increases with the number of amplifiers used. The three op amp, state variable active filter, *Figure 1*, is most popular for 2nd order designs.

A major shortcoming of this type of filter is that resonant frequency accuracy is only as good as the capacitors used. In high volume production, to minimize filter tuning procedures, costly, low-tolerance, low-drift capacitors are required. Furthermore, these filters use a fair number of components; 3 op amps, 7 resistors and 2 capacitors for each 2nd order section. Even the best single amplifier 2nd order filter realizations require 3 to 5 resistors and 2 capacitors.

To offer designers an attractive alternative to these types of active filters, a device would have to:

1) eliminate critical capacitors entirely

2) minimize overall parts count

3) provide easy tunability of filter parameters

4) allow for the design of all five filter responses and,

5) simplify design equations.

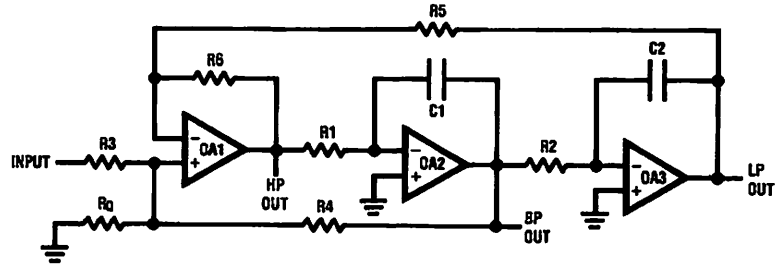
These are the design objectives behind the development of the MF10. Recent advances in sample-data techniques permit the construction of an op amp integrator on a monolithic substrate without the need for any external capacitors (see page 11 "The Switched Capacitor Integrator—How it Works"). The integrator is a key factor in filter designs for establishing the overall filter time constant and, therefore, its resonant frequency. The MF10 contains, in one 20-pin DIP package, all of the necessary active and reactive components to construct two complete 2nd order state variable type active filters, *Figure 2*. The only external requirements are for resistors to establish the desired filter parameters.

Basic Circuit Description

To keep the device as universal as possible, the outputs of each section of each filter are brought out. This allows designs for all five filtering functions: lowpass, bandpass, highpass, allpass and bandreject or notch filters. With two independent 2nd order sections in one package, cascading to achieve 4th order responses can easily be accomplished. Additionally, any of the classical filter response types such as Butterworth, Chebyshev, Bessel and Cauer can be implemented.

Between the output of the summing op amp and the input of the first integrator there is a unique 3-input summing stage where two of the inputs are subtracted from the third. One of the (−) inputs is brought out to serve as the signal input for some filter configurations. The other (−) input is connected through an internal switch to either the lowpass output or analog ground depending upon the desired filter implementation. The direction of this input connection is common to both halves of the MF10 and is controlled by the voltage level on the $S_{A/B}$ input terminal.

Basic Circuit Description (Continued)



$$f_o = \frac{1}{2\pi}\sqrt{\frac{R6}{R5}}\sqrt{\frac{1}{R1\,R2\,C1\,C2}}$$

$$Q = \left(\frac{1 + \frac{R4}{R3} + \frac{R4}{RQ}}{1 + \frac{R6}{R5}}\right)\sqrt{\frac{R6\,R1\,C1}{R5\,R2\,C2}}$$

FIGURE 1. The Universal State Variable 2nd Order Active Filter (note the complexity of design equations and the number of critical external components)

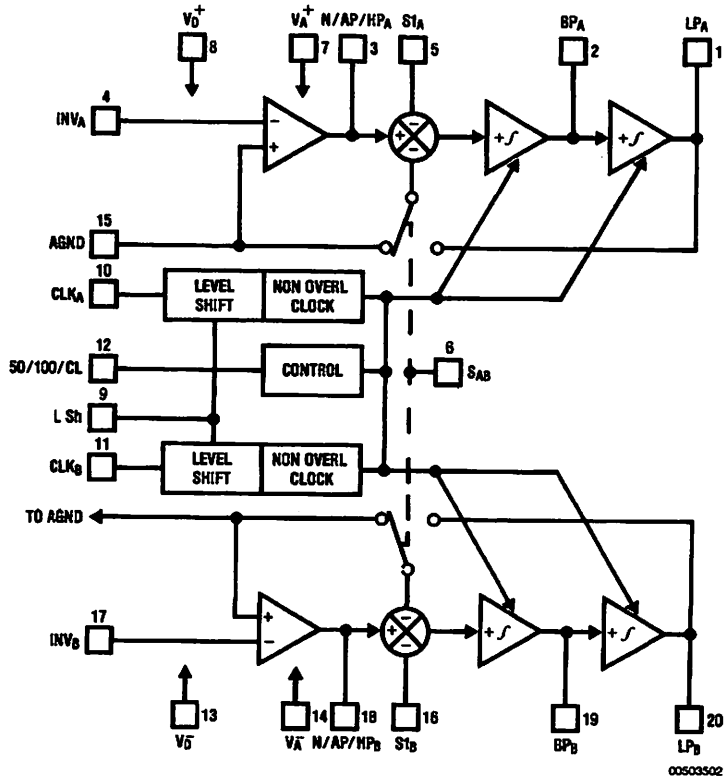


FIGURE 2. Block Diagram of the MF10

When tied to V_D^+ [the (+) supply], the switch connects the lowpass output, and when tied to V_D^- [the (–) supply], the connection to ground is made. In some applications one half of the MF10 may require that both of the (–) inputs to this summer be connected to ground, while the other side requires one to be connected to the lowpass output and the

Basic Circuit Description (Continued)

ther to ground. For this, the $S_{A/B}$ control should be tied to e (−) supply and the connection to the lowpass output ould be made externally to the $S1_A$ ($S1_B$) pin.

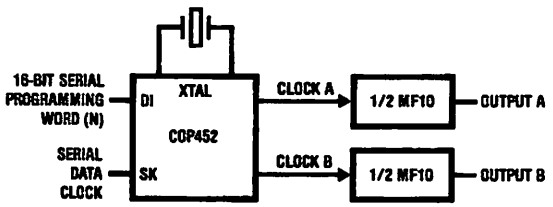
clock with close to 50% duty cycle is required to control the sonant frequency of the filter. Either TTL or CMOS logic mpatible clocks can be accommodated, whether the F10 is powered from split supplies or a single supply, by mply grounding the level shift (L Sh) control pin.

ne resonant frequency of each filter is directly controlled by clock. A tri-level control pin sets the ratio of the clock equency to the center frequency (the 50/100/CL pin) for th halves. When this pin is tied to V^+ the center frequency ill be 1/50 of the clock frequency. When tied to mid-supply tential (i.e., ground, when biased from split supplies) pro-des 100 to 1 clock to center frequency operation. When is pin is tied to V^- a power saving supply current limiter uts down operation and rolls back the supply current by %.

lter center frequency accuracy and stability are only as od as the clock provided. Standard crystal oscillators, mbined with digital counters, can provide very stable cks for specific filter frequencies. A relatively new device m National's COPS™ family of microcontrollers and pe-herals, the COP452 programmable frequency generator/ unter, finds a unique use with the MF10, *Figure 3*. This low st device can generate two independent 50% duty cycle ck frequencies. Each clock output is programmed via a -bit serial data word (N). This allows over 64,000 different ck frequencies for the MF10 from a single crystal.

ne MF10 is intended for use with center frequencies up to kHz, and is guaranteed to operate with clocks up to 1 Hz. This means that for center frequencies greater than 10 z, the 50 to 1 clock control should be used. The effect of ing 100 to 1 or 50 to 1 clock to center frequency ratio anifests itself in the number of "stair-steps" apparent in the tput waveform. The MF10 closely approximates the time d frequency domain response of continuous filters (RC tive filters, for example) but does so using sampling tech-ques. The clock to center frequency control determines the mber of samples taken (1 per clock cycle) in one cycle of e center frequency. Therefore, as shown in the photo of *gure 4*, 100 to 1 clocking provides a smoother looking tput as it has twice as many samples per cycle. For most dio applications, the audible effects of these step edges d the clock frequency component in the output are negli-ble as they are beyond 20 kHz. To obtain a cleaner output veform, a simple passive RC lowpass can be added to the tput to serve as a smoothing filter without affecting the F10 filtering action.

veral of the modes of operation (discussed in a later ction) allow altering of the clock to center frequency ratio an external resistor ratio. This can be used to obtain nter frequencies of values other than 1/50 or 1/100 of the ck frequency. In multiple stage, staggered tuned filters, the center frequency of each stage can be set independently with resistors to allow the overall filter to be controlled by just one clock frequency.



$$f_{CLK} = \frac{f_{XTAL}}{2(1+N)}$$

00503503

FIGURE 3. A Programmable Dual Clock Generator

All of the rules of sampling theory apply when using the MF10. The sampling rate, or clock frequency, should be at least twice the maximum input frequency to produce the best equivalent to a continuous time filter. High frequency components in the input signal that approach the clock frequency will generate aliasing signals which appear at the output of the lower frequency filter and are indistinguishable from valid passband signals. Bandlimiting the input signal to attenuate these potential aliasing frequencies is the best preventative measure. In most applications, aliasing will not be a problem as the clock frequency is much higher than the passband of interest. In the event that a much higher clock frequency is required, the modes of operation which utilize external resistor ratios to increase the clock to center frequency ratio can extend the clock frequency to greater than 100 times the center frequency. By using a higher clock frequency, the aliasing frequencies are correspondingly higher. The limiting factor, with regard to increasing the clock to center frequency ratio, has to do with increased DC offsets at the various outputs.

The Basic Filter Configurations

There are six basic configurations (or modes of operation) for the 2nd order sections in the MF10 to realize a wide variety of filter responses. In all cases, no external capacitors are required. Design is a simple matter of establishing a few resistor ratios to set the desired passband gain and Q and generating a clock for the proper resonant frequency. Each 2nd order section can be treated in a modular fashion, with regard to individual center frequency, Q and gain, when cascading either the two sections within a package or several packages for very high order filters. This individuality of sections is important in implementing the various response characteristics such as Butterworth, Chebyshev, etc.

The following is a general summary of design hints common to all modes of operation.

The Basic Filter Configurations

(Continued)

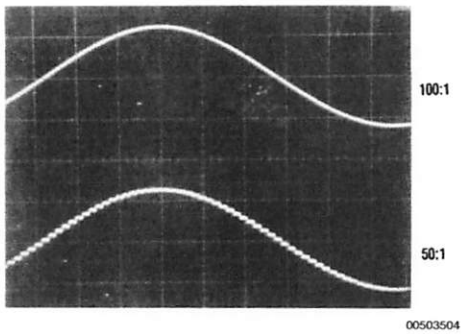


FIGURE 4. The Sampled-Data Output Waveform

1. The maximum supply voltage for the MF10 is ±7V or just +14V for single supply operation. The minimum supply to properly bias the part is 8V.
2. The maximum swing at any of the outputs is typically within 1V of either supply rail.
3. The internal op amps can source 3 mA and sink 1.5 mA. This is an important criterion when selecting a minimum resistor value.
4. The maximum clock frequency is typically 1.5 MHz.
5. To insure the proper filter response, the f_oxQ product of each stage must be realizable by the MF10. For center frequencies less than 5 kHz, the f_oxQ product can be as high as 300 kHz (Q must be less than or equal to 150). A 3 kHz bandpass filter, for example, could have a Q as high as 100 with just one section. For center frequencies less than 20 kHz, the allowable f_oxQ product is limited to 200 kHz. A 10 kHz bandpass design using a single section should have a Q no larger than 20.
6. Center frequency matching from part to part for a given clock frequency is typically ±0.2%. Center frequency drift with temperature (excluding any clock frequency drift) is typically ±10 ppm/°C with 50:1 switching and ±100 ppm/°C for 100:1.
7. Q accuracy from part to part is typically ±2% with a temperature coefficient of ±500 ppm/°C.
8. The expressions for circuit dynamics given with each of the modes are important. They determine the voltage swing at each output as a function of the circuit Q. A high Q bandpass design can generate a significant peak in the response at the lowpass output at the center frequency.
9. Both sides of the MF10 are independent, except for supply voltages, analog ground, clock to center frequency ratio setting and internal switch setting for the three input summing stage.

In the following descriptions of the filter configurations, f_o is the filter center frequency, H_o is the passband gain and Q is the quality factor of the complex pole pair and is equal to f_o/BW where BW is the −3 dB bandwidth measured at the bandpass output.

MODE 1A: Non-Inverting Bandpass, Inverting Bandpass, Lowpass

This is a minimum external component configuration (only 2 resistors) useful for low Q lowpass and bandpass applications. The non-inverting bandpass output is necessary for minimum phase filter designs.

DESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

$$Q = \frac{R3}{R2}$$

$$H_{OLP} = -1$$

$$H_{OBP_1} = -\frac{R3}{R2}$$

$$H_{OBP_2} = 1 \text{ (non-inverting)}$$

CIRCUIT DYNAMICS

$H_{OBP1} = -Q$ (this is the reason for the low Q recommendation)

$H_{OLP\ (peak)} = Q \times H_{OLP}$

MODE 1: Notch, Bandpass and Lowpass

With the addition of just one more external resistor, the output dynamics are improved over Mode 1A to allow bandpass designs with a much higher Q. The notch output features equal gain above and below the notch frequency.

DESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

$$f_{notch} = f_o$$

$$Q = \frac{R3}{R2}$$

$$H_{OLP} = -\frac{R2}{R1}$$

$$H_{OBP} = -\frac{R3}{R1}$$

$$H_{ON} = -\frac{R2}{R1} \text{ as } f \longrightarrow 0 \text{ and as } f \longrightarrow \frac{f_{CLK}}{2}$$

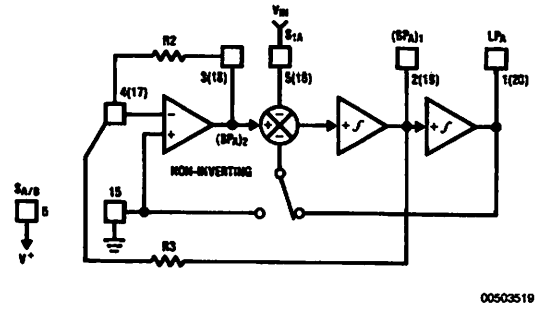
CIRCUIT DYNAMICS

$H_{OBP} = H_{OLP} \times Q = H_{ON} \times Q$

$H_{OLP\ (peak)} = Q \times H_{OLP}$ (if the DC gain of the LP output is too high, a high Q value could cause clipping at the lowpass output resulting in gain non-linearity and distortion at the bandpass output).

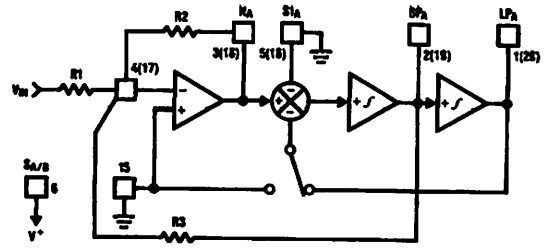
IODE 1: Notch, Bandpass and Lowpass (Continued)

MODE 1A



00503519

MODE 1



00503520

IODE 2: Notch (with $f_n \leq f_o$), andpass and Lowpass

iis configuration allows tuning of the clock to center fre- ency ratio to values greater than 100 to 1 or 50 to 1. The tch output is useful for designing elliptic highpass filters cause the frequency of the required complex zeros (f_{notch}) less than the frequency of the complex poles (f_o).

ESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100}\sqrt{1+\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50}\sqrt{1+\frac{R2}{R4}}$$

$$f_n = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

$$Q = \frac{\sqrt{R_2/R_4+1}}{R_2/R_3}$$

$$H_{OLP} = \frac{-\frac{R2}{R1}}{1+\frac{R2}{R4}}$$

$$H_{OBP} = -\frac{R3}{R1}$$

$$H_{ON_1}\,(\text{as } f \rightarrow 0) = \frac{-\frac{R2}{R1}}{1+\frac{R2}{R4}}$$

$$H_{ON_2}\left(\text{as } f \rightarrow \frac{f_{CLK}}{2}\right) = -\frac{R2}{R1}$$

RCUIT DYNAMICS

$$H_{OBP} = Q\sqrt{H_{OLP} \times H_{ON_2}} = Q\sqrt{H_{ON_1} \times H_{ON_2}}$$

MODE 3: Highpass, Bandpass and Lowpass

This configuration is the classical state variable filter (the circuit of *Figure 1*) implemented with only 4 external resistors. This is the most versatile mode of operation, since the clock to center frequency ratio can be externally tuned either above or below the 100 to 1 or 50 to 1 values. The circuit is suitable for multiple stage Chebyshev filters controlled by a single clock.

DESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100}\sqrt{\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50}\sqrt{\frac{R2}{R4}}$$

$$Q = \sqrt{\frac{R2}{R4}} \times \frac{R3}{R2}$$

$$H_{OHP} = -\frac{R2}{R1}$$

$$H_{OBP} = -\frac{R3}{R1}$$

$$H_{OLP} = -\frac{R4}{R1}$$

CIRCUIT DYNAMICS

$$H_{OHP} = H_{OLP}\left(\frac{R2}{R4}\right)$$

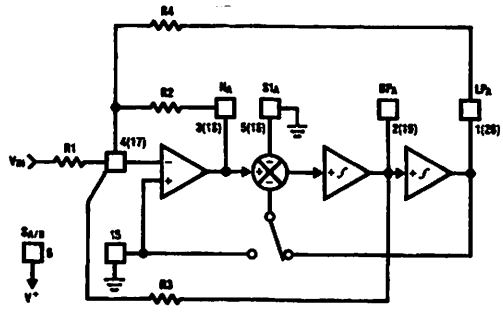
$$H_{OLP\,(peak)} = Q \times H_{OLP}$$

$$H_{OBP} = Q\sqrt{H_{OHP} \times H_{OLP}}$$

$$H_{OHP\,(peak)} = Q \times H_{OHP}$$

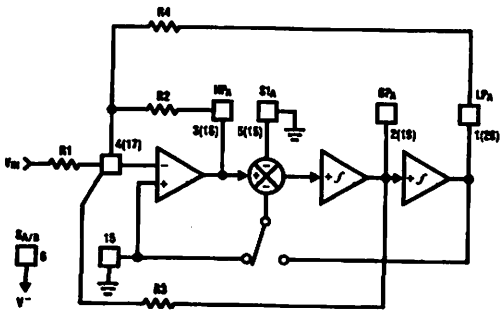
MODE 3: Highpass, Bandpass and Lowpass (Continued)

MODE 2



00503525

MODE 3



00503526

MODE 3A: Highpass, Bandpass, Lowpass and Notch

A notch output is created from the circuit of Mode 3 by summing the highpass and lowpass outputs through an external op amp. The ratio of the summing resistors R_h and R_l adjusts the notch frequency independent of the center frequency. For elliptic filter designs, each stage combines a complex pole pair (at f_o) with a complex zero pair (at f_{notch}) and this configuration provides easy tuning of each of these frequencies for any response type. When cascading several stages of the MF10 the external op amp is needed only at the final output stage. The summing junction for the intermediate stages can be the inverting input of the MF10 internal op amp.

IODE 3A: Highpass, Bandpass, owpass and Notch (Continued)

ESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100}\sqrt{\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50}\sqrt{\frac{R2}{R4}}$$

$$Q = \sqrt{\frac{R2}{R4}} \times \frac{R3}{R2}$$

$$f_{notch} = \frac{f_{CLK}}{100}\sqrt{\frac{R_h}{R_l}} \text{ or } \frac{f_{CLK}}{50}\sqrt{\frac{R_h}{R_l}}$$

$$H_{OHP} = -\frac{R2}{R1}$$

$$H_{OLP} = -\frac{R4}{R1}$$

$$H_{OBP} = -\frac{R3}{R1}$$

$$H_{ON}\,(\text{at } f = f_o) = \left|Q\left(\frac{R_g}{R_l}H_{OLP} - \frac{R_g}{R_h}H_{OHP}\right)\right|$$

$$H_{ON_l}\,(\text{as } f \rightarrow 0) = \frac{R_g}{R_l} \times H_{OLP}$$

$$H_{ON_h}\left(\text{as } f \rightarrow \frac{f_{CLK}}{2}\right) = \frac{R_g}{R_h} \times H_{OHP}$$

00503527

IODE 4: Allpass, Bandpass and owpass

ilizing the $S1_A$ ($S1_B$) terminal as a signal input, an allpass nction can be obtained. An allpass can provide a linear phase change with frequency which results in a constant time delay. This configuration restricts the gain at the allpass output to be unity.

DESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

f_z (frequency of complex zero pair) = f_o

$$Q = \frac{R3}{R2}$$

$$Q_z \text{ (Q of complex zero pair)} = \frac{R3}{R1}$$

$$H_{OAP} = -\frac{R2}{R1} = -1$$

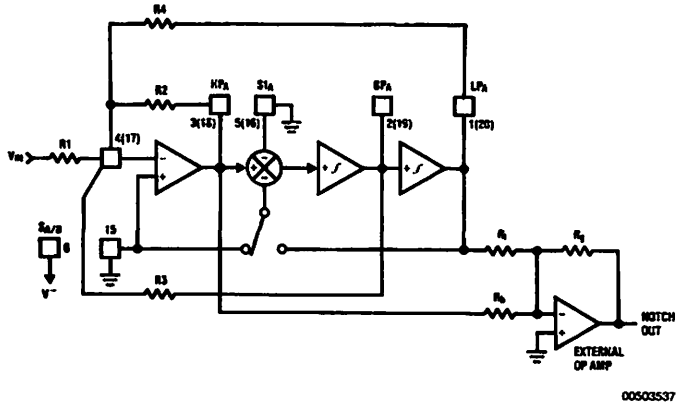
$$H_{OLP} = -\left(\frac{R2}{R1} + 1\right) = -2$$

$$H_{OBP} = -\left(1 + \frac{R2}{R1}\right)\frac{R3}{R2} = -2\frac{R3}{R2}$$

CIRCUIT DYNAMICS

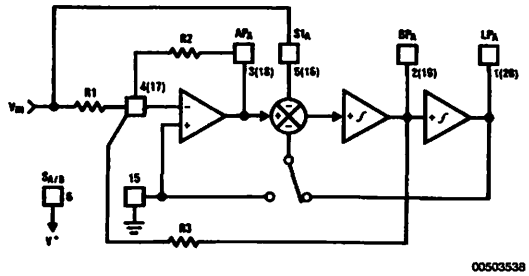
$H_{OBP} = H_{OLP} \times Q = (H_{OAP} + 1)Q$

MODE 3A



00503537

MODE 4



00503538

MODE 5: Complex Zeros (C.z), Bandpass and Lowpass

This mode features an improved allpass design over that of Mode 4, in that it maintains a more constant amplitude with frequency at the complex zeros (C.z) output. The frequencies of the pole pair and zero pair are resistor tunable.

DESIGN EQUATIONS

$$f_o = \frac{f_{CLK}}{100}\sqrt{1+\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50}\sqrt{1+\frac{R2}{R4}}$$

$$f_z = \frac{f_{CLK}}{100}\sqrt{1-\frac{R1}{R4}} \text{ or } \frac{f_{CLK}}{50}\sqrt{1-\frac{R1}{R4}}$$

$$Q = \frac{R3}{R2}\sqrt{1+\frac{R2}{R4}}$$

$$Q_z = \frac{R3}{R1}\sqrt{1-\frac{R1}{R4}}$$

$$H_{O(C.z)} \text{ as } f \rightarrow 0 = \frac{R2(R4-R1)}{R1(R2+R4)}$$

$$H_{O(C.z)} \text{ as } f \rightarrow \frac{f_{CLK}}{2} = \frac{R2}{R1}$$

$$H_{OBP} = \frac{R3}{R2}\left(1+\frac{R2}{R1}\right)$$

$$H_{OLP} = \frac{R4}{R1}\left(\frac{R2+R1}{R2+R4}\right)$$

MODE 6A: Single Pole, Highpass and Lowpass

By using only one of the internal integrators, this mode is useful for creating odd-ordered cascaded filter responses by providing a real pole that is clock tunable to track the resonant frequency of other 2nd order MF10 sections. The corner frequency is resistor tunable.

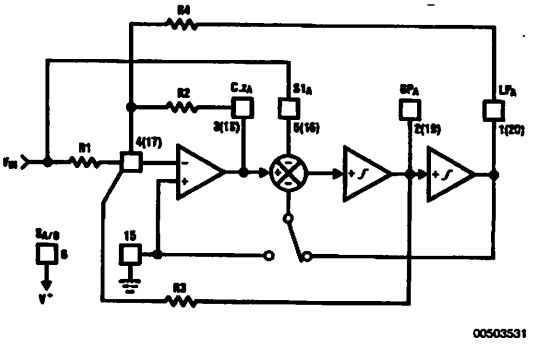
DESIGN EQUATIONS

$$f_c \text{ (cut-off frequency)} = \frac{f_{CLK}}{100}\left(\frac{R2}{R3}\right) \text{ or } \frac{f_{CLK}}{50}\left(\frac{R2}{R3}\right)$$

$$H_{OLP} = -\frac{R3}{R1}$$

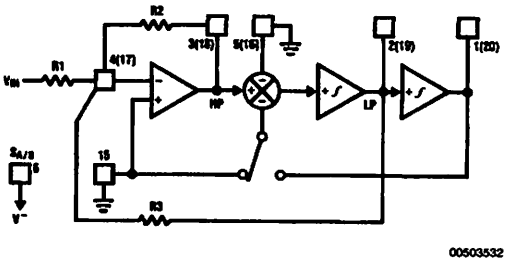
$$H_{OHP} = -\frac{R2}{R1}$$

MODE 5



00503531

MODE 6A



00503532

MODE 6B: Single Pole Lowpass (Inverting and Non- Inverting)

This mode utilizes only one of the integrators for a single pole lowpass, and the input op amp as an inverting amplifier, to provide non-inverting lowpass output. Again, this mode is useful for designing odd-ordered lowpass filters.

DESIGN EQUATIONS

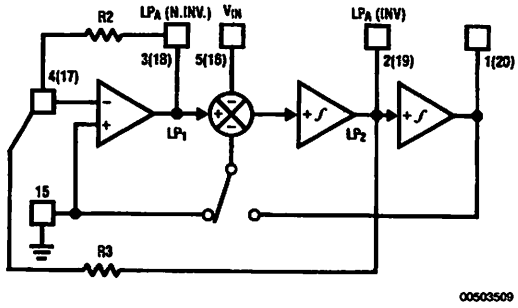
$$f_c \text{ (cut-off frequency)} = \frac{f_{CLK}}{100}\left(\frac{R2}{R3}\right) \text{ or } \frac{f_{CLK}}{50}\left(\frac{R2}{R3}\right)$$

$$H_{OLP} \text{ (inverting output)} = -\frac{R3}{R2}$$

$$H_{OLP} \text{ (non-inverting output)} = +1$$

IODE 6B: Single Pole Lowpass (Inverting and Non- Inverting) (Continued)

MODE 6B



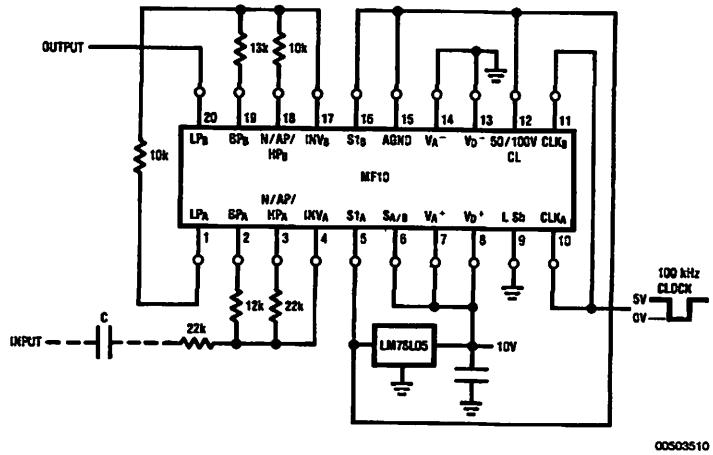
00503509

ome Specific Application Examples

ır single-supply operation, it is important for several termi- ıls to be biased to half supply. A single-supply design for a h order 1 kHz Butterworth lowpass (24 dB/octave or 80 l/decade rolloff) is shown using Mode 1 in *Figure 5*. Note at the analog ground terminal (pin 15), the summer inputs I_A and $S1_B$ (pins 5 and 16) and the clock switching control ı (pin 12) are all biased to $V_{CC}/2$. For symmetrical split pply operation these pins would be grounded. An input upling capacitor is optional, as it is needed only if the input ınal is not also biased to $V_{CC}/2$. For a two-stage Butter- ırth response, both stages have the same corner fre- ency, hence the common clock for both sides. The resistor lues shown are the nearest 5% tolerance values used to t the overall gain of the filter to unity and to set the required of the first stage (side A) to 0.504 and the second stage de B) Q to 1.306 for a flat passband response.

unique advantage of the switched capacitor design of the =10 is illustrated in *Figure 6*. Here the MF10 serves double ty in a data acquisition system as an input filter for simple ndlimiting or anti-aliasing and, as a sample and hold to ow larger amplitude, higher frequency input signals. By ting OFF the applied clock, the switched capacitor integrators will hold the last sampled voltage value. The droop rate of the output voltage during the hold time is approximately 0.1 mV per ms.

A useful non-filtering application of the MF10 is shown in *Figure 7*. In this circuit, the MF10, together with an LM311 comparator, are used as a resonator to generate stable amplitude sine and cosine outputs without using AGC circuitry. The MF10 operates as a Q of 10 bandpass filter which will ring at its resonant frequency in response to a step input change. This ringing signal is fed to the LM311 which creates a square wave input signal to the bandpass to regenerate the oscillation. The bandpass output is the filtered fundamental frequency of a 50% duty cycle square wave. A 90° phase shifted signal of the same amplitude is available at the lowpass output through the second integrator in the MF10. The frequency of oscillation is set by the center frequency of the filter as controlled by the clock and the 50:1/100: 1 control pin. The output amplitude is set by the peak to peak swing of the square wave input, which in this circuit is defined by the back to back diode clamps at the LM311 output.



00503510

FIGURE 5. Only 6 resistors required for this 4th order, 1 kHz Butterworth lowpass filter. This example also illustrates single-supply biasing.

Some Specific Application Examples (Continued)

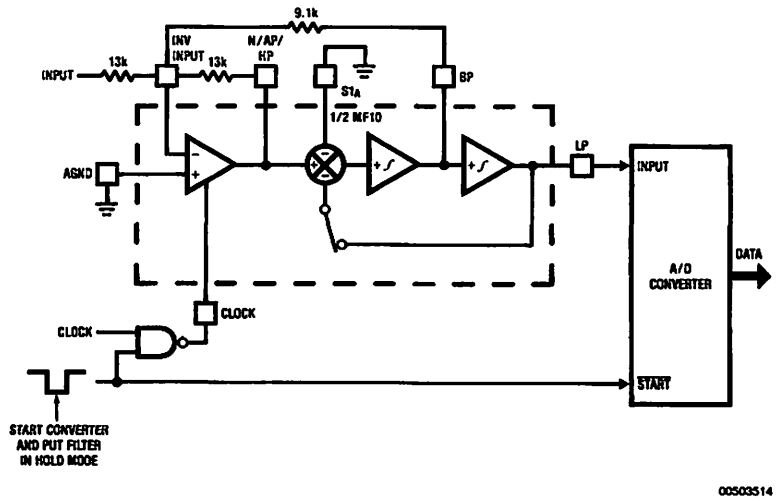


FIGURE 6. An MF10 as an Input Filter and Sample/Hold

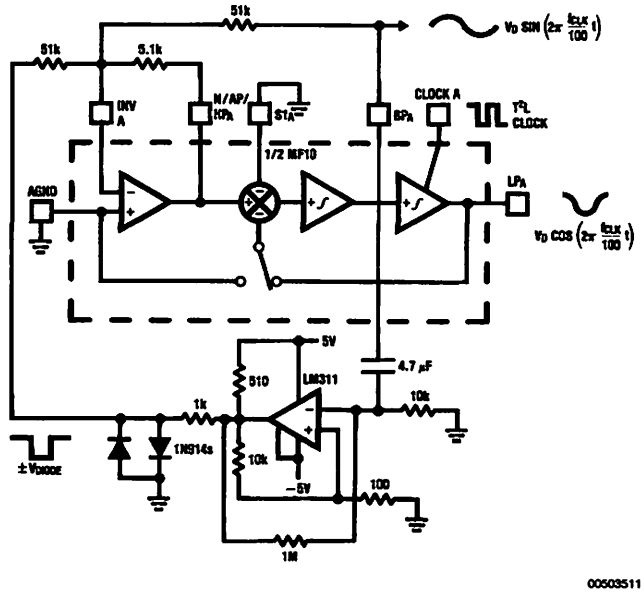


FIGURE 7. Generating Quadrature Sine Waves from a T^2L Clock

Finally, as a graphic illustration of the simplicity of filter implementation using the MF10, *Figure 8* is a complete 300 baud, full-duplex modem filter. The filter is an 8th order, 1 dB ripple Chebyshev bandpass which functions as both an 1170 Hz originate filter and a 2125 Hz answer filter. Control of answer or originate operation is set by the logic level at the 50/100/CL input so that only one clock frequency is required. The overall filter gain is 22 dB.

Construction of this filter on a printed circuit board would obviously be more compact than an RC active filter approach and much more cost effective for the level of precision required. An even more attractive implementation from a space savings point of view would be a hybrid circuit approach. A film resistor array connecting to two MF10 die could produce the entire filter in one package requiring only 7 external connections for input, output, supplies, etc.

ome Specific Application Examples (Continued)

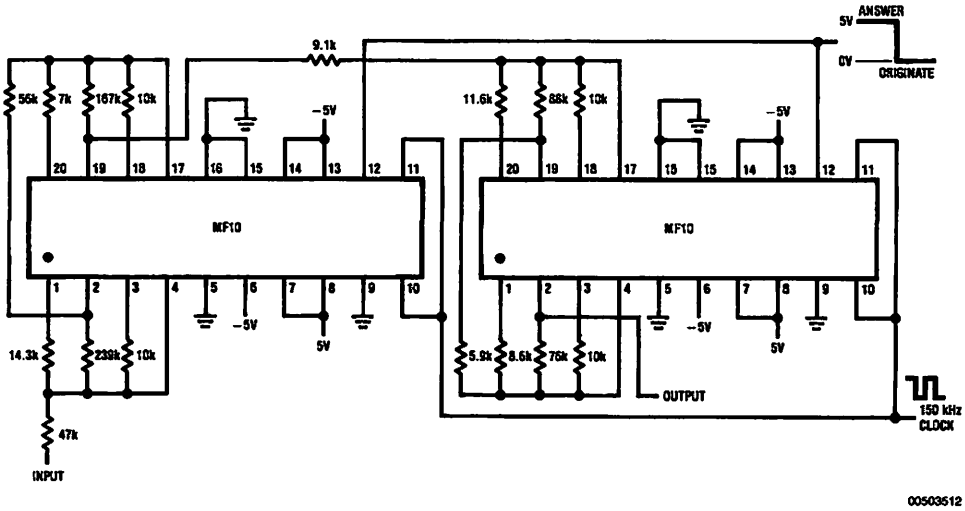


FIGURE 8. A Complete Full-Duplex 300 Baud Modem Filter

he Switched Capacitor Intergrator—How it Works

ıe most important feature of the MF10 is that it requires no ternal capacitors, yet can implement filters over a wide nge of frequencies. A clock is used to control the time nstant of two non-inverting integrators. To feel comfortable th the operation of the MF10, it is important to understand w this control is accomplished.

s easiest to discuss an inverting integrator (*Figure 9*) and w its input resistor can be replaced by 2 switches and a pacitor (*Figure 10*). In *Figure 9* the current which flows ough feedback capacitor C is equal to V_{IN}/R and the cuit time constant is RC. This time constant accuracy pends on the absolute accuracy of two completely differ- t discrete components. In *Figure 10*, switches S1 and S2 ɜ alternately closed by the clock. When switch S1 is closed 2 is opened), capacitor C1 charges up to V_{IN}. At the end of lf a clock period, the charge on C1 (QC1) is equal to ɣxC1. When the clock changes state, S1 opens and S2 ıses. During this half of the clock period all of the charge C1 gets transferred to the feedback capacitor C2.

e amount of charge transferred from the input, V_{IN}, to the mming junction [the (−) input] of the op amp during one mplete clock period is $V_{IN}C1$. Recall that electrical current defined as the amount of charge that passes through a nduction path during a specific time interval (1 ampere=1 ulomb per second). For this circuit, the current which flows ough C2 to the output is:

$$I=\frac{\Delta Q}{\Delta t}=\frac{V_{IN}C1}{T}=V_{IN}C1\,f_{CLK}$$

where T is equal to the clock period.

The effective resistance from V_{IN} to the (−) input is therefore:

$$R=\frac{V_{IN}}{1}=\frac{1}{C1\,f_{CLK}}$$

This means that S1, S2 and C1, when clocked in *Figure 10*, act the same as the resistor in *Figure 9* to yield a clock tunable time constant of:

$$\tau=\frac{C2}{C1\,f_{CLK}}$$

Note that the time constant of the switched capacitor integrator is dependent on a *ratio* of two capacitor values, which, when fabricated on the same die, is very easy to control. This can provide precise filter resonant frequency control both from part to part and with changes in temperature.

The actual integrators used in the MF10 are non-inverting, requiring a slightly more elegant switching scheme, as shown in *Figure 11*. In this circuit, $S1_A$ and $S1_B$ are closed together to charge C1 to V_{IN}. Then $S2_A$ and $S2_B$ are closed to connect C1 to the summing junction with the capacitor plates reversed, to provide the non-inverting operation. If V_{IN} is positive, V_{OUT} will move positive as C2 acquires the charge from C1.

The Switched Capacitor Intergrator—How it Works (Continued)

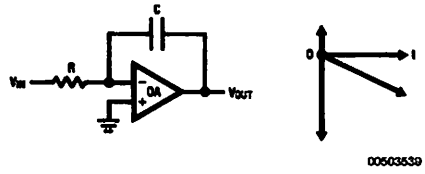


FIGURE 9.

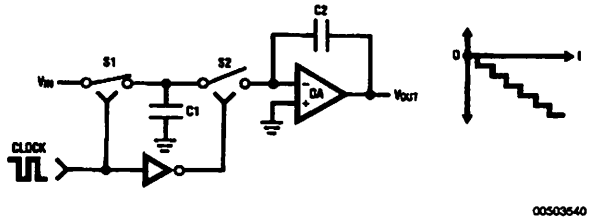


FIGURE 10.

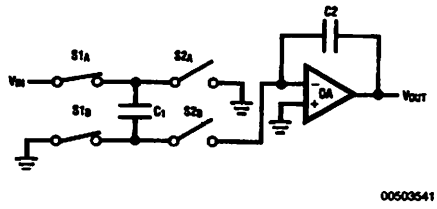


FIGURE 11. The Non-Inverting Integrator Used in the MF10

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor
Americas Customer Support Center
Email: new.feedback@nsc.com
Tel: 1-800-272-9959

www.national.com

National Semiconductor
Europe Customer Support Center
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor
Asia Pacific Customer Support Center
Fax: 65-6250 4466
Email: ap.support@nsc.com
Tel: 65-6254 4466

National Semiconductor
Japan Customer Support Center
Fax: 81-3-5639-7507
Email: nsj.crc@jksmtp.nsc.com
Tel: 81-3-5639-7560

National Semiconductor

May 2001

MF10
Universal Monolithic Dual Switched Capacitor Filter

General Description

The MF10 consists of 2 independent and extremely easy to use, general purpose CMOS active filter building blocks. Each block, together with an external clock and 3 to 4 resistors, can produce various 2nd order functions. Each building block has 3 output pins. One of the outputs can be configured to perform either an allpass, highpass or a notch function; the remaining 2 output pins perform lowpass and bandpass functions. The center frequency of the lowpass and bandpass 2nd order functions can be either directly dependent on the clock frequency, or they can depend on both clock frequency and external resistor ratios. The center frequency of the notch and allpass functions is directly dependent on the clock frequency, while the highpass center frequency depends on both resistor ratio and clock. Up to 4th order functions can be performed by cascading the two 2nd order building blocks of the MF10; higher than 4th order functions can be obtained by cascading MF10 packages.

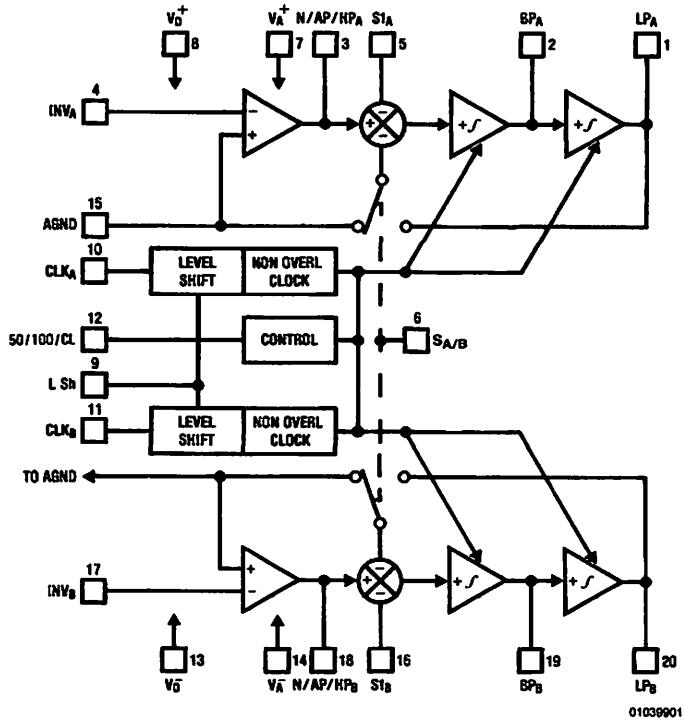
Any of the classical filter configurations (such as Butterworth, Bessel, Cauer and Chebyshev) can be formed.

For pin-compatible device with improved performance refer to LMF100 datasheet.

Features

- Easy to use
- Clock to center frequency ratio accuracy ±0.6%
- Filter cutoff frequency stability directly dependent on external clock quality
- Low sensitivity to external component variation
- Separate highpass (or notch or allpass), bandpass, lowpass outputs
- f_O x Q range up to 200 kHz
- Operation up to 30 kHz
- 20-pin 0.3" wide Dual-In-Line package
- 20-pin Surface Mount (SO) wide-body package

System Block Diagram



Package in 20 pin molded wide body surface mount and 20 pin molded DIP.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage ($V^+ - V^-$)	14V
Voltage at Any Pin	$V^+ + 0.3V$
	$V^- - 0.3V$
Input Current at Any Pin (Note 2)	5 mA
Package Input Current (Note 2)	20 mA
Power Dissipation (Note 3)	500 mW
Storage Temperature	150°C
ESD Susceptability (Note 11)	2000V
Soldering Information	
N Package: 10 sec	260°C
SO Package:	
Vapor Phase (60 Sec.)	215°C
Infrared (15 Sec.)	220°C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" (Appendix D) for other methods of soldering surface mount devices.

Operating Ratings (Note 1)

Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
MF10ACN, MF10CCN	$0°C \le T_A \le 70°C$
MF10CCWM	$0°C \le T_A \le 70°C$

Electrical Characteristics

$V^+ = +5.00V$ and $V^- = -5.00V$ unless otherwise specified. **Boldface limits apply for T_{MIN} to T_{MAX};** all other limits $T_A = T_J = 25°C$.

Symbol	Parameter		Conditions		MF10ACN, MF10CCN, MF10CCWM			Units
					Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
$V^+ - V^-$	Supply Voltage	Min					**9**	V
		Max					**14**	V
I_S	Maximum Supply Current		Clock Applied to Pins 10 & 11 No Input Signal		8	12	**12**	mA
f_O	Center Frequency Range	Min	$f_O \times Q < 200$ kHz		0.1		**0.2**	Hz
		Max			30		**20**	kHz
f_{CLK}	Clock Frequency Range	Min			5.0		**10**	Hz
		Max			1.5		**1.0**	MHz
f_{CLK}/f_O	50:1 Clock to Center Frequency Ratio Deviation	MF10A	Q = 10 Mode 1	V_{pin12} = 5V f_{CLK} = 250 KHz	±0.2	±0.6	**±0.6**	%
		MF10C			±0.2	±1.5	**±1.5**	%
f_{CLK}/f_O	100:1 Clock to Center Frequency Ratio Deviation	MF10A	Q = 10 Mode 1	V_{pin12} = 0V f_{CLK} = 500 kHz	±0.2	±0.6	**±0.6**	%
		MF10C			±0.2	±1.5	**±1.5**	%
	Clock Feedthrough		Q = 10 Mode 1		10			mV
	Q Error (MAX) (Note 4)		Q = 10 Mode 1	V_{pin12} = 5V f_{CLK} = 250 kHz	±2	±6	**±6**	%
				V_{pin12} = 0V f_{CLK} = 500 kHz	±2	±6	**±6**	%
H_{OLP}	DC Lowpass Gain		Mode 1 R1 = R2 = 10k		0	±0.2	**±0.2**	dB
V_{OS1}	DC Offset Voltage (Note 5)				±5.0	±20	**±20**	mV
V_{OS2}	DC Offset Voltage	Min	V_{pin12} = +5V	$S_{A/B} = V^+$	−150	−185	**−185**	mV

lectrical Characteristics (Continued)

$^+$ = +5.00V and V^- = −5.00V unless otherwise specified. **Boldface limits apply for T_{MIN} to T_{MAX}**; all other limits $T_A = T_J$ =
5°C.

nbol	Parameter		Conditions		MF10ACN, MF10CCN, MF10CCWM			Units
					Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
	(Note 5)	Max	(f_{CLK}/f_O = 50)			−85	−85	
		Min	V_{pin12} = +5V	$S_{A/B}$ = V^-	−70			mV
		Max	(f_{CLK}/f_O = 50)					
:3	DC Offset Voltage	Min	V_{pin12} = +5V	All Modes	−70	−100	−100	mV
	(Note 5)	Max	(f_{CLK}/f_O = 50)			−20	−20	
:2	DC Offset Voltage (Note 5)		V_{pin12} = 0V (f_{CLK}/f_O = 100)	$S_{A/B}$ = V^+	−300			mV
			V_{pin12} = 0V (f_{CLK}/f_O = 100)	$S_{A/B}$ = V^-	−140			mV
:3	DC Offset Voltage (Note 5)		V_{pin12} = 0V (f_{CLK}/f_O = 100)	All Modes	−140			mV
T	Minimum Output Voltage Swing	BP, LP Pins	R_L = 5k		±4.25	±3.8	±3.8	V
		N/AP/HP Pin	R_L = 3.5k		±4.25	±3.8	±3.8	V
W	Op Amp Gain BW Product				2.5			MHz
	Op Amp Slew Rate				7			V/µs
	Dynamic Range(Note 6)		V_{pin12} = +5V (f_{CLK}/f_O = 50)		83			dB
			V_{pin12} = 0V (f_{CLK}/f_O = 100)		80			dB
	Maximum Output Short Circuit Current (Note 7)	Source			20			mA
		Sink			3.0			mA

ogic Input Characteristics

oldface limits apply for T_{MIN} to T_{MAX}; all other limits $T_A = T_J$ = 25°C

Parameter		Conditions	MF10ACN, MF10CCN, MF10CCWM			Units
			Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
MOS Clock	Min Logical "1"	V^+ = +5V, V^- = −5V,		+3.0	+3.0	V
ut Voltage	Max Logical "0"	V_{LSh} = 0V		−3.0	−3.0	V
	Min Logical "1"	V^+ = +10V, V^- = 0V,		+8.0	+8.0	V
	Max Logical "0"	V_{LSh} = +5V		+2.0	+2.0	V
L Clock	Min Logical "1"	V^+ = +5V, V^- = −5V,		+2.0	+2.0	V
ut Voltage	Max Logical "0"	V_{LSh} = 0V		+0.8	+0.8	V

Logic Input Characteristics (Continued)

Boldface limits apply for T_{MIN} to T_{MAX}; all other limits $T_A = T_J = 25°C$

Parameter		Conditions	MF10ACN, MF10CCN, MF10CCWM			Units
			Typical (Note 8)	Tested Limit (Note 9)	Design Limit (Note 10)	
	Min Logical "1"	$V^+ = +10V$, $V^- = 0V$,		+2.0	**+2.0**	V
	Max Logical "0"	$V_{LSh} = 0V$		+0.8	**+0.8**	V

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: When the input voltage (V_{IN}) at any pin exceeds the power supply rails ($V_{IN} < V^-$ or $V_{IN} > V^+$) the absolute value of current at that pin should be limited to 5 mA or less. The 20 mA package input current limits the number of pins that can exceed the power supply boundaries with a 5 mA current limit to four.

Note 3: The maximum power dissipation must be derated at elevated temperatures and is dictated by T_{JMAX}, θ_{JA}, and the ambient temperature, T_A. The maximum allowable power dissipation at any temperature is $P_D = (T_{JMAX} - T_A)/\theta_{JA}$ or the number given in the Absolute Maximum Ratings, whichever is lower. For this device, $T_{JMAX} = 125°C$, and the typical junction-to-ambient thermal resistance of the MF10ACN/CCN when board mounted is 55°C/W. For the MF10AJ/CCJ, this number increases to 95°C/W and for the MF10ACWM/CCWM this number is 66°C/W.

Note 4: The accuracy of the Q value is a function of the center frequency (f_O). This is illustrated in the curves under the heading "Typical Performance Characteristics".

Note 5: V_{OS1}, V_{OS2}, and V_{OS3} refer to the internal offsets as discussed in the Applications Information Section 3.4.

Note 6: For ±5V supplies the dynamic range is referenced to 2.82V rms (4V peak) where the wideband noise over a 20 kHz bandwidth is typically 200 µV rms for the MF10 with a 50:1 CLK ratio and 280 µV rms for the MF10 with a 100:1 CLK ratio.

Note 7: The short circuit source current is measured by forcing the output that is being tested to its maximum positive voltage swing and then shorting that output to the negative supply. The short circuit sink current is measured by forcing the output that is being tested to its maximum negative voltage swing and then shorting that output to the positive supply. These are the worst case conditions.

Note 8: Typicals are at 25°C and represent most likely parametric norm.

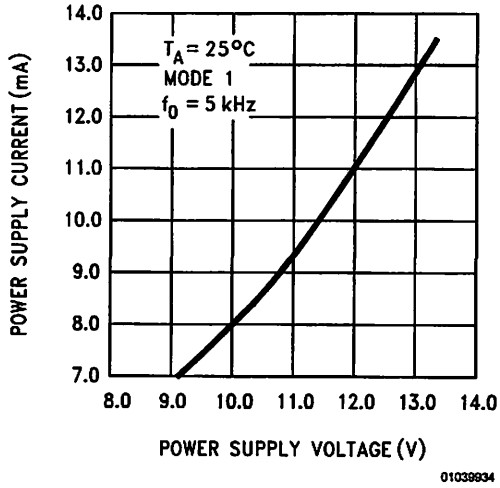
Note 9: Tested limits are guaranteed to National's AOQL (Average Outgoing Quality Level).

Note 10: Design limits are guaranteed but not 100% tested. These limits are not used to calculate outgoing quality levels.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

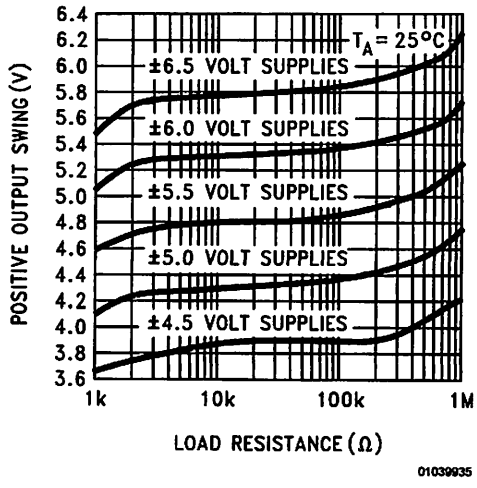
Typical Performance Characteristics

Power Supply Current vs. Power Supply Voltage



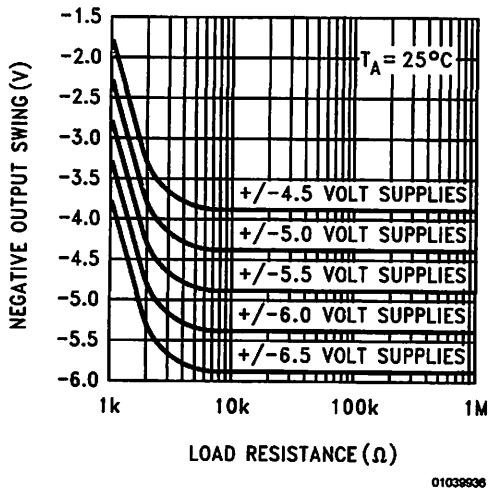
01039934

Positive Output Voltage Swing vs. Load Resistance (N/AP/HP Output)



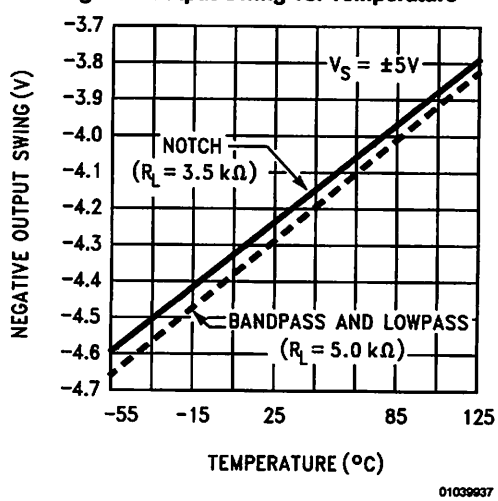
01039935

Negative Output Voltage Swing vs. Load Resistance (N/AP/HP Output)



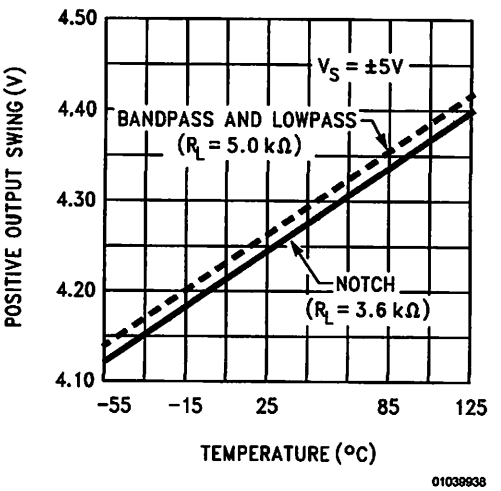
01039936

Negative Output Swing vs. Temperature



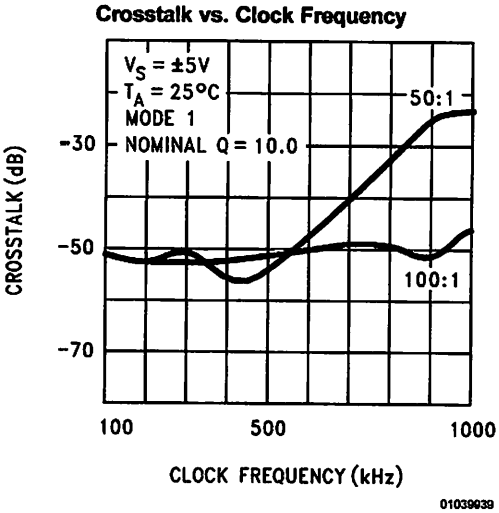
01039937

Positive Output Swing vs. Temperature



01039938

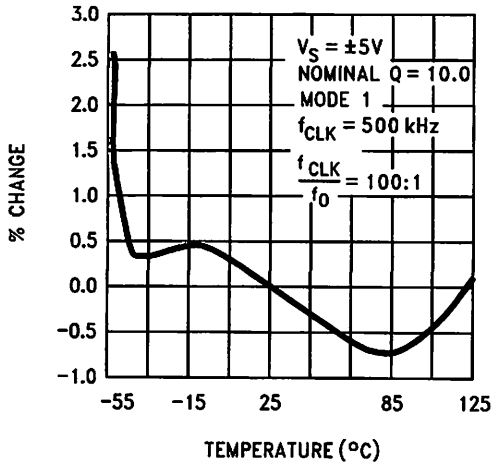
Crosstalk vs. Clock Frequency



01039939

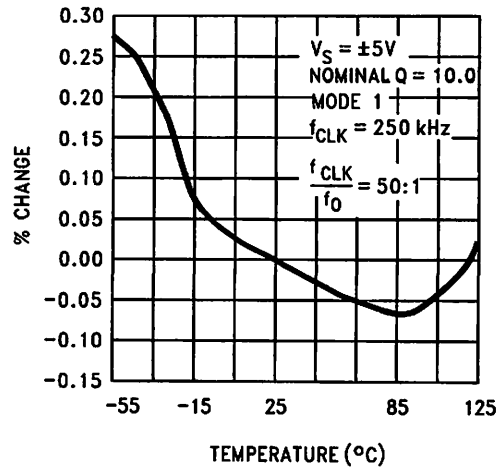
Typical Performance Characteristics (Continued)

Q Deviation vs. Temperature



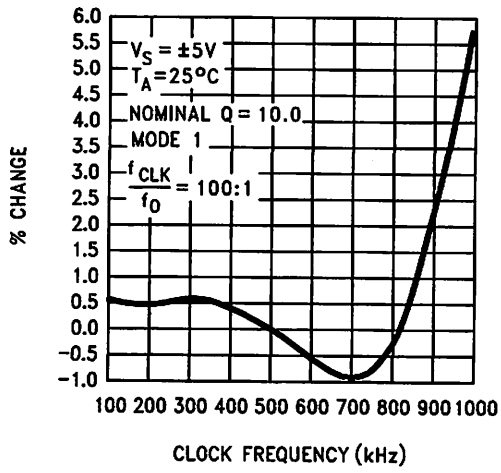
01039940

Q Deviation vs. Temperature



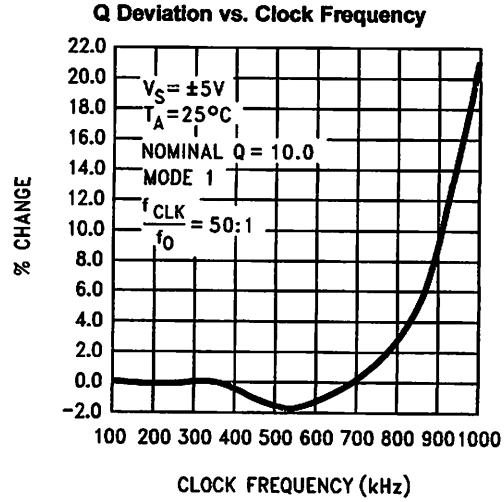
01039941

Q Deviation vs. Clock Frequency



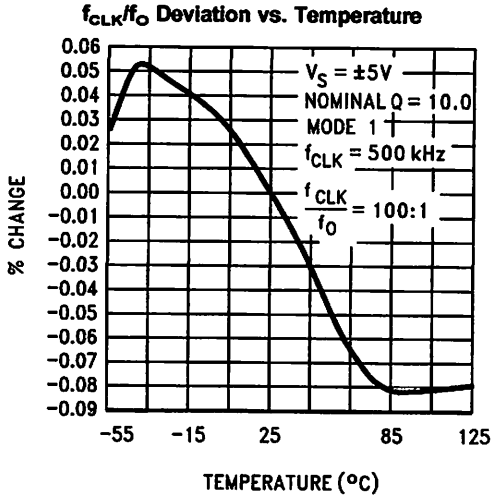
01039942

Q Deviation vs. Clock Frequency



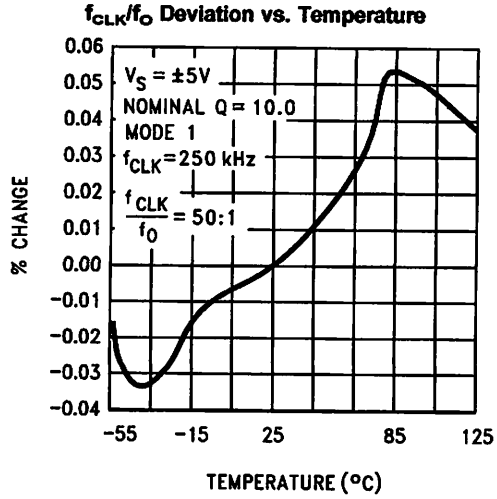
01039943

f_{CLK}/f_O Deviation vs. Temperature



01039944

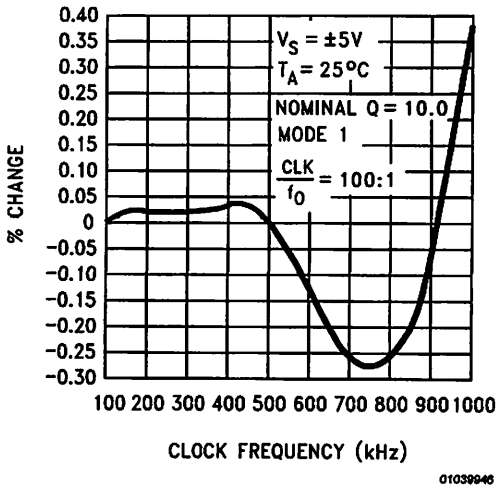
f_{CLK}/f_O Deviation vs. Temperature



01039945

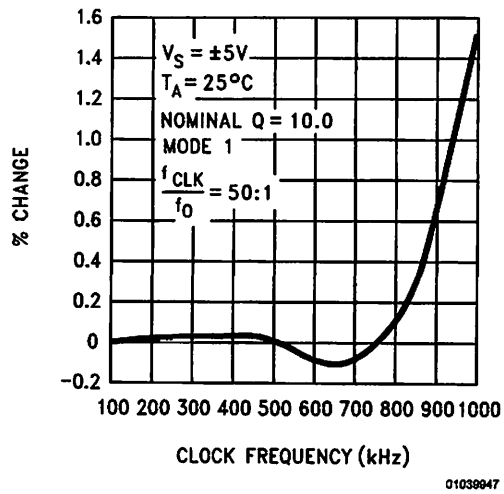
Typical Performance Characteristics (Continued)

f_{CLK}/f_O Deviation vs. Clock Frequency



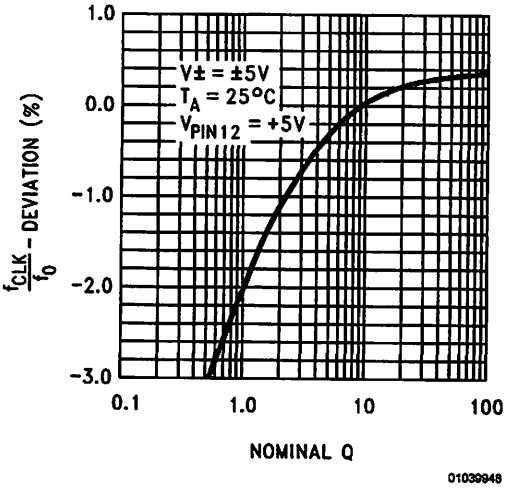
01039946

f_{CLK}/f_O Deviation vs. Clock Frequency



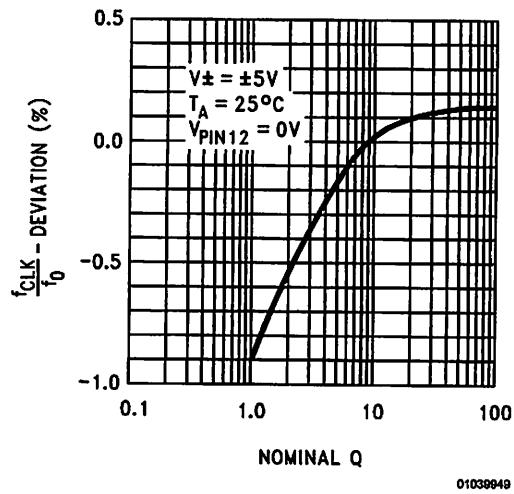
01039947

Deviation of f_{CLK}/f_O vs. Nominal Q



01039948

Deviation of f_{CLK}/f_O vs. Nominal Q



01039949

Pin Descriptions

LP(1,20), BP(2,19), N/AP/HP(3,18)
The second order lowpass, bandpass and notch/allpass/highpass outputs. These outputs can typically sink 1.5 mA and source 3 mA. Each output typically swings to within 1V of each supply.

INV(4,17)
The inverting input of the summing op-amp of each filter. These are high impedance inputs, but the non-inverting input is internally tied to AGND, making INV_A and INV_B behave like summing junctions (low impedance, current inputs).

S1(5,16)
S1 is a signal input pin used in the allpass filter configurations (see modes 4 and 5). The pin should be driven with a source impedance of less than 1 kΩ. If S1 is not driven with a signal it should be tied to AGND (mid-supply).

$S_{A/B}$(6)
This pin activates a switch that connects one of the inputs of each filter's second summer to either AGND ($S_{A/B}$ tied to V^-) or to the lowpass (LP) output ($S_{A/B}$ tied to V^+). This offers the flexibility needed for configuring the filter in its various modes of operation.

V_A^+(7),V_D^+(8)
Analog positive supply and digital positive supply. These pins are internally connected through the IC substrate and therefore V_A^+ and V_D^+ should be derived from the same power supply source. They have been brought out separately so they can be bypassed by separate capacitors, if desired. They can be externally tied together and bypassed by a single capacitor.

V_A^-(14), V_D^-(13)
Analog and digital negative supplies. The same comments as for V_A^+ and V_D^+ apply here.

LSh(9)
Level shift pin; it accommodates various clock levels with dual or single supply operation. With dual ±5V supplies, the MF10 can be driven with CMOS clock levels (±5V) and the LSh pin should be tied to the system ground. If the same supplies as above are used but only TTL clock levels, derived from 0V to +5V supply, are available, the LSh pin should be tied to the system ground. For single supply operation (0V and +10V) the V_A^-, V_D^- pins should be connected to the system ground, the AGND pin should be biased at +5V and the LSh pin should also be tied to the system ground for TTL clock levels. LSh should be biased at +5V for CMOS clock levels in 10V single-supply applications.

CLKA(10), CLKB(11)
Clock inputs for each switched capacitor filter building block. They should both be of the same level (TTL or CMOS). The level shift (LSh) pin description discusses how to accommodate their levels. The duty cycle of the clock should be close to 50% especially when clock frequencies above 200 kHz are used. This allows the maximum time for the internal op-amps to settle, which yields optimum filter operation.

50/100/CL(12)
By tying this pin high a 50:1 clock-to-filter-center-frequency ratio is obtained. Tying this pin at mid-supplies (i.e. analog ground with dual supplies) allows the filter to operate at a 100:1 clock-to-center-frequency ratio. When the pin is tied low (i.e., negative supply with dual supplies), a simple current limiting circuit is triggered to limit the overall supply current down to about 2.5 mA. The filtering action is then aborted.

AGND(15)
This is the analog ground pin. This pin should be connected to the system ground for dual supply operation or biased to mid-supply for single supply operation. For a further discussion of mid-supply biasing techniques see the Applications Information (Section 3.2). For optimum filter performance a "clean" ground must be provided.

1.0 Definition of Terms

f_{CLK}: the frequency of the external clock signal applied to pin 10 or 11.

f_O: center frequency of the second order function complex pole pair. f_O is measured at the bandpass outputs of the MF10, and is the frequency of maximum bandpass gain. (*Figure 1*)

f_{notch}: the frequency of minimum (ideally zero) gain at the notch outputs.

f_z: the center frequency of the second order complex zero pair, if any. If f_z is different from f_O and if Q_Z is high, it can be observed as the frequency of a notch at the allpass output. (*Figure 10*)

Q: "quality factor" of the 2nd order filter. Q is measured at the bandpass outputs of the MF10 and is equal to f_O divided by the −3 dB bandwidth of the 2nd order bandpass filter (*Figure 1*). The value of Q determines the shape of the 2nd order filter responses as shown in *Figure 6*.

Q_Z: the quality factor of the second order complex zero pair, if any. Q_Z is related to the allpass characteristic, which is written:

$$H_{AP}(s) = \frac{H_{OAP}\left(s^2 - \frac{s\omega_O}{Q_Z} + \omega_O^2\right)}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

where $Q_Z = Q$ for an all-pass response.

H_{OBP}: the gain (in V/V) of the bandpass output at $f = f_O$.

.0 Definition of Terms (Continued)

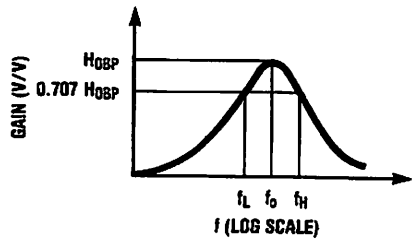
H_{OLP}: the gain (in V/V) of the lowpass output as f → 0 Hz (*Figure 2*).

H_{OHP}: the gain (in V/V) of the highpass output as f → $f_{CLK}/2$ (*Figure 3*).

H_{ON}: the gain (in V/V) of the notch output as f → 0 Hz and as f → $f_{CLK}/2$, when the notch filter has equal gain above and below the center frequency (*Figure 4*). When the low-frequency gain differs from the high-frequency gain, as in modes 2 and 3a (*Figure 11* and *Figure 8*), the two quantities below are used in place of H_{ON}.

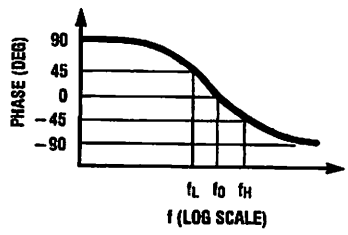
H_{ON1}: the gain (in V/V) of the notch output as f → 0 Hz.

H_{ON2}: the gain (in V/V) of the notch output as f → $f_{CLK}/2$.



01039905

(a)



01039906

(b)

$$H_{BP}(s) = \frac{H_{OBP}\frac{\omega O}{Q}S}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

$$Q = \frac{f_O}{f_H - f_L};\ f_O = \sqrt{f_L f_H}$$

$$f_L = f_O\left(\frac{-1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right)$$

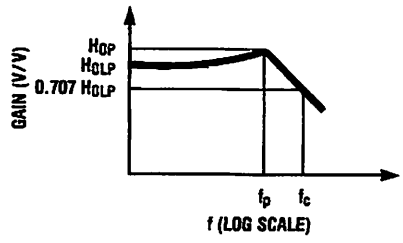
$$f_H = f_O\left(\frac{1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right)$$

$$\omega_O = 2\pi f_O$$

01039956

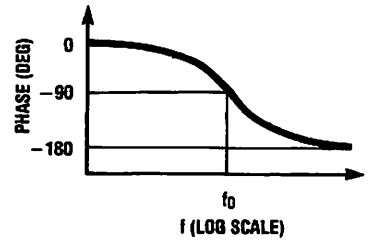
FIGURE 1. 2nd-Order Bandpass Response

1.0 Definition of Terms (Continued)



01039907

(a)



01039908

(b)

$$H_{LP}(s) = \frac{H_{OLP}\omega_O^2}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

$$f_C = f_O \times \sqrt{\left(1 - \frac{1}{2Q^2}\right) + \sqrt{\left(1 - \frac{1}{2Q^2}\right)^2 + 1}}$$

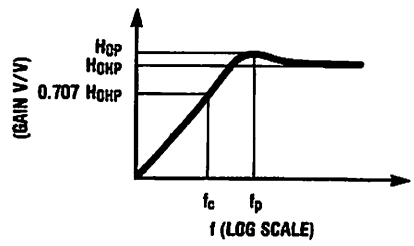
$$f_p = f_O \sqrt{1 - \frac{1}{2Q^2}}$$

$$H_{OP} = H_{OLP} \times \frac{1}{\frac{1}{Q}\sqrt{1 - \frac{1}{4Q^2}}}$$

01039957

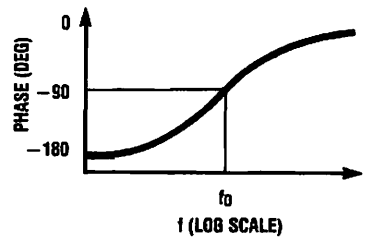
FIGURE 2. 2nd-Order Low-Pass Response

.0 Definition of Terms (Continued)



01039909

(a)



01039910

(b)

$$H_{HP}(s) = \frac{H_{OHP}s^2}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

$$f_C = f_O \times \left[\sqrt{\left(1 - \frac{1}{2Q^2}\right) + \sqrt{\left(1 - \frac{1}{2Q^2}\right)^2 + 1}}\right]^{-1}$$

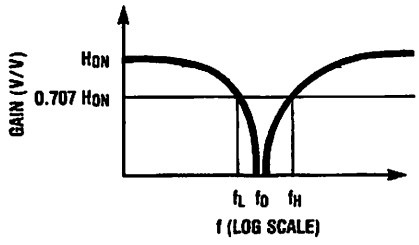
$$f_P = f_O \times \left[\sqrt{1 - \frac{1}{2Q^2}}\right]^{-1}$$

$$H_{OP} = H_{OHP} \times \frac{1}{\frac{1}{Q}\sqrt{1 - \frac{1}{4Q^2}}}$$

01039958

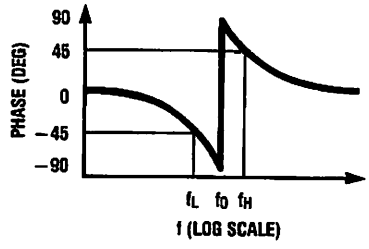
FIGURE 3. 2nd-Order High-Pass Response

1.0 Definition of Terms (Continued)



01039911

(a)



01039912

(b)

$$H_N(s) = \frac{H_{ON}(s^2 + \omega_O^2)}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

$$Q = \frac{f_O}{f_H - f_L};\ f_O = \sqrt{f_L f_H}$$

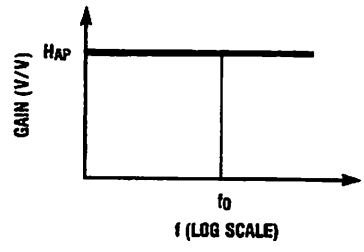
$$f_L = f_O\left(\frac{-1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right)$$

$$f_H = f_O\left(\frac{1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right)$$

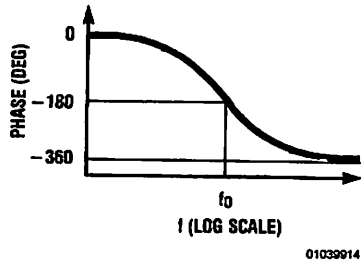
01039960

FIGURE 4. 2nd-Order Notch Response

.0 Definition of Terms (Continued)



(a)



(b)

$$H_{AP}(s) = \frac{H_{OAP}\left(s^2 - \frac{s\omega_O}{Q} + \omega_O^2\right)}{s^2 + \frac{s\omega_O}{Q} + \omega_O^2}$$

01039961

FIGURE 5. 2nd-Order All-Pass Response

1.0 Definition of Terms (Continued)

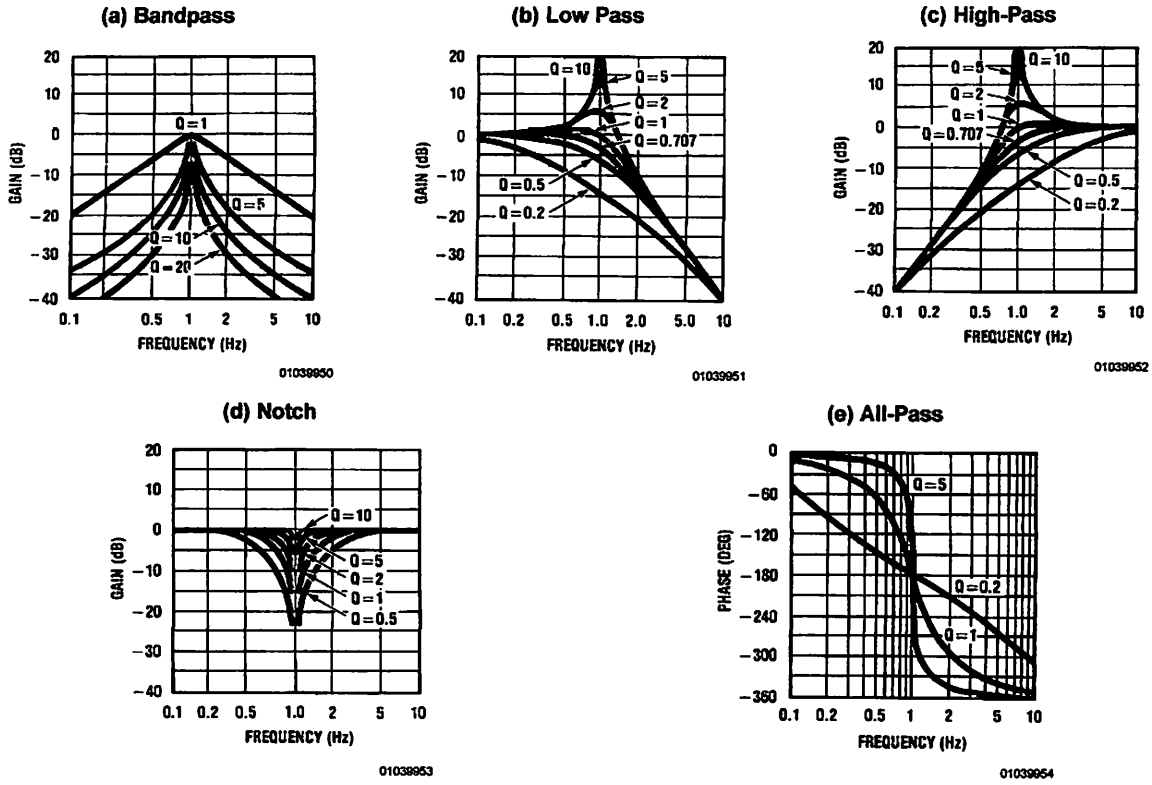


FIGURE 6. Response of various 2nd-order filters as a function of Q. Gains and center frequencies are normalized to unity.

2.0 Modes of Operation

The MF10 is a switched capacitor (sampled data) filter. To fully describe its transfer functions, a time domain approach is appropriate. Since this is cumbersome, and since the MF10 closely approximates continuous filters, the following discussion is based on the well known frequency domain. Each MF10 can produce a full 2nd order function. See *Table 1* for a summary of the characteristics of the various modes.

MODE 1: Notch 1, Bandpass, Lowpass Outputs:

$f_{notch} = f_O$ **(See *Figure 7*)**

f_O= center frequency of the complex pole pair

$$= \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

f_{notch}= center frequency of the imaginary zero pair = f_O.

$$H_{OLP} = \text{Lowpass gain (as } f \rightarrow 0) = -\frac{R2}{R1}$$

$$H_{OBP} = \text{Bandpass gain (at } f = f_O) = -\frac{R3}{R1}$$

$$H_{ON} = \text{Notch output gain as } \left.\begin{matrix} f \rightarrow 0 \\ f \rightarrow f_{CLK}/2 \end{matrix}\right\} = \frac{-R_2}{R_1}$$

$$Q = \frac{f_O}{BW} = \frac{R3}{R2}$$

= quality factor of the complex pole pair

BW = the −3 dB bandwidth of the bandpass output.

Circuit dynamics:

$$H_{OLP} = \frac{H_{OBP}}{Q} \text{ or } H_{OBP} = H_{OLP} \times Q$$

$$= H_{ON} \times Q.$$

$H_{OLP(peak)} \cong Q \times H_{OLP}$ (for high Q's)

MODE 1a: Non-Inverting BP, LP (See *Figure 8*)

$$f_O = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

$$Q = \frac{R3}{R2}$$

$H_{OLP} = -1$; $H_{OLP(peak)} \cong Q \times H_{OLP}$ (for high Q's)

$$H_{OBP_1} = -\frac{R3}{R2}$$

H_{OBP_2}= 1 (Non-Inverting)

Circuit Dynamics: $H_{OBP1} = Q$

Note: V_{IN} should be driven from a low impedance (<1 kΩ) source.

.0 Modes of Operation (Continued)

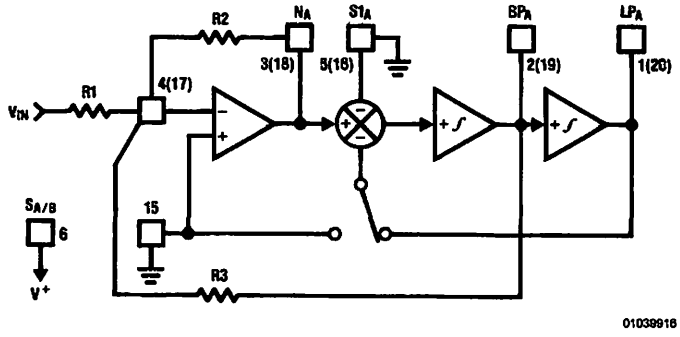


FIGURE 7. MODE 1

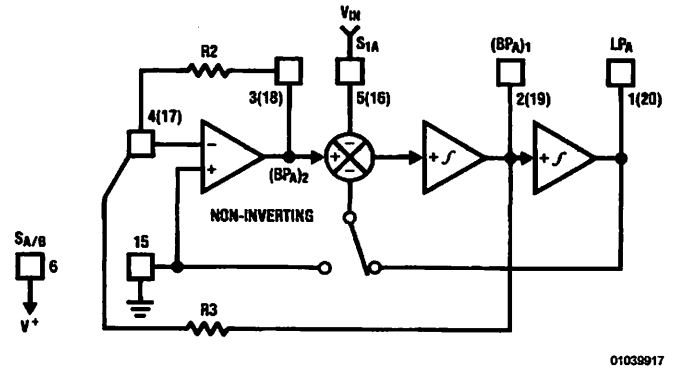


FIGURE 8. MODE 1a

)DE 2: Notch 2, Bandpass, Lowpass: $f_{notch} < f_o$
ee *Figure 9*)

$$)\quad = \text{center frequency}$$

$$= \frac{f_{CLK}}{100}\sqrt{\frac{R2}{R4}+1} \text{ or } \frac{f_{CLK}}{50}\sqrt{\frac{R2}{R4}+1}$$

$$_{notch} = \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

$$)\quad = \text{quality factor of the complex pole pair}$$

$$= \frac{\sqrt{R2/R4+1}}{R2/R3}$$

$$_{OLP} = \text{Lowpass output gain (as } f \rightarrow 0)$$

$$= -\frac{R2/R1}{R2/R4+1}$$

$$_{OBP} = \text{Bandpass output gain (at } f = f_O) = -R3/R1$$

$$_{ON_1} = \text{Notch output gain (as } f \rightarrow 0)$$

$$= -\frac{R2/R1}{R2/R4+1}$$

$$_{ON_2} = \text{Notch output gain}\left(\text{as } f \rightarrow \frac{f_{CLK}}{2}\right) = -R2/R1$$

$$\text{ilter dynamics: } H_{OBP} = Q\sqrt{H_{OLP}\,H_{ON_2}} = \sqrt{H_{ON1}\,H_{ON2}}$$

MODE 3: Highpass, Bandpass, Lowpass Outputs (See *Figure 10*)

$$f_O = \frac{f_{CLK}}{100}\times\sqrt{\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50}\times\sqrt{\frac{R2}{R4}}$$

$$Q = \text{quality factor of the complex pole pair}$$

$$= \sqrt{\frac{R2}{R4}}\times\frac{R3}{R2}$$

$$H_{OHP} = \text{Highpass Gain}\left(\text{as } f \rightarrow \frac{f_{CLK}}{2}\right) = -\frac{R2}{R1}$$

$$H_{OBP} = \text{Lowpass Gain}\left(\text{at } f = f_O\right) = -\frac{R3}{R1}$$

$$H_{OLP} = \text{Lowpass Gain}\left(\text{as } f \rightarrow 0\right) = -\frac{R4}{R1}$$

$$\text{Circuit dynamics: } \frac{R2}{R4} = \frac{H_{OHP}}{H_{OLP}};$$

$$H_{OBP} = \sqrt{H_{OHP}\times H_{OLP}}\times Q$$

$$H_{OLP(peak)} \cong Q \times H_{OLP} \text{ (for high Q's)}$$

$$H_{OHP(peak)} \cong Q \times H_{OHP} \text{ (for high Q's)}$$

2.0 Modes of Operation (Continued)

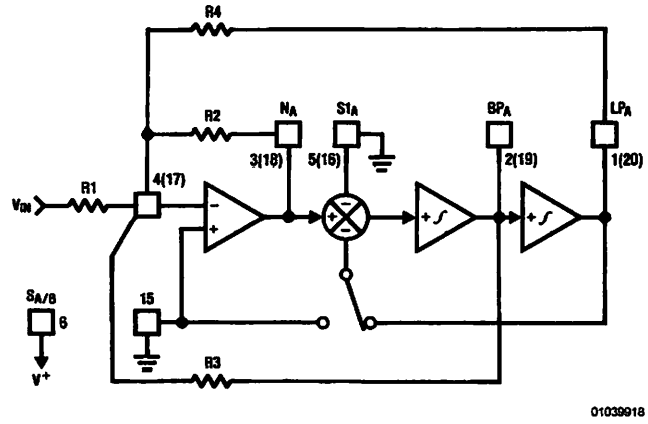
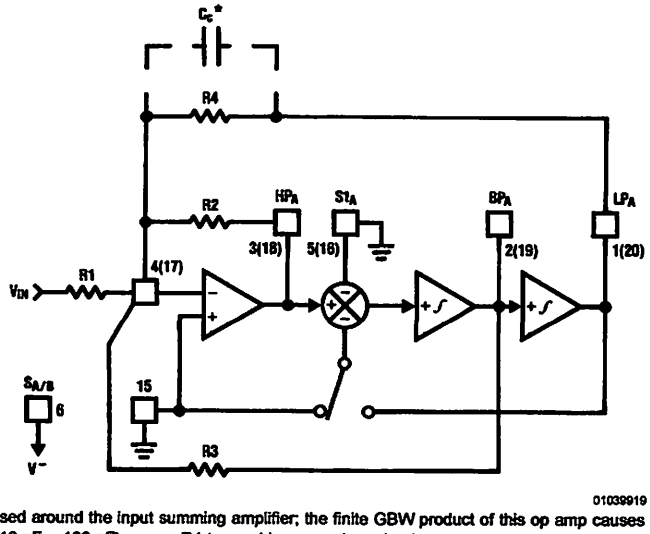


FIGURE 9. MODE 2



*In Mode 3, the feedback loop is closed around the input summing amplifier; the finite GBW product of this op amp causes a slight Q enhancement. If this is a problem, connect a small capacitor (10 pF – 100 pF) across R4 to provide some phase lead.

FIGURE 10. MODE 3

.0 Modes of Operation (Continued)

)DE 3a: HP, BP, LP and Notch with External Op Amp ee *Figure 11*)

$$f_O = \frac{f_{CLK}}{100} \times \sqrt{\frac{R2}{R4}} \text{ or } \frac{f_{CLK}}{50} \times \sqrt{\frac{R2}{R4}}$$

$$Q = \sqrt{\frac{R2}{R4}} \times \frac{R3}{R2}$$

$$H_{OHP} = -\frac{R2}{R1}$$

$$H_{OBP} = -\frac{R3}{R1}$$

$$H_{OLP} = -\frac{R4}{R1}$$

f_n = notch frequency $= \frac{f_{CLK}}{100}\sqrt{\frac{R_h}{R_l}}$ or $\frac{f_{CLK}}{50}\sqrt{\frac{R_h}{R_l}}$

H_{ON} = gain of notch at

$$f = f_O = \left\| Q\left(\frac{R_g}{R_l} H_{OLP} - \frac{R_g}{R_h} H_{OHP}\right)\right\|$$

H_{n1} = gain of notch (as $f \rightarrow 0$) $= \frac{R_g}{R_l} \times H_{OLP}$

H_{n2} = gain of notch $\left(\text{as } f \rightarrow \frac{f_{CLK}}{2}\right)$

$$= -\frac{R_g}{R_h} \times H_{OHP}$$

MODE 4: Allpass, Bandpass, Lowpass Outputs(See *Figure 12*)

f_O = center frequency

$$= \frac{f_{CLK}}{100} \text{ or } \frac{f_{CLK}}{50}$$

f_z* = center frequency of the complex zero $\approx f_O$

$$Q = \frac{f_O}{BW} = \frac{R3}{R2}$$

Q_Z = quality factor of complex zero pair $= \frac{R3}{R1}$

For AP output make R1 = R2

H_{OAP}* = Allpass gain $\left(\text{at } 0 < f < \frac{f_{CLK}}{2}\right) = -\frac{R2}{R1} = -1$

H_{OLP} = Lowpass gain (as $f \rightarrow 0$)

$$= -\left(\frac{R2}{R1} + 1\right) = -2$$

H_{OBP} = Bandpass gain (at $f = f_O$)

$$= -\frac{R3}{R2}\left(1 + \frac{R2}{R1}\right) = -2\left(\frac{R3}{R2}\right)$$

Circuit Dynamics: $H_{OBP} = (H_{OLP}) \times Q = (H_{OAP} + 1)Q$

*Due to the sampled data nature of the filter, a slight mismatch of f_z and f_O occurs causing a 0.4 dB peaking around f_O of the allpass filter amplitude response (which theoretically should be a straight line). If this is unacceptable, Mode 5 is recommended.

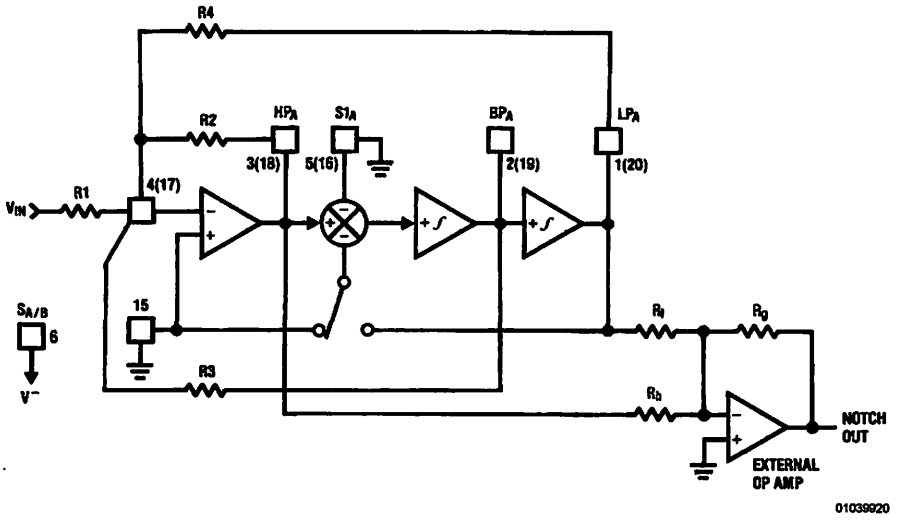


FIGURE 11. MODE 3a

2.0 Modes of Operation (Continued)

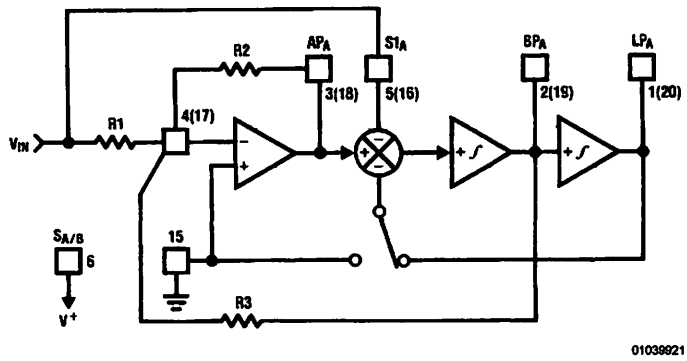


FIGURE 12. MODE 4

MODE 5: Numerator Complex Zeros, BP, LP (See *Figure 13*)

$$f_O = \sqrt{1 + \frac{R2}{R4}} \times \frac{f_{CLK}}{100} \text{ or } \sqrt{1 + \frac{R2}{R4}} \times \frac{f_{CLK}}{50}$$

$$f_Z = \sqrt{1 - \frac{R2}{R4}} \times \frac{f_{CLK}}{100} \text{ or } \sqrt{1 - \frac{R1}{R4}} \times \frac{f_{CLK}}{50}$$

$$Q = \sqrt{1 + R2/R4} \times \frac{R3}{R2}$$

$$Q_Z = \sqrt{1 - R1/R4} \times \frac{R3}{R1}$$

$H_{0_{Z1}}$ = gain at C.Z. output (as f $\longrightarrow$ 0 Hz)

$$= \frac{-R2(R4 - R1)}{R1(R2 + R4)}$$

$H_{0_{Z2}}$ = gain at C.Z. output $\left(\text{as f} \longrightarrow \frac{f_{CLK}}{2}\right) = \frac{-R2}{R1}$

$$H_{OBP} = -\left(\frac{R2}{R1} + 1\right) \times \frac{R3}{R2}$$

$$H_{OLP} = -\left(\frac{R2 + R1}{R2 + R4}\right) \times \frac{R4}{R1}$$

MODE 6a: Single Pole, HP, LP Filter (See *Figure 14*)

f_C = cutoff frequency of LP or HP output

$$= \frac{R2}{R3} \frac{f_{CLK}}{100} \text{ or } \frac{R2}{R3} \frac{f_{CLK}}{50}$$

$$H_{OLP} = -\frac{R3}{R1}$$

$$H_{OHP} = -\frac{R2}{R1}$$

MODE 6b: Single Pole LP Filter (Inverting and Non-Inverting) (See *Figure 15*)

f_C = cutoff frequency of LP outputs

$$\cong \frac{R2}{R3} \frac{f_{CLK}}{100} \text{ or } \frac{R2}{R3} \frac{f_{CLK}}{50}$$

H_{OLP1} = 1 (non-inverting)

$$H_{OLP2} = -\frac{R3}{R2}$$

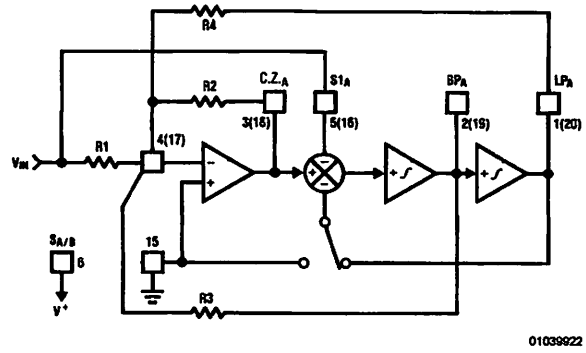
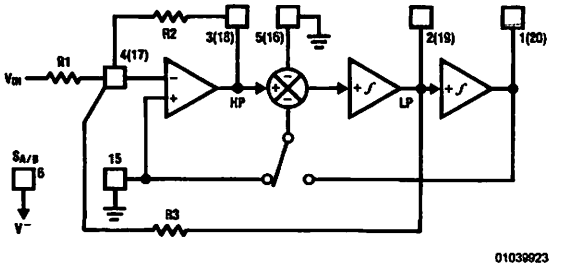


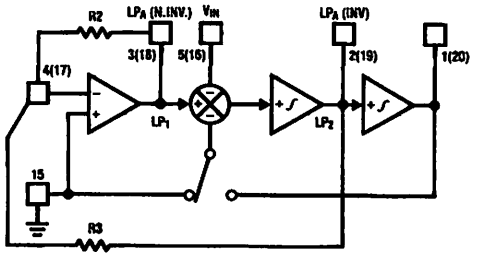
FIGURE 13. MODE 5

0 Modes of Operation (Continued)



01039923

FIGURE 14. MODE 6a



01039924

FIGURE 15. MODE 6b

TABLE 1. Summary of Modes. Realizable filter types (e.g. low-pass) denoted by asterisks. Unless otherwise noted, gains of various filter outputs are inverting and adjustable by resistor ratios.

e	BP	LP	HP	N	AP	Number of Resistors	Adjustable f_{CLK}/f_O	Notes
	*	*		*		3	No	
	(2) $H_{OBP1} = -Q$ $H_{OBP2} = +1$	$H_{OLP} + 1$				2	No	May need input buffer. Poor dynamics for high Q.
	*	*		*		3	Yes (above $f_{CLK}/50$ or $f_{CLK}/100$)	
	*	*	*			4	Yes	Universal State-Variable Filter. Best general-purpose mode.
	*	*	*	*		7	Yes	As above, but also includes resistor-tuneable notch.
	*	*			*	3	No	Gives Allpass response with $H_{OAP} = -1$ and $H_{OLP} = -2$.
	*	*			*	4		Gives flatter allpass response than above if $R_1 = R_2 = 0.02R_4$.
		*	*			3		Single pole.
		(2) $H_{OLP1} = +1$ $H_{OLP2} = \frac{-R3}{R2}$				2		Single pole.

3.0 Applications Information

The MF10 is a general-purpose dual second-order state variable filter whose center frequency is proportional to the frequency of the square wave applied to the clock input (f_{CLK}). By connecting pin 12 to the appropriate DC voltage, the filter center frequency f_O can be made equal to either $f_{CLK}/100$ or $f_{CLK}/50$. f_O can be very accurately set (within ±6%) by using a crystal clock oscillator, or can be easily varied over a wide frequency range by adjusting the clock frequency. If desired, the f_{CLK}/f_O ratio can be altered by external resistors as in *Figures 9, 10, 11, 13, 14, 15*. The filter Q and gain are determined by external resistors.

All of the five second-order filter types can be built using either section of the MF10. These are illustrated in *Figure 1* through *Figure 5* along with their transfer functions and some related equations. *Figure 6* shows the effect of Q on the shapes of these curves. When filter orders greater than two are desired, two or more MF10 sections can be cascaded.

3.1 DESIGN EXAMPLE

In order to design a second-order filter section using the MF10, we must define the necessary values of three parameters: f_0, the filter section's center frequency; H_0, the passband gain; and the filter's Q. These are determined by the characteristics required of the filter being designed.

As an example, let's assume that a system requires a fourth-order Chebyshev low-pass filter with 1 dB ripple, unity gain at DC, and 1000 Hz cutoff frequency. As the system order is four, it is realizable using both second-order sections of an MF10. Many filter design texts include tables that list the characteristics (f_O and Q) of each of the second-order filter sections needed to synthesize a given higher-order filter. For the Chebyshev filter defined above, such a table yields the following characteristics:

f_{0A} = 529 Hz Q_A = 0.785

f_{0B} = 993 Hz Q_B = 3.559

For unity gain at DC, we also specify:

H_{0A} = 1

H_{0B} = 1

The desired clock-to-cutoff-frequency ratio for the overall filter of this example is 100 and a 100 kHz clock signal is available. Note that the required center frequencies for the two second-order sections will not be obtainable with clock-to-center-frequency ratios of 50 or 100. It will be necessary to adjust

$$\frac{f_{CLK}}{f_0}$$

externally. From *Table 1*, we see that Mode 3 can be used to produce a low-pass filter with resistor-adjustable center frequency.

In most filter designs involving multiple second-order stages, it is best to place the stages with lower Q values ahead of stages with higher Q, especially when the higher Q is greater than 0.707. This is due to the higher relative gain at the center frequency of a higher-Q stage. Placing a stage with lower Q ahead of a higher-Q stage will provide some attenuation at the center frequency and thus help avoid clipping of signals near this frequency. For this example, stage A has the lower Q (0.785) so it will be placed ahead of the other stage.

For the first section, we begin the design by choosing a convenient value for the input resistance: R_{1A} = 20k. The absolute value of the passband gain H_{OLPA} is made equal to 1 by choosing R_{4A} such that: $R_{4A} = -H_{OLPA}R_{1A} = R_{1A}$ = 20k. If the 50/100/CL pin is connected to mid-supply for nominal 100:1 clock-to-center-frequency ratio, we find R_{2A} by:

$$R_{2A} = R_{4A}\frac{f_{0A}^2}{(f_{CLK}/100)^2} = 2 \times 10^4 \times \frac{(529)^2}{(1000)^2} = 5.6\text{k and}$$

$$R_{3A} = Q_A\sqrt{R_{2A}R_{4A}} = 0.785\sqrt{5.6 \times 10^3 \times 2 \times 10^4} = 8.3\text{k}$$

The resistors for the second section are found in a similar fashion:

$$R_{1B} = 20\text{k}$$

$$R_{4B} = R_{1B} = 20\text{k}$$

$$R_{2B} = R_{4B}\frac{f_{0B}^2}{(f_{CLK}/100)^2} = 20\text{k}\frac{(993)^2}{(1000)^2} = 19.7\text{k}$$

$$R_{3B} = Q_B\sqrt{R_{2B}R_{4B}} = 3.559\sqrt{1.97\times10^4\times2\times10^4} = 70.6\text{k}$$

The complete circuit is shown in *Figure 16* for split ±5V power supplies. Supply bypass capacitors are highly recommended.

0 Applications Information (Continued)

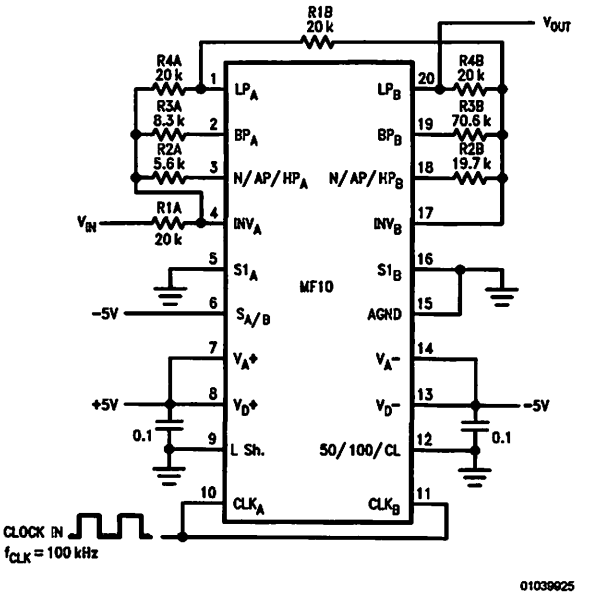


FIGURE 16. Fourth-Order Chebyshev Low-Pass Filter from Example in 3.1. ±5V Power Supply. 0V–5V TTL or –5V ±5V CMOS Logic Levels.

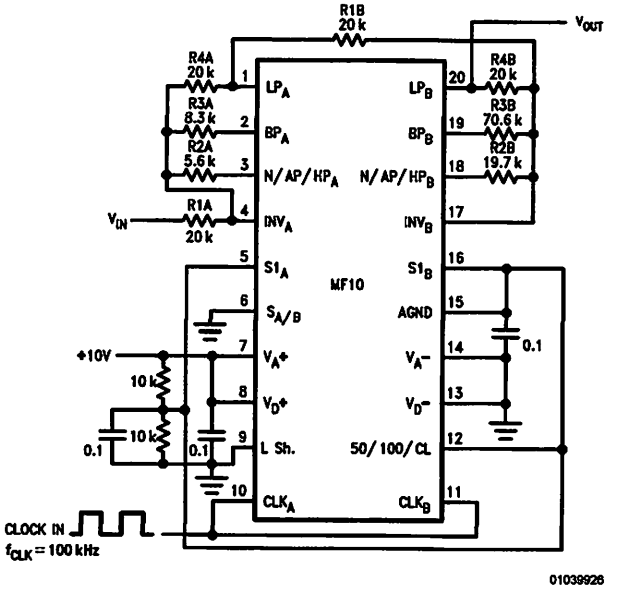
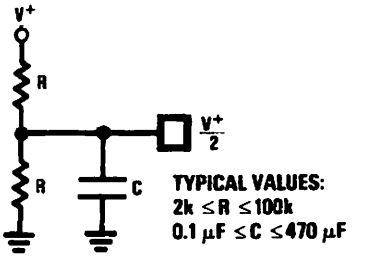
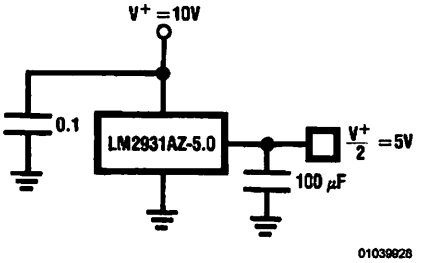


FIGURE 17. Fourth-Order Chebyshev Low-Pass Filter from Example in 3.1. Single +10V Power Supply. 0V–5V TTL Logic Levels. Input Signals Should be Referred to Half-Supply or Applied through a Coupling Capacitor.

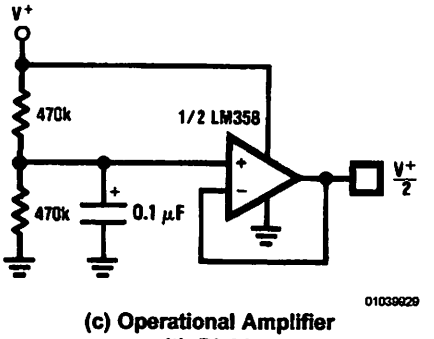
3.0 Applications Information (Continued)



(a) Resistive Divider with Decoupling Capacitor



(b) Voltage Regulator



(c) Operational Amplifier with Divider

FIGURE 18. Three Ways of Generating V+/2 for Single-Supply Operation

0 Applications Information

(Continued)

SINGLE SUPPLY OPERATION

e MF10 can also operate with a single-ended power sup-. *Figure 17* shows the example filter with a single-ended wer supply. V_A^+ and V_D^+ are again connected to the sitive power supply (8V to 14V), and V_A^- and V_D^- are nnected to ground. The A_{GND} pin must be tied to $V^+/2$ for gle supply operation. This half-supply point should be y "clean", as any noise appearing on it will be treated as input to the filter. It can be derived from the supply voltage h a pair of resistors and a bypass capacitor (*Figure 18a*), a low-impedance half-supply voltage can be made using a ee-terminal voltage regulator or an operational amplifier *gure 18b* and *Figure 18c*). The passive resistor divider h a bypass capacitor is sufficient for many applications, vided that the time constant is long enough to reject any wer supply noise. It is also important that the half-supply erence present a low impedance to the clock frequency, at very low clock frequencies the regulator or op-amp proaches may be preferable because they will require aller capacitors to filter the clock frequency. The main wer supply voltage should be clean (preferably regulated) d bypassed with 0.1 µF.

DYNAMIC CONSIDERATIONS

e maximum signal handling capability of the MF10, like t of any active filter, is limited by the power supply volt-es used. The amplifiers in the MF10 are able to swing to hin about 1V of the supplies, so the input signals must be t small enough that none of the outputs will exceed these its. If the MF10 is operating on ±5V, for example, the puts will clip at about 8 V_{p-p}. The maximum input voltage ltiplied by the filter gain should therefore be less than $_{p-p}$.

le that if the filter Q is high, the gain at the lowpass or hpass outputs will be much greater than the nominal filter n (*Figure 6*). As an example, a lowpass filter with a Q of will have a 20 dB peak in its amplitude response at f_O. If nominal gain of the filter H_{OLP} is equal to 1, the gain at f_O be 10. The maximum input signal at f_O must therefore be s than 800 mV_{p-p} when the circuit is operated on ±5V plies.

o note that one output can have a reasonable small age on it while another is saturated. This is most likely for ircuit such as the notch in Mode 1 (*Figure 7*). The notch put will be very small at f_O, so it might appear safe to ly a large signal to the input. However, the bandpass will e its maximum gain at f_O and can clip if overdriven. If one put clips, the performance at the other outputs will be graded, so avoid overdriving any filter section, even ones se outputs are not being directly used. Accompanying *ure 7* through *Figure 15* are equations labeled "circuit amics", which relate the Q and the gains at the various puts. These should be consulted to determine peak circuit ns and maximum allowable signals for a given applica-

OFFSET VOLTAGE

MF10's switched capacitor integrators have a higher ivalent input offset voltage than would be found in a cal continuous-time active filter integrator. *Figure 19* ws an equivalent circuit of the MF10 from which the put DC offsets can be calculated. Typical values for these ets with $S_{A/B}$ tied to V^+ are:

V_{os1} = opamp offset = ±5 mV

V_{os2} = −150 mV @ 50:1: −300 mV @ 100:1

V_{os3} = −70 mV @ 50:1: −140 mV @ 100:1

When $S_{A/B}$ is tied to V^-, V_{os2} will approximately halve. The DC offset at the BP output is equal to the input offset of the lowpass integrator (V_{os3}). The offsets at the other outputs depend on the mode of operation and the resistor ratios, as described in the following expressions.

Mode 1 and Mode 4

$$V_{OS(N)} = V_{OS1}\left(\frac{1}{Q} + 1\left\|H_{OLP}\right\|\right) - \frac{V_{OS3}}{Q}$$

$$V_{OS(BP)} = V_{OS3}$$

$$V_{OS(LP)} = V_{OS(N)} - V_{OS2}$$

Mode 1a

$$V_{OS(N.INV.BP)} = \left(1 + \frac{1}{Q}\right)V_{OS1} - \frac{V_{OS3}}{Q}$$

$$V_{OS(INV.BP)} = V_{OS3}$$

$$V_{OS(LP)} = V_{OS(N.INV.BP)} - V_{OS2}$$

3.0 Applications Information

(Continued)

Mode 2 and Mode 5

$$V_{OS(N)} = \left(\frac{R2}{R_p} + 1\right) V_{OS1} \times \frac{1}{1 + R2/R4}$$

$$+ V_{OS2} \frac{1}{1 + R4/R2} - \frac{V_{OS3}}{Q\sqrt{1 + R2/R4}}$$

$$R_p = R1//R3//R4$$

$$V_{OS(BP)} = V_{OS3}$$

$$V_{OS(LP)} = V_{OS(N)} - V_{OS2}$$

Mode 3

$$V_{OS(HP)} = V_{OS2}$$

$$V_{OS(BP)} = V_{OS3}$$

$$V_{OS(LP)} = V_{OS1}\left[1 + \frac{R4}{R_p}\right] - V_{OS2}\left(\frac{R4}{R2}\right)$$

$$-V_{OS3}\left(\frac{R4}{R3}\right)$$

$$Rp = R1//R2//R3$$

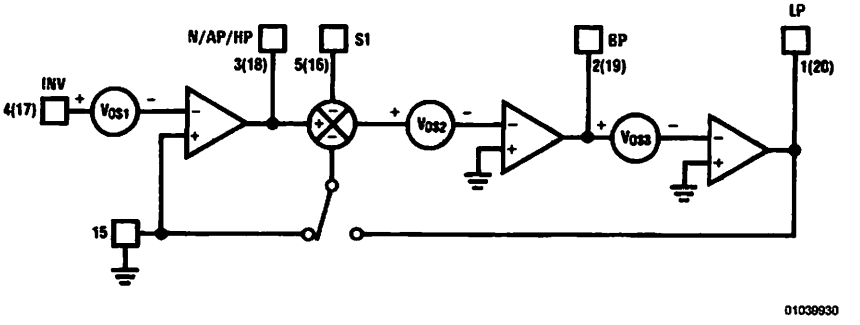


FIGURE 19. MF10 Offset Voltage Sources

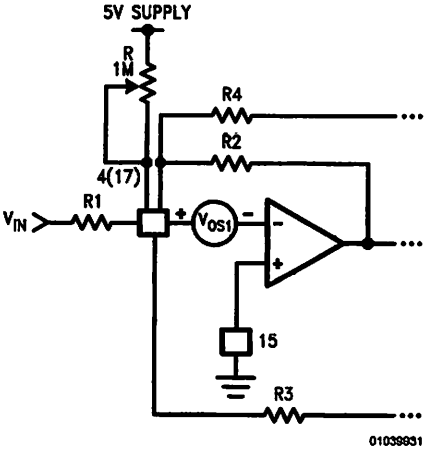


FIGURE 20. Method for Trimming V_{OS}

For most applications, the outputs are AC coupled and DC offsets are not bothersome unless large signals are applied to the filter input. However, larger offset voltages will cause clipping to occur at lower AC signal levels, and clipping at

.0 Applications Information

(Continued)

y of the outputs will cause gain nonlinearities and will ange f_O and Q. When operating in Mode 3, offsets can come excessively large if R2 and R4 are used to make $_{LK}/f_O$ significantly higher than the nominal value, especially Q is also high. An extreme example is a bandpass filter ving unity gain, a Q of 20, and f_{CLK}/f_O = 250 with pin 12 d to ground (100:1 nominal). R4/R2 will therefore be equal 6.25 and the offset voltage at the lowpass output will be out +1V. Where necessary, the offset voltage can be justed by using the circuit of *Figure 20*. This allows adjust- ent of V_{OS1}, which will have varying effects on the different tputs as described in the above equations. Some outputs nnot be adjusted this way in some modes, however $_{OS(BP)}$ in modes 1a and 3, for example).

5 SAMPLED DATA SYSTEM CONSIDERATIONS

e MF10 is a sampled data filter, and as such, differs in ny ways from conventional continuous-time filters. An portant characteristic of sampled-data systems is their ect on signals at frequencies greater than one-half the mpling frequency. (The MF10's sampling frequency is the me as its clock frequency.) If a signal with a frequency eater than one-half the sampling frequency is applied to e input of a sampled data system, it will be "reflected" to a quency less than one-half the sampling frequency. Thus, input signal whose frequency is $f_s/2$ + 100 Hz will cause e system to respond as though the input frequency was 2 − 100 Hz. This phenomenon is known as "aliasing", and can be reduced or eliminated by limiting the input signal spectrum to less than $f_s/2$. This may in some cases require the use of a bandwidth-limiting filter ahead of the MF10 to limit the input spectrum. However, since the clock frequency is much higher than the center frequency, this will often not be necessary.

Another characteristic of sampled-data circuits is that the output signal changes amplitude once every sampling period, resulting in "steps" in the output voltage which occur at the clock rate (*Figure 21*). If necessary, these can be "smoothed" with a simple R–C low-pass filter at the MF10 output.

The ratio of f_{CLK} to f_C (normally either 50:1 or 100:1) will also affect performance. A ratio of 100:1 will reduce any aliasing problems and is usually recommended for wideband input signals. In noise sensitive applications, however, a ratio of 50:1 may be better as it will result in 3 dB lower output noise. The 50:1 ratio also results in lower DC offset voltages, as discussed in Section 3.4.

The accuracy of the f_{CLK}/f_O ratio is dependent on the value of Q. This is illustrated in the curves under the heading "Typical Performance Characteristics". As Q is changed, the true value of the ratio changes as well. Unless the Q is low, the error in f_{CLK}/f_O will be small. If the error is too large for a specific application, use a mode that allows adjustment of the ratio with external resistors.

It should also be noted that the product of Q and $f_O$should be limited to 300 kHz when f_O < 5 kHz, and to 200 kHz for f_O > 5 kHz.

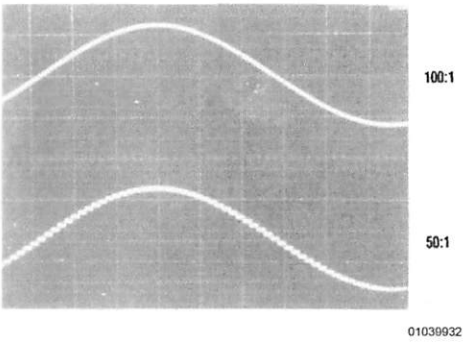
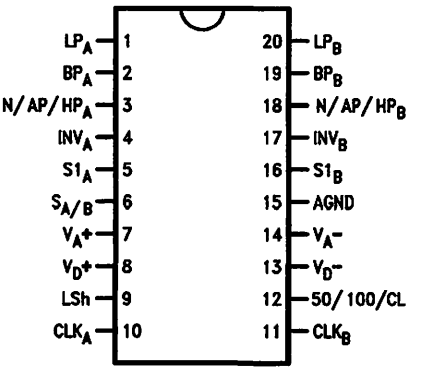


FIGURE 21. The Sampled-Data Output Waveform

3.0 Applications Information (Continued)

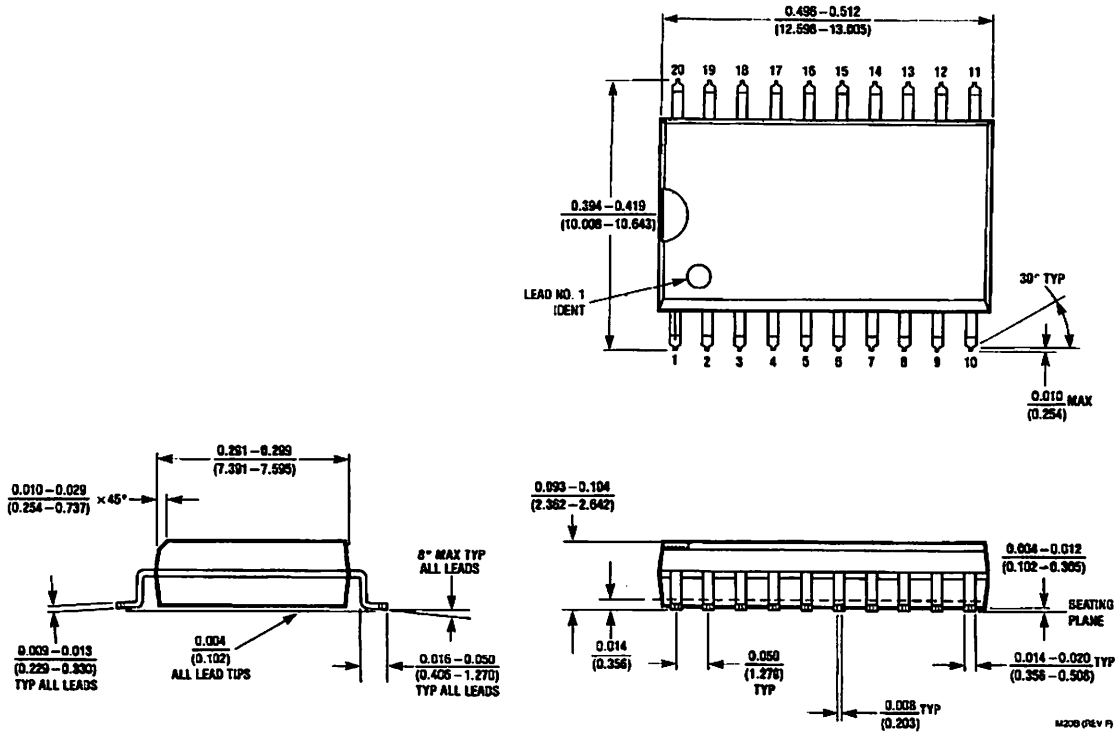
Connection Diagram

Surface Mount and Dual-In-Line Package

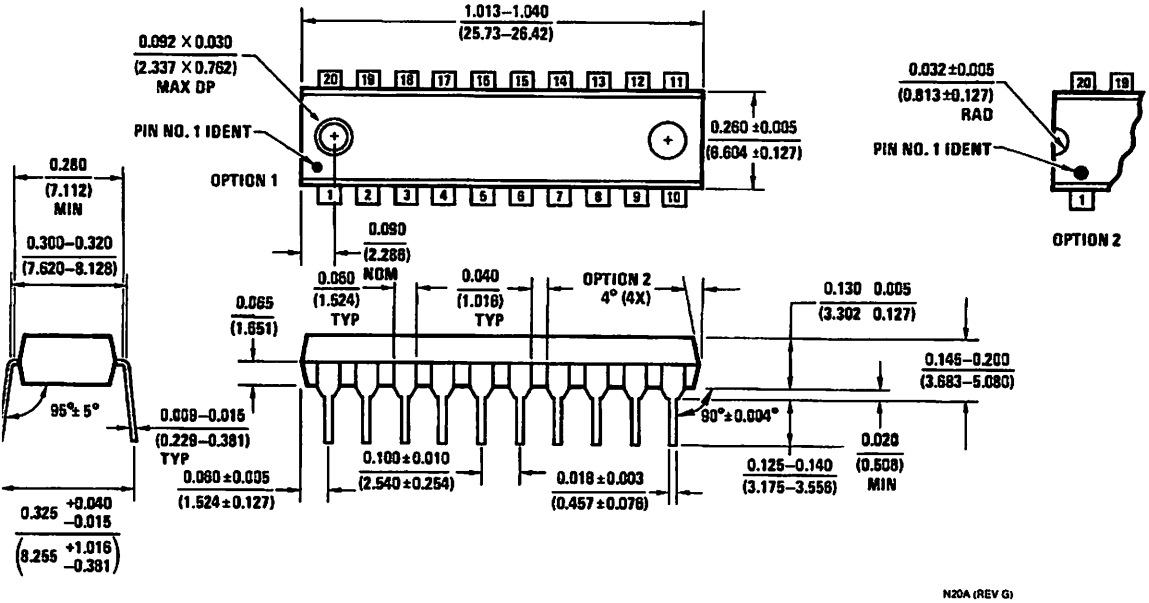


Top View
Order Number MF10CCWM
See NS Package Number M20B
Order Number MF10ACN or MF10CCN
See NS Package Number N20A

hysical Dimensions inches (millimeters)
less otherwise noted



Molded Package (Small Outline) (M)
Order Number MF10ACWM or MF10CCWM
NS Package Number M20B



20-Lead Molded Dual-In-Line Package (N)
Order Number MF10ACN or MF10CCN
NS Package Number N20A

Notes

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

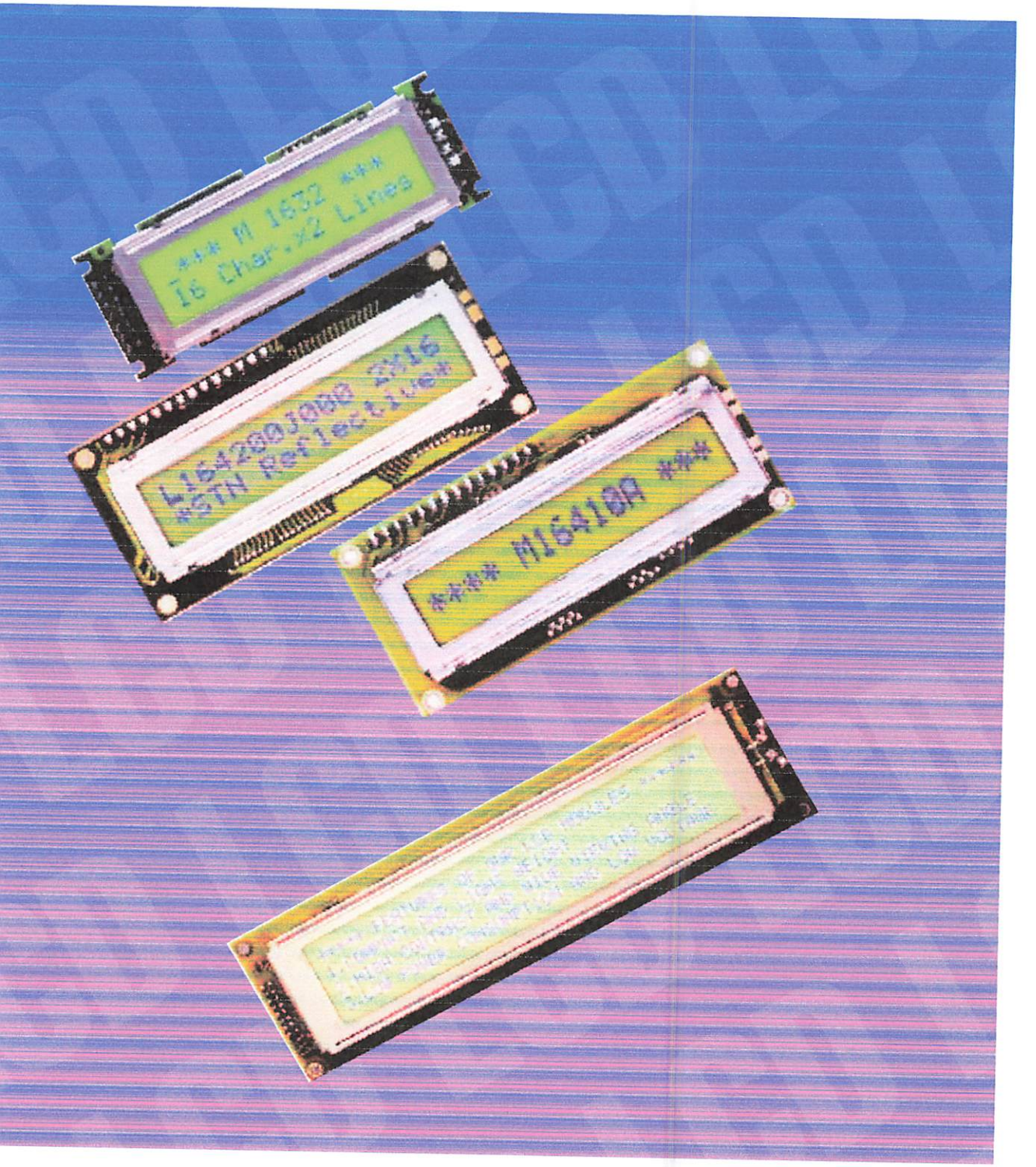
National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

LCM

Liquid Crystal Display Modules

Seiko Instruments GmbH

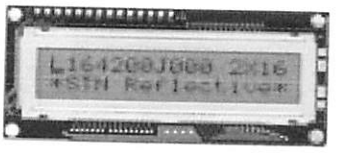


Dot Matrix Liquid Crystal Display Modules

HARACTER TYPE

- **FEATURES :**
  - Slim, light weight and low power consumption
  - High contrast and wide viewing angle
  - Built-in controller for easy interfacing
  - LCD modules with built-in EL or LED backlight

M1641



L1642

L1614

M1632

L1652

L2012

• SPECIFICATIONS :

: Standard products : Products of optional specification

r Format (character x line)		16 x 1	16 x 2	16 x 2	16 x 2	16 x 4	20 x 2
		M1641	M1632	L1642	L1652	L1614	L2012
		M16410AS	M16320AS	L164200J000S	L165200J200S	L161400J000S	L201200J000S
ght		M16419DWS	M16329DWS	L164221J000S	L165221J200S	L161421J000S	L201221J000S
light		M16417DYS	M16327DYS	L1642B1J000S	L1652B1J200S	L1614B1J000S	L2012B1J000S
(wide temp)		M16410CS	M16320CS	L164200L000S	L165200L200S	L161400L000S	L201200L000S
light (wide temp)		M16417JYS	M16327JYS	L1642B1L000S	L1652B1L200S	L1614B1L000S	L2012B1L000S
font		5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
	Reflective	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
	EL backlight	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
nm	LED backlight	80,0 x 36,0 x 15,8	80,0 x 30,0 x 15,8	80,0 x 36,0 x 15,8	122,0 x 44,0 x 15,8	87,0 x 60,0 x 15,8	116,0 x 37,0 x 15,8
rea (HxV) mm		64,5 x 13,8	62,0 x 16,0	64,5 x 13,8	99,0 x 24,0	61,8 x 25,2	83,0 x 18,6
size (HxV) mm *1		3,07 x 5,73	2,78 x 4,27	2,95 x 3,80	4,84 x 8,06	2,95 x 4,15	3,20 x 4,85
HxV) mm		0,55 x 0,75	0,50 x 0,55	0,50 x 0,55	0,92 x 1,10	0,55 x 0,55	0,60 x 0,65
pply voltage (VDD-VSS) V		+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
nsumption	IDD	1,5	2,0	1,6	2,0	2,7	2,0
	ILC *4	0,2	0,2	0,3	0,4	1,1	0,4
ethod (duty)		1/16	1/16	1/16	1/16	1/16	1/16
l		KS0066 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent
temperature (°C)	normal temp.	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50
	wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
nperature (°C)	normal temp.	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
	wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
Reflective		25	25	25	50	50	40
EL backlight		30	30	30	55	55	45
LED backlight		35	40	35	65	65	60
Model		5S	5S	5S	5C	5A	5A
Power supply (V)		+ 5.0	+ 5.0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
current consumption (mA) *3		10	10	10	35	45	45
Forward current consumption (mA)		100	112	100	240	200	154
Forward input voltage (V,typ.)		+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

g cursor

rnal temperature compensation

EL backlight

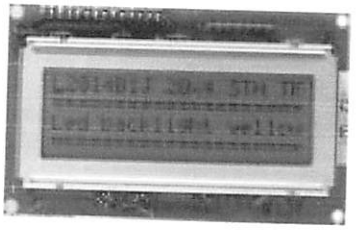
normal temperature range

H : Horizontal V : Vertical T : Thickness (max)

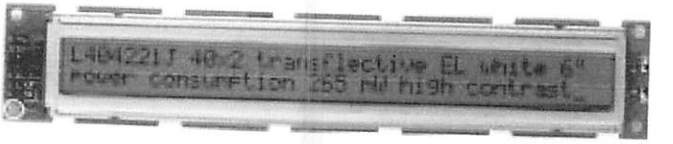
icy is one of continues improvements we reserve the right to change the specifications for the products in the catalogue without notice.

L2022

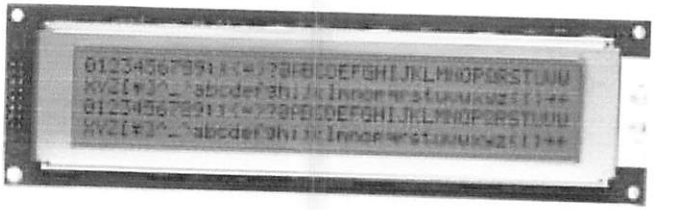
L2432



L2014



L4042



M4024

• SPECIFICATIONS :

: Standard products : Products of optional specification

aracter Format (character x line)		20 x 2	20 x 4	24 x 2	40 x 2	40 x 4
odel		L2022	L2014	L2432	L4042	M4024
flective		-	L201400J000S	L243200J000S	L404200J000S	M40240AS
backlight		-	L201421J000S	L243221J000S	L404221J000S	M40249DWS
) backlight		-	L2014B1J000S	L2432B1J000S	L4042B1J000S	M40247DYS
lective (wide temp)		L202200P000S	L201400L000S	L243200L000S	L404200L000S	M40240CS
) backlight (wide temp)		L2022B1P000S	L2014B1L000S	L2432B1L000S	L4042B1L000S	M40247JYS
aracter font		5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
dule	Reflective	180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
	EL backlight	180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
VxT) mm	LED backlight	180,0 x 40,0 x 14,8	98,0 x 60,0 x 15,8	118,0 x 36,0 x 15,8	182,0 x 33,5 x 16,3	190,0 x 54,0 x 16,3
wing area (HxV) mm		149,0 x 23,0	76,0 x 25,2	94,5 x 17,8	154,4 x 15,8	147,0 x 29,5
racter size (HxV) mm *1		6,00 x 9,66	2,95 x 4,15	3,20 x 4,85	3,20 x 4,85	2,78 x 4,27
size (HxV) mm		1,12 x 1,12	0,55 x 0,55	0,60 x 0,65	0,60 x 0,65	0,50 x 0,55
er supply voltage (VDD-VSS) V		+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
ent consumption	IDD	4,2	2,9	2,5	3,0	8,0
typ)	ILC *4	2,6	1,2	0,5	1,0	3,0
ing method (duty)		1/16	1/16	1/16	1/16	1/16
-in LSI		KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent
ating temperature (°C)	normal temp.	-	0 to + 50	0 to + 50	0 to + 50	0 to + 50
	wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
ge temperature (°C)	normal temp.	-	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
	wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
nt	Reflective	80	55	40	70	90
p.)	EL backlight	-	60	45	75	105
	LED backlight	110	70	60	95	140
	Model	-	5A	5A	5C	5D
ers	Power supply (V)	+ 5.0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
	current consumption (mA) *3	-	45	45	25	80
	Forward current consumption (mA)	320	240	150	260	480
ght	Forward input voltage (V,typ.)	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

cluding cursor
th external temperature compensation
luding EL backlight
sed on normal temperature range

H : Horizontal V : Vertical T : Thickness (max)

Dot Matrix Liquid Crystal Display Modules

GRAPHIC TYPE

• FEATURES :

- •Wide viewing angle and high contrast
- •Full dot configuration fits any application
- •Slim, light weight and low power consumption
- •Available in STN and FSTN

• SPECIFICATIONS :

at (HxV,dot)			97 x 32	128 x 32	128 x 64	128 x 64
			Y97031	G1213	G1216	G1226
e	Reflective	built-in RAM	-	-	-	-
ode)	Reflective wide temp.	built-in RAM	-	G121300N000S	G121600N000S	-
	LED backlight	built-in RAM	-	-	-	G1226B1J000S
	LED backlight wide temp	built-in RAM	-	G1213B1N000S	G1216B1N000S	-
e	Transmissive	-	-	-	-	-
ode)	with CFL backlight	built-in controller	-	-	-	-
	Transflective	built-in RAM	Y97031LF60W	-	-	-
ze	Reflective (no backlight)		47,5 x 65,4 x 2,1	75,0 x 41,5 x 6,8	75,0 x 52,7 x 6,8	-
T)	LED backlight		-	75,0 x 41,5 x 8,9	75,0 x 52,7 x 8,9	93,0 x 70,0 x 11,4
	CFL backlight		-	-	-	-
rea (HxV) mm			43,5 x 23,9	60,0 x 21,3	60,0 x 32,5	70,7 x 38,8
H x V) mm			0,35 x 0,48	0,40 x 0,48	0,40 x 0,40	0,44 x 0,44
H x V) mm			0,39 x 0,52	0,43 x 0,51	0,43 x 0,43	0,48 x 0,48
pply voltage (V)		(VDD - VSS)	+ 5,0	+ 5,0	+ 5,0	+ 5,0
		(VLC - VSS)	-	- 8,0	- 8,1	-8,2
nsumption		IDD	0,10	2,0	2,0	3,0
		IDD (built-in controller)	-	-	-	-
		ILC	-	1,8	1,8	2,0
Driving method (duty)			1/33	1/64	1/64	1/64
		Driver	SED1530 or equivalent	HD61202 HD61203 or equivalent	HD61202 HD61203 or equivalent	KS0107 KS0108 or equivalent
		Controller	-	-	-	-
temperature range (°C)			- 20 to + 70	- 20 to + 70	- 20 to + 70	0 to + 50
mperature range (°C)			- 30 to + 80	- 30 to + 80	- 30 to + 80	- 20 to + 60
	Reflective (Transflective no backlight)		10	23	35	-
	LED backlight		-	35	45	72
	CFL backlight		-	-	-	-
ght	Forward current consumption (mA)		-	40	90	125
	Forward input voltage (V, typ.)		-	3,8	4,1	4,1
CFL	Mode		-	-	-	-
	Power supply voltage (V)		-	-	-	-
	Current consumption (mA, typ.)		-	-	-	-

DC/DC converter (single power source)

n external temperature compensation circuit

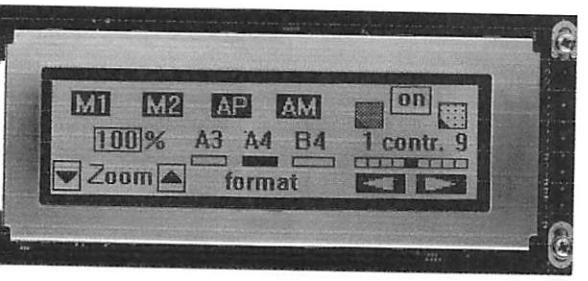
olicy is one of continues improvements we reserve the right to change the specifications of the products in the catalogue without notice.

format (HxV,dot)			240 x 64	240 x 128	320 x 200	320 x 240	640 x 200
el			G2446	G242C	G321D	G324E	G649D
type	Reflective	built-in RAM	-	-	-	-	-
y mode)	Reflective wide temp.	built-in RAM	-	-	-	-	-
	LED backlight	built-in RAM	-	-	-	-	-
	LED backlight wide temp	built-in RAM	-	-	-	-	-
N type	Transmissive	-	G2446X5R1A0S	G242CX5R1ACS	G321DX5R1A0S	G324EX5R1A0S	G649DX5R010S
V mode)	with CFL backlight	built-in controller	G2446X5R1ACS	G242CX5R1A0S	G321DX5R1ACS	G324EX5R1ACS	-
	Transflective	built-in RAM	-	-	-	-	-
ule size	Reflective (no backlight)			-	-	-	-
V x T)	LED backlight			-	-	-	-
	CFL backlight			-	-	-	-
ing area (HxV) mm			191,0 x 79,0 x 15,1	180,0 x 110,0 x 15,1	166,0 x 134,0 x 15,1	166,0 x 134,0 x 15,1	260,0 x 122,0 x 15,7
ize (H x V) mm			134,0 x 41,0	134,0 x 76,0	128,0 x 110,0	128,0 x 110,0	216,0 x 83,0
itch (H x V) mm			0,49 x 0,49	0,47 x 0,47	0,34 x 0,48	0,32 x 0,39	0,30 x 0,36
r supply voltage (V)		(VDD - VSS)	0,53 x 0,53	0,51 x 0,51	0,38 x 0,52	0,36 x 0,43	0,33 x 0,39
		(VLC - VSS)	+ 5,0	+ 5,0	+ 5,0	+ 5,0	+ 5,0
nt consumption		IDD	*1	*1	-24,0	-24,0	-24,0
		IDD (built-in controller)	12	30	8	7,5	11
typ.)		ILC	15	40	23	23	-
Driving method (duty)			-	-	6	6,5	9
n LSI		Driver	1/64	1/128	1/200	1/240	1/200
			MSM5298 MSM5299 or equivalent	KS0103 KS0104 or equivalent	MSM5298 MSM5299 or equivalent	HD66204 HD66205 or equivalent	MSM5298 MSM5299 or equivalent
		Controller	SED1330FB	SED1330FB	SED1330FB	SED1330FB	-
ting temperature range (°C)			0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50
ge temperature range (°C)			- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
t	Reflective (Transflective no backlight)		-	-	-	-	-
.)	LED backlight		-	-	-	-	-
	CFL backlight		200	280	350	350	420
acklight	Forward current consumption (mA)		-	-	-	-	-
	Forward input voltage (V, typ.)		-	-	-	-	-
	Mode		4800210	4800210	4800210	4800210	4800120
r for CFL	Power supply voltage (V)		+ 5,0	+ 5,0	+ 5,0	+ 5,0	+ 12,0
	Current consumption (mA, typ.)		250	350	365	365	390

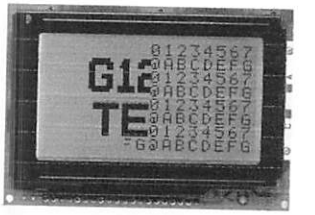
ilt-in DC/DC converter (single power source)

e with external temperature compensation

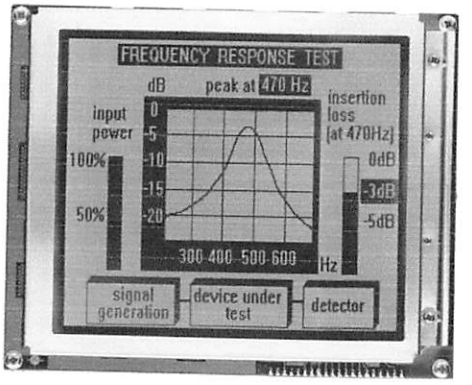
our policy is one of continous improvemets, we reserve the right to change the specifications of the products in the catalogue without notice.



G2446

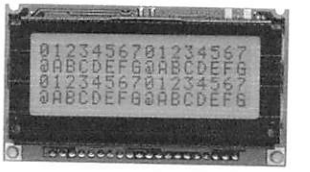


G1226

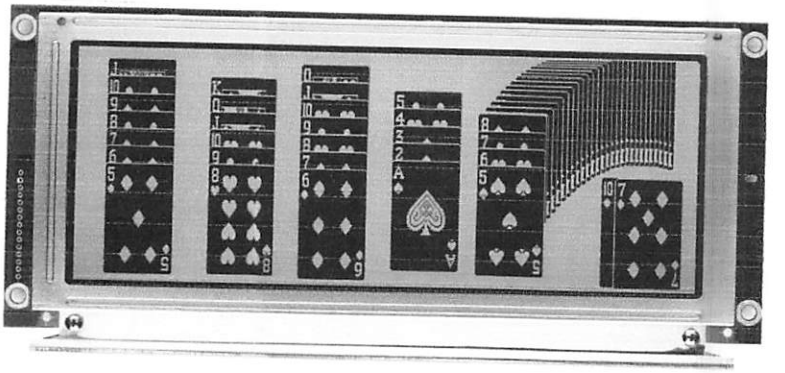


G321D

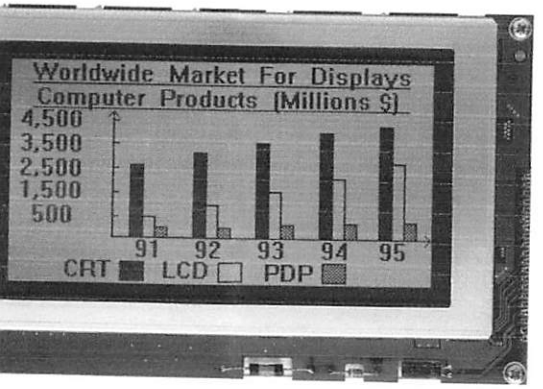
G1216



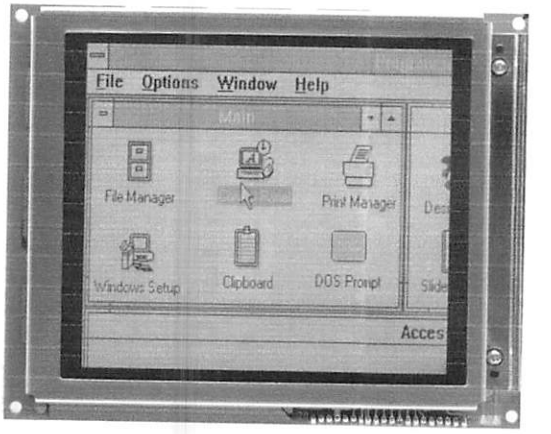
G1213



G649D



G242C



G324E

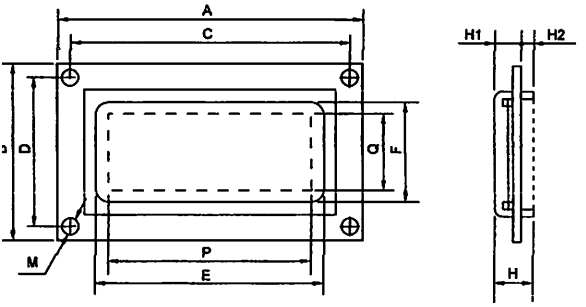
IECK LIST FOR CUSTOM DESIGNED LCD MODULE

ompany_____ **2. Application**_____ **3. Customer Specified Part No.**_____

esign

New ☐Modified : Manufacturer_____ , Part No._____ , Remarks _____
Equivalent: Manufacturer_____, Part No._____, Remarks _____

CM Dimensions



A x B : Module size _____ x _____ mm
E x F : Viewing area _____ x _____ mm
P x Q : Active display area _____ x _____ mm
C : Length between mounting holes _____ mm
D : Length between mounting holes _____ mm
M : Diameter of mounting hole _____ mm
H : Total thickness _____ mm
H1 : Upper thickness _____ mm
H2 : Lower thickness _____ mm

isplay Contents

Character type:_____characters_____lines
Character font_____x_____dots + cursor
Character pitch_____x_____mm
Dot pitch_____x_____mm
Dot size_____x_____mm
Graphics (Full dot) type:_____x_____dots
Dot pitch_____x_____mm
Dot size_____x_____mm
Segment type:_____digits_____lines
Others _____

CD Panel

ewing angle: ☐6 o'clock ☐12 o'clock☐____o'clock
pe: ☐TN ☐FSTN (Black and white)
☐STN (☐Yellow green ☐Gray ☐Blue)
Chromaticity coordinates
(_____ ≦ x ≦_____ , _____ ≦ y ≦_____)
☐Positive type ☐ Negative type
☐Reflective ☐ Transflective ☐ Transmissive
☐Others_____
ay scale: ☐Yes_____ gray scale ☐No
eferential specifications:
☐Response time t_{on} ____ ms (°C) t_{off} ____ ms (°C)
☐Viewing angle ____ deg. (°C) ☐ Contrast ____ (°C)
☐Others_____
D surface finishing:
☐Normal ☐Anti-glare☐_____
larizer color: ☐Normal (neutral gray) ☐Red
☐Green ☐ Blue ☐_____

riving Method

ltiplexing:1/_____ duty, 1/_____ bias
me frequency: _____ Hz
:
D driver: ☐Specified ☐Unspecified
Segment driver _____(Manufacturer _____)
Common driver _____(Manufacturer _____)
ntroller: ☐Internal ☐External
Type No._____(Manufacturer _____)
PU: ☐Internal ☐External
Type No._____(Manufacturer _____)
M: ☐Internal ☐External
ype No. /Memory size____(Kbit) (Manufacturer _____)

Power Supply

Single power supply: ☐ 5V ☐_____V
2 power supplies
For logic: (V_{DD}-Vss) : ☐ 5V ☐_____V
For LC drive: (V_{LC}-Vss) : ☐_____V

11. Temperature Compensation Circuit

☐Internal ☐External ☐Unnecessary
Compensation range: ☐ 0°C to 50°C ☐ ___°C to ___°C

12. Current Consumption

For logic: typ._____mA, max._____mA
For LC drive: typ._____mA, max._____mA
Others ( ) : typ._____mA, max._____mA

13. Contrast Adjustment

☐Internal ☐External ☐Unnecessary
Method: ☐Temp. compensation circuit ☐Volume ☐____

14. Temperature Range

Operating temperature range: ☐ 0°C to 50°C ☐____°C to____°C
Storage temperature range: ☐ - 20°C to 60°C ☐____°C to____°C

15. Input/Output Terminals

Specifying allocation: ☐Yes ☐No
Specifying position: ☐Yes ☐No

16. Weight

typ._____ g, max._____ g

17. Connector

☐ Internal ☐ External ☐ Unnecessary
Type No._____(Manufacturer _____)

18. Backlight

☐Internal ☐External ☐Unnecessary
☐EL: ☐Green ☐White ☐_____
☐LED: ☐Yellow green ☐Amber ☐_____
☐CFL: ☐White ☐_____
☐Incandescent lamp ☐Others_____
☐Backlight type ☐Edge backlight type
Brightness: _____cd/m²
Inverter: ☐Internal ☐External ☐Unnecessary
Power supply voltage_____V
Current consumption (backlight included)_____ mA
Brightness control: ☐Yes ☐No

19. Others

_____
_____
_____

20. Schedule

Estimate: _____
Sample: Delivery_____, Quantity:_____pcs
Mass production: Target price: _____
Delivery _____, Total quantity:_____pcs
Quantity per month_____pcs

uid Crystal Displays

ECK LIST FOR CUSTOM DESIGNED LCD

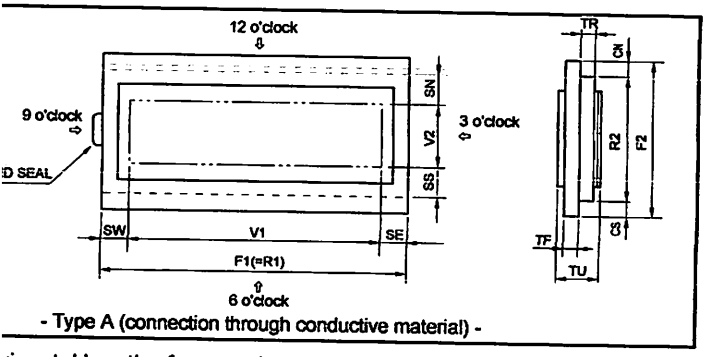
mpany_____ **2. Application**_____ **3. Customer Specified Part No.**_____

sign

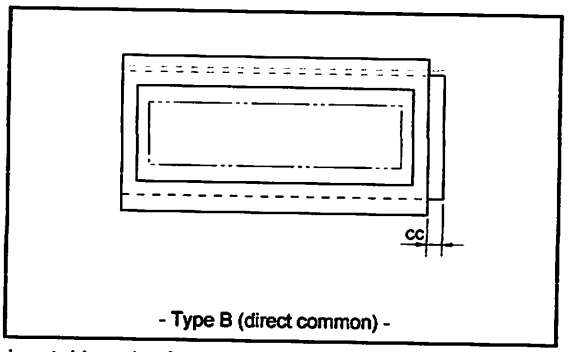
lew ☐Modified: Manufacturer _____ , Part No._____ , Remarks_____

quivalent: Manufacturer_____ , Part No._____ , Remarks_____

nel Dimensions



- Type A (connection through conductive material) -



- Type B (direct common) -

rizontal length of upper glass _____ mm

rtical length of upper glass_____ mm

rizontal length of lower glass_____ the same as F1

ertical length of lower glass_____ mm

generally longer than F2 when terminals are with pin.

***: Thickness of glass_____ mm

ndard type: 1.1 mm or 0.7 mm

ickness of LCD_____ mm

al: ☐Right ☐Left ☐Right or Left

V1 :Horizontal length of viewing area _____ mm

V2 :Vertical length of viewing area _____ mm

CN**: Terminal length_____ mm

CS**:Terminal length_____ mm

**CN or CS=0 in case of one side terminal type.

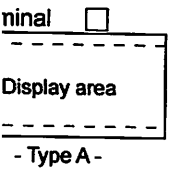
CC: Terminal length_____ mm

SE,SW,SN,SS : Seal width

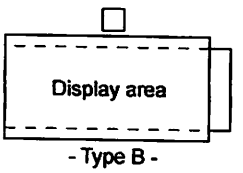
(According to design or manufacturing condition:
about 2.0 mm to 4.0 mm)

el Form

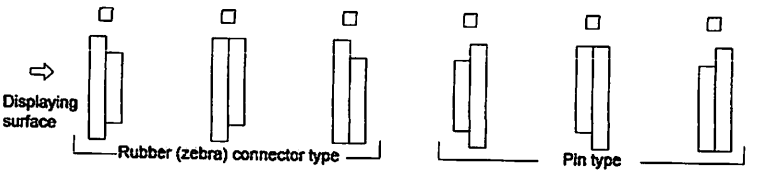
ninal ☐



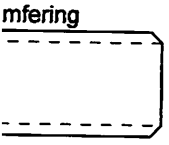
- Type A -



- Type B -

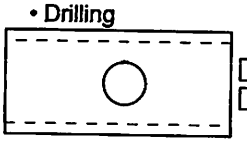


mfering



☐ Yes ☐ No

• Drilling



☐ Yes ☐ No

ilay Mode

ng angle : ☐6 o'clock ☐12 o'clock ☐_____o'clock

☐TN ☐FSTN (Black and white)

STN: (☐ Yellow green ☐ Gray ☐ Blue)

romaticity coordinates (____≦x≦____, ____≦y≦____)

'ositive type ☐Negative type

eflective ☐Transflective ☐Transmissive

ential specifications:

esponse time t_{on} ____ ms (____ °C) t_{off} ____ ms (____ °C)

iewing angle ____ deg. (____ °C) ☐Contrast ____ (____ °C)

thers_____

rizer

e finishing: ☐Normal ☐Anti-glare ☐_____

☐Normal (neutral gray) ☐Red ☐Green

☐Blue ☐_____

olarizer : ☐Attached type ☐Separate type

olarizer : ☐Attached type ☐Separate type

ng Method

☐Multiplexing: (1/_____duty, 1/_____bias)

ting voltage (V_{opr}) :_____ V

frequency:_____ Hz

g IC:_____(Manufacturer _____)

t consumption:_____ μA

10. Temperature Range

Operating temperature range

☐With temperature compensation circuit (or volume)

(☐ 0°C to 50°C ☐_____°C to_____°C)

☐Without temperature compensation circuit

(☐ 0°C to 50°C ☐_____°C to_____°C)

Storage temperature range

(☐ - 20°C to 60°C ☐_____°C to_____°C)

11. Terminal Connecting Method

☐Rubber connector (Zebra rubber)

☐Pin: ☐DIL ☐SIL ☐_____

Pitch (☐2.54 ☐_____mm) Length (_____mm)

☐Heat seal: ☐Equipped ☐Unnecessary

12. Others

Print (Characters, lines, masks etc.) : ☐Yes ☐No

Protective film:

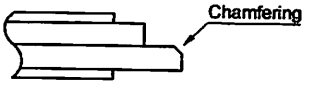
☐Yes (Color: ☐Red ☐Translucent ☐Transparent) ☐No

Chamfering (for heat-seal connector) :

☐Yes (Position:_____)

(Quantity:_____)

☐No



13. Schedule

Estimate :_____

Sample : Delivery_____ , Quantity :_____pcs

Mass production : Target price :_____

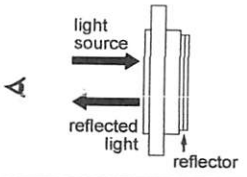
Delivery _____ , Total quantity:_____pcs

Quantity per month:_____pcs

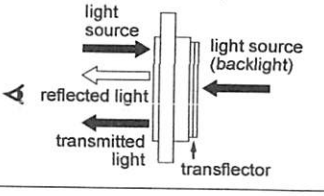
quid Crystal Display Modules

EFLECTIVE/TRANSFLECTIVE/TRANSMISSIVE LCD

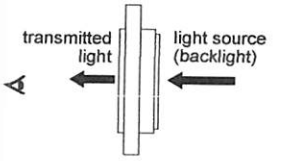
Reflective LCD
flector bonded to the rear polarizer
ects the incoming ambient light. Low power
sumption because no backlight is required.



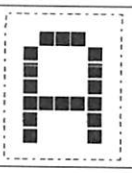
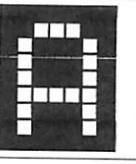
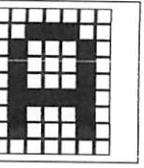
❷ Transflective LCD
Transflector bonded to the rear polarizer reflects light from the front as well as enabling lights to pass through the back. Used with backlight off in bright light and with it on in low light to reduce power consumption.



❸ Transmissive LCD
Without reflector or transflector bonded to the rear polarizer. Backlight required. Most common is transmissive negative image.



OSITIVE/NEGATIVE MODE

Positive type	
Negative type	
Negative type (inverse image) (when data is inverted)	

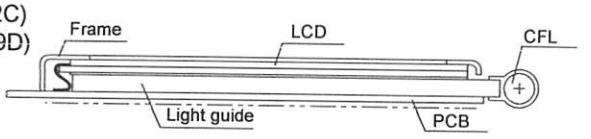
TYPE/STN TYPE/FSTN TYPE

	(Background/dot color) Gray/Black	TN(Twisted Nematic) type is most conventional and economical. It is used for static drive LCD and low-duty drive LCD (watch,calculator, etc.)
	Yellowgreen/Dark blue	STN (Super Twisted Nematic) type has a higher twist angle, and thus provides clear visibility and wider viewing angle. This is suitable especially for high-duty drive LCD.
	Gray/Dark blue	
	White/Blue	
N	White/Black	FSTN (Film Super Twisted Nematic) type utilizes RCF (Retardation Control Film) to remove the coloring of STN LCD. Thus FSTN type provides easy-to-read black-and-white display.

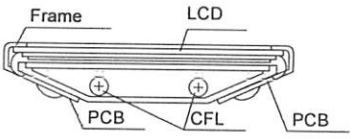
RUCTURE AND FEATURE OF LCD MODULE WITH BACKLIGHT

L (Cold Cathode Fluorescent Lamp) backlight
atures: high brightness, long service life, inverter required

dge backlight type
(G2446,G242C)
(G321D,G649D)

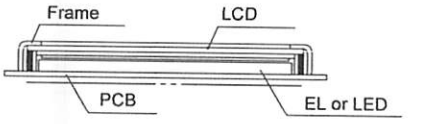


acklight type



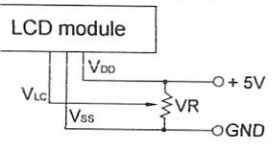
EL (Electroluminescent Lamp) backlight
LED (Light Emitting Diode) backlight

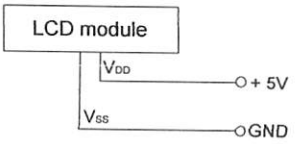
Features: EL: thin, inverter required
LED: long service life, low voltage driving, no inverter required

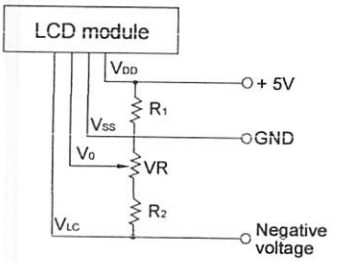


WER SUPPLY

racter modules (single power supply) • G2446,G242C (Built-in DC-DC conv.) • G321D, G324E and G649D

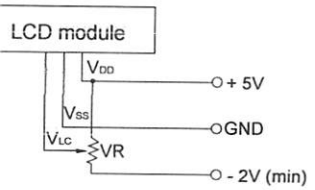


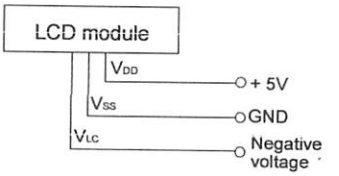




Note 1:Contrast can be adjusted by VR.
Note 2:For module with backlight, power supply for backlight is necessary.

aracter Modules(Dual power supply) • Y1206 and G1226





·Negative voltage should be variable for contrast adjustment.

ecautions

ty Instructions

he LCD panel is damaged, be careful not to get the uid crystal in your mouth and not to be injured by ıshed glasses.

ou should swallow the liquid crystal, first, wash your ıuth thoroughly with water, then, drink a lot of water d induce vomiting, and then, consult a physician.

he liquid crystal should get in your eye, flush your e with running water for at least fifteen minutes.

he liquid crystal touches your skin or clothes, re-ve it and wash the affected part of your skin or thes with soap and running water.

or CFL backlight is driven by a high voltage with an erter. Do not touch the connection part or the wir- pattern of the inverter.

not use inverters without a load or in the short-cir- t mode.

e the LCD module within the rated voltage to pre- nt overheating and/or damage. Also, take steps to sure that the connector does not come off.

lling Precautions

ce the LCD panel has glass substrate, avoid apply- mechanical shock or pressure on the module. Do drop, bend, twist or press the module.

not soil or damage LCD panel terminals.

ce the polarizer is made of easily-scratched mate- , be careful not to touch or place objects on the dis- y surface.

ep the display surface clean. Do not touch it with ır skin.

OS LSI is used in the LCD module. Be careful of ic electricity.

not disassemble the module or remove the liquid stal panel or the panel frame.

not damage the film surface of the EL lamp; other- e the lamp will be damaged by humidity.

set an EL lamp in an LCD module, push the EL p with its emitting side up, without pushing the rub- connectors too hard. If you damage them, the LCD dule may not work properly.

ting and Designing

rotect the polarizer and the LCD panel, cover the lay surface with a transparent plate (e.g., acrylic lass) with a small gap between the transparent plate the display surface.

p the module dry. Avoid condensation to prevent transparent electrodes from being damaged.

e LCD panel with AC waveform in which DC ele- nt is not included to prevent deterioration in the LCD el.

trast of LCD varies depending on the ambient tem- ıture. To offer the optimum contrast, LC drive volt- should be adjusted. LCD driven in a high duty must be provided with drive voltage adjustment nod.

nt a LCD module with the specified mounting part/ s.

- Design the equipment so that input signal is not applied to the LCD module while power supply voltage is not applied to it.
- Do not locate the CFL tube and the lamp lead wire close to a metal plate or a plated part inside the equipment. Otherwise stray capacity causes a drop in voltage, decreasing the brightness and the ability to start-up.

Cleaning

- Do not wipe the polarizer with a dry cloth, as it may scratch the surface.
- Wipe the LCD panel gently with a soft cloth soaked with a petroleum benzine.
- Do not use ketonic solvents (ketone and acetone)or aromatic solvents (toluene and xylene), as they may damage the polarizer.

Storing

- Store the LCD panel in a dark place, where the temperature is 25°C±10°C and the relative humidity below 65%. If possible, store the LCD panel in the packaging situation when it was delivered.
- Do not store the module near organic solvents or corrosive gases.
- Keep the module (including accessories) safe from vibration, shock and pressure.
- Use an LCD module with built-in EL backlight within six months of delivery.
- EL backlight is easily affected by environmental conditions such as temperature and humidity; the quality may deteriorate if stored for an extended period of time. Contact Seiko Instruments GmbH for details.
- Some parts of the backlight and the inverter generate heat. Take care so that the heat does not affect the liquid crystal or any other parts.
- Dust particles attached to the surface of the LCD or the surface of the backlight degrade the display quality. Be careful to keep dust out in designing the structure as well as in handling the module.
- Black or white air-bubbles may be produced if the LCD panel is stored for long time in the lower temperature or mechanical shocks are applied onto the LCD panel.

On This Brochure

- Seiko Instruments GmbH reserves the right to make changes without notice to the specifications and materials contained herein.
- The colors of the products reproduced herein may be different from the actual colors. Check color on actual products before using the product.
- The information contained herein shall not be reproduced in whole or in part without the express written consent of Seiko Instruments GmbH
- The products described herein are designed for consumer equipment and cannot be used as part of any device or equipment which influences the human body or requires a significantly high reliability, such as physical exercise equipment, medical equipment, disaster prevention equipment, gas related equipment, vehicles, aircraft and equipment mounted on vehicles.

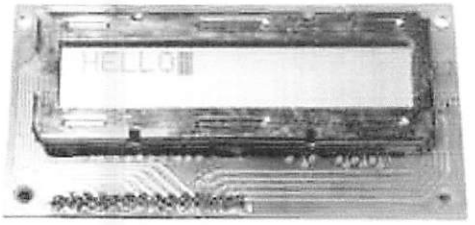
HOW TO USE
INTELLIGENT L.C.D.S

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

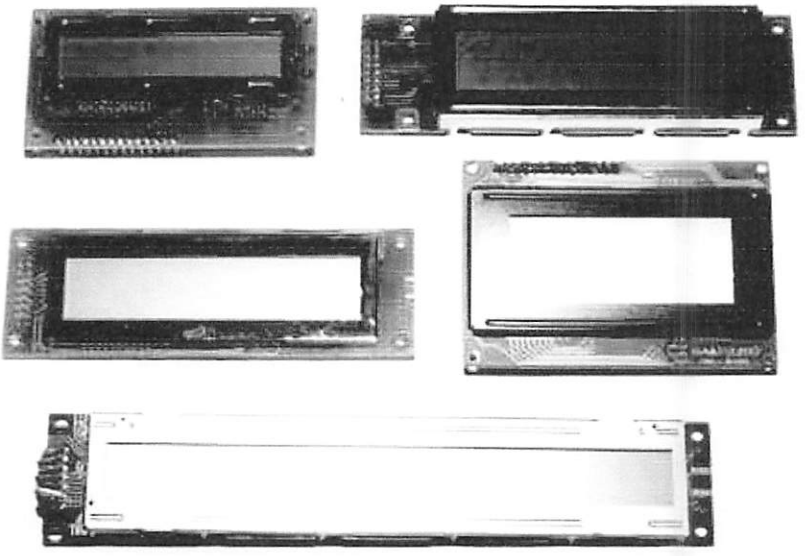
This article deals with the character-based l.c.d. modules which use the Hitachi HD44780 (or compatible) controller chip, as do most modules available to the hobbyist. Of course, these modules are not quite as advanced as the latest generation, full size, full colour, back-lit types used in today's laptop 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 Z80 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.

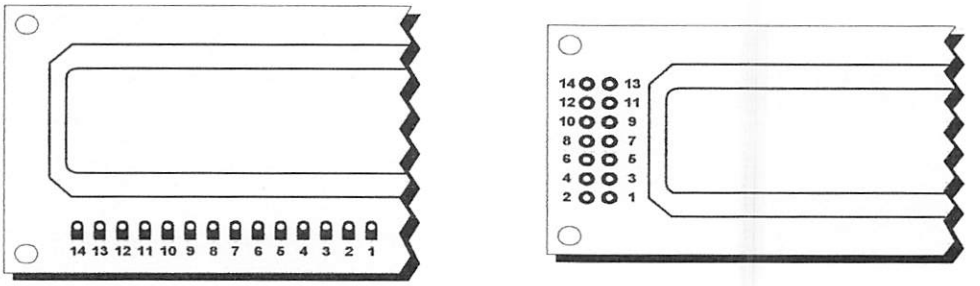


Figure 1: Pinouts of the two basic l.c.d formats.

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.

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 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.

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.

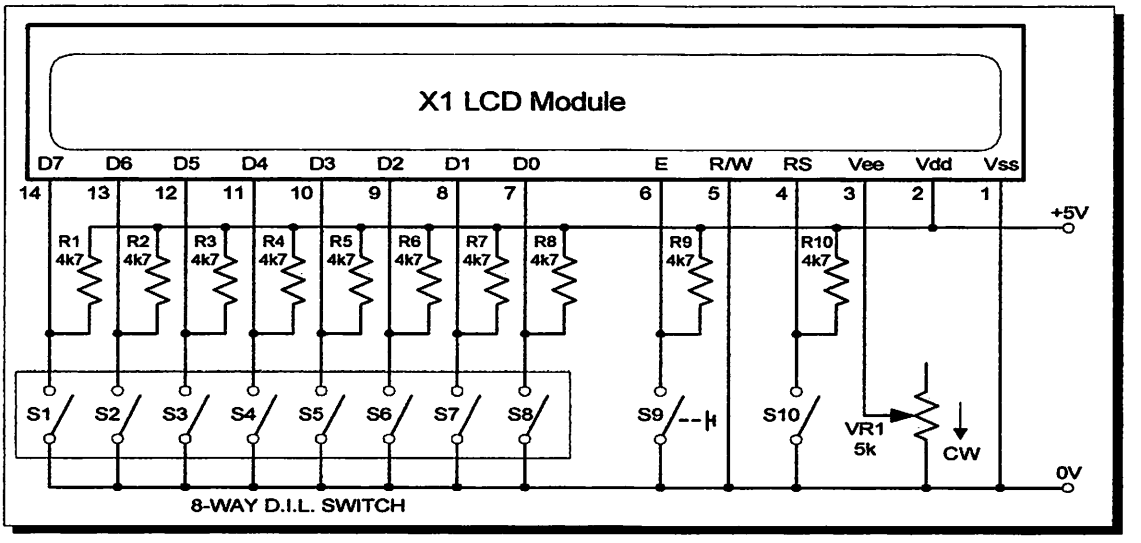


Figure 2: Circuit diagram for an l.c.d. experimental rig.

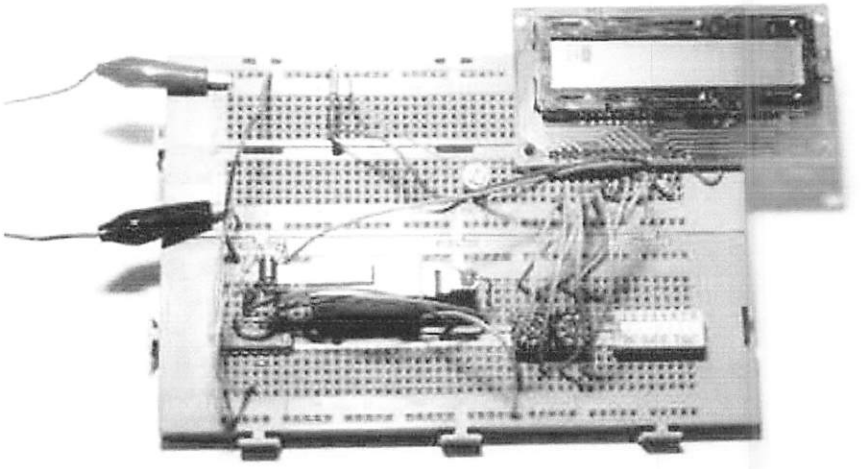
In Figure 2 is shown the circuit diagram of an l.c.d. experimentation rig. The circuit can be wired-up on a "plug-in" style prototyping board, using d.i.l. (dual-in-line) switches for the data lines (S1 to S8), a toggle switch for the RS input (S10), and a momentary action switch (or microswitch) for 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.

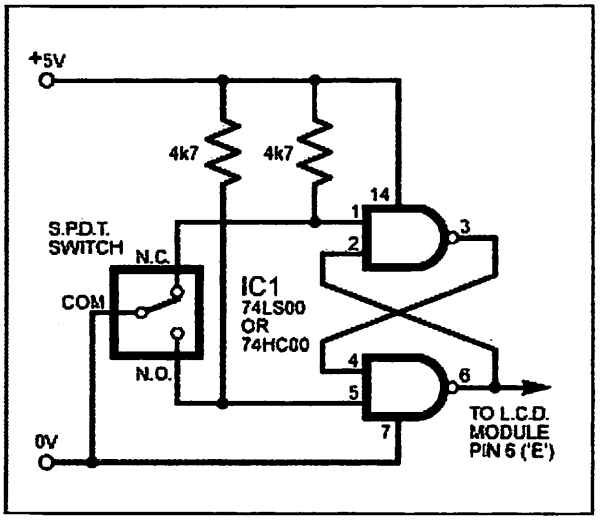


Figure 3. Switch debounce circuit.

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.

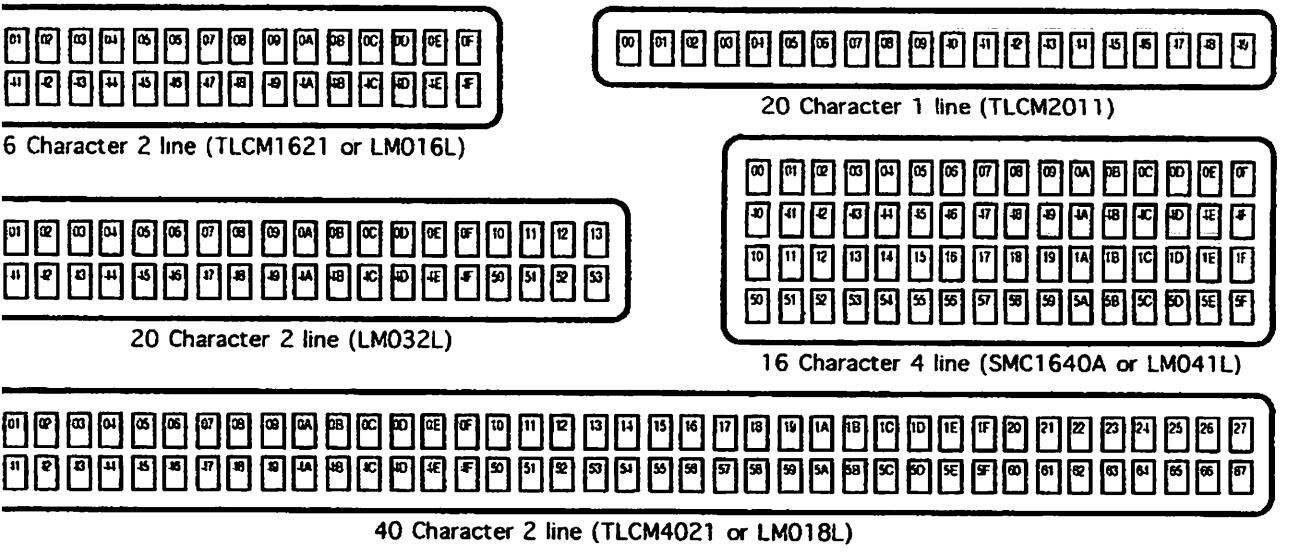


Figure 4: Examples of the relationship between addresses and display locations for typical module formats

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 ($00 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.

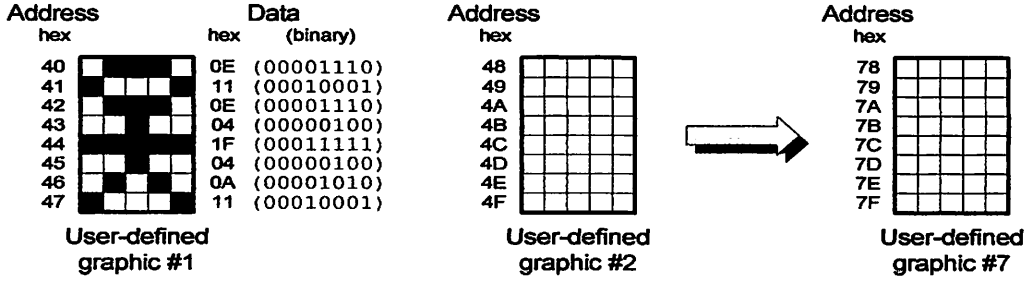


Figure 5: Showing how the CGRAM addresses correspond to individual pixels.

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!

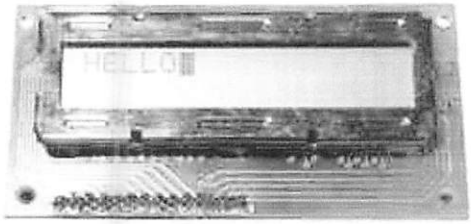
Well, that's about it, really. You've made it this far, so now you know everything there is to know about l.c.d. modules. Well, almost everything!

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



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 micro-processor, 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.

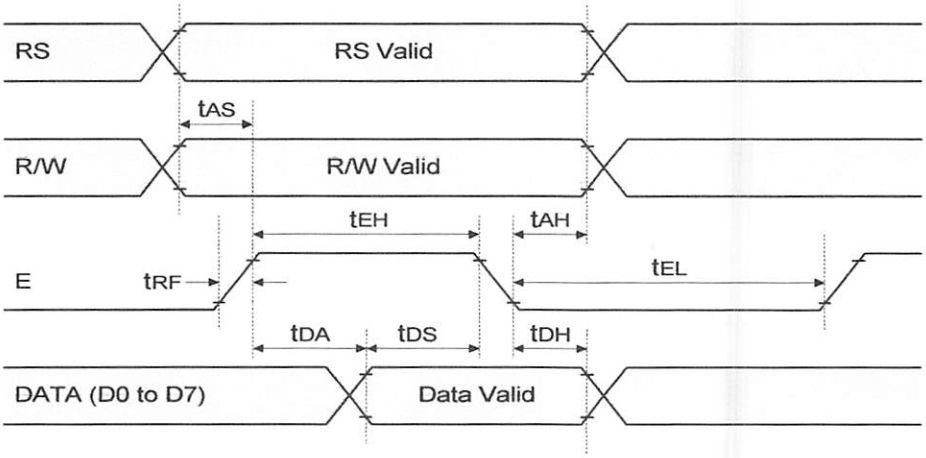


Figure 6: HD44780 timing diagram

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.

'e have published several PIC-based rojects in recent month's which are well 'orth studying, along with their software 'stings. 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 commands 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 micro-controller. The circuit diagram of the modified arrangement is shown in Figure 7. There's no longer any need for the debounce circuit, the microcontroller provides very clean output signals. It is not essential to use the PIC16C84 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 one or two 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 need to be programmed and erased during the course of experimentation. Other versions of the microcontroller cannot be erased so easily, indeed some cannot be erased at all (those referred to as OTP, One-Time Programmable devices, for example).

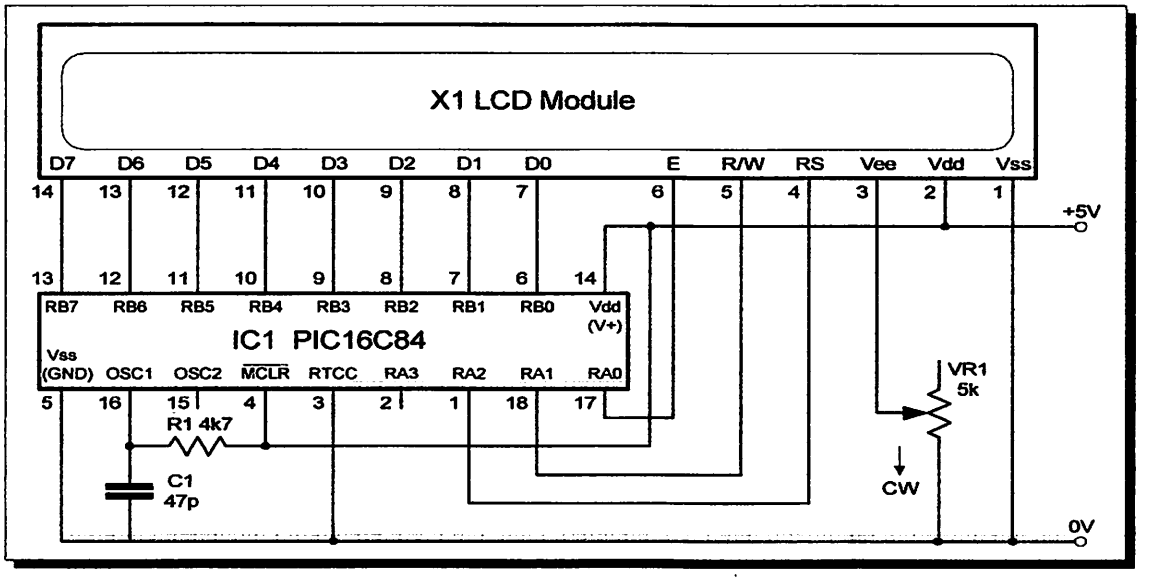
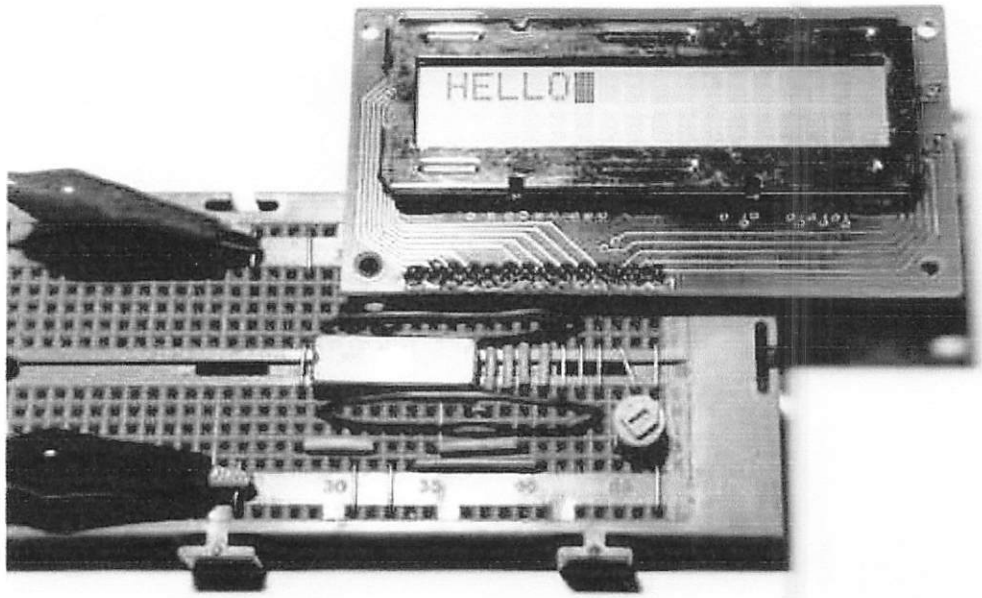


Figure 7: Circuit diagram for interfacing a PIC16C84 microcontroller to an l.c.d. module.

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.)

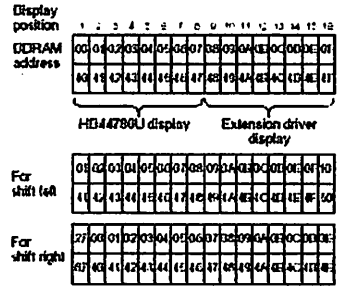
M1632 MODULE LCD 16 X 2 BARIS (M1632)

Deskripsi:

M1632 adalah merupakan modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya y rendah. Modul ini dilengkapi dengan mikrokontroler yang didisain khusus untuk mengendalikan L(Mikrokontroler HD44780 buatan Hitachi yang berfungsi sebagai pengendali LCD ini mempunyai CGR((Character Generator Read Only Memory), CGRAM (Character Generator Random Access Memory) DDRAM (Display Data Random Access Memory).

DDRAM

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



Gambar 1
DDRAM M1632 (diambil dari data sheet HD44780)

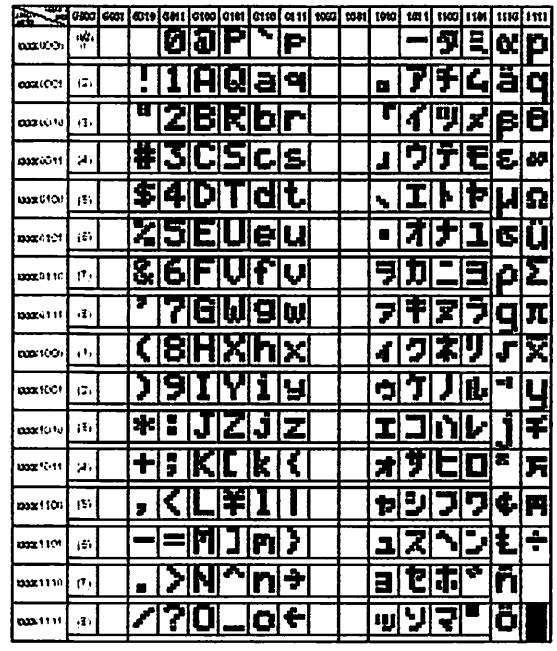
CGRAM

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

CGROM

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

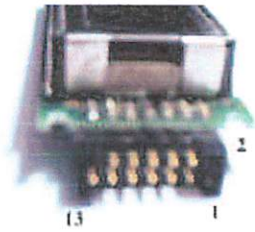
Pada gambar 2, tampak terlihat pola-pola karakter yang tersimpan dalam lokasi-lokasi tert dalam CGROM. Pada saat HD44780 akan menampilkan data 41H yang tersimpan pada DDRAM, m HD44780 akan mengambil data di alamat 41H (0100 0001) yang ada pada CGROM yaitu pola karakter A



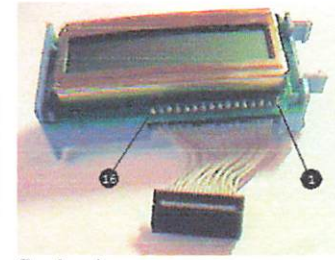
Gambar 2
Hubungan antara CGROM dan DDRAM (diambil dari data sheet HD44780)

Pin Out

No	Nama Pin	Deskripsi
1	VCC	+5V
2	GND	0V
3	VEE	Tegangan Kontras LCD
4	RS	Register Select, 0 = Register Perintah, 1 = Register I
5	R/W	1 = Read, 0 = Write
6	E	Enable Clock LCD, logika 1 setiap kali pengiriman : pembacaan data
7	D0	Data Bus 0
8	D1	Data Bus 1
9	D2	Data Bus 2
10	D3	Data Bus 3
11	D4	Data Bus 4
12	D5	Data Bus 5
13	D6	Data Bus 6
14	D7	Data Bus 7
15	Anoda (Kabel coklat untuk LCD Hitachi)	Tegangan positif backlight
16	Katoda (Kabel merah untuk LCD Hitachi)	Tegangan negatif backlight



Gambar 3
Pin Out M1632 LCD Hitachi



Gambar 4
Pin Out LCD M1632 Standard

Register

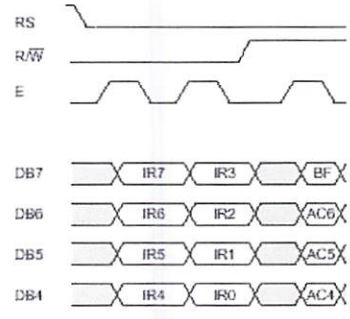
HD44780, mempunyai dua buah Register yang aksesnya diatur dengan menggunakan kaki Pada saat RS berlogika 0, maka register yang diakses adalah Register Perintah dan pada saat RS berlog 1, maka register yang diakses adalah Register Data

Register Perintah

Register ini adalah register di mana perintah-perintah dari mikrokontroler ke HD44780 pada : proses penulisan data atau tempat status dari HD44780 dapat dibaca pada saat pembacaan data.

Penulisan Data ke Register Perintah

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



Gambar 5
Timing diagram Penulisan Data ke Register Perintah Mode 4 bit Interface

Tabel 1
Perintah-perintah M1632

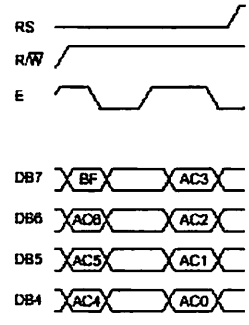
Perintah	D7	D6	D5	D4	D3	D2	D1	D0	Deskripsi
Hapus Display	0	0	0	0	0	0	0	1	Hapus Display dan DDRAM
Posisi Awal	0	0	0	0	0	0	1	X	Set Alamat DDRAM di 0
Set Mode	0	0	0	0	0	1	I/D	S	Atur arah pergeseran cursor dan displ

Display On/OFF	0	0	0	0	1	D	C	B	Atur display (D) On/OFF, cursor ON/OFF, Blinking (B)
Geser Cursor/Display	0	0	0	1	S/C	R/L	X	X	Geser Cursor atau display tanpa men alamat DDRAM
Set Fungsi	0	0	1	DL	N	F	X	X	Atur panjang data, jumlah baris tampil, dan font karakter
Set Alamat CGRAM	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Data dapat dibaca atau ditulis se alamat diatur
Set Alamat DDRAM	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Data dapat dibaca atau ditulis se alamat diatur

X = diabaikan
I/D 1=Increment, 0=Decrement
S 0=Display tidak geser
S/C 1=Display Shift, 0=Geser Cursor
R/L 1=Geser Kiri, 0=Geser Kanan
DL 1=8 bit, 0=4bit
N 1=2 baris, 0=1 baris
F 1=5x10, 0=5x8
D 0=Display OFF, 1=Display ON
C 0=Cursor OFF, 1=Cursor ON
B 0=Blinking OFF, 1=Blinking ON

Pembacaan Data dari Register Perintah

Proses pembacaan data pada register perintah biasa digunakan untuk melihat status busy dari L atau membaca Address Counter. RS diatur pada logika 0 untuk akses ke Register Perintah, R/W diatur p logika 1 yang menunjukkan proses pembacaan data. 4 bit nibble tinggi dibaca dengan diawali pulsa log 1 pada E Clock dan kemudian 4 bit nibble rendah dibaca dengan diawali pulsa logika 1 pada E Clo Untuk Mode 8 bit interface, pembacaan 8 bit (nibble tinggi dan rendah) dilakukan sekaligus dengan diav sebuah pulsa logika 1 pada E Clock.



Gambar 6
Timing Diagram Pembacaan Register Perintah Mode 4 bit Interface

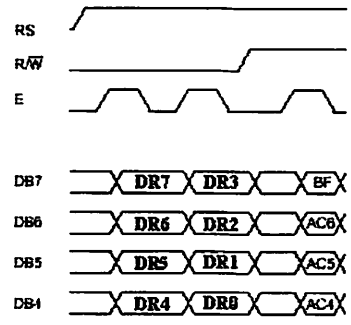
Register Data

Register ini adalah register di mana mikrokontroler dapat menuliskan atau membaca data ke a dari DDRAM. Penulisan data pada register ini akan menempatkan data tersebut ke DDRAM sesuai den alamat yang telah diatur sebelumnya

Penulisan Data ke Register Data

Penulisan data pada Register Data dilakukan untuk mengirimkan data yang akan ditampilkan pa LCD. Proses diawali dengan adanya logika 1 pada RS yang menunjukkan akses ke Register Data, kondisi R/W diatur pada logika 0 yang menunjukkan proses penulisan data. Data 4 bit nibble tinggi (bit 7 hingga

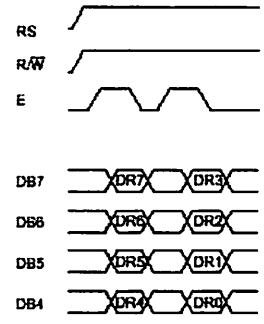
bit 4) dikirim dengan diawali pulsa logika 1 pada sinyal E Clock dan kemudian diikuti 4 bit nibble rendah (bit 3 hingga bit 0) yang juga diawali pulsa logika 1 pada sinyal E Clock.



Gambar 7
Timing Diagram Penulisan Data ke Register Data Mode 4 bit Interface

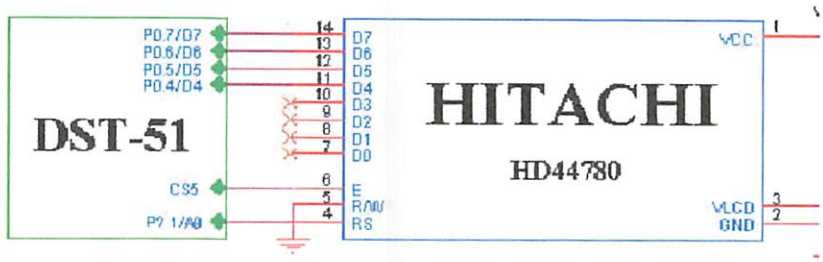
Pembacaan Data dari Register Data

Pembacaan data dari Register Data dilakukan untuk membaca kembali data yang tampil p LCD. Proses dilakukan dengan mengatur RS pada logika 1 yang menunjukkan adanya akses ke Regi Data. Kondisi R/W diatur pada logika tinggi yang menunjukkan adanya proses pembacaan data. Data 4 nibble tinggi (bit 7 hingga bit 4) dibaca dengan diawali adanya pulsa logika 1 pada E Clock dan dilanjut dengan data 4 bit nibble rendah (bit 3 hingga bit 0) yang juga diawali dengan pulsa logika 1 pada E Clock

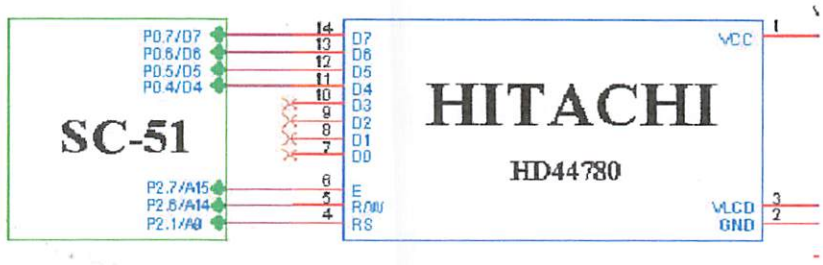


Gambar 8
Timing Diagram Pembacaan Data dari Register Data Mode 4 bit Interface

Antar muka LCD dengan mikrokontroler



Gambar 9
Antar muka dengan Modul DST-51



Gambar 10
Antar Muka dengan Modul SC-51 atau AT8951

Program
Rutin-rutin Program untuk DST-51 yang diassembly dengan ALDS atau ASM51
Rutin-rutin Program untuk SC-51/AT8951 yang diassembly dengan ALDS atau ASM51
Rutin delay yang diassembly dengan ALDS atau ASM51
Datasheet HD44780