LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLER AT89S52

SKRIPSI

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

Disusun Oleh :
Raditya Aspri Putra
00.17.169

Diperiksa dan disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Cahyo C. ,Msc.	**M. Ashar ,ST ,MT.**
NIP. 103 040 0412	**NIP. 103 050 0408**

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono ,MT.
NIP.Y. 103 950 0274

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

KATA PENGANTAR

Dengan memanjatkan puji syukur kepada Tuhan Yang Maha Esa, yang telah memberikan berkah dan karunia-Nya, akhirnya penyusun dapat menyelesaikan skripsi yang berjudul **"Perancangan dan pembuatan Simulator Akord Gitar menggunakan Mikrokontroler AT89S52".** Laporan skripsi ini merupakan salah satu persyaratan kelulusan Strata 1 jurusan Teknik Elektro Program Studi Elektronika, Institut Teknologi Nasional Malang.

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

1. Kedua orang tua atas dukungan material dan do'a restunya
2. Ir. F. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro S-1
3. Dr. Cahyo C. , Msc. , selaku Dosen Pembimbing I
4. M. Ashar, ST, Msc. ,selaku Dosen Pembimbing II
5. Ir. H. Sidik Noertjahjono, MT, selaku Dekan Institut Teknologi Nasional Malang
6. Prof. Dr. Ir. Abraham Lomi, MSEE, selaku Rektor Institut Teknologi Nasional Malang
7. Teman – teman yang telah membantu terselesaikannya skripsi ini

Namun karena keterbatasan waktu, kemampuan serta faktor lain yang dihadapi sehingga menyebabkan laporan skripsi ini tidak lepas dari banyaknya kekurangan. Karena itu sejumlah koreksi dan masukan diperlukan guna kesempurnaan laporan skripsi ini. Semoga laporan skripsi dari pemikiran yang

sederhana ini akan menjadi cikal bakal dari karya yang lebih inovatif dan dapat bermanfaat untuk semua orang.

Malang, maret 2009

Penyusun

PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52

Raditya Aspri Putra
Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1
Konsentrasi Teknik Elektronika
Jl. Raya Karanglo Km.2
Phone : 085649698468
e-mail : 3lk4@doramail.com
Pembimbing I : Dr. Cahyo C. ,Msc.
Pembimbing II : M. Ashar, ST, MT.

ABSTRAK

Gitar merupakan alat musik yang banyak digemari masyarakat, baik orang dewasa, remaja, maupun anak-anak. Bahkan orang yang sudah berumur pun banyak yang hobi memainkan gitar. Harga gitar di pasaran sangat bervariasi, bahkan terbilang murah sehingga semua golongan dapat memilikinya. Di samping itu, gitar adalah alat musik yang sangat harmonis sehingga dapat digunakan untuk mengiringi berbagai aliran lagu. Akord gitar merupakan pengetahuan dasar yang harus dimengerti oleh para pemula dalam menguasai permainan gitar.

Alat ini menggunakan mikrokontroler AT89S52 sebagai kontrol utamanya yaitu membaca fungsi inputan dari keypad lalu mengeluarkan hasilnya ke LED matriks dan LCD. Bagian input berupa keypad yang terdiri dari 12 tombol yang mewakili 12 nada akord, sedangkan bagian outputnya berupa LED yang disusun secara matriks untuk menampilkan posisi jari dan sebuah LCD display untuk memberi keterangan akord apa yang sedang di tampilkan oleh LED matriks. Alat penampil akord gitar ini dapat menampilkan 36 variasi akord. Setiap tombol input dapat ditekan 3 kali untuk menampilkan 3 variasi akord gitar yang berbeda.

Kata kunci : akord gitar, AT89S52

PERANCANGAN DAN PENGUJIAN SIMULATOR ARGORO GITAR MENGGUNAKAN MIKROKONTROLLER ATMEGA32

Hadiyu Aspri Putra
Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1
Konsentrasi Teknik Elektronika
Jl. Raya Karanglo Km.2
Phone : 085604098468
e-mail :
Pembimbing I : Ir. Cahyo C., MSc.
Pembimbing II : M. Aslam, ST., MT.

ABSTRAK

Gitar merupakan alat musik yang banyak digemari masyarakat, baik orang kecil, remaja, maupun anak-anak. Dalam arena yang sudah bermutu pun banyak yang telah memainkan gitar. Hanya saja, alat ini pasaran sangat berpengaruh. Dalam teknik permainan gitar [illegible] Pembimbing II. Untuk alat ini mempunyai [illegible] dapat digunakan untuk mengatur berbagai alat musik [illegible] alat ini mempunyai dua buah dimensi yang hanya dimengerti oleh pemain [illegible] gitar.

Alat ini menggunakan mikrokontroler ATMEGA32 sebagai kontrol utama yaitu menerima input dari [illegible] dan gelombang analog ke LED matriks dan LCD. Dengan input 2 tombol yang terdiri dari 12 tombol yang masing-masing 12 nada akan [illegible] dan [illegible] yang di atur secara matriks untuk mengeluarkan pada 1 dari delapan nada. Alat ini menggunakan 2 buah [illegible] yang akan [illegible] matriks. Alat ini menggunakan dua buah [illegible] dan [illegible] untuk output dimana 1 buah untuk mengatur [illegible] dan yang [illegible].

Kata kunci : akord gitar, ATMEGA32

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Gitar merupakan alat musik yang banyak digemari masyarakat, baik orang dewasa, remaja, maupun anak-anak. Bahkan orang yang sudah berumur pun banyak yang hobi memainkan gitar. Harga gitar di pasaran sangat bervariasi, bahkan terbilang murah sehingga semua golongan dapat memilikinya. Di samping itu, gitar adalah alat musik yang sangat harmonis sehingga dapat digunakan untuk mengiringi berbagai aliran lagu.

Setiap orang yang ingin belajar bermain gitar pasti menginginkan jalan termudah untuk melakukannya. Salah satu hal terpenting dalam permainan gitar adalah akord gitar. Akord gitar berfungsi sebagai ritem atau pengiring lagu dalam permainan gitar. Akord gitar ini biasanya dapat ditemukan pada bagian belakang buku-buku lagu.

Untuk seorang pemula yang sedang belajar bermain gitar, tentulah tidak mudah untuk mempelajari akord gitar sambil membalik-balik buku. Oleh karena itu, pada kesempatan penyusunan skripsi ini penulis mencoba membuat sebuah alat simulator akord gitar yang praktis dalam pemakaiannya, dengan harapan dapat membantu para pemula dalam menguasai permainan gitar, dengan menggunakan mikrokontroler AT89S52 sebagai komponen utamanya dan keypad sebagai masukannya untuk menentukan akord apa yang ingin ditampilkan, sebagai tampilannya digunakan LED matriks 6x8 dan LCD display.

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Gitar merupakan alat musik yang banyak digemari masyarakat baik orang dewasa, remaja, maupun anak-anak. Bahkan orang yang sudah berumur pun banyak yang lebih memainkan gitar. Harga gitar di pasaran sangat bervariasi, bahkan terhitung murah sehingga semua golongan dapat memilikinya. Di samping itu, gitar adalah alat musik yang sangat harmonis sehingga dapat digunakan untuk mengiringi berbagai aliran lagu.

Setiap orang yang ingin belajar bermain gitar pasti menginginkan jalan termudah untuk melakukannya. Salah satu hal terpenting dalam permainan gitar adalah akord gitar. Akord gitar berfungsi sebagai iringan atau pengiring lagu dalam permainan gitar. Akord gitar ini biasanya dapat ditemukan pada bagian belakang buku-buku lagu.

Untuk seorang pemula yang sedang belajar bermain gitar, tentulah tidak mudah untuk memperlajari akord gitar sambil membolak-balik buku. Oleh karena itu, pada kesempatan penyusunan skripsi ini penulis menerapkan membuat sebuah alat simulator akord gitar yang praktis dalam penggunaannya, dengan harapan dapat membantu para pemula dalam mengenali permainan gitar, dengan menggunakan mikrokontroler AT89S52 sebagai komponen utamanya dan keypad sebagai masukannya untuk menentukan akord apa yang ingin ditampilkan, sebagai tampilannya digunakan LED matriks 6x8 dan LCD display.

1.2. Tujuan Penulisan

Adapun tujuan dari penulisan skripsi ini adalah sebagai berikut:

1. Merancang dan membuat simulator akord gitar menggunakan mikrokontroller AT89S52.
2. Membantu orang yang pertama kali untuk belajar menggunakan gitar.

1.3. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan pada bagian sebelumnya, maka permasalahannya adalah bagaimana merancang dan membuat alat yang dapat mensimulasikan akord gitar supaya dapat di mengerti oleh pemula. Sehubungan dengan hal itu, maka skripsi ini diberi judul **PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52**

Berdasar hal tersebut diatas maka dibuatlah rumusan masalah sebagai berikut:

1. Bagaimana membuat rangkaian penampil simulasi akor gitar berbasis mikrokontroler AT89S52
2. Bagaimana membuat program assembler simulasi akord yang bisa di tampilkan ke LED matrik dan LCD dalam mikrokontroler AT89S52

1.4. Batasan Masalah

Dalam menyusun skripsi ini diperlukan suatu batasan masalah agar tidak menyimpang dari ruang lingkup yang akan dibahas. Adapun batasan masalahnya:

1. Mikrokontroller yang digunakan adalah AT89S52
2. Hanya mensimulasikan akord : Mayor , Minor , Augment
3. Masukan menggunakan keypad dengan tampilan berupa LED Matriks dan LCD
4. Tidak membahas suara dari gitar
5. Tidak membahas catu daya
6. Hanya menampilkan sampai 8 tanda fret

1.5. Metodologi

Metodologi yang dipakai dalam pembuatan skripsi ini adalah:

1. Studi literatur

   Dengan mencari referensi – referensi yang berhubungan dengan perancangan dan pembuatan alat yang akan dibuat.

2. Perancangan dan Pembuatan alat

   Meliputi pembuatan diagram alir alat lalu diteruskan dalam pembuatan PCB, perakitan komponen serta penyolderan. Pembuatan diagram alir perangkat lunak lalu diteruskan dalam pembuatan perangkat lunak.

3. Pengujian sistem / alat

   Dengan melakukian pengujian perblok rangkaian dan kerja seluruh sistem pada alat.

4. Penyusunan laporan skripsi

   Mendokumentasikan referensi – referensi yang didapat, diagram alir alat serta gambar rangkaian alat dan diagram alir perangkat lunak serta

perangkat lunaknya, dan hasil pengujian alat baik perblok maupun secara keseluruhan ke dalam bentuk laporan yang tersusun secara sistematis.

1.6. Sistimatika Penulisan

Penulisan skripsi ini terbagi menjadi lima bab dengan sistematika :

BAB I PENDAHULUAN

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

BAB II TEORI PENUNJANG

Membahas teori – teori dasar penunjang perancangan dan pembuatan alat.

BAB III PERANCANGAN DAN PEMBUATAN ALAT

Membahas tentang perancangan alat baik perangkat keras maupun perangkat lunak dan cara kerja blok diagram.

BAB IV PENGUJIAN ALAT

Mencakup pembahasan tentang proses pengujian alat yang terdiri dari peralatan yang digunakan, langkah kerja, dan analisa hasil pengujian.

BAB V PENUTUP

Berisi kesimpulan dan saran.

BAB II

TEORI PENDUKUNG

2.1 Mikrokontroler AT89S52

Perbedaan mendasar antara mikrokontroller dan mikroprosesor adalah mikrokontroller selain memiliki CPU juga dilengkapi dengan memori input-output yang merupakan kelengkapan sebagai sistem minimum mikrokomputer sehingga sebuah mikrokontoller dapat dikatakan sebagai mikrokomputer dalam keping tunggal *(single chip Microcomputer)* yang dapat berdiri sendiri.

Mikrokontroler AT89S52 adalah mikrokontroler CMOS 8 bit keluaran Atmel dengan 8K *bytes Flash* PEROM (*Programmable and Eraseable Read Only Memory*). Memori ini digunakan untuk menyimpan perintah (instruksi) berstandar MCS-51, sehingga memungkinkan mikrokontroler ini untuk bekerja dalam mode *single chip operation* (mode operasi keping tunggal) yang tidak memerlukan *external memory* (memori luar) untuk menyimpan *source code* tersebut. Fitur lain dari AT89S52 adalah 256 Byte Memori RAM internal, 32 jalur Input – Output (4 buah port paralel I/O)

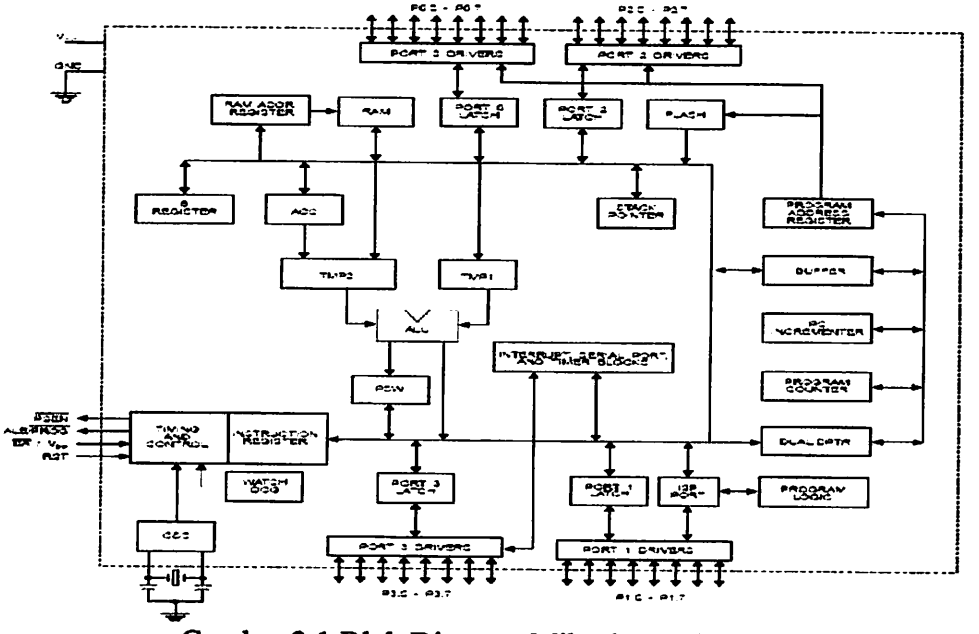
Dalam sistem mikrontroller terdapat dua hal yang mendasar, yaitu: perangkat keras dan perangkat lunak yang keduanya saling terkait dan mendukung. Perangkat lunak berfungsi untuk mengatur dan mengendalikan keseluruhan sistem perangkat keras yang telah dibuat. Sebuah mikrokontroler tidak akan bekerja bila tidak diberikan program kepadanya. Program tersebut memberitahukan apa yang harus dilakukan oleh mikrokontroler.

2.1.1 Arsitektur Mikrokontroler AT89S52

Mikrokontroller AT89S52 mempunyai fitur sebagai berikut:

1. Kompatibel dengan mikrokontroler MCS-51
2. 8 Kbyte Downloadable Flas Memory
3. 3 level program memori lock
4. 256 byte RAM internal
5. 32 pin I/O yang dapat dipakai semua
6. 3 buah timer/ counter 16 bit.
7. Programmable Wacthdog Timer
8. Dual Data Pointer
9. Frekuensi kerja 0 sampai 33 MHz
10. Tegangan operasi 4 Volt sampai 5,5 Volt

Gambar di dibawah ini merupakan blok diagram dari mikrokontroler AT89S52:

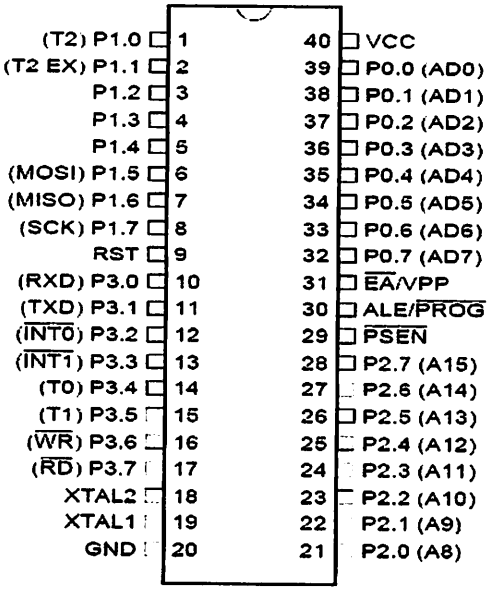


Gambar 2.1 Blok Diagram Mikrokontroler AT89S52

Pada gambar blok diagram AT89C51 terlihat bahwa mikrokontroler ini mempunyai empat port I/O, akumulator, register, RAM internal, stack pointer, Arithmetic Logic Unit (ALU), dan rangkaian oscilator yang membuat mikrokontroler ini dapat beroperasi hanya dengan sekeping IC.

Secara fisik, mikrokontroler AT89C51 mempunyai 40 pin, 32 pin di antaranya adalah pin untuk port masukan/keluaran paralel. Satu port I/O paralel terdiri dari 8 pin, dengan demikian 32 pin tersebut membentuk 4 buah port paralel, yang masing-masing dikenal dengan Port 0, Port 1, Port 2 dan Port 3.

Gambar dibawah menunjukkan susunan pin yang dimiliki oleh AT89S52:



Gambar 2.2 Susunan Pin AT89S52

Berikut penjelasan dari masing-masing pin:

a) Port 1 (pin 1 sampai pin 8)

Port 1 terdapat pada pin 1 sampai pin 8 yang berfungsi sebagai general purpose I/O dengan lebar 8 bit. Beberapa pin pada port 1 juga memiliki fungsi khusus seperti dapat dilihat pada tabel 2.1:

Tabel 2.1 fungsi khusus pin pada port 1

Port Pin	Fungsi Alternatif
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

b) RST (pin 9)

pin ini berfungsi sebagai input untuk melakukan reset terhadap mikrokontroler dan jika RST bernilai high selama minimal 2 machine cycle maka reset akan aktif yang menyebabkan nilai internal egister akan kembali seperti awal mulai bekerja.

c) Port 3 (pin 10 sampai pin 17)

Port 3 merupakan port 8 bit dua arah dengan pull-up internal. Keluaran dari port 3 ini dapat mendayai atau menerima masukan sebanyak 4 masukan TTL. Saat logika 1 dituliskan pada port ini, maka port ini akan dibuat tinggi oleh pull-up internalnya dan port ini dapat dipakai sebagai masukan. Port 3 menerima beberapa sinyal control ketika pemrograman flash memori dan verifikasi. Selaian sebagai port pararel biasa, port 3 juga memiliki fungsi

khusus. Fungsi khusus pada port 3 ini diperlihatkan pada tabel 2.2

Tabel 2.2 fungsi Alternatif port 3

Pin	Fungsi Alternatif
P3.0	RXD Untuk menerima data *port* serial
P3.1	TXD Untuk mengirim data *port* serial
P3.2	INT0 Masukan Interupsi 0
P3.3	INT1 Masukan Interupsi 1
P3.4	T0 Masukan Timer/counter 0
P3.4	T1 Masukan Timer/counter 1
P3.6	WR Jalur penulisan data memori luar
P3.7	RD Jalur pembacaan data memori luar

d) Xtal 1 dan Xtal 2 (pin 19 dan pin 18)

Pin 18 merupakan keluaran dari rangkaian osilasi mikrokontroler. (Xtal 2)

Pin 19 merupakan masukan untuk rangkaian osilasi mikrokontroler. (Xtal 1)

e) GND (pin 20)

Berfungsi sebagai pentanahan (Ground) dari sumber tegangan.

f) Port 2 (pin 21 sampai 28)

Port 1 terdapat pada pin 21 sampai pin 28 yang berfungsi sebagai general purpose I/O dengan lebar 8 bit. Saat pengambilan data dari program memori *external* atau selama pengaksesan data memori *external* yang menggunakan alamat 16 bit, *Port* 2 berfungsi sebagai saluran alamat tinggi (A8 – A15).

Akan tetapi, saat mengakses data memori *external* yang menggunakan alamat 8 bit, *Port* 2 mengeluarkan isi P2 pada *Special Function Register*.

g) PSEN (pin 29)

Program Store Enable adalah pulsa pengaktifan untuk membaca program memori luar. Saat mikrokontroler melaksanakan instruksi dari program memori luar, PSEN akan aktifkan dua kali tiap siklus mesin, kecuali saat mengakses data memori luar

h) ALE (pin 30)

Pin ini dapat berfungsi sebagai *Address Latch Enable* (ALE) yang menahan *low bytes address* pada saat mengakses memori *external*. Sedangkan pada saat *Flash Programming* (PROG) berfungsi sebagai pulsa input selama proses pemrograman.

i) EA (pin 31)

Pada kondisi *low*, pin ini akan berfungsi sebagai *External Access Enable* (EA) yaitu mikrokontroler akan menjalankan program yang ada pada memori *external*. Jika berkondisi *high*, pin ini akan berfungsi untuk menjalankan program yang ada pada memori *internal*. Pin ini juga berfungsi sebagai masukan tegangan pemrograman selama proses pemrograman.

j) Port 0 (pin 32 sampai pin 39)

Pin 32 sampai 39 ialah *Port* 0 yang merupakan saluran I/O 8 bit *open drain* dan dapat juga digunakan sebagai multipleks bus alamat rendah dan bus data selama adanya akses ke memori program *external*. Saat proses pemrograman dan verifikasi, *Port* 0 digunakan sebagai saluran data.

k) VCC (pin 40)

Sebagai tempat sumber tegangan sebesar +5V

2.1.2 Struktur Memori

AT89S52 mempunyai struktur memori yang terdiri atas:

- RAM internal, memori sebesar 256 bytes yang digunakan untuk menyimpan variabel atau data yang bersifat sementara.
- *Special Function Register* (Register Fungsi Khusus), memori yang berisi register-register yang

  mempunyai fungsi-fungsi khusus yang disediakan oleh mikrokontroler seperti *timer*, serial *port* dan lain-lain.
- *Flash* PEROM, memori yang digunakan untuk menyimpan instruksi-instruksi MCS-51.

  AT89S52 mempunyai struktur memori yang terpisah antara RAM *internal* dan *Flash* PEROM-nya. RAM *Internal* dialamati oleh *RAM Address Register* (Register Alamat RAM) sedangkan *Flash* PEROM yang menyimpan perintah-perintah MCS-51 dialamati oleh *Program Address Register* (Register Alamat Program). Dengan adanya struktur memori yang terpisah tersebut, walaupun RAM *Internal* dan *Flash* PEROM mempunyai alamat awal yang sama, yaitu alamat 00H, namun secara fisik kedua memori ini tidak saling berhubungan.

2.1.2.1 RAM Internal

RAM internal terdiri atas :

- Register Bank

AT89S52 mempunyai 8 buah register yang terdiri atas R0 hingga R7. Kedelapan

buah register ini selalu terletak pada alamat 00H hingga 07H pada setiap kali sistem direset. Namun, posisi R0 hingga R7 dapat dipindah ke *Bank* 1 (08H hingga 0FH), *Bank* 2 (10H hingga 17H) atau *Bank*3 (18H hingga 1FH) dengan mengatur bit RS0 dan RS1 pada *Program Status Word* (PSW).

- 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 di-set, *clear*, AND dan OR. Dengan adanya sistem *Bit addressable RAM*, proses yang seharusnya dijalankan dengan tiga *cycle* dapat digantikan dengan sebuah instruksi yang hanya membutuhkan satu *cycle* saja.

- 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 operan merupakan bilangan yang menunjukkan lokasi yang dialamati. Sedangkan pengalamatan secara tak langsung pada lokasi dari RAM ini adalah akses data dari memori ketika alamat memori tersebut tersimpan dalam suatu register R0 atau R1. R0 dan R1 adalah dua buah register pada mikrokontroler berasitektur MCS-51 yang dapat digunakan sebagai *pointer* dari sebuah lokasi memori pada RAM *Internal*.

2.1.2.2 Register Fungsi Khusus

AT89S52 mempunyai 21 *Special Function Register* (Register Fungsi Khusus) yang terletak pada alamat 80H hingga FFH. Beberapa dari register - register ini juga dapat dialamati dengan pengalamatan bit. Register Fungsi

Khusus pada AT89S52 antara lain:

- Akumulator

ACC atau akumulator digunakan sebagai register utama dalam proses aritmatik dan penyimpanan data sementara. Dalam instruksi pemrograman akumulator dituliskan sebagai A.

- Port 0, Port 1, Port 2, Port 3

AT89S52 mempunyai 4 buah *port*, yaitu *Port* 0, *Port* 1, *Port* 2 dan *Port* 3 yang terletak pada alamat 80H, 90H, A0H, dan B0H. Semua *port* ini dapat diakses dengan pengalamatan secara bit sehingga dapat dilakukan perubahan output pada tiap-tiap pin dari *port* ini tanpa mempengaruhi pin-pin yang lainnya. Jika digunakan memori *external* ataupun fungsi-fungsi spesial, seperti *External Interrupt*, Serial ataupun *External Timer*, *Port* 0, *Port* 2, dan *Port* 3 tidak dapat digunakan sebagai *port* dengan fungsi umum. Untuk itu disediakan *Port* 1 yang dikhususkan untuk *port* dengan fungsi umum.

- Register B

Register B digunakan bersama akumulator untuk proses aritmatika, selain itu dapat juga difungsikan sebagai register biasa. Register ini juga bersifat *Bit addressable.*

- Stack Pointer

Stack Pointer merupakan sebuah register 8 bit yang terletak di alamat 81H. Isi dari *Stack Pointer* ini merupakan alamat dari data yang disimpan di *stack*. Jika tidak dilakukan perubahan pada isi dari Register *Stack Pointer*, maka isi register ini adalah 07H sehingga penyimpanan data ke *stack* yang pertama kali adalah

pada alamat 08H.

- Data Pointer

Data Pointer atau DPTR merupakan register 16 bit dan terletak pada alamat 82H untuk DPL dan 83H untuk DPH. DPTR biasa digunakan untuk mengakses *source code* ataupun data yang terletak di memori *external*.

- Serial Data Buffer

AT89S52 mempunyai sebuah *on chip serial port* yang dapat digunakan untuk berkomunikasi dengan peralatan lain yang menggunakan serial *port* juga *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 98

2.1.2.3 Flash PEROM

AT89S52 mempunyai 8 K *bytes Flash* PEROM (*Programmable and Eraseable Read Only Memory*), yaitu ROM yang dapat ditulis ulang atau dihapus menggunakan sebuah perangkat *programmer*. *Flash* PEROM dalam AT89S52 menggunakan *Atmel's High-Density Non Volatile Technology* yang mempunyai kemampuan untuk ditulis ulang hingga 1000 kali dan berisikan perintah standard MCS-51. 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* PEROM-nya. Namun, jika pin EA/VP berlogika nol, mikrokontroler akan aktif berdasarkan program yang ada pada memori *external*. Untuk keamanan program yang ada dalam *Flash* PEROM, AT89S52 mempunyai fasilitas *Lock Bit Protection* yang

terdiri atas:

- Lock Bit 1, instruksi MOVC yang dieksekusi dari memori external untuk membaca isi Flash PEROM tidak dapat dilakukan.
- *Lock Bit* 2, sama dengan *Lock Bit* 1, tetapi isi dari *Flash* PEROM tidak dapat di-*verivy* oleh 89C51 *programmer*.
- *Lock Bit* 3, sama dengan *Lock Bit* 2, tetapi akses ke memori *external* tidak dapat dilakukan.

2.1.3 Metode Pengalamatan

Data atau operan bisa berada di tempat yang berbeda sehingga dikenal beberapa cara untuk mengakses data atau operan tersebut yang dinamakan sebagai mode pengalamatan (*addressing mode*). Mode pengalamatan dalam MCS-51 antara lain yaitu:

1. Mode pengalamatan segera (immediate addressing mode)

   Mode pengalamatan ini terjadi pada sebuah perintah ketika nilai operan merupakan data yang akan diproses. Biasanya operan tersebut selalu diawali dengan tanda '#' seperti pada contoh berikut:

   MOV A,#05H

   Operan yang digunakan pada mode pengalamatan segera juga dapat berupa bilangan bertanda mulai dari –256 hingga +256.

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

   Mode pengalamatan 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. Hanya data pada RAM *internal* dan SFR yang dapat diproses dengan menggunakan mode pengalamatan langsung ini.

3. 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 tetapi dititipkan ke register lain, misalnya:

MOV A,@R0

Dalam instruksi ini register serbaguna R0 dipakai untuk menyimpan lokasi memori, sehingga instruksi ini mempunyai arti memori yang alamat lokasinya tersimpan dalam R0 isinya disalin ke Akumulator. Tanda '@' dipakai untuk menandai lokasi memori yang tersimpan di dalam R0.

Baik *internal* maupun *external* RAM dapat diakses menggunakan mode pengalamatan ini. Register alamat untuk 8 bit yang dapat dipakai adalah R0 dan R1 dari *bank* register, atau *Stack Pointer*. Pada pengalamatan 16 bit dapat menggunakan register DPTR (*Data Pointer*).

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

Mode pengalamatan ini menjadikan register serba guna R0 sampai R7 sebagai tempat penyimpan data yang praktis dan kerjanya sangat cepat, misalnya:

MOV A,R5

Instruksi ini mempunyai arti bahwa data dalam register serbaguna R5 disalin ke Akumulator.

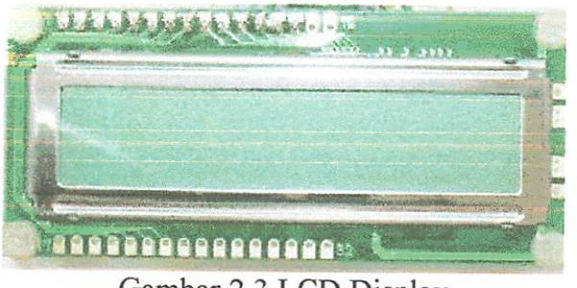
5. Mode pengalamatan kode tidak langsung (*code indirect addressing mode*)

Untuk keperluan ini, MCS-51 mempunyai cara penyebutan data dalam memori program yang dilakukan secara tak langsung, misalnya:

MOVC A,@A+DPTR

Dalam instruksi ini, istruksi 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).

2.2 LCD Display



Gambar 2.3 LCD Display

LCD display sering digunakan sebagai keluaran untuk tampilan grafik atau teks pada aplikasi mikrokontroler atau pun mikroprosesor. LCD display adalah sebuah modul terintegrasi yang tersusun dari bahan Liquid Cristal dependent dan beberapa bahan lain yang terintegrasi menjadi satu modul. LCD display bisa menampilkan beberapa karakter berdasar perintah yang diinginkan melalui metode pemrograman dari sebuah mikrokontroller eksternal. Untuk

menggabungkan modul ini dengan IC terprogram (mikrokontroller) tersebut kita perlu mengetahui fungsi dari masing-masing pin yang dimiliki oleh modul LCD tersebut.

Untuk fungsi masing-masing pin kaki LCD display dapat dilihat pada Tabel 2.3

Tabel 2.3 daftar pin LCD display beserta fungsinya

PIN	Name	Function
1	VSS	Ground voltage
2	VCC	+5V
3	VEE	Contrast voltage
4	RS	Register Select : 0 = Instruction Register; 1 = Data Register
5	R/W	Read/ Write, to choose write or read mode 0 = write mode; 1 = read mode
6	E	Enable 0 = start to lacht data to LCD character 1= disable
7 - 14	DB0 – DB7	Data bus : DB0 = LSB ; DB7 = MSB
15	A	Back Plane Light +
16	K	Back Plane Light - (Ground voltage)

Display karakter pada LCD diatur oleh pin E, RS. Jalur E dinamakan Enable. Jalur ini digunakan untuk memberitahu LCD bahwa anda sedang mengirimkan sebuah data baik data instruksi maupun data karakter yang akan ditampilkan ke LCD. Pengiriman data terjadi ketika terjadi transisi dari sinyal high "1" ke low "0" pada pin E.

Jalur RS adalah jalur Register Select, Ketika RS berlogika low "0", data akan dianggap sebagi sebua perintah atau instruksi khusus (seperti clear screen, posisi kursor dll). Ketika RS berlogika high "1", data yang dikirim adalah data text yang akan ditampilkan pada display LCD.

Jalur RW adalah jalur kontrol Read/ Write. Ketika RW berlogika low (0), maka informasi pada bus data akan dituliskan pada layar LCD. Ketika RW berlogika high "1", maka program akan melakukan pembacaan memori dari LCD. Sedangkan pada aplikasi umum pin RW selalu diberi logika low "0". Pada akhirnya, bus data terdiri dari 4 atau 8 jalur (bergantung pada mode operasi yang dipilih oleh user). Pada kasus bus data 8 bit, jalur diacukan sebagai DB0 s/d DB7 sedangkan untuk mode 4 bit bus data yang digunakan adalah DB4 s/d DB7.

Beberapa perintah dasar yang harus dipahami dalam inisialisasi LCD adalah sebagai berikut:

- Function Set: Mengatur interface lebar data, jumlah dari baris dan ukuran font karakter

Tabel 2.4 Bit Instruksi Function Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

CATATAN:

X : Don't care

DL: Mengatur lebar data

DL=1, Lebar data interface 8 bit (DB7 s/d DB0)

DL=0, Lebar data interface 4 bit (DB7 s/d DB4)

Ketika menggunakan lebar data 4 bit, data harus dikirimkan dua kali

N: Pengaktivan baris

N=0, 1 baris ; N=1, 2 baris

F: Penentuan ukuran font karakter

F=0, 5x7 ; F=1, 5x8

- Entry Mode Set : Mengatur increment/ decrement dan mode geser

Tabel 2.5 Bit Instruksi Entry Mode Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

Catatan:

I/D: Increment/ decrement dari alamat DDRAM dengan 1 ketika kode karakter dituliskan ke DDRAM.

I/D = "0", decrement

I/D= "1", increment

S: Geser keseluruhan display kekanan dan kekiri

S=1, geser kekiri atau kekanan bergantung pada I/D

S=0, display tidak bergeser

- Display On/ Off Cursor : Mengatur status display ON atau OFF, cursor ON/ OFF dan fungsi Cursor Blink

Tabel 2.6 Bit Instruksi Display On/Off Cursor

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

D : Mengatur display

D = 1, Display is ON ; D = 0, Display is OFF

Pada kasus ini data display masih tetap berada di DDRAM, dan dapat ditampilkan kembali secara langsung dengan mengatur D=1.

C : Menampilkan kursor

C = 1, kursor ditampilkan

C = 0, kursor tidak ditampilkan

B : Karakter ditunjukkan dengan kursor yang berkedip

B=1, kursor blink

- Clear Display : Perintah untuk hapus layar

Tabel 2.7 Bit perintah Clear Display

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

- Geser Kursor dan Display: perintah untuk geser posisi kursor atau display ke kanan atau kekiri tanpa menulis atau baca data display. Fungsi ini digunakan untuk koreksi atau pencarian display

Tabel 2.8 Bit Perintah Geser Kursor dan Display

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

Catatan : x = Dont care

Tabel 2.9 Nilai S/C dan R/L

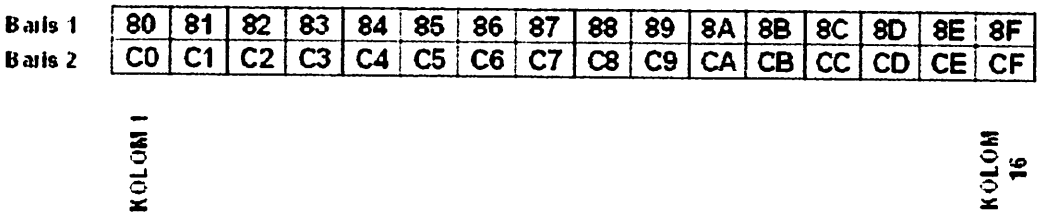
S/C	R/L	Note
0	0	Shift cursor position to the left
0	1	Shift cursor position to the right
1	0	Shift the entire display to the left
1	1	Shift the entire display to the right

- Posisi Kursor: perintah untuk menentukan posisi kursor tampilan

Tabel 2.10 Bit Perintah Posisi Kursor

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	Set	Set	Set	Set	Set	Set	Set	Set

Keterangan: nilai DB7 s/d DB0 dapat dilihat pada Gambar 2. 4



Gambar 2.4 Letak posisi Kursor pada LCD Dysplay 2 x 16

Dari gambar 2.4 dapat ditemtukan nilai DB7 s/d DB0 untuk perintah menentukan posisi kursor. Misalnya kita ingin menampilkan karakter pada baris 1 maka nilai DB7 s/d DB0 dapat dilihat pada tabel 2.11

Tabel 2.10 nilai DB7 s/d DB0 untuk menampilkan karakter pada baris 1

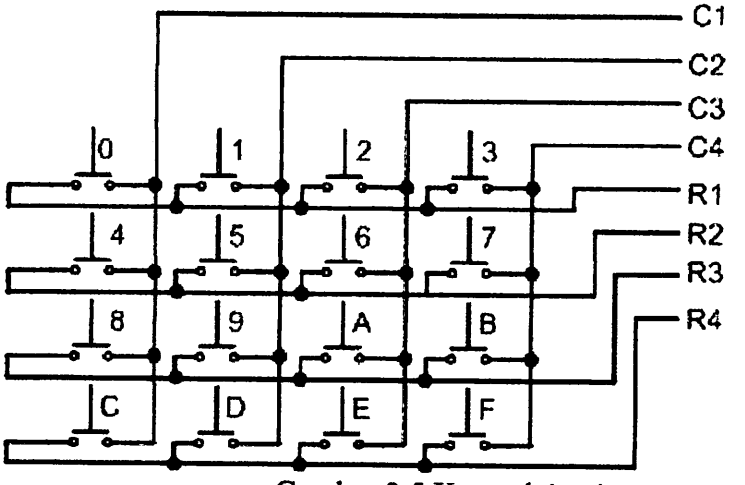
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	0	0	0	0	0	0

2.3 Keypad

keypad biasa digunakan sebagai inputan oleh mikrokontroler, keypad sebenarnya terdiri dari beberapa tombol / switches yang dihubungkan dengan kolom dan baris seperti dapat dilihat pada gambar 2.3. dimana tombol push button terdiri dari 2 kaki, yang lalu satu kaki dihubungkan dengan kolom yang sama dan kaki lainnya dihubungkan dengan baris yang sama dengan tombol yang lain, jadi ketika terjadi penekanan tombol salah satu kolom dan salah satu

baris akan terhubung singkat. Karena itulah walaupun keypad tersebut terdiri dari 16 tombol tapi kita hanya butuh 8 pin saja untuk mengaturnya yaitu 4 pin baris dan 4 pin kolom.

Nantinya dalam mikrokontroler untuk bisa melakukan pendeteksian tombol bisa dilakukan dengan scanning dalam baris yang berfungsi sebagai nible output dan deteksi penekanan tombol dalam kolom yang berfungsi sebagai nible input atau sebaliknya. Dalam scanning baris salah satu baris di beri sinyal nol (low) dan mikrokontroler akan membaca apa ada penekanan tombol dengan melihat sinyal low pada kolom di baris tersebut.



Gambar 2.5 Keypad 4 x 4

2.4 Akord Gitar

2.4.1 Pengertian Akord

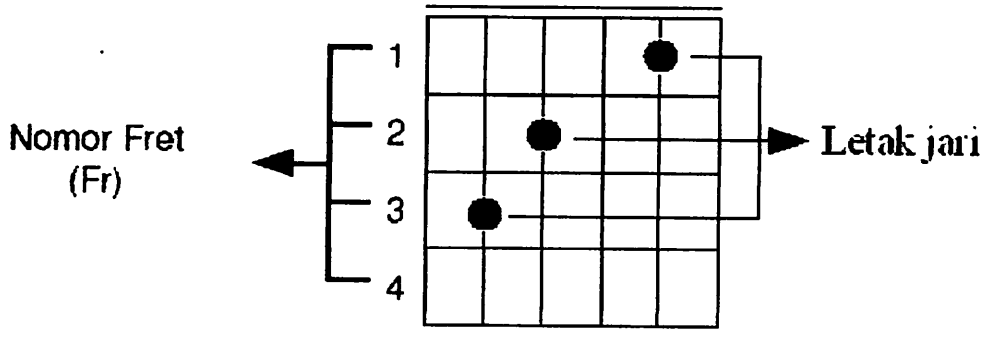
Akord adalah kumpulan tiga nada atau lebih yang bila dimainkan secara bersamaan terdengar harmonis. Akord bisa dimainkan secara terputus-putus ataupun secara bersamaan. Akord ini digunakan untuk mengiringi suatu lagu. Ketika Anda menekan tiga tuts piano C, E dan G secara bersamaan, ini berarti

anda sudah memainkan akord. Contoh alat musik lainnya yang bisa memainkan akord adalah gitar (akustik dan listrik), organ, electone.

Akord itu banyak macamnya. Antara lain akord mayor, akord minor, akord dominan septim, akord diminished, akord augmented, akord minor 6, akord mayor 7, akord suspended dan masih banyak yang lainnya. Akord yang paling sering dipakai dalam suatu lagu yang sederhana adalah akord mayor, akord minor dan akord dominan septim. Akord lainnya digunakan untuk memperindah atau mengubah kualitas suatu lagu. Penyisipan akord yang berbeda akan memberikan efek rasa yang berbeda dalam iringan suatu lagu.

2.4.2 Cara Membaca Akord

Contoh cara membaca akord gitar dapat dilihat pada gambar 2.4 dimana bulatan hitam adalah letak posisi jari kita dalam fret dan senar gitar dan nomor 1,2,3, dan seterusnya adalah nomor dari fret



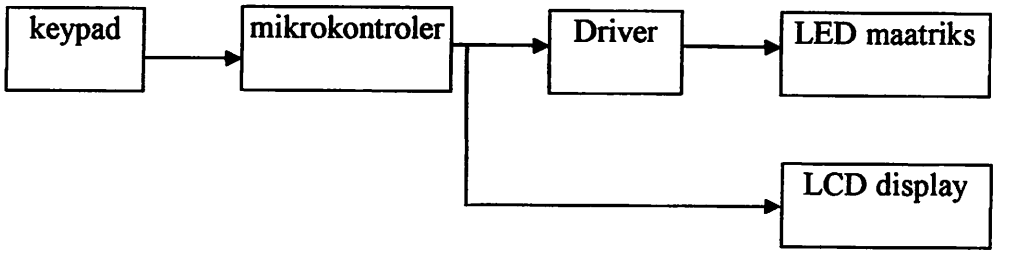
Gambar 2.6 cara baca akord dan letak posisi jari

BAB III

PERANCANGAN ALAT

3.1 Pendahuluan

Alat yang dirancang dalam skripsi ini adalah penampil simulasi akord gitar melalui LED matriks dan LCD. Inputan berasal dari keypad yang dibaca oleh mikrokontroler lalu mikrokontroler akan menyalakan LED matriks untuk mensimulasikan posisi jari dari kord gitar, dan menampilkan keterangan kord yang ditampilkan oleh LED matris di LCD. Adapun diagram blok dari alat yang akan dibuat dapat dilihat pada gambar 3.1 :



Gambar 3.1 Blok Diagram Rangkaian

Fungsi dari masing-masing blok diagram adalah sebagai berikut:

- Keypad

  Disini keypad berfungsi sebagai masukan dari alat yang menentukan akord apa yang akan ditampilkan.

- Mikrokontroler AT89S52

  Untuk memproses masukan dari keypad lalu menampilkannya ke LCD display dan LED Matriks

- LCD Display

  Untuk tampilan teks dari akord yang ditampilkan

- LED Matriks display

  Mensimulasikan posisi penekanan jari dari akord

3.2 Perancangan Perangkat Keras

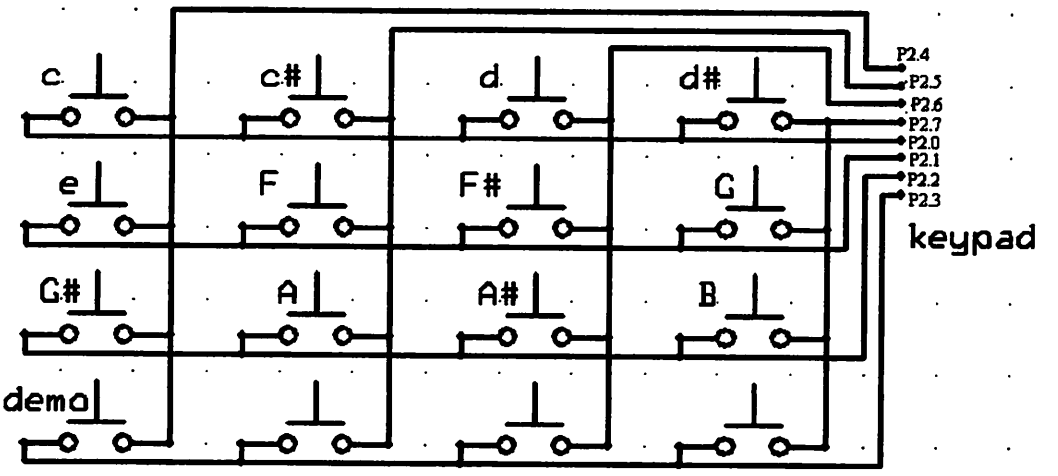
Langkah berikutnya adalah membuat perangkat keras dari rangkaian setiap blok. Rangkaian-rangkaian yang akan dibuat yaitu:

- Rangkaian keypad untuk dihubungkan dengan mikrokontroler sebagai input
- Rangkaian minimum sistem mikrokontroler AT89S52
- Menghubungkan rangkaian LED Matriks dan driver ke mikrokontroler
- Membuat rangkaian LCD Display yang dihubungkan ke mikrokontroler

3.2.1 Rangkaian Keypad

Rangkaian *keypad* berfungsi sebagai masukan bagi sistem mikrokontroler. Rangkaian *keypad* ini dirancang terdiri dari 16 tombol (untuk menandakan 12 nada pada akord dan 1 buat demo) yang disusun dalam bentuk matriks 4x4, seperti terlihat pada Gambar 3,2, sehingga meskipun terdiri dari 16 tombol hanya diperlukan 8 pin saja buat mengaturnya yaitu 4 pin buat baris dan 4 pin buat kolom.Pola pembacaan keypad adalah apabila ada penekanan salah satu tombol maka ada satu kolom dan satu baris yang terhubung singkat . Dengan demikian kita bisa melakukan scanning di baris dengan cara memberikan sinyal low '0' di salah satu baris dan identifikasi penekanan tombol di kolom dengan cara

membaca apakah ada sinyal low '0' di salah satu kolom pada baris yang di beri sinyal low '0' tersebut yang menandakan adanyak penekanan tombol atau sebaliknya dengan melakukan scanning kolom yaitu memberikan sinyal low '0' pada salah satu kolom dan membaca apa ada sinyal low '0' pada baris di kolom tersebut yang menandakan adanya penekanan tombol.



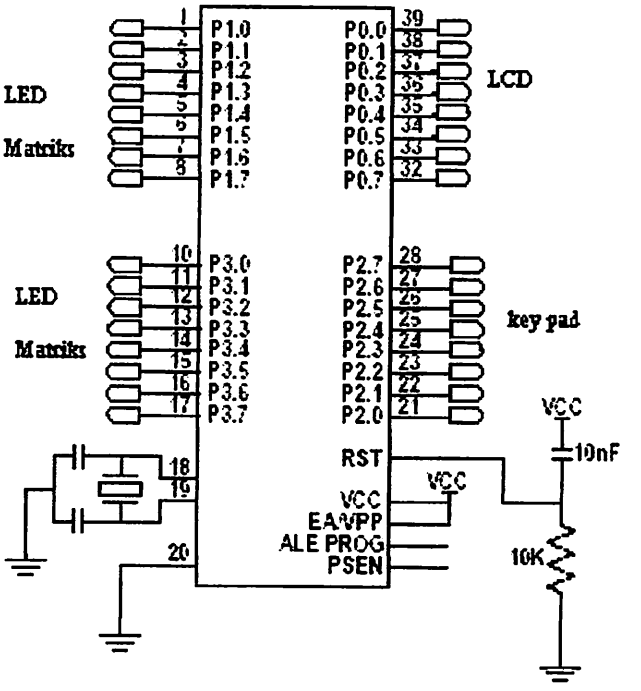
Gambar 3.2 Keypad

Cara kerja dari keypad diatas ada 2 yaitu :

- scan baris : pertama kita kasih logika low pada R1 bila ada penekanan tombol pada baris satu kolom satu maka C1 juga akan low bila kita tekan baris satu kolom dua maka C2 yang low dan seterusnya.
- scan kolom : pertama kita beri logika low pada C1 bila ada penekanan pada baris satu kolom satu maka R1 akan low, bila kita tekan baris dua kolom satu maka R2 yang low dan seterusnya.

3.2.2 Rangkaian Mikrokontroler

Rangkaian mikrokontroler yang akan digunakan adalah rangkaian single chip dengan memory internal, dengan port 2 sebagai masukan dari keypad, port 3 sebagai data display LED matriks, port satu buat proses scanning data display LED matriks, dan port 0 sebagai keluaran buat display LCD. Gambar rangkaian mikrokonroler dapat dilihat pada gambar 3.3 :



Gambar 3.3 Rangkaian Mikrokontroler

Adapun komponen pembentuk minimum sistem mikrokontroler adalah sebagai berikut:

- Rangkaian *clock*

  Kecepatan proses yang dilakukan oleh mikrokontroler ditentukan oleh sumber *clock* (pewaktuan) yang mengendalikan mikrokontroler tersebut.

Sistem yang dirancang akan menggunakan osilator internal yang sudah tersedia di dalam mikrokontroler AT89S52. Untuk menentukan frekuensi osilatornya, cukup dengan cara menghubungkan kristal pada pin XTAL1 dan XTAL2 serta dua buah kapasitor ke ground. Dengan menggunakan kristal maka dapat dihitung waktu yang diperlukan untuk satu siklus mesin

f = 12 Mhz

sehingga $T = \frac{1}{f}$

$$T = \frac{1}{12.000.000}$$

$$T = 0{,}0833 \mu S$$

Maka untuk satu siklus mesin dari mikrokontroler besarnya adalah :

$$Time = 12 \times T$$

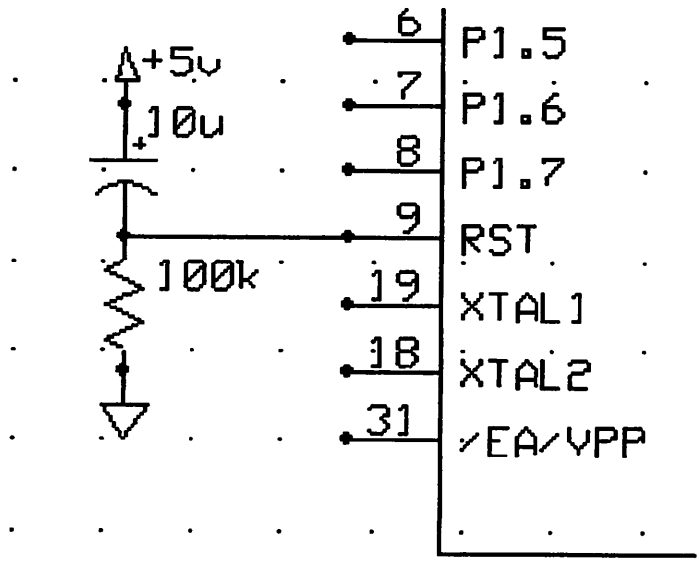
$$Time = 12 \times 0{,}0833 \mu S$$

$$Time = 1 \mu S$$

- Rangkaian reset

Reset pada mikrokontroler merupakan masukan aktif high '1', pulsa transisi dari rendah '0' ke tinggi '1' akan mereset mikrokontroler dan menjalankan program pada alamat 0000h.

Rangkaian reset bertujuan agar mikrokontroler dapat menjalankan proses dari awal. Rangkaian reset untuk mikrokontroler dirancang agar mempunyai kemampuan power on reset, yaitu reset yang terjadi pada saat sistem dinyalakan untuk pertama kalinya. Reset juga dapat dilakukan secara manual dengan menekan tombol reset yang berupa switch push button.



Gambar 3.4 Rangkaian Reset

Frekuensi dari rangkain reset diatas adalah :

$$fo = \frac{1}{RC}$$

$$fo = \frac{1}{10^5 x 10^{-5}}$$

$$fo = 1Hz$$

Maka periode *clock*:

$$T = \frac{1}{fo}$$

$$T = \frac{1}{1Hz}$$

$$T = 1\,\text{detik}$$

Sesuai dengan *datasheet* AT89S52 telah dicantumkan bahwa $t_{reset(min)}$ adalah sebesar 2 siklus mesin. Setiap 1 siklus mesin membutuhkan 12 siklus *clock*, maka untuk mencapai 2 siklus mesin diperlukan 24 siklus *clock*. Adapun perhitungan $t_{reset(min)}$ untuk 24 siklus *clock* adalah sebagai berikut :

$$t_{reset(\min)} = \frac{24}{f_{crystal}}$$

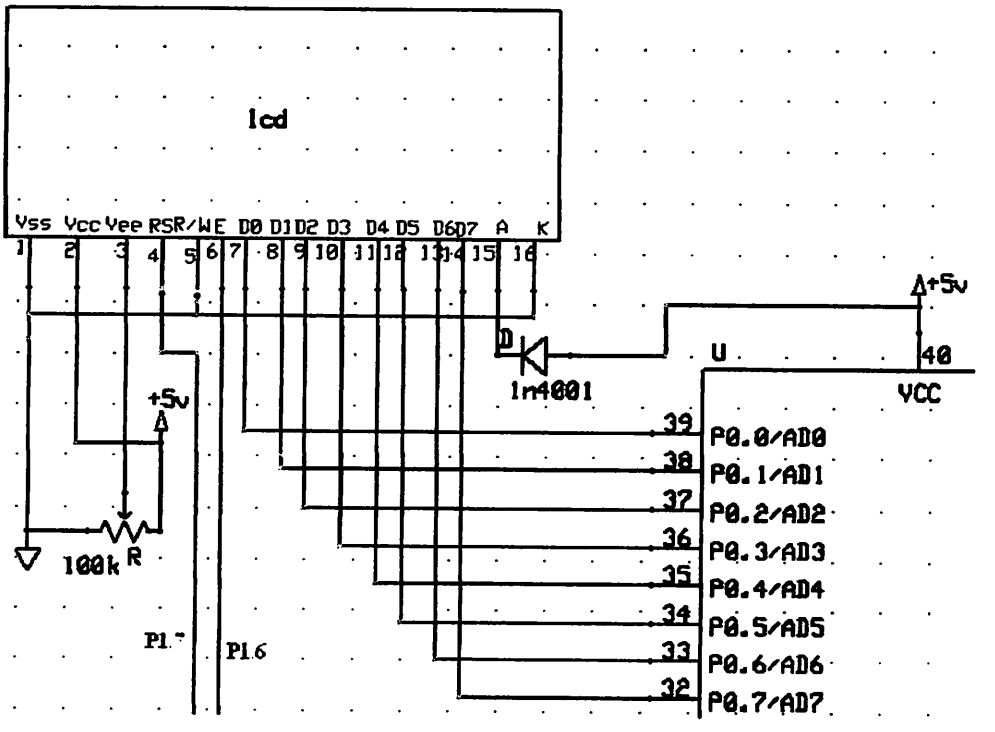
$$t_{reset(\min)} = \frac{24}{12.000.000}$$

$$t_{reset(\min)} = 0{,}000002 \det ik$$

Berdasarkan hasil perhitungan diatas, maka komponen resistor 100KΩ dan kapasitor 10μF dapat dijadikan sebagai komponen pembentuk rangkaian *power on reset*

3.2.3 Rangkaian LCD

Rangkaian LCD ini untuk memberi keterangan akord apa yang di tampilkan oleh LED matriks.



Gambar 3.5 Rangkaian LCD

Keterangan Gambar:

1. D0 sampai D7: kaki pin 7 sampai dengan pin 14 di hubungkan ke port 0 mikrokontroler
2. VSS : pin 1 di hubungkan ke ground
3. VCC: pin 2 di hubungkan ke sumber tegangan +5V
4. VEE: pin 3 pengaturan kontras di hubungkan ke variabel resistor
5. RS: pin 4 di hubungkan ke P1.0 mikrokontroler
6. R/W: pin 5 di hubungkan ke Ground
7. E (enable): di hubungkan ke P1.1 mikrokontroler
8. A : backlight plus di hubungkan ke dioda
9. K : backlight minus di hubungkan ke ground

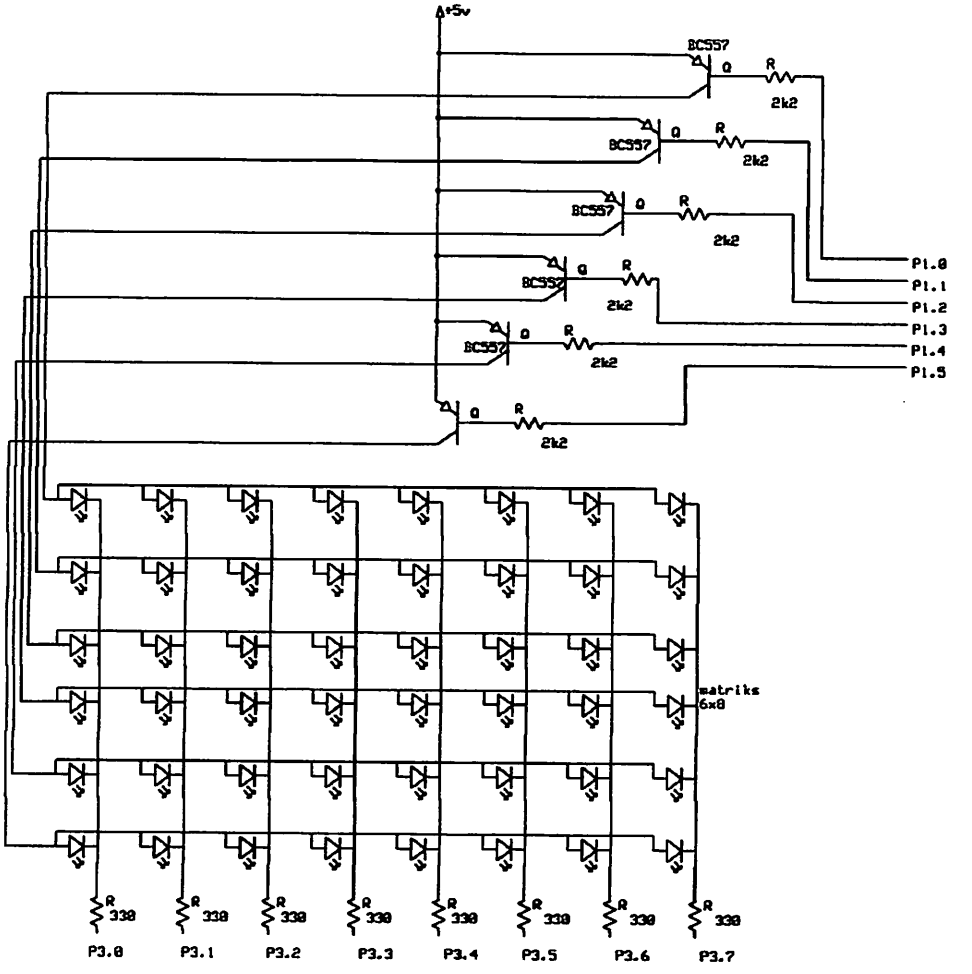
3.2.4 Rangkaian LED Matriks

Rangkaian *display* berfungsi untuk menampilkan variasi akord gitar sesuai dengan penekanan tombol yang dilakukan pada rangkaian *keypad*. *Switching transistor* dipakai untuk melakukan proses *scanning* dengan cara menghidupkan satu per satu transistor ini dengan cepat. Transistor yang dipakai disini adalah transistor PNP BC557 yang merupakan transistor untuk keperluan umum (*general purpose transistor*). Basis dari transistor ini dihubungkan dengan pin-pin keluaran mikrokontroler (P1.0...P1.5) melalui resistor 2,2KΩ dan emiter transistor dihubungkan dengan sumber tegangan Vcc (+5V).

Cara kerja rangkaian *display* ini adalah sebagai berikut:

Pada saat pertama kali mikrokontroler diaktifkan atau pada kondisi sesudah

reset, semua *port* AT89S51 adalah dalam kondisi '*high*'. Pada kondisi ini, maka semua *switching* transistor dalam keadaan '*off*', sehingga semua LED tidak menyala. Apabila kemudian mikrokontroler mengirimkan sinyal '*low*' pada basis transistor, maka sekarang transistor dalam kondisi '*on*'. Dalam keadaan ini, nyala tidak-nya LED tergantung pada sinyal yang dikirimkan pada bagian katoda-nya (*Port 3* pada mikrokontroler). Jika sinyal yang dikirimkan pada LED tersebut adalah '*high*' (logika 1), maka LED dan segment yang bersangkutan tidak akan menyala. Namun jika sinyal yang dikirimkan adalah '*low*' (logika 0), maka LED dan *segment* yang bersangkutan akan menyala.



Gambar 3.6 Rangkaian LED matriks

3.3 Perancangan Program Untuk mikrokontroler

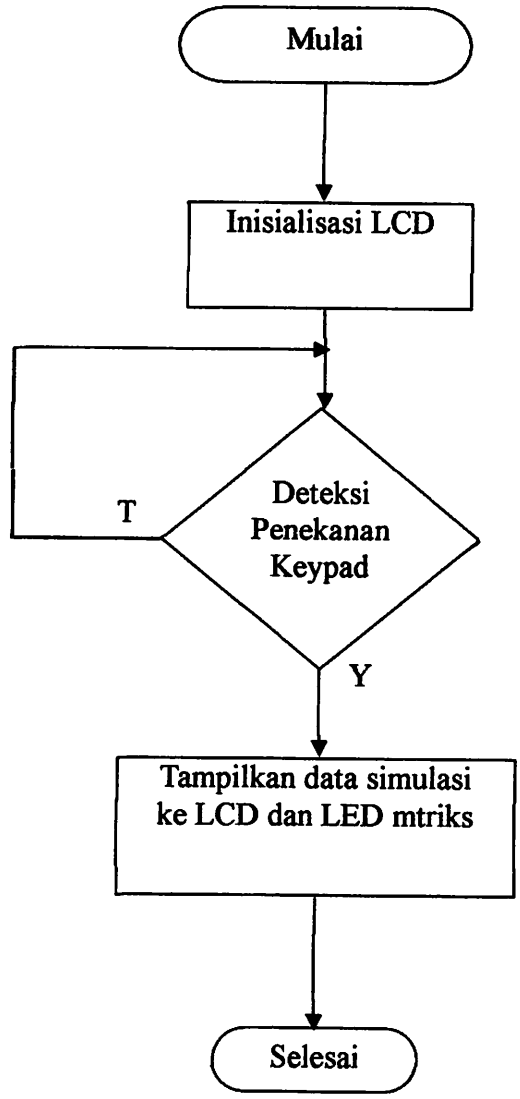
Setelah melakukan perancangan rangkaian perangkat keras sistem, maka langkah selanjutnya adalah membuat program untuk menjalankan alat tersebut. Program untuk mikrokontroler AT89C51 ini dibuat menggunakan bahasa *assembler* ASM51. Sesudah di-*compile* menjadi file hexa (*.hex), program ini kemudian di*download* ke dalam mikrokontroler menggunakan *mikrokontroller programmer*.

Untuk mempermudah pembuatan program ini, maka untuk tiap-tiap program dibuat terlebih dahulu sebuah diagram alir yang menggambarkan urutan kerja dari program tersebut. Setelah itu, barulah menyusun program sesuai dengan diagram alir yang telah dibuat.

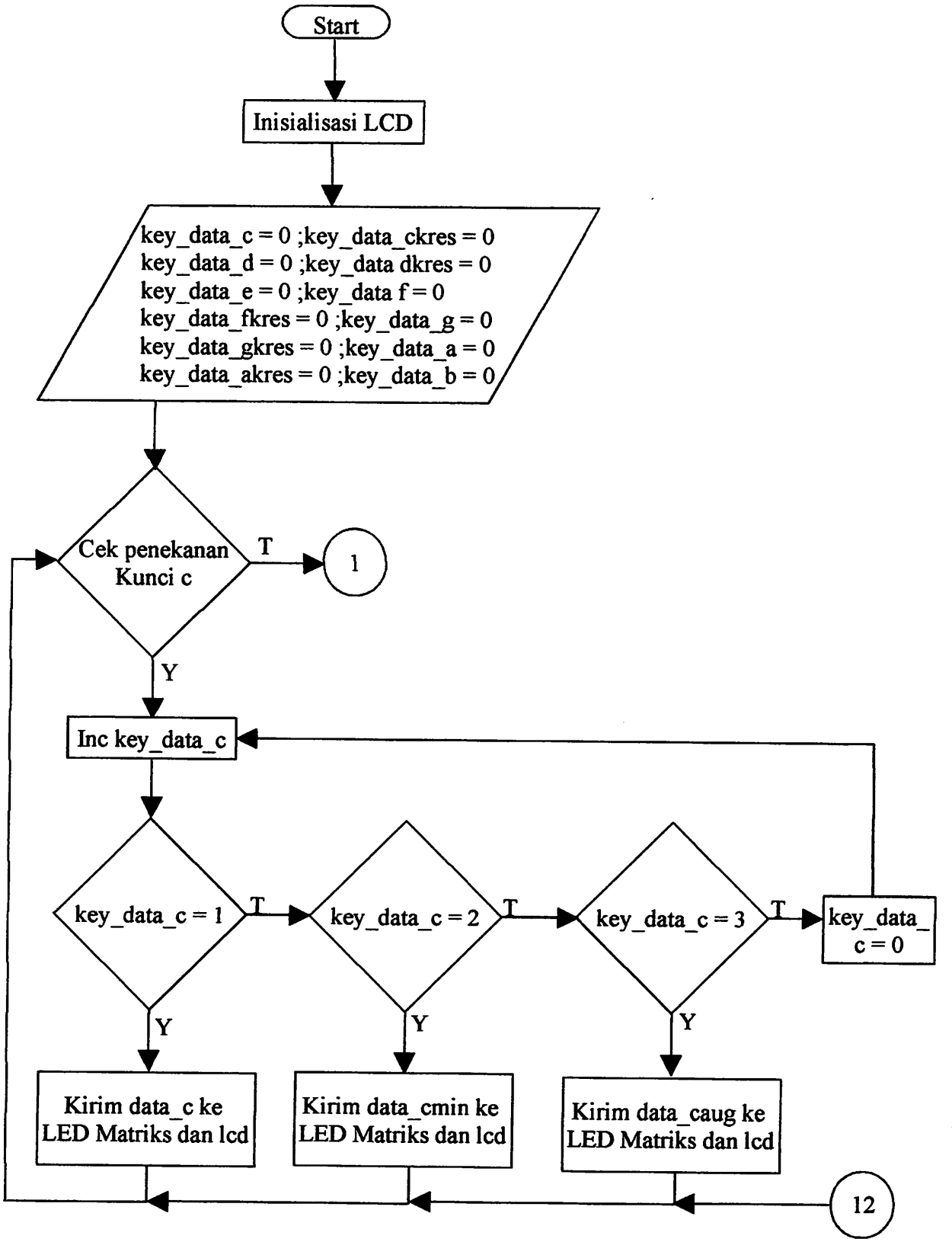
3.3.1 Diagram Alir program

Untuk mendukung agar perangkat keras berfungsi sesuai dengan perencanaan, maka diperlukan perangkat lunak sebagai penunjangnya, yang berfungsi untuk mengatur dan mengendalikan keseluruhan sistem perangkat keras yang telah dibuat. Sebuah mikrokontroler tidak akan bekerja bila tidak diberikan program kepadanya. Program tersebuat memberitahukan apa yang harus dilakukan oleh mikrokontroler. Sistem aplikasi mikrokontroler AT89S52 ini dapat mengatur dan mengendalikan keseluruhan sistem apabila ada urutan instruksi yang mendefinisikan secara jelas urutan tugas yang harus dikerjakan. Urutan instruksi ini sangat penting untuk didefinisikan, karena mikrokontroler bekerja secara pasti berdasarkan urutan instruksi ini. Susunan logika perancangan yang salah tidak dapat diketahui oleh mikrokontroler. Selama

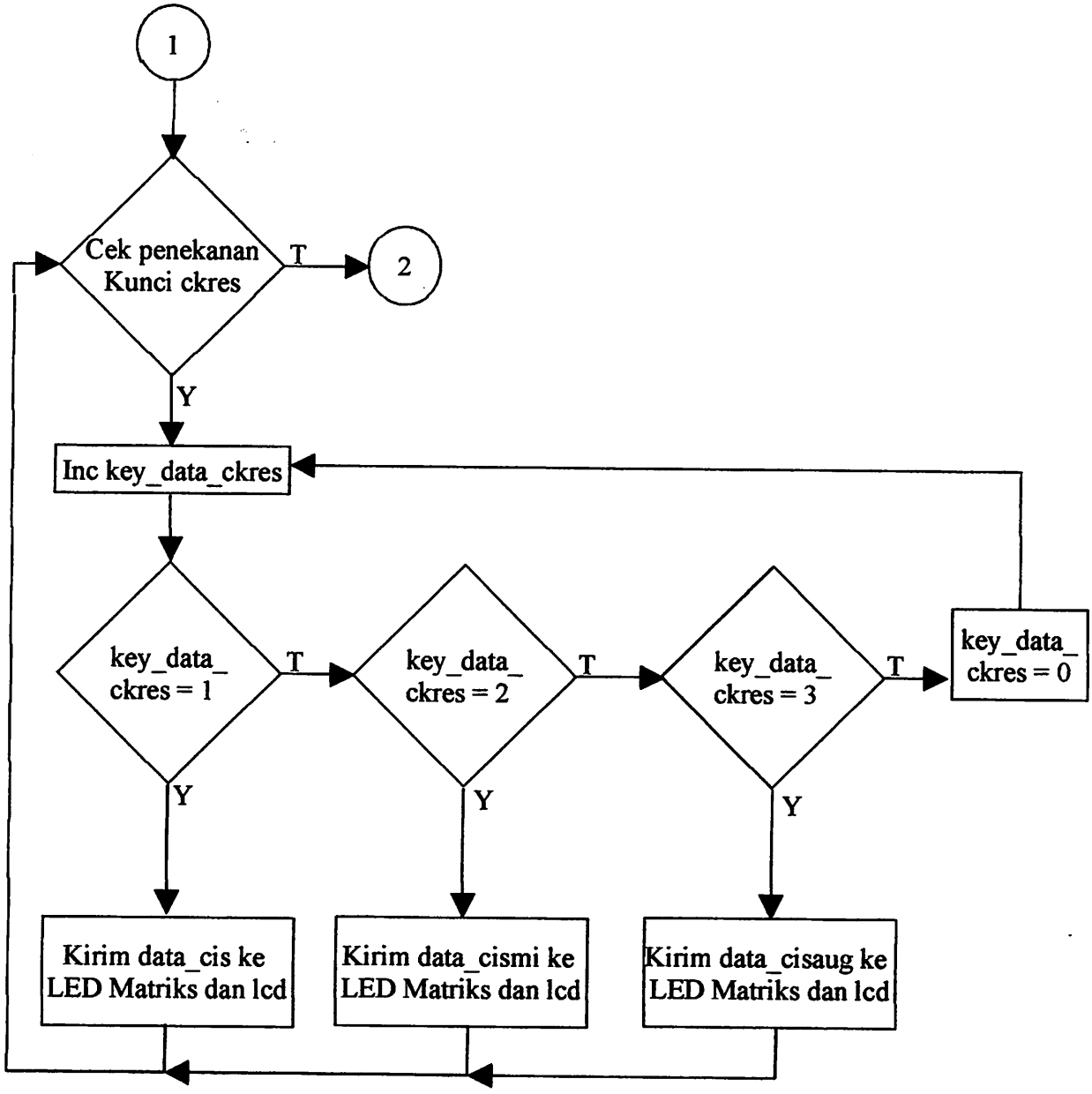
instruksi yang diterima sesuai dengan aturannya, mikrokontroler akan tetap mengerjakan instruksi tersebut. Kesalahan seperti ini baru diketahui ketika kerja sistem aplikasi tidak sesuai dengan spesifikasi awal. Oleh karena itu, perancangan perangkat keras sangat menentukan keberhasilan pembuatan perangkat lunak. Berikut ini diagram alir dari perangkat lunak yang akan di gunakan oleh sistem



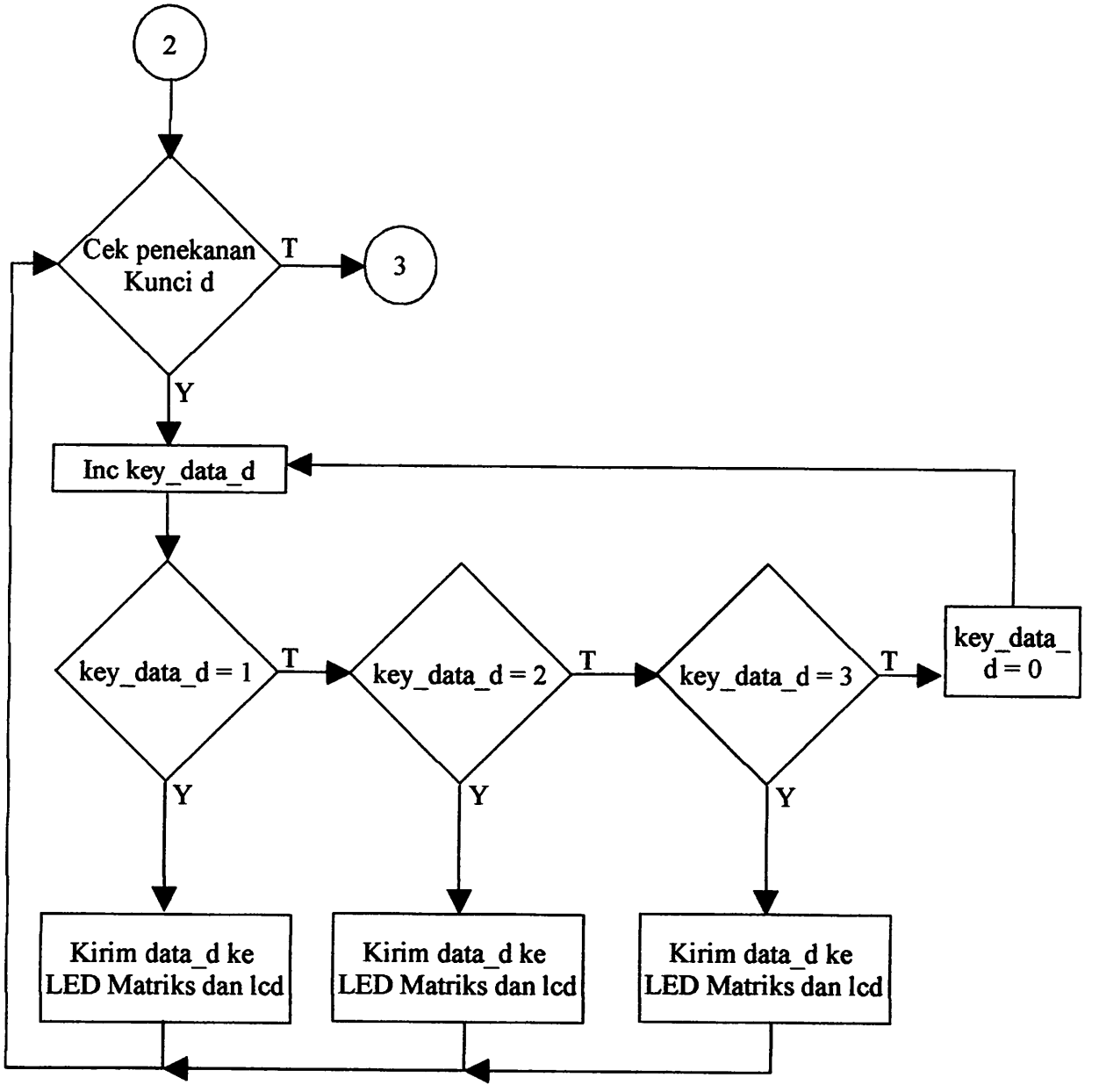
Gambar 3.7 Diagram Alir Alat



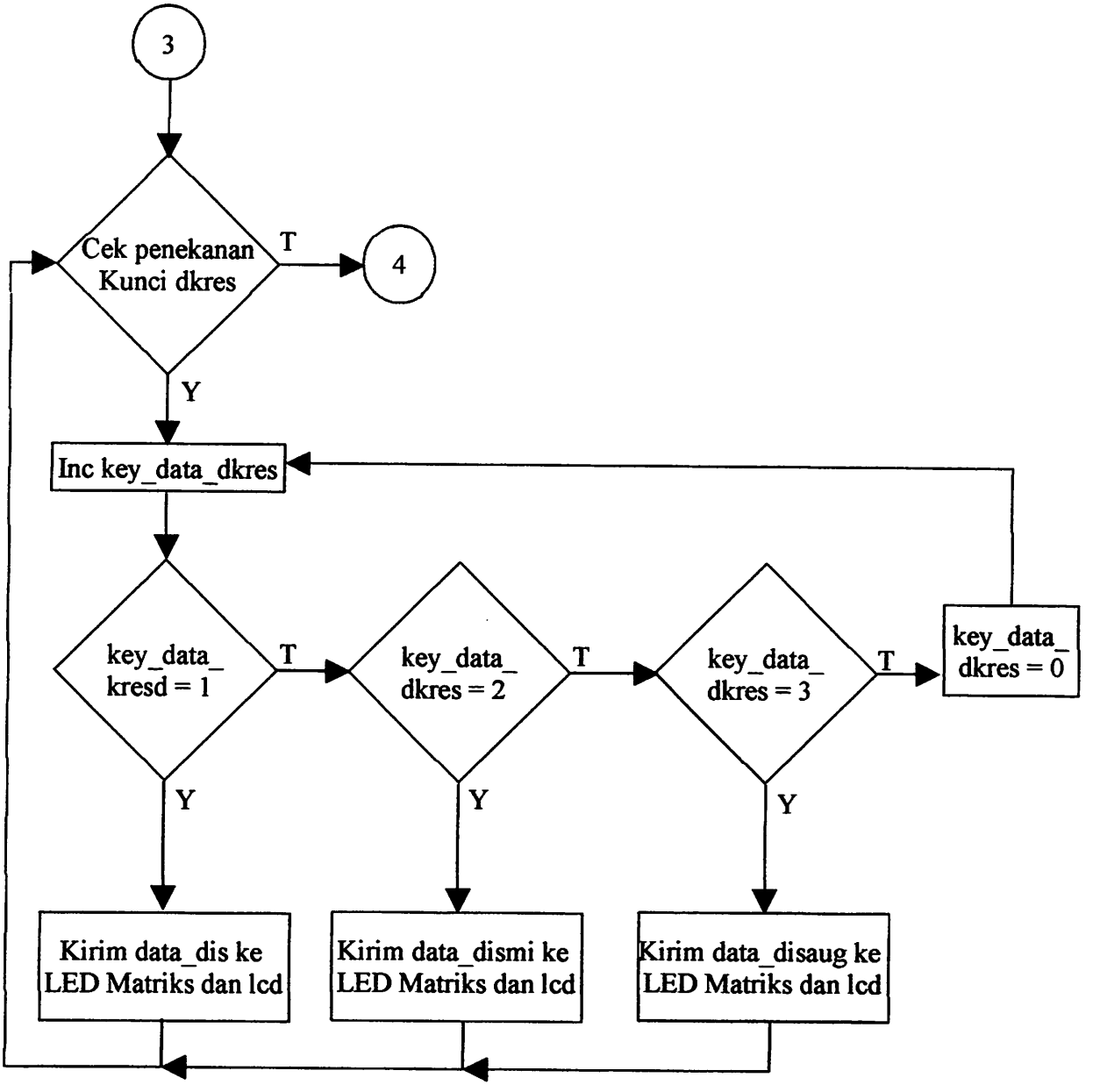
Gambar 3.8 Diagram Alir Program Untuk Cek Keypad C



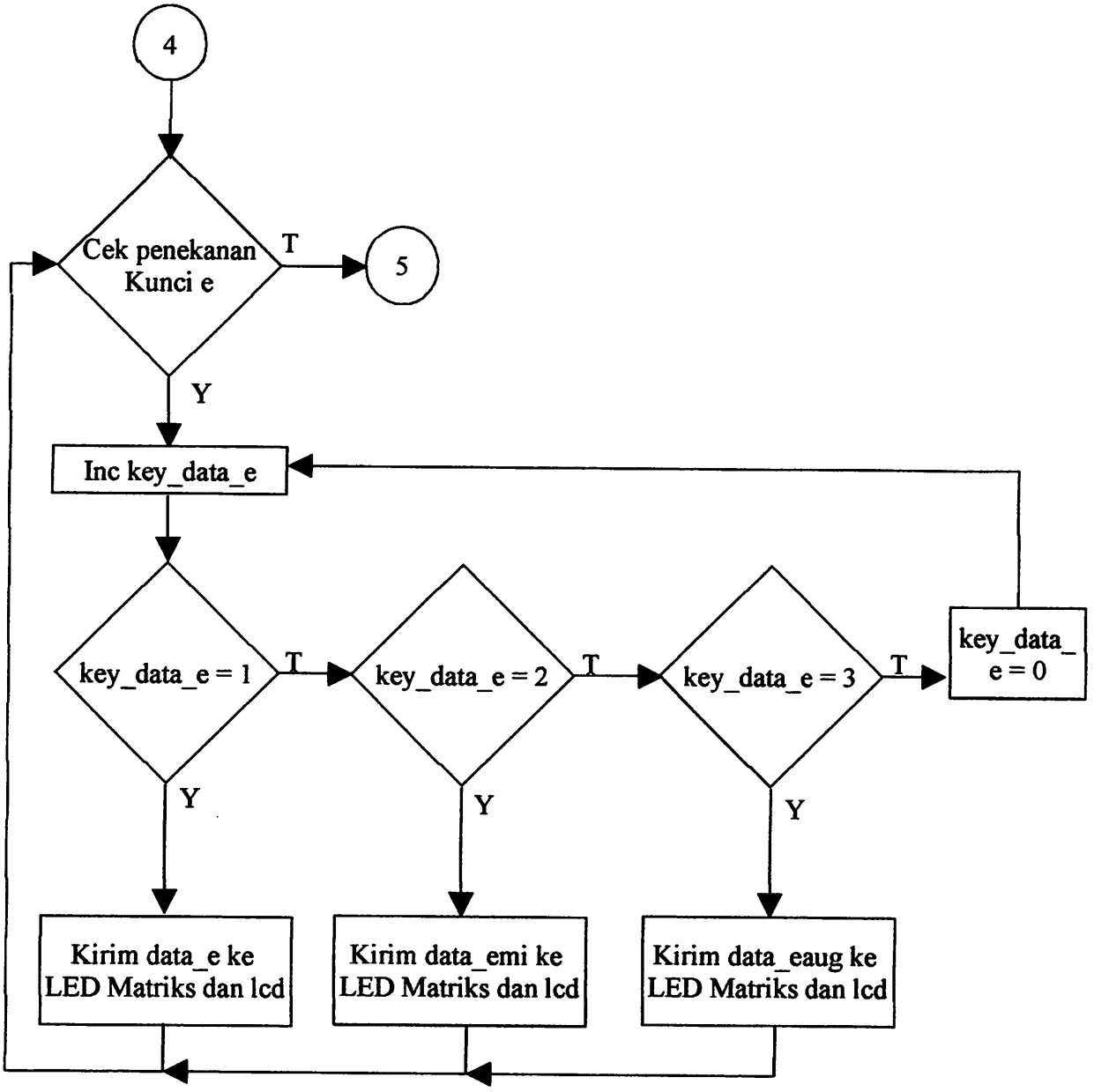
Gambar 3.9 Diagram Alir Program Untuk Cek Keypad C#



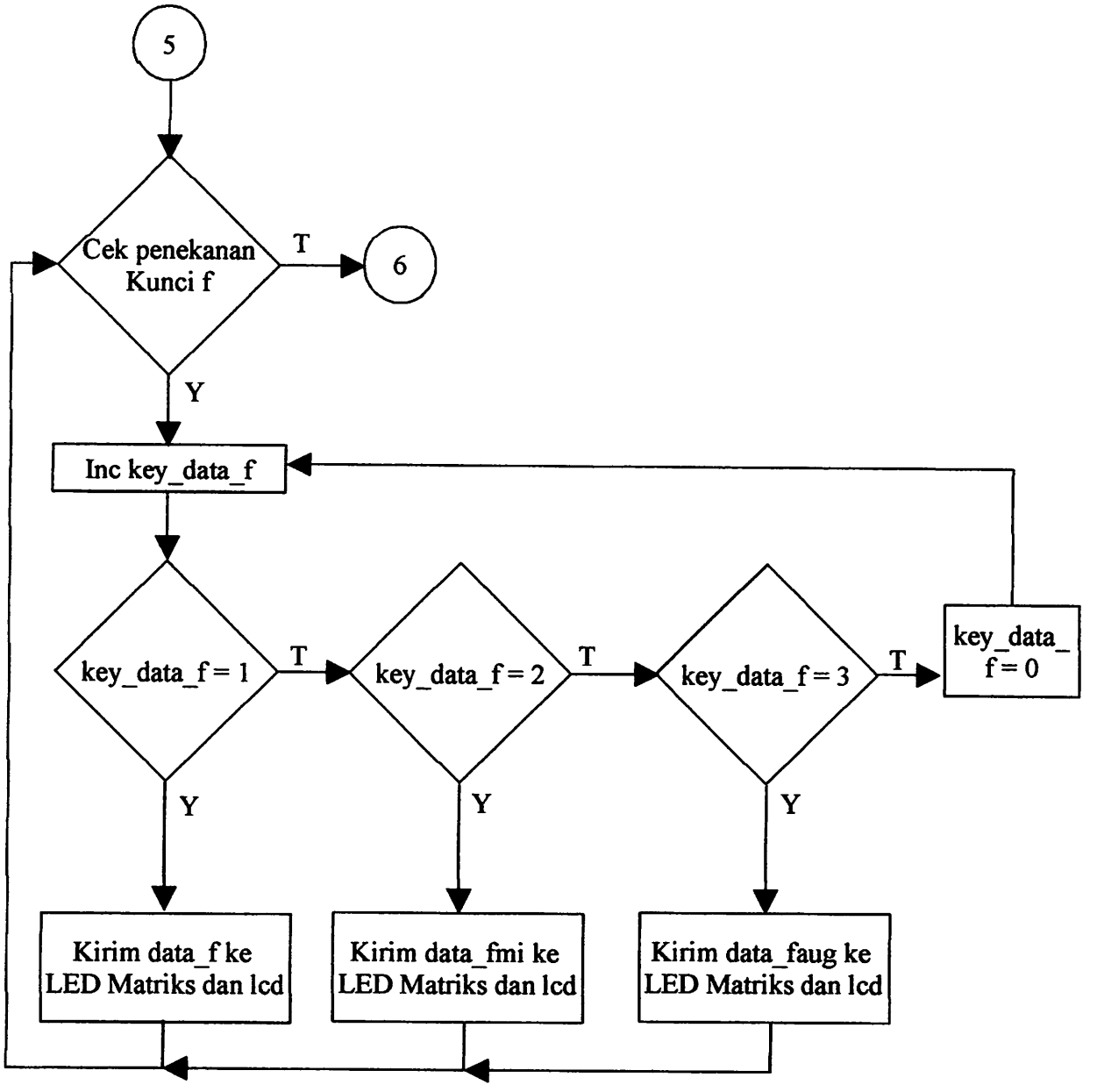
Gambar 3.10 Diagram Alir Program Untuk Cek Keypad D



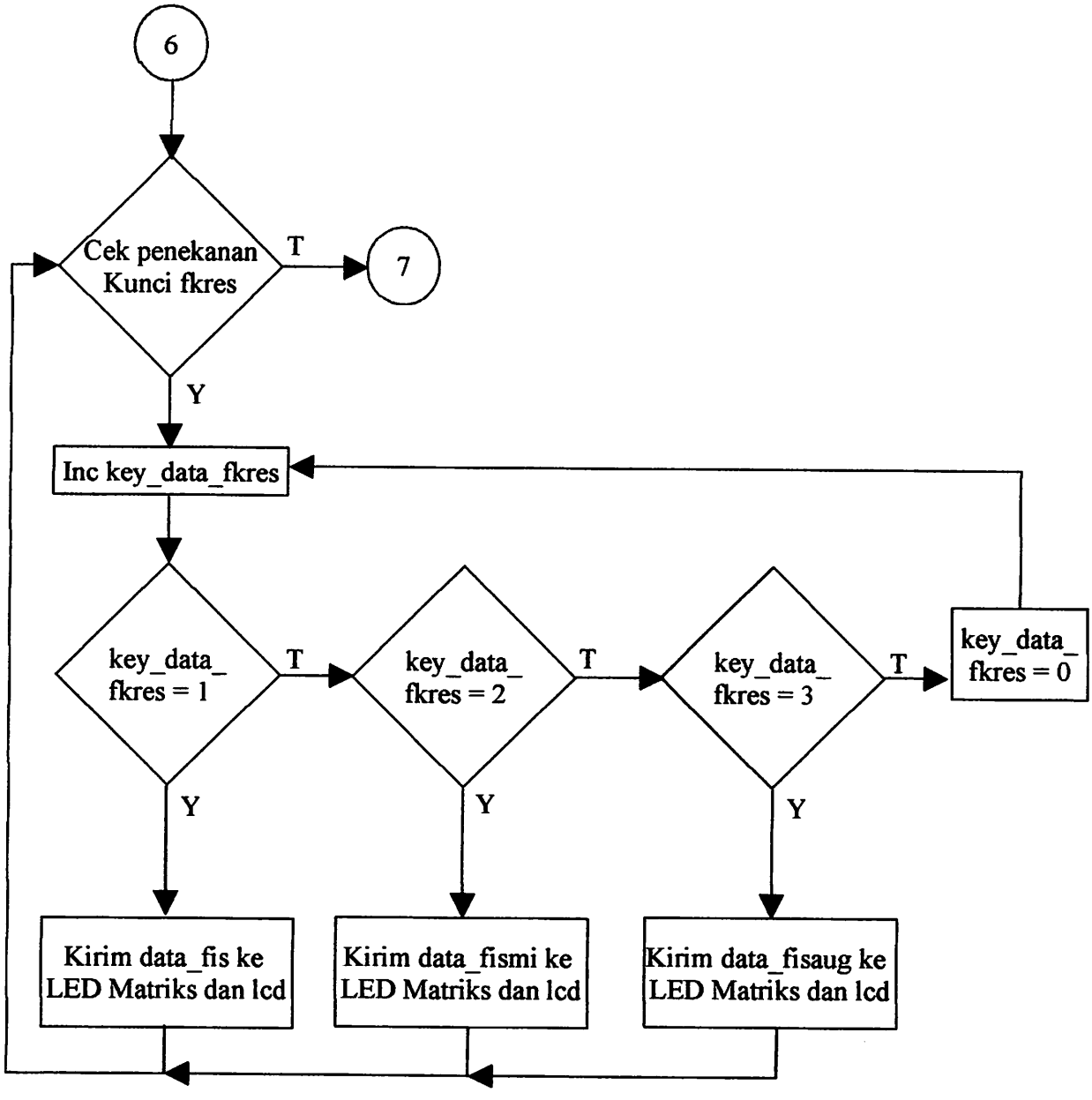
Gambar 3.11 Diagram Alir Program Untuk Cek Keypad D#



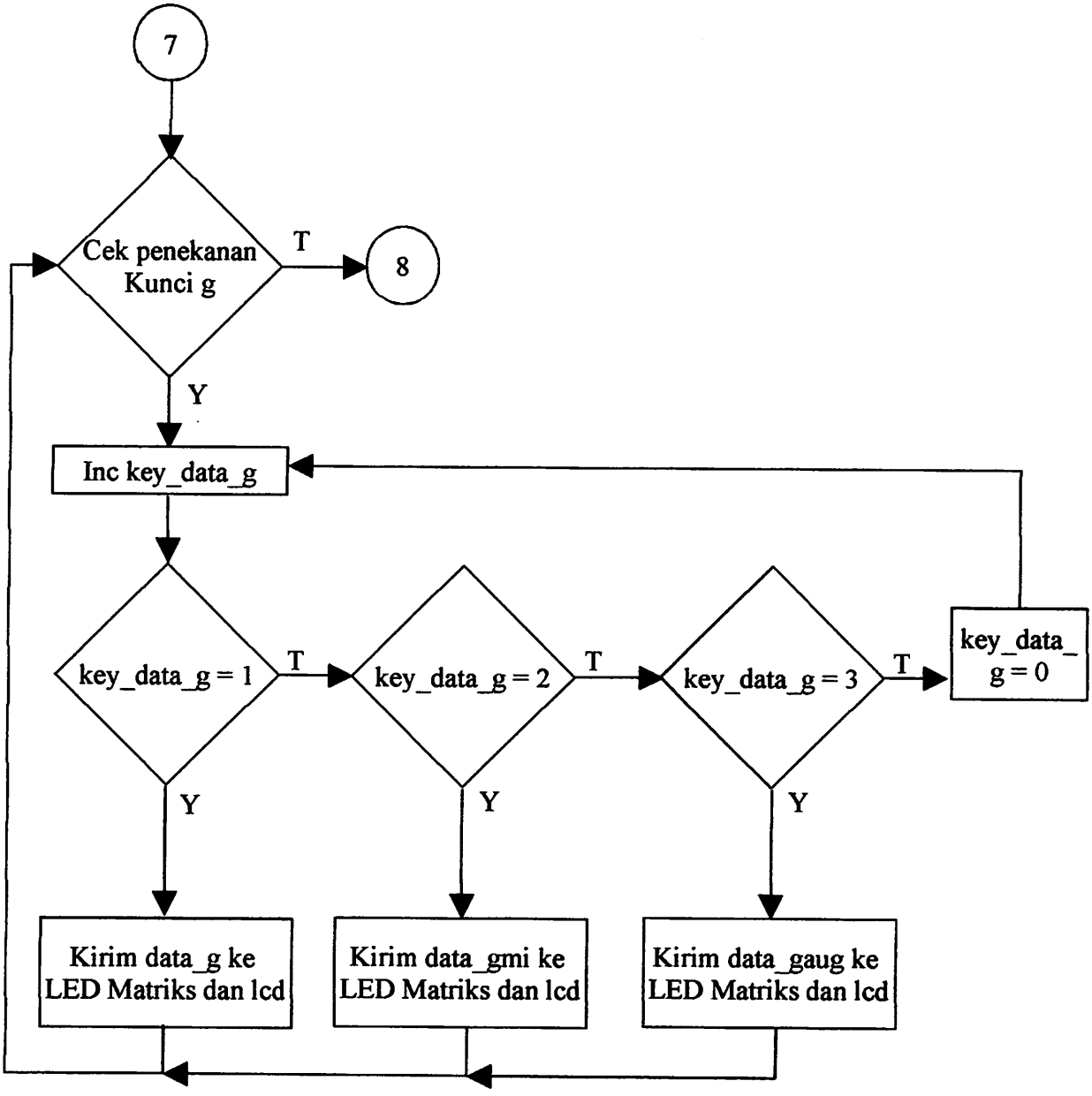
Gambar 3.12 Diagram Alir Program Untuk Cek Keypad E



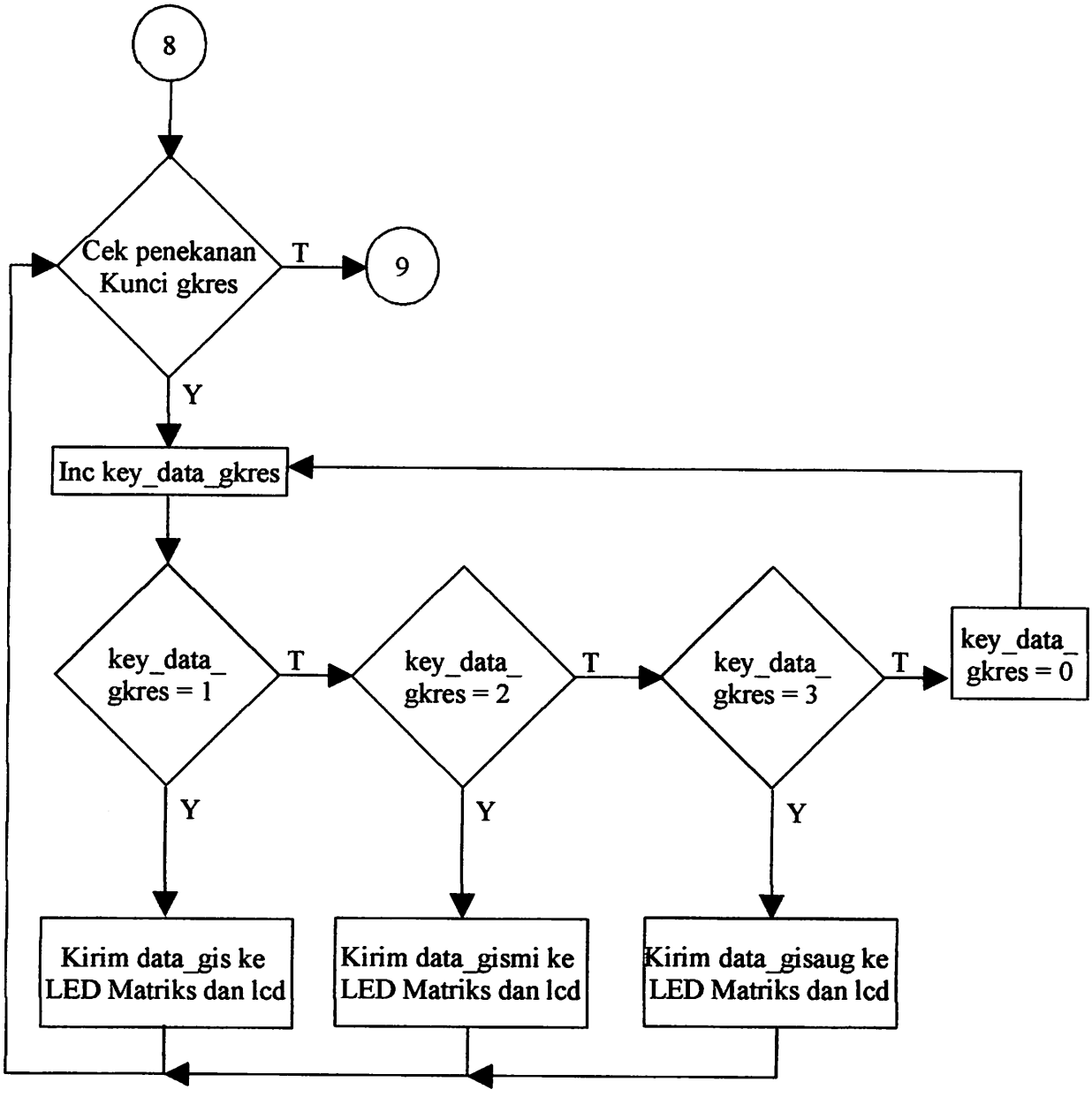
Gambar 3.13 Diagram Alir Program Untuk Cek Keypad F



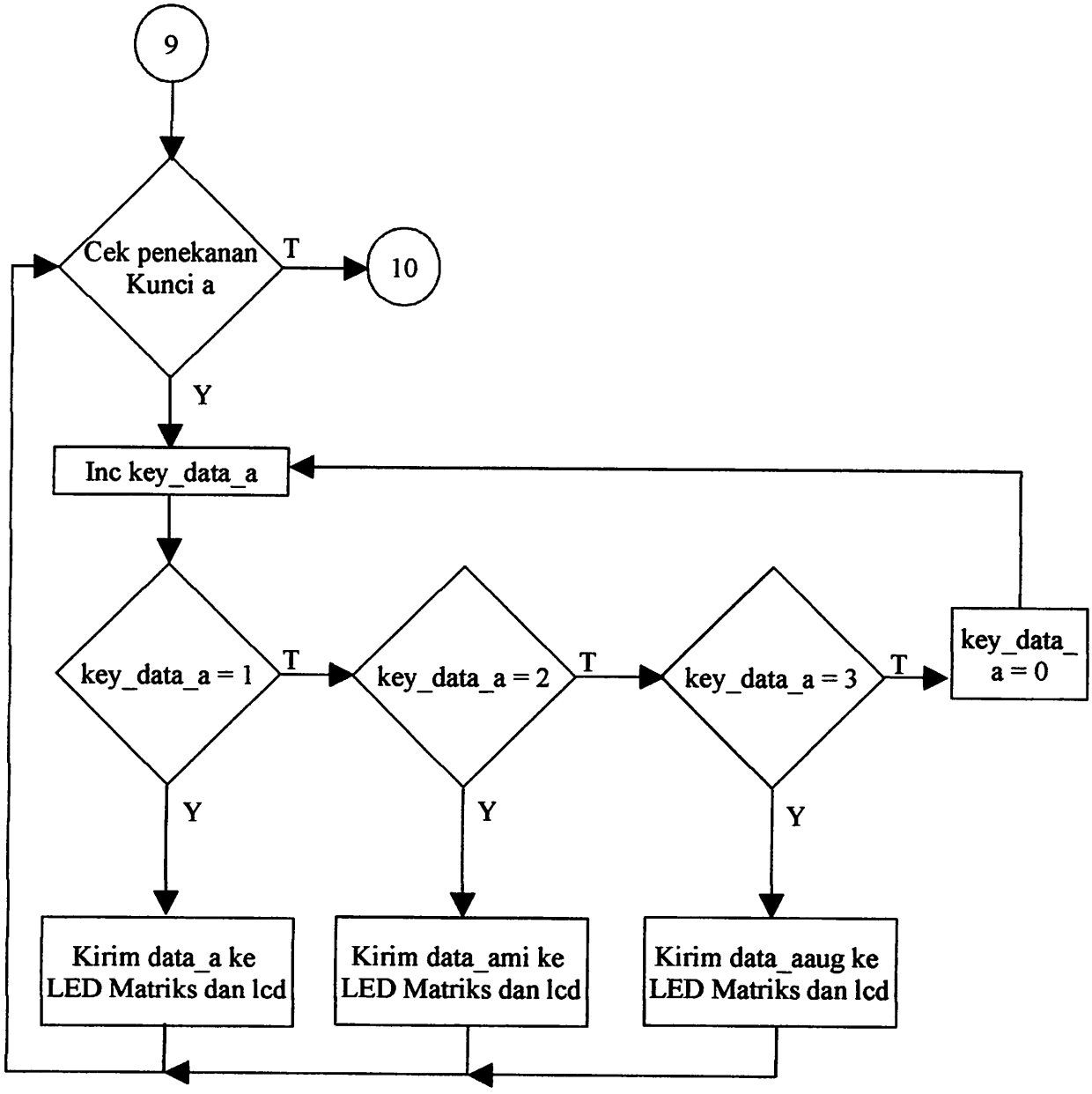
Gambar 3.14 Diagram Alir Program Untuk Cek Keypad F#



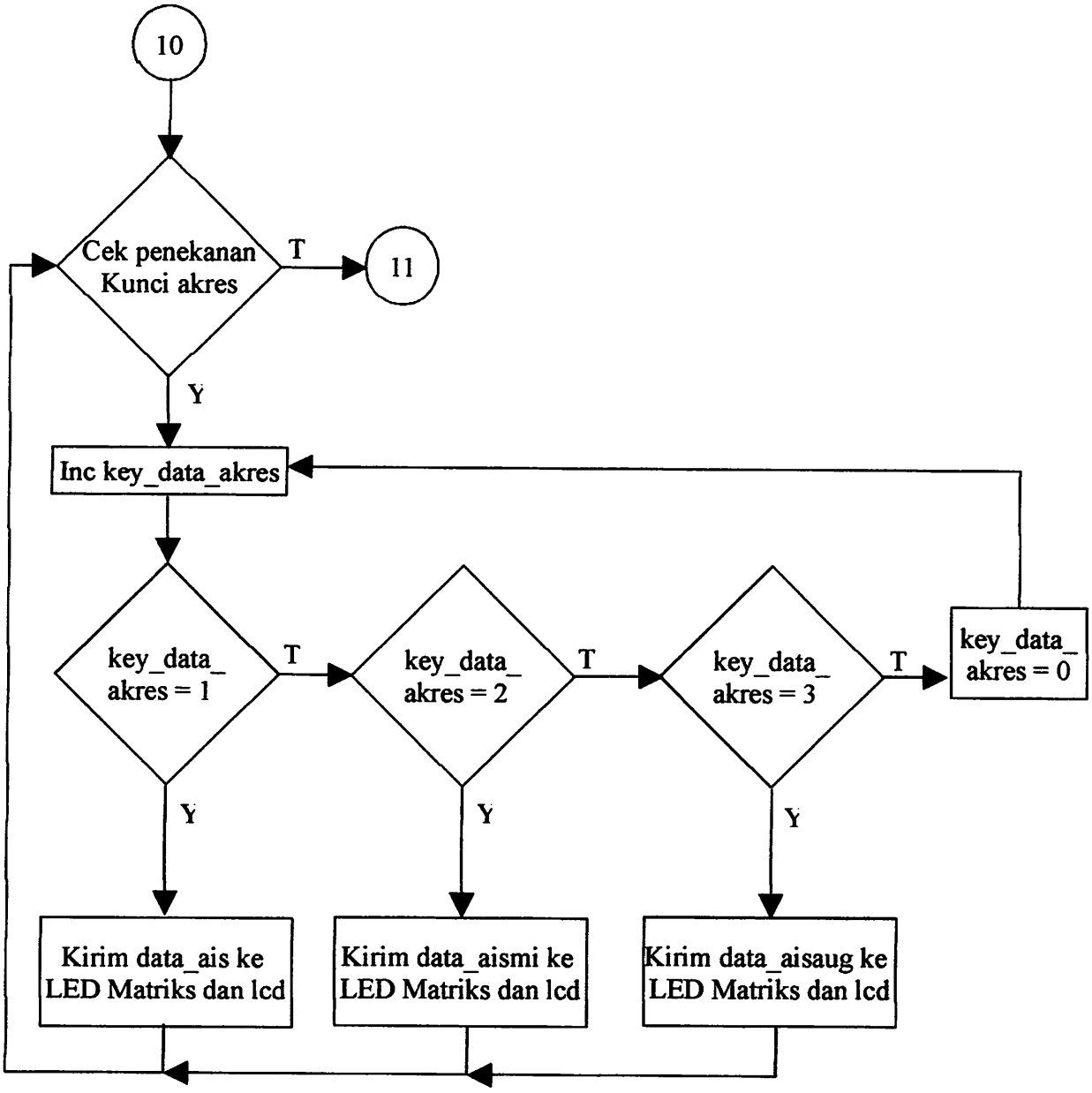
Gambar 3.15 Diagram Alir Program Untuk Cek Keypad G



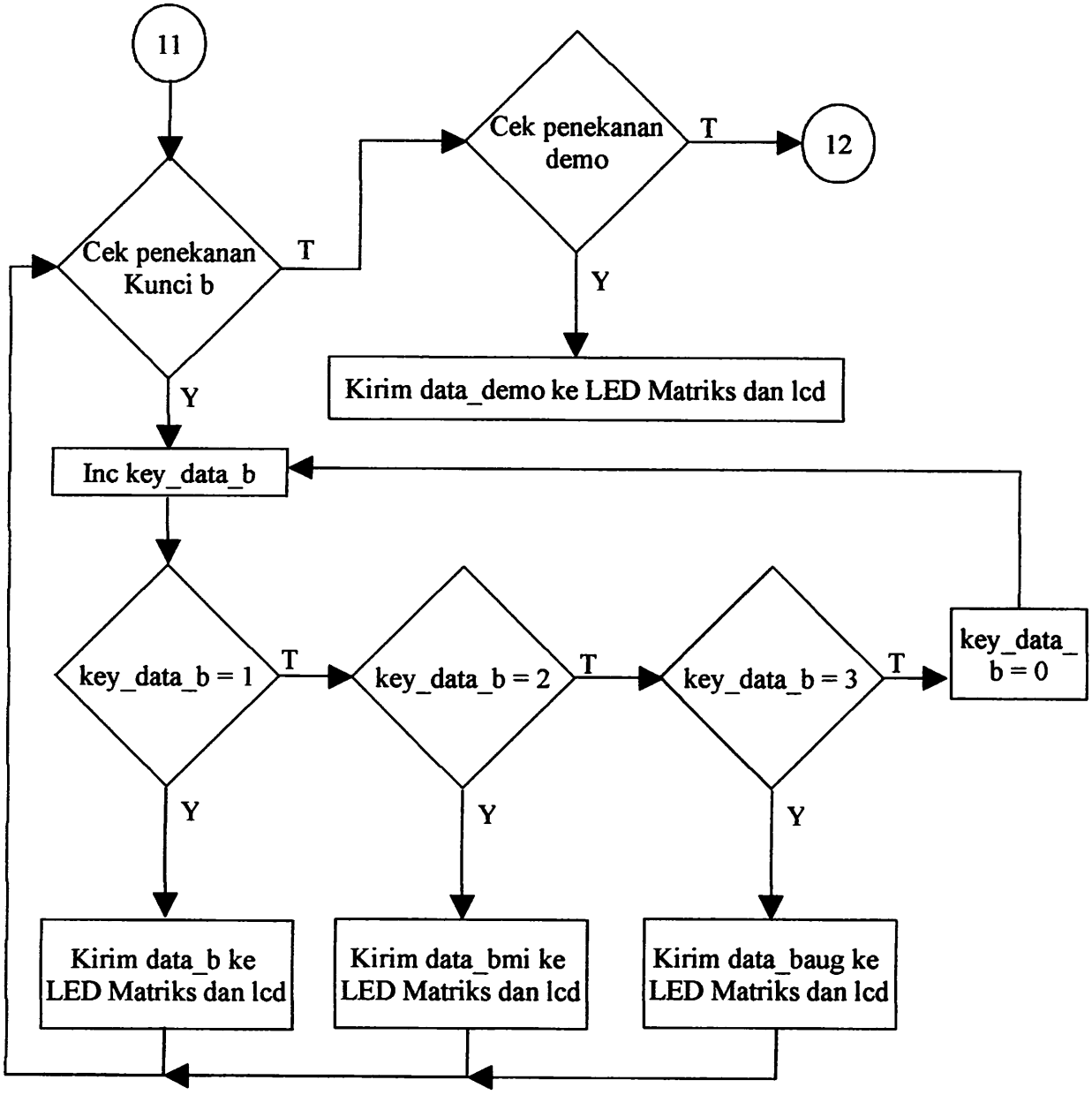
Gambar 3.16 Diagram Alir Program Untuk Cek Keypad G#



Gambar 3.17 Diagram Alir Program Untuk Cek Keypad A



Gambar 3.18 Diagram Alir Program Untuk Cek Keypad A#



Gambar 3.19 Diagram Alir Program Untuk Cek Keypad B dan Demo

3.3.2 Program Untuk Membaca Keypad

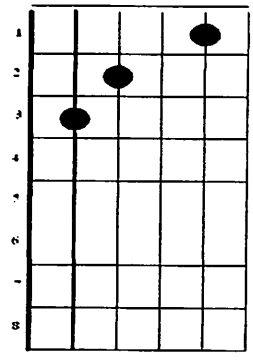
Program untuk membaca keypad ini berfungsi untuk memeriksa tombol pada keypad apakah ditekan atau tidak. Yang nantinya sebagai masukan buat mikrokontroller untuk diproses dan mikrokontroler akan menampilkan simulasi akord gitar dari pembacaan penekanan tombol keypad.

3.3.3 Program Buat LCD

Program buat LCD berfungsi untuk menjelaskan kord apa yang sedang di tampilkan oleh LED matriks sesuai dengan masukan dari keypad.

3.3.4 Program Buat LED matriks

Program ini berfungsi untuk menampilkan akord gitar pada *display* sesuai dengan tombol yang ditekan. Untuk dapat membuat program ini, maka data-data yang akan dikirimkan ke *port* keluaran mikrokontroler (*Port* 3) harus ditentukan terlebih dahulu. Sebagai contoh dapat dilihat gambar dan tabel dibawah adalah data-data yang harus dikirimkan ke *port* keluaran mikrokontroler untuk menampilkan akord C Mayor.



Gambar 3.20 Tampilan Akord C Mayor

Dari gambar 3.8 dapat ditentukan data yang akan dikirim ke port tiga untuk menyalakan LED matriks, data dapat dilihat pada tabel 3.1 :

Tabel 3.1 Data Untuk Menampilkan Akord C Mayor

Tampilan simulasi kord	Data yang dikirim ke Port 3
C	11111111b (0ffh)
	11011111b (0dfh)
	10111111b (0bfh)
	11111111b (0ffh)
	01111111b (07fh)
	11111111b (0ffh)
	11111111b (0ffh)

Dengan cara yang sama juga ditentukan data-data yang harus dikirimkan untuk menampilkan akord-akord lainnya.

3.4 Cara Kerja Alat

Cara kerja alat penampil akord gitar ini adalah sebagai berikut: Jika salah satu tombol ditekan satu kali (misalnya tombol 'C'), maka *display* akan menampilkan akord C Mayor. Jika tombol tersebut ditekan sekali lagi, maka *display* akan menampilkan akord Minor dari tombol tersebut, demikian seterusnya hingga semua variasi akord ditampilkan. Jika tombol yang ditekan berbeda dengan tombol sebelumnya, maka *display* akan menampilkan akord Mayor dari tombol yang ditekan.

BAB IV

PENGUJIAN ALAT

4.1 Pendahuluan

Sebelum kita menghubungkan semua rangkaian sesuai dengan blok diagram alat, ada baiknya kita menguji rangkaian per blok lebih dulu agar diketahui ada kerusakan atau tidak. Pengujian ini dilakukan secara bertahap dari satu rangkaian ke rangkaian berikutnya.

4.2. Pengujian Keypad

Tujuan pengujian keypad adalah untuk mengetahui apakah salah satu tombol dari keypad ada yang rusak atau tidak

Peralatan yang digunakan :

- Keypad
- Multi meter

Prosedur pengujian :

1. Mengatur multimeter pada posisi pengukuran *Ohm*
2. Menghubungkan multimeter pada kaki pin kolom 1 dan baris 1 keypad
3. Menekan tombol 1 pada keypad dan mengukur nilai resistansinya
4. Menghubungkan multimeter pada kaki pin kolom 2 baris 1 keypad
5. Menekan tombol 2 pada keypad dan mengukur nilai resistansinya
6. Menghubungkan multimeter pada kaki pin kolom 3 baris 1 keypad
7. Menekan tombol 3 pada keypad dan mengukur nilai resistansinya

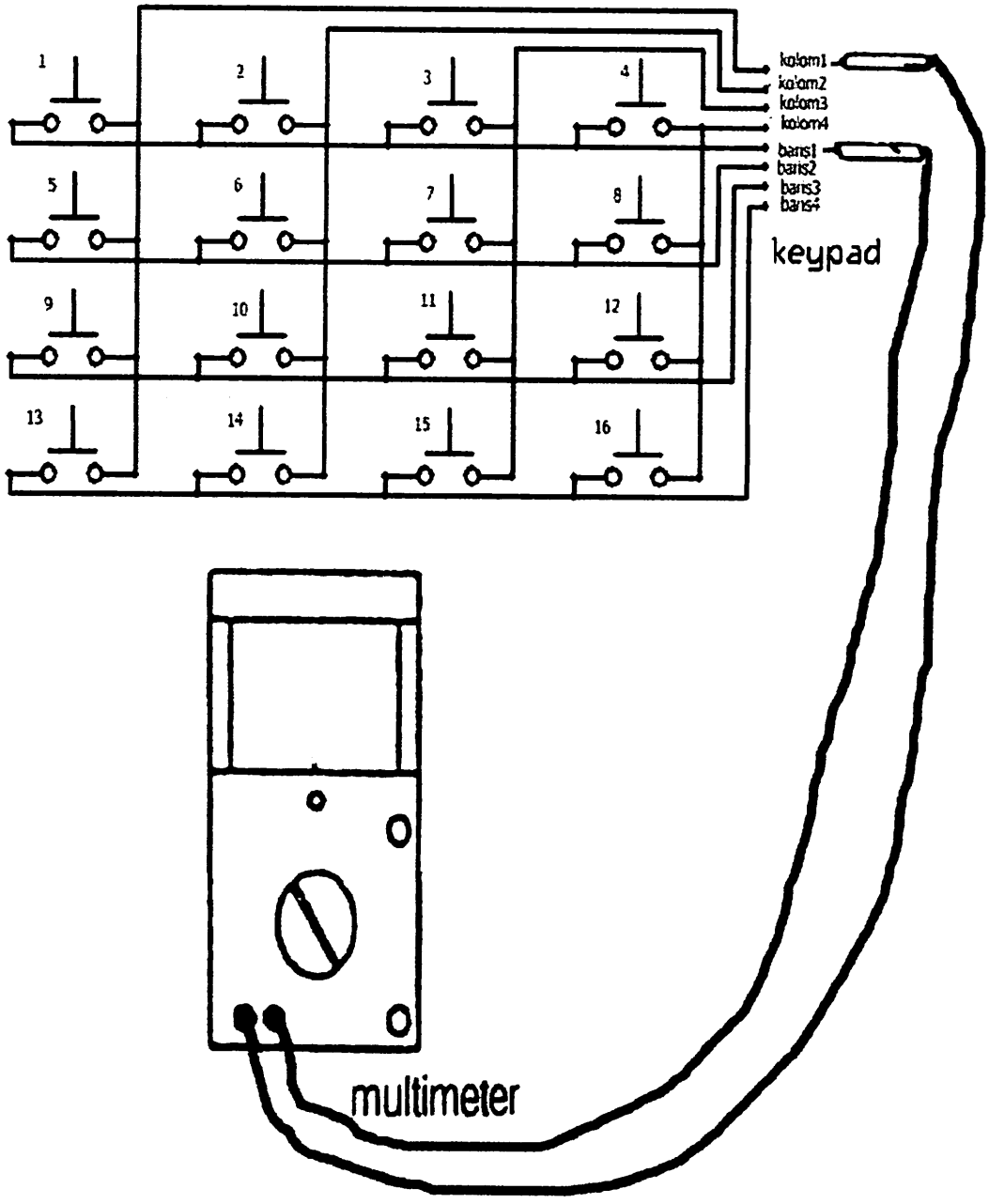
8. Menghubungkan multimeter pada kaki pin kolom 4 baris 1 keypad
9. Menekan tombol 4 pada keypad dan mengukur nilai resistansinya
10. Menghubungkan multimeter pada kaki pin kolom 1 baris 2 keypad
11. Menekan tombol 5 pada keypad dan mengukur nilai resistansinya
12. Menghubungkan multimeter pada kaki pin kolom 2 baris 2 keypad
13. Menekan tombol 6 pada keypad dan mengukur nilai resistansinya
14. Menghubungkan multimeter pada kaki pin kolom 3 baris 2 keypad
15. Menekan tombol 7 pada keypad dan mengukur nilai resistansinya
16. Menghubungkan multimeter pada kaki pin kolom 4 baris 2 keypad
17. Menekan tombol 8 pada keypad dan mengukur nilai resistansinya
18. Menghubungkan multimeter pada kaki pin kolom 1 baris 3 keypad
19. Menekan tombol 9 pada keypad dan mengukur nilai resistansinya
20. Menghubungkan multimeter pada kaki pin kolom 2 baris 3 keypad
21. Menekan tombol 10 pada keypad dan mengukur nilai resistansinya
22. Menghubungkan multimeter pada kaki pin kolom 3 baris 3 keypad
23. Menekan tombol 11 pada keypad dan mengukur nilai resistansinya
24. Menghubungkan multimeter pada kaki pin kolom 4 baris 3 keypad
25. Menekan tombol 12 pada keypad dan mengukur nilai resistansinya
26. Menghubungkan multimeter pada kaki pin kolom 1 baris 4 keypad
27. Menekan tombol 13 pada keypad dan mengukur nilai resistansinya
28. Menghubungkan multimeter pada kaki pin kolom 2 baris 4 keypad
29. Menekan tombol 14 pada keypad dan mengukur nilai resistansinya
30. Menghubungkan multimeter pada kaki pin kolom 3 baris 4 keypad

31. Menekan tombol 15 pada keypad dan mengukur nilai resistansinya

32. Menghubungkan multimeter pada kaki pin kolom 4 baris 4 keypad

33. Menekan tombol 16 pada keypad dan mengukur nilai resistansinya

34. Mencatat hasil pengukuran ke dalam table 4.1



Gambar 4.1 Pengujian Keypad

Tabel 4.1 Hasil Pengujian Keypad

No.	Tombol yang ditekan	Nilai resistansi yang terukur
1	Tombol 1	22,1 Ω
2	Tombol 2	22,1 Ω
3	Tombol 3	21,7 Ω
4	Tombol 4	23,7 Ω
5	Tombol 5	19 Ω
6	Tombol 6	19,1 Ω
7	Tombol 7	19,1 Ω
8	Tombol 8	21 Ω
9	Tombol 9	17,3 Ω
10	Tombol 10	17,2 Ω
11	Tombol 11	17,3 Ω
12	Tombol 12	19,2 Ω
13	Tombol 13	14,3 Ω
14	Tombol 14	12,8 Ω
15	Tombol 15	12,8 Ω
16	Tombol 16	14,3 Ω

Dari hasil pengujian keypad dapat di simpulkan bahwa keypad dalam kondisi baik, yaitu nilai resistansinya cukup kecil.

4.3. Pengujian LCD

Tujuan pengujian LCD:

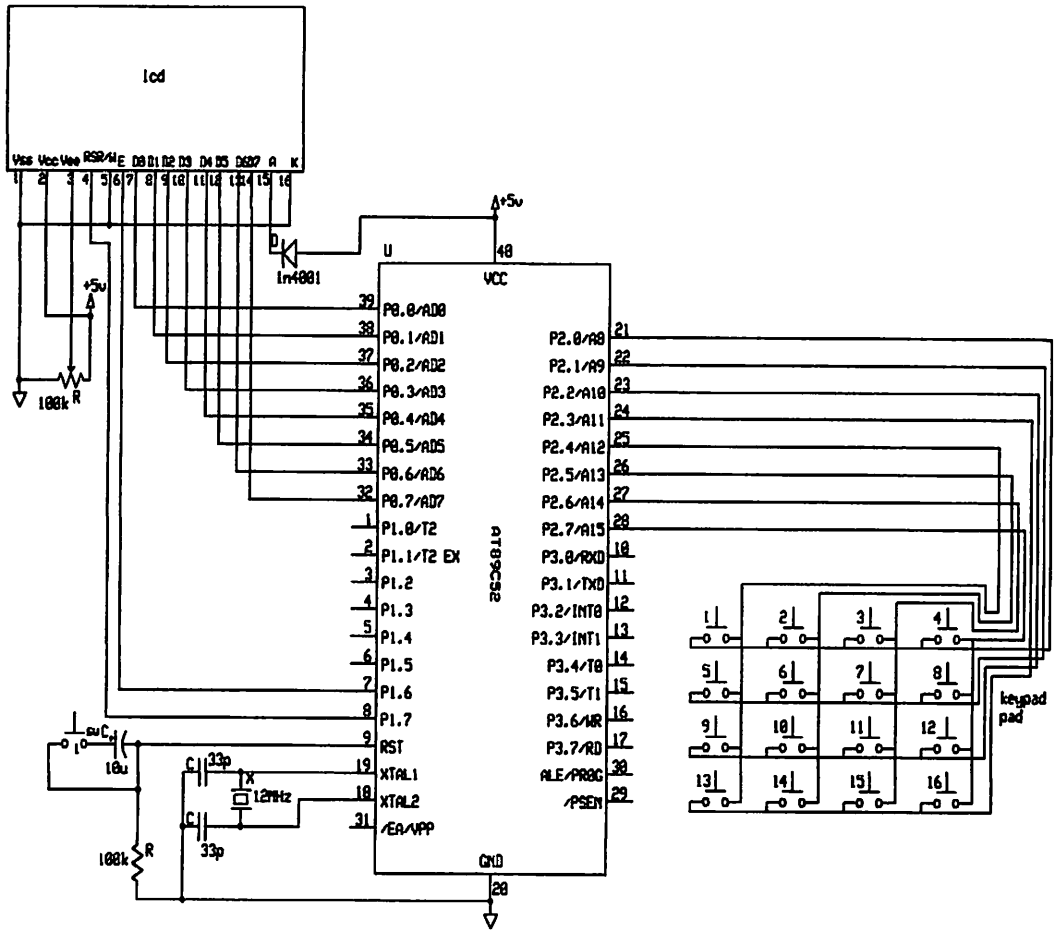
- Untuk mengetahui apakah LCD dalam kondisi baik atau rusak. Hal ini dapat diketahui dengan melihat LCD apakah dapat menampilkan karakter pada baris satu dan baris dua sesuai dengan program assembler yang dimasukkan ke dalam mikrokontroler
- Untuk mengetahui apakah rangkaian LCD dan keypad ke mikrokontroler sudah dapat bekerja dengan baik atau belum. Hal ini dapat diketahui ketika kita menekan tombol 1 pada keypad keluaran di LCD harus sesuai dengan program assembler yang di masukkan ke dalam mikrokontroler yang menyatakan bahwa tombol satu telah ditekan dan ketika kita menekan tombol 2 pada keypad maka tampilan pada LCD akan berubah sesuai dengan program assembler yang dimasukkan ke dalam mikrokontroler yang menyatakan tombol dua keypad telah ditekan begitu seterusnya untuk tombol – tombol yang lainnya.

Peralatan yang digunakan :

- Personal komputer
- Downloader buat mikrokontroler
- Mikrokontroler AT89S52
- Keypad
- LCD
- Power Supply +5 volt

Prosedur pengujian :

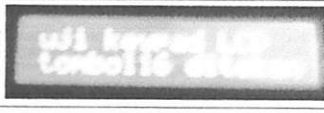
- Membuat Program dalam bahasa Assembler untuk pengambilan data ketika keypad ditekan dan menampilkannya ke LCD, lalu memasukkan program tersebut ke Mikrokontroler AT89S52.
- Membuat rangkaian sesuai dengan gambar 4.1



Gambar 4.2 Rangkaian Pengujian LCD

- Menghidupkan Power supply dan memberi catu +5v ke rangkaian
- Menekan Keypad dan melihat Hasil keluarannya di LCD
- Mencatat dan memasukkan hasilnya ke dalam tabel hasil pengujian rangkaian LCD

Tabel 4.2 Hasil Pengujian Rangkaian LCD

Tombol yang ditekan	Tampilan LCD
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

4.4. Pengujian LED matriks

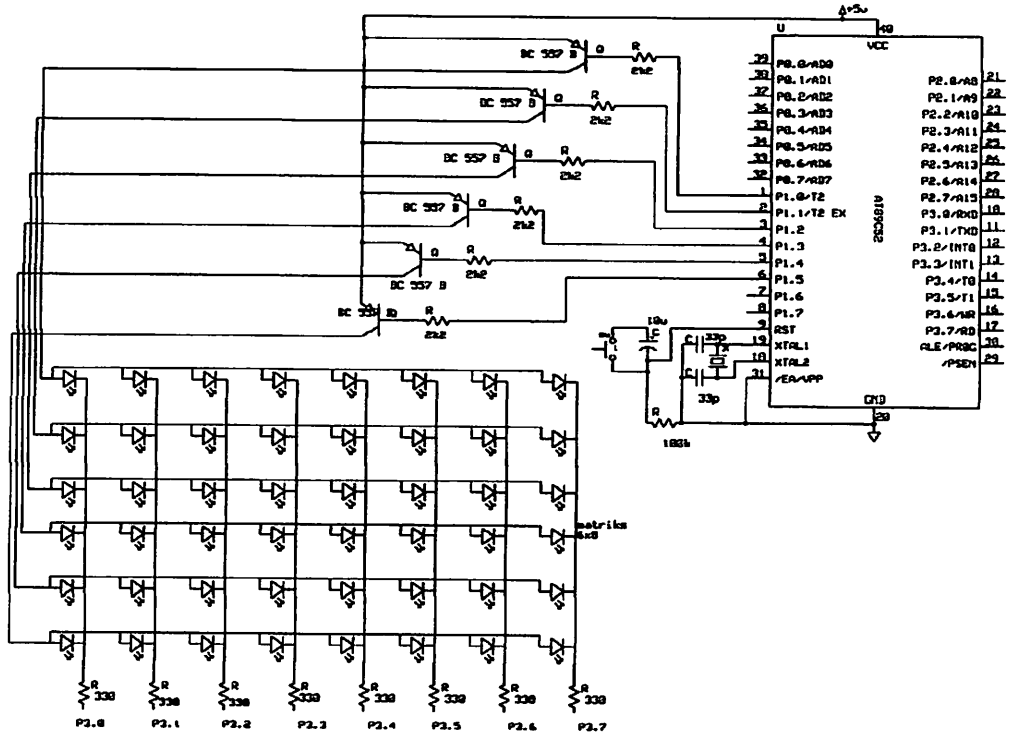
Bertujuan untuk mengetahui apakah rangkaian dot matriks dapat menampilkan kord dengan baik.

Peralatan yang digunakan :

- Personal Komputer
- Mikrokontroler AT89S52
- Rangkaian Dot Matriks
- Power Supply

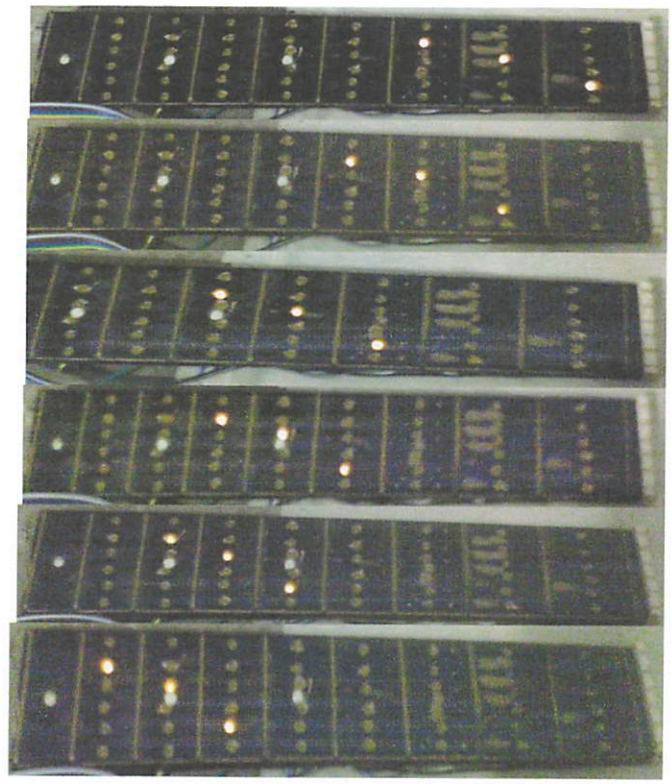
Prosedur Pengujian :

- Membuat Program dalam bahasa Asembler untuk menampilkan kord C mayor dan bergeser dalam Dot Matriks lalu memasukkannya ke mikrokontroler.
- Membuat rangkaian sesuai dengan gambar berikut :



Gambar 4.3 Rangkaian Pengujian LED Matriks

- Menghidupkan Power Supply dan memberikan catu +5v ke rangkaian
- Melihat tampilannya di Dot Matriks



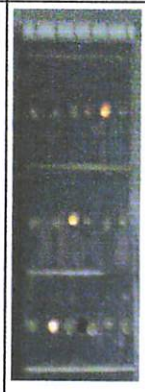
Gambar 4.4 Tampilan Akord C dengan 5 Kali Pergeseran

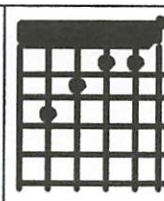
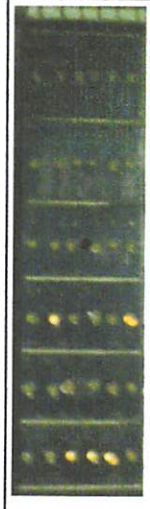
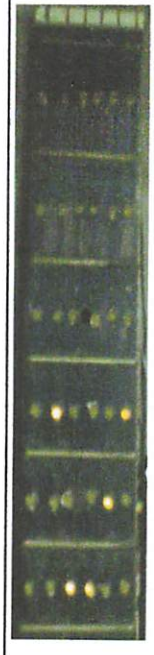
4.5. Pengujian Alat Lengkap

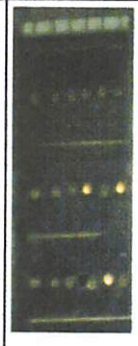
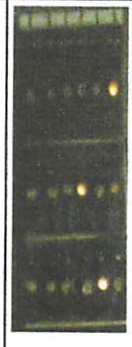
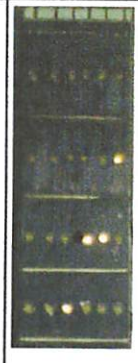
Setelah semua rangkaian sudah berjalan dengan baik, langkah selanjutnya adalah menggabungkan seluruh blok rangkaian dan memasukkan program utama simulasi akord gitar ke mikrokontroler lalu menguji alat secara keseluruhan. Pengujian ini dilakukan dengan cara menekan tombol pada *keypad* satu per satu hingga semua variasi akord gitar ditampilkan. Jika semua variasi akord gitar sudah ditampilkan dengan benar, berarti alat sudah berjalan dengan baik. Jika ada

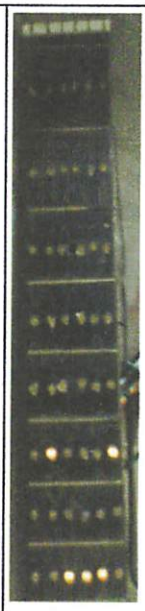
tampilan akord yang salah, periksa kembali data masing-masing akord pada program mikrokontroler dan perbaiki data yang salah. Hasil pengujian yang dilakukan terhadap alat secara keseluruhan dapat dilihat pada tabel 4.3:

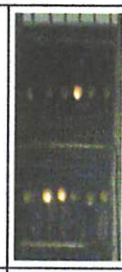
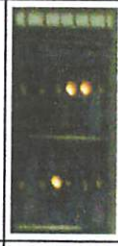
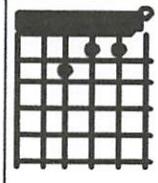
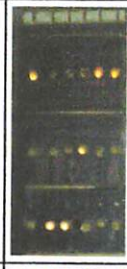
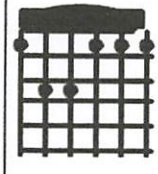
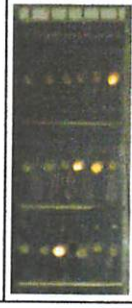
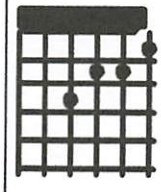
Tabel 4.3 Hasil Pengujian Pada Alat secara Keseluruhan

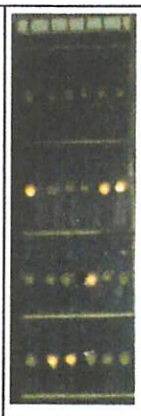
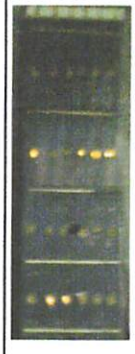
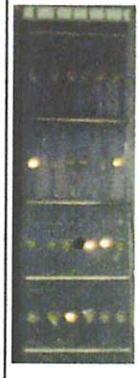
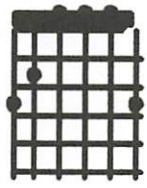
Tombol yang dipilih	Jumlah penekanan	Tampilan LED Matriks	Data Akord Gitar	Tampilan di LCD	Ket.
C	1 kali				sesuai
	2 kali				sesuai

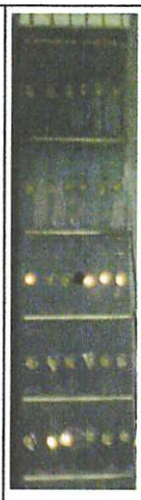
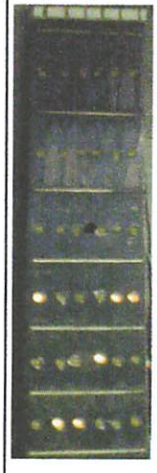
	3 kali				sesuai
C#	1 kali				sesuai
	2 kali				sesuai

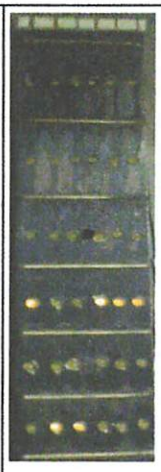
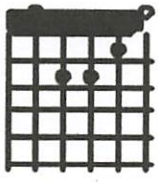
	3 kali				sesuai
D	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai

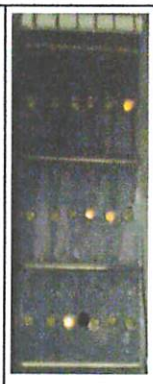
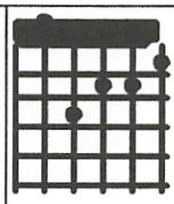
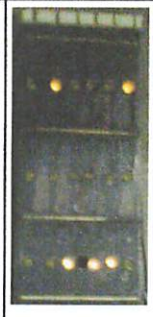
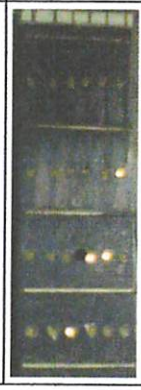
D#	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai

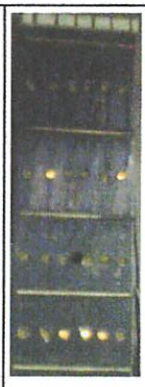
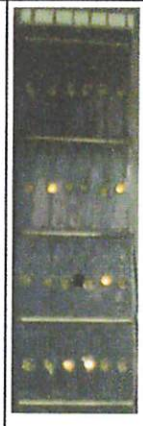
E	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai
F	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai

F#	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai
G	1 kali				sesuai

	2 kali				sesuai
	3 kali				sesuai
G#	1 kali				sesuai

	2 kali				sesuai
	3 kali				sesuai
A	1 kali				sesuai
	2 kali				sesuai

	3 kali				sesuai
A#	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai

B	1 kali				sesuai
	2 kali				sesuai
	3 kali				sesuai

Dari hasil pengujian alat secara keseluruhan dapat disimpulkan bahwa alat simulasi penampil akord gitar sudah dapat mensimulasikan tampilan dari variasi akord – akord gitar dengan baik, dan sudah sesuai dengan data akord – akord gitar yang di dapat dari datasheet akord gitar (Guitar chord Chart)

BAB V

PENUTUP

5.1. Kesimpulan

Dari hasil perancangan, pembuatan serta pengujian dapat diambil kesimpulan sebagai berikut :

1. Pada pengujian per blok alat dapat disimpulkan bahwa tiap blok rangkaian bisa bekerja dengan baik
2. Dari hasil pengujian alat secara keseluruhan alat sudah bisa menampilkan simulasi akord gitar
3. Hasil tampilan simulasi akord sudah sesuai dengan datasheet akord gitar

5.2. Saran

1. Untuk keypad bisa menggunakan keypad 3x4 dengan menghilangkan fungsi demo.
2. Bila menginginkan dapat mensimulasikan akord sampai fret ke 24 sebaiknya menggunakan memori eksternal buat menyimpan data akord gitar dan PPI untuk menambah kekurangan port.
3. Sebaiknya senar gitar jangan dibuat kencang agar tidak membebani gagang fret yang sudah dilubangi buat tempat LED untuk menjaga agar gagang fret tidak patah.
4. Untuk mendapatkan segi portabilitas alat bisa dilepas dari gitarnya agar lebih kecil.

BAB 7

PENUTUP

5.1. Kesimpulan

Dari hasil pencacangan, pembuatan serta pengujian dapat diambil kesimpulan sebagai berikan :

1. Pada pengujian per blok alat dapat disimpulkan bahwa tiap blok rangkaian bisa bekerja dengan baik
2. Dari hasil pengujian alat secara keseluruhan alat sudah bisa menampilkan simulasi akord gitar
3. Hasil tampilan simulasi akord sudah sesuai dengan datasheet akord gitar

5.2. Saran

1. Untuk keypad bisa menggunakan keypad 3x4 dengan mengbilangkan tunggi demo.
2. Bila menginginkan dapat memsimulasikan akord sempai fret ke 24 sebaiknya menggunakan memori eksternal buat menginpan data akord gitar dan PPI untuk menambah keluarangan por.
3. Sebaiknya sour gitar jangan dibuat kurang dari tidak membahani gagang fret yang sudah dibuangi buat tempat LED untuk mengga gagang fret tidak pasah.
4. Untuk mendapatkan segi portabilitas alat bisa dilepas dari jiarnya agar lebih kecil.

DAFTAR PUSTAKA

Paulus Andi Nalwan,2003, Panduan Praktis Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51, Elex Media Komputindo, Jakarta

Amos, S. W. ,1997, Kamus Elektronika, Elex Media Komputindo, Jakarta

2004, Laporan Praktikum Dasar Elektronioka, Institut Teknologi Nasional Malang

AT89S52 datasheet

BC557 datasheet

myTutorialCafe.com

ElectronicLab.com

www.chordie.com

linkpluskord.com

en.wikipedia.org

id.wikipedia.org

Futurlec.com

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo Km.2
Malang

BERITA ACARA UJIAN SKRIPSI

Nama : Raditya Aspri Putra
N.I.M. : 00.17.169
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika
Judul Skripsi : Perancangan dan Pembuatan Simulator Akord Gitar Menggunakan Mikrokontroler AT89S52

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada:

Hari : Selasa
Tanggal : 17 Maret 2009
Dengan Nilai : 81,25 (A)

Panitia Majelis Penguji :

Ketua	Sekretaris
Ir. H. Sidik Noertjahjono, MT	Ir. F. Yudi Limpraptono MT.
NIP.Y. 1028700163	NIP.Y. 1039500274

Anggota Penguji :

Penguji Pertama	Penguji Kedua
I Komang Somawirata ST. MT.	Irmalia Suryani Faradisa ST. MT.
NIP.P. 1030100361	NIP.P. 1030100365

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANG LO, KM 2
MALANG

FORMULIR PERBAIKAN SKRIPSI

NAMA : RADITYA ASPRI PUTRA

N.I.M. : 00.17.169

MASA BIMBINGAN : 11 AGUSTUS 2008 s/d 11 FEBRUARI 2009

JUDUL SKRIPSI : PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52

NO.	TANGGAL	URAIAN	PARAF
1.	17 Maret 2009	Lampiran yang Tidak Perlu Dihilangkan	

Disetujui,
Penguji I

I Komang Somawirata ,ST ,MT
NIP.P. 1030100361

Mengetahui,

Dosen Pembimbing I

Dr. Cahyo C. ,Msc.
NIP. 1030400412

Dosen Pembimbing II

M. Ashar ,ST ,MT.
NIP. 1030500408

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 : RADITYA ASPRI P.
NIM : 00-17-169
Perbaikan meliputi :

* Lampiran yg tdk perlu di hilangkan !

Malang, 17-3 2009

(Ir. Choirul S., ST, MT)

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANG LO, KM 2
MALANG

FORMULIR PERBAIKAN SKRIPSI

NAMA : RADITYA ASPRI PUTRA

N.I.M. : 00.17.169

MASA BIMBINGAN : 11 AGUSTUS 2008 s/d 11 FEBRUARI 2009

JUDUL SKRIPSI : PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52

NO.	TANGGAL	URAIAN	PARAF
1.	17 Maret 2009	Tata Tulis Laporan Diperbaiki	

Disetujui,
Penguji II

Irmalia Suryani Faradisa ,ST ,MT
NIP.P. 1030100365

Mengetahui,

Dosen Pembimbing I

Dr. Cahyo C. ,Msc.
NIP. 1030400412

Dosen Pembimbing II

M. Ashar ,ST ,MT.
NIP. 1030500408

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 : Radityá
N I M : 0017169
Perbaikan meliputi :

Cara penulisan laporan diperbaiki

Malang, 200

( )

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

NI (PERSERO) MALANG
NK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 15 Agustus 2008

Nomor : ITN-252/7/TA /2008
Lampiran :
Perihal : *Bimbingan Skripsi*

Kepada :Yth. Sdr. **DR. CAHYO CHRYSDIAN, MSc** *)
Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skipsi untuk mahasiswa:

Nama	: **RADITYA ASPRI PUTRA**
Nim	: **0017169**
Fakultas	: **Teknologi Industri**
Jurusan	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Elektronika S-I**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu 6 (enam) bulan, terhitung mulai tanggal:

11 AGUSTUS 2008 S/D 11 PEBRUARI 2009

Adapun tugas tersebut merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik, Jurusan Elektro apabila lewat dari batas waktu tsb. Maka, skripsinya akan digugurkan.
Demikian atas perhatian serta kerjasama yang baik kami ucapkan terima kasih

Ketua Jurusan
Teknik Elektro S-1

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

Tindasan:
3. *)Perpanjangan
4. Mahasiswa yang Bersangkutan
5. Arsip

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANG LO, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

NAMA : RADITYA ASPRI PUTRA
N.I.M. : 00.17.169
MASA BIMBINGAN : 11 AGUSTUS 2008 s/d 11 FEBRUARI 2009
JUDUL SKRIPSI : PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52

NO.	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	30/09/01	Demo Alat	
2.	02/09/02	Revisi paper Seminar hasil	
3.	07/09/02	Revisi Pengujian LCD Matriks	
4.	15/09/02	Bab V Penutup <Tambah Saran>	
5.	20/09/02	Acc. Makalah <paper> Seminar hasil	
6.	15/09/03	Acc. Laporan Skripsi	
7.	16/09/03	Acc. Komprehensif.	
8.			
9.			
10.			

Malang,
Dosen Pembimbing

Dr. Cahyo C. ,Msc.
NIP : 1030400412

Form S-4b

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
II (PERSERO) MALANG
NK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 15 Agustus 2008

Nomor : ITN-253/7/TA /2008
Lampiran :
Perihal : *Bimbingan Skripsi*

Kepada :Yth. Sdr. M. ASHAR, ST, MT *)
Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skipsi untuk mahasiswa:

Nama	: **RADITYA ASPRI PUTRA**
Nim	: **0017169**
Fakultas	: **Teknologi Industri**
Jurusan	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Elektronika S-I**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu 6 (enam) bulan, terhitung mulai tanggal:

11 AGUSTUS 2008 S/D 11 PEBRUARI 2009

Adapun tugas tersebut merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik, Jurusan Elektro apabila lewat dari batas waktu tsb. Maka, skripsinya akan digugurkan.
Demikian atas perhatian serta kerjasama yang baik kami ucapkan terima kasih

Ketua Jurusan
Teknik Elektro S-1

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

Tindasan:
3. *)Perpanjangan
4. Mahasiswa yang Bersangkutan
5. Arsip

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANG LO, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

NAMA : RADITYA ASPRI PUTRA
N.I.M. : 00.17.169
MASA BIMBINGAN : 11 AGUSTUS 2008 s/d 11 FEBRUARI 2009
JUDUL SKRIPSI : PERANCANGAN DAN PEMBUATAN SIMULATOR AKORD GITAR MENGGUNAKAN MIKROKONTROLLER AT89S52

NO.	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	31/01'09	Bab I, II, III (Revisi)	
2.	5/02'09	Bab IV (Pengujian) !	
3.		Bab Pengujian tambah	
4.		Pengujian Error	
5.		Bandingkan dg Manual	
6.		Pengujian Nada	
7.		Bab V (kesimpulan)	
8.		Makalah Seminar	
9.		Komputer	
10.			

Malang,
Dosen Pembimbing

M. Ashar, ST, Msc.
NIP : 1030500408

Form S-4b

Program untuk Pengujian Rangkaian LCD :

```
;-------------------------
org     0h
;-------------------------
mulai:          call    init_lcd
                call    atas
keypad:                 mov     p2,#0feh
key1:           jb      p2.4,key2
                jnb     p2.4,$
lcdkey1:        mov     dptr,#data_key1
                call    lcd_baris2
key2:           jb      p2.5,key3
                jnb     p2.5,$
lcdkey2:        mov     dptr,#data_key2
                call    lcd_baris2
key3:           jb      p2.6,key4
                jnb     p2.6,$
lcdkey3:        mov     dptr,#data_key3
                call    lcd_baris2
key4:           jb      p2.7,baris2
                jnb     p2.7,$
lcdkey4:        mov     dptr,#data_key4
                call    lcd_baris2
;-------------------------
baris2:         mov     p2,#0fdh
key5:           jb      p2.4,key6
                jnb     p2.4,$
lcdkey5:        mov     dptr,#data_key5
                call    lcd_baris2
key6:           jb      p2.5,key7
                jnb     p2.5,$
lcdkey6:        mov     dptr,#data_key6
                call    lcd_baris2
key7:           jb      p2.6,key8
                jnb     p2.6,$
lcdkey7:        mov     dptr,#data_key7
                call    lcd_baris2
key8:           jb      p2.7,baris3
                jnb     p2.7,$
lcdkey8:        mov     dptr,#data_key8
                call    lcd_baris2
;-------------------------
baris3:         mov     p2,#0fbh
key9:           jb      p2.4,key10
                jnb     p2.4,$
```

```
lcdkey9:    mov   dptr,#data_key9
            call  lcd_baris2
key10:      jb    p2.5,key11
            jnb   p2.5,$
lcdkey10:   mov   dptr,#data_key10
            call  lcd_baris2
key11:      jb    p2.6,key12
            jnb   p2.6,$
lcdkey11:   mov   dptr,#data_key11
            call  lcd_baris2
key12:      jb    p2.7,baris4
            jnb   p2.7,$
lcdkey12:   mov   dptr,#data_key12
            call  lcd_baris2
;-------------------------
baris4:     mov   p2,#0f7h
key13:      jb    p2.4,key14
            jnb   p2.4,$
lcdkey13:   mov   dptr,#data_key13
            call  lcd_baris2
key14:      jb    p2.5,key15
            jnb   p2.5,$
lcdkey14:   mov   dptr,#data_key14
            call  lcd_baris2
key15:      jb    p2.6,key16
            jnb   p2.6,$
lcdkey15:   mov   dptr,#data_key15
            call  lcd_baris2
key16:      jb    p2.7,mulaia
            jnb   p2.7,$
lcdkey16:   mov   dptr,#data_key16
            call  lcd_baris2
mulaia:           ljmp  keypad
;-----------------------------------------------
delay: mov  r4,#0ffh
delay1:mov  r5,#0fh
            djnz  r5,$
            djnz  r4,delay1
            ret
;-----------------------------------------------
;=============================================:
;       SUBROUTINE LCD                        :
;=============================================:
init_lcd:   mov   a,#03fh
            call  instruksi
            mov   a,#0eh
```

```
                call    instruksi
                mov     a,#06h
                call    instruksi
                mov     a,#01h
                call    instruksi
                ret
;===============================================:
instruksi:      clr     p1.7    ;rs
                mov     p0,a
                setb    p1.6
                clr     p1.6
                call    delay
                ret
;===============================================:
tulis_data:     setb    p1.7    ;rs
                mov     p0,a
                setb    p1.6
                clr     p1.6
                call    delay
                ret
;===============================================:
;lcd atas
;===============================================:
atas:           mov     dptr,#data_atas
                mov     r2,#16
                mov     a,#80h
                call    instruksi
tulis1:         clr     a
                movc    a,@a+dptr
                inc     dptr
                call    tulis_data
                djnz    r2,tulis1
                ret
;===============================================:
;lcd baris2
;===============================================:
lcd_baris2:     mov     r7,#16
                mov     a,#0c0h
                call    instruksi
tulis2:         clr     a
                movc    a,@a+dptr
                inc     dptr
                call    tulis_data
                djnz    r7,tulis2
                ret
;===============================================:
```

```
data_key1:    db    'tombol1 ditekan '
data_key2:    db    'tombol2 ditekan '
data_key3:    db    'tombol3 ditekan '
data_key4:    db    'tombol4 ditekan '
data_key5:    db    'tombol5 ditekan '
data_key6:    db    'tombol6 ditekan '
data_key7:    db    'tombol7 ditekan '
data_key8:    db    'tombol8 ditekan '
data_key9:    db    'tombol9 ditekan '
data_key10:   db    'tombol10 ditekan'
data_key11:   db    'tombol11 ditekan'
data_key12:   db    'tombol12 ditekan'
data_key13:   db    'tombol13 ditekan'
data_key14:   db    'tombol14 ditekan'
data_key15:   db    'tombol15 ditekan'
data_key16:   db    'tombol16 ditekan'
data_atas:    db    'uji keypad LCD  '
end
```

Program untuk Pengujian Rangkaian LED matriks :

```
org     0h
mulai:      mov     r0,#00h
            mov     r3,#0feh
            mov     r6,#00h
kord_c:             mov   dptr,#data_c
            call    tampil
            inc     r6
            cjne    r6,#04fh,kord_c
            mov     r6,#00h
geser1:     mov     dptr,#c_geser1
            call    tampil
            inc     r6
            cjne    r6,#04fh,geser1
            mov     r6,#00h
geser2:     mov     dptr,#c_geser2
            call    tampil
            inc     r6
            cjne    r6,#04fh,geser2
            mov     r6,#00h
geser3:     mov     dptr,#c_geser3
            call    tampil
            inc     r6
            cjne    r6,#04fh,geser3
            mov     r6,#00h
geser4:     mov     dptr,#c_geser4
            call    tampil
            inc     r6
            cjne    r6,#04fh,geser4
            mov     r6,#00h
geser5:     mov     dptr,#c_geser5
            call    tampil
            inc     r6
            cjne    r6,#04fh,geser5
            mov     r6,#00h
            jmp     mulai
;-----------------------------------------------
tampil:     clr     a
            movc    a,@a+dptr
            inc     dptr
            mov     p3,a
            mov     p1,r3
            call    delay
            mov     a,r3
            rl      a
```

```
                mov     r3,a
                inc     r0
                cjne    r0,#07h,tampil
                mov     r0,#00h
                mov     r3,#0feh
                ret
;-----------------------------------------------
delay:  mov     r4,#0ffh
delay1:mov      r5,#06h
                djnz    r5,$
                djnz    r4,delay1
                ret
;-------------------------------------------------------
data_c:         DB      0ffh,0dfh,0bfh,0ffh,07fh,0ffh,0ffh
c_geser1:       DB      0ffh,0efh,0dfh,0ffh,0bfh,0ffh,0ffh
c_geser2:       DB      0ffh,0f7h,0efh,0ffh,0dfh,0ffh,0ffh
c_geser3:       DB      0ffh,0fbh,0f7h,0ffh,0efh,0ffh,0ffh
c_geser4:       DB      0ffh,0fdh,0fbh,0ffh,0f7h,0ffh,0ffh
c_geser5:       DB      0ffh,0feh,0fdh,0ffh,0fbh,0ffh,0ffh
end
```

Program Lengkap Alat Penampil Simulasi Akord Gitar :

```
key_datac       equ     70h
bouncing        equ     71h
key_data_ckres          equ     72h
key_data_d      equ     73h
key_data_dkres          equ     74h
key_data_e      equ     75h
key_data_f      equ     76h
key_data_fkres          equ     77h
key_data_g      equ     78h
key_data_gkres          equ     79h
key_data_a      equ     7Ah
key_data_akres          equ     7Bh
key_data_b      equ     7Ch
;---------------------------------
                org     0h
;---------------------------------
mulai:          call    init_lcd
                mov     r0,#00h
                mov     r3,#0feh
                mov     r6,#00h
                mov     key_datac,#00h
                mov     bouncing,#0ffh
                mov     key_data_ckres,#00h
                mov     key_data_d,#00h
                mov     key_data_dkres,#00h
                mov     key_data_e,#00h
                mov     key_data_f,#00h
                mov     key_data_fkres,#00h
                mov     key_data_g,#00h
                mov     key_data_gkres,#00h
                mov     key_data_a,#00h
                mov     key_data_akres,#00h
                mov     key_data_b,#00h
;-----------------------------------------------
;baca isi keypad
;-----------------------------------------------
keypad:                 mov     p2,#0feh                ;baris1
kunci_c:        jb      p2.4,kunci_cis
                djnz    bouncing,kunci_c
                jnb     p2.4,$
                inc     key_datac
                mov     r1,key_datac
                cjne    r1,#04h,macam_c
                mov     r1,#01h
```

```
macam_c:    cjne    r1,#01h,c_minor
c_mayor:    call    atas
            mov     dptr,#data_c
            call    tampil
            inc     r6
            cjne    r6,#04fh,c_mayor
            mov     r6,#00h
            ljmp    keypad
c_minor:    cjne    r1,#02h,c_aug
c_min1:             call    atas
            mov     dptr,#data_cmin
            call    tampil
            inc     r6
            cjne    r6,#04fh,c_min1
            mov     r6,#00h
            ljmp    keypad
c_aug:      cjne    r1,#03h,awalc
c_aug1:             call    atas
            mov     dptr,#data_caug
            call    tampil
            inc     r6
            cjne    r6,#04fh,c_aug1
            mov     r6,#00h
awalc:      mov     key_datac,#00h
            ljmp    keypad
;-----------------------------------------------
kunci_cis:  jb      p2.5,kunci_d
            djnz    bouncing,kunci_cis
            jnb     p2.5,$
            inc     key_data_ckres
            mov     r1,key_data_ckres
            cjne    r1,#04h,macam_ckres
            mov     r1,#01h
macam_ckres:cjne    r1,#01h,ckres_minor
ckres_mayor: call   atas
            mov     dptr,#data_cis
            call    tampil
            inc     r6
            cjne    r6,#04fh,ckres_mayor
            mov     r6,#00h
            ljmp    keypad
ckres_minor: cjne   r1,#02h,ckres_aug
ckres_min1: call    atas
            mov     dptr,#data_cismi
            call    tampil
            inc     r6
```

```
                cjne    r6,#04fh,ckres_min1
                mov     r6,#00h
                ljmp    keypad
ckres_aug:      cjne    r1,#03h,awal_ckres
ckres_aug1:     call    atas
                mov     dptr,#data_cisaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,ckres_aug1
                mov     r6,#00h
awal_ckres:     mov     key_data_ckres,#00h
                ljmp    keypad
;-----------------------------------------------
kunci_d:        jb      p2.6,kunci_dis
                djnz    bouncing,kunci_d
                jnb     p2.6,$
                inc     key_data_d
                mov     r1,key_data_d
                cjne    r1,#04h,macam_d
                mov     r1,#01h
macam_d:        cjne    r1,#01h,d_minor
d_mayor:        call    atas
                mov     dptr,#data_d
                call    tampil
                inc     r6
                cjne    r6,#04fh,d_mayor
                mov     r6,#00h
                ljmp    keypad
d_minor:        cjne    r1,#02h,d_aug
d_min1:                 call    atas
                mov     dptr,#data_dmi
                call    tampil
                inc     r6
                cjne    r6,#04fh,d_min1
                mov     r6,#00h
                ljmp    keypad
d_aug:          cjne    r1,#03h,awal_d
d_aug1:                 call    atas
                mov     dptr,#data_daug
                call    tampil
                inc     r6
                cjne    r6,#04fh,d_aug1
                mov     r6,#00h
awal_d:                 mov     key_data_d,#00h
                ljmp    keypad
;-----------------------------------------------
```

```
kunci_dis:      jb      p2.7,kunci_e
                djnz    bouncing,kunci_dis
                jnb     p2.7,$
                inc     key_data_dkres
                mov     r1,key_data_dkres
                cjne    r1,#04h,macam_dkres
                mov     r1,#01h
macam_dkres:    cjne    r1,#01h,dkres_minor
dkres_mayor:    call    atas
                mov     dptr,#data_dis
                call    tampil
                inc     r6
                cjne    r6,#04fh,dkres_mayor
                mov     r6,#00h
                ljmp    keypad
dkres_minor:    cjne    r1,#02h,dkres_aug
dkres_min1:     call    atas
                mov     dptr,#data_dismi
                call    tampil
                inc     r6
                cjne    r6,#04fh,dkres_min1
                mov     r6,#00h
                ljmp    keypad
dkres_aug:      cjne    r1,#03h,awal_dkres
dkres_aug1:     call    atas
                mov     dptr,#data_disaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,dkres_aug1
                mov     r6,#00h
awal_dkres:     mov     key_data_dkres,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_e:        mov     p2,#0fdh                ;baris2
                jb      p2.4,kunci_f
                djnz    bouncing,kunci_e
                jnb     p2.4,$
                inc     key_data_e
                mov     r1,key_data_e
                cjne    r1,#04h,macam_e
                mov     r1,#01h
macam_e:        cjne    r1,#01h,e_minor
e_mayor:        call    atas
                mov     dptr,#data_e
                call    tampil
                inc     r6
```

```
                cjne    r6,#04fh,e_mayor
                mov     r6,#00h
                ljmp    keypad
e_minor:        cjne    r1,#02h,e_aug
e_min1:                 call    atas
                mov     dptr,#data_emi
                call    tampil
                inc     r6
                cjne    r6,#04fh,e_min1
                mov     r6,#00h
                ljmp    keypad
e_aug:          cjne    r1,#03h,awal_e
e_aug1:                 call    atas
                mov     dptr,#data_eaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,e_aug1
                mov     r6,#00h
awal_e:                 mov   key_data_e,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_f:        jb      p2.5,kunci_fis
                djnz    bouncing,kunci_f
                jnb     p2.5,$
                inc     key_data_f
                mov     r1,key_data_f
                cjne    r1,#04h,macam_f
                mov     r1,#01h
macam_f:        cjne    r1,#01h,f_minor
f_mayor:        call    atas
                mov     dptr,#data_f
                call    tampil
                inc     r6
                cjne    r6,#04fh,f_mayor
                mov     r6,#00h
                ljmp    keypad
f_minor:        cjne    r1,#02h,f_aug
f_min1:                 call    atas
                mov     dptr,#data_fmi
                call    tampil
                inc     r6
                cjne    r6,#04fh,f_min1
                mov     r6,#00h
                ljmp    keypad
f_aug:          cjne    r1,#03h,awal_f
f_aug1:                 call    atas
```

```
                mov     dptr,#data_faug
                call    tampil
                inc     r6
                cjne    r6,#04fh,f_aug1
                mov     r6,#00h
awal_f:         mov     key_data_f,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_fis:      jb      p2.6,kunci_g
                djnz    bouncing,kunci_fis
                jnb     p2.6,$
                inc     key_data_fkres
                mov     r1,key_data_fkres
                cjne    r1,#04h,macam_fkres
                mov     r1,#01h
macam_fkres:    cjne    r1,#01h,fkres_minor
fkres_mayor:    call    atas
                mov     dptr,#data_fis
                call    tampil
                inc     r6
                cjne    r6,#04fh,fkres_mayor
                mov     r6,#00h
                ljmp    keypad
fkres_minor:    cjne    r1,#02h,fkres_aug
fkres_min1:     call    atas
                mov     dptr,#data_fismi
                call    tampil
                inc     r6
                cjne    r6,#04fh,fkres_min1
                mov     r6,#00h
                ljmp    keypad
fkres_aug:      cjne    r1,#03h,awal_fkres
fkres_aug1:     call    atas
                mov     dptr,#data_fisaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,fkres_aug1
                mov     r6,#00h
awal_fkres:     mov     key_data_fkres,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_g:        jb      p2.7,kunci_gis
                djnz    bouncing,kunci_g
                jnb     p2.7,$
                inc     key_data_g
                mov     r1,key_data_g
```

```
                cjne    r1,#04h,macam_g
                mov     r1,#01h
macam_g:        cjne    r1,#01h,g_minor
g_mayor:        call    atas
                mov     dptr,#data_g
                call    tampil
                inc     r6
                cjne    r6,#04fh,g_mayor
                mov     r6,#00h
                ljmp    keypad
g_minor:        cjne    r1,#02h,g_aug
g_min1:                 call    atas
                mov     dptr,#data_gmi
                call    tampil
                inc     r6
                cjne    r6,#04fh,g_min1
                mov     r6,#00h
                ljmp    keypad
g_aug:          cjne    r1,#03h,awal_g
g_aug1:                 call    atas
                mov     dptr,#data_gaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,g_aug1
                mov     r6,#00h
awal_g:                 mov     key_data_g,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_gis:      mov     p2,#0fbh                ;baris3
                jb      p2.4,kunci_a
                djnz    bouncing,kunci_gis
                jnb     p2.4,$
                inc     key_data_gkres
                mov     r1,key_data_gkres
                cjne    r1,#04h,macam_gkres
                mov     r1,#01h
macam_gkres:    cjne    r1,#01h,gkres_minor
gkres_mayor:    call    atas
                mov     dptr,#data_gis
                call    tampil
                inc     r6
                cjne    r6,#04fh,gkres_mayor
                mov     r6,#00h
                ljmp    keypad
gkres_minor:    cjne    r1,#02h,gkres_aug
gkres_min1:     call    atas
```

```
                mov     dptr,#data_gismi
                call    tampil
                inc     r6
                cjne    r6,#04fh,gkres_min1
                mov     r6,#00h
                ljmp    keypad
gkres_aug:      cjne    r1,#03h,awal_gkres
gkres_aug1:     call    atas
                mov     dptr,#data_gisaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,gkres_aug1
                mov     r6,#00h
awal_gkres:     mov     key_data_gkres,#00h
                ljmp    keypad
;-------------------------------------------------
kunci_a:        jb      p2.5,kunci_ais
                djnz    bouncing,kunci_a
                jnb     p2.5,$
                inc     key_data_a
                mov     r1,key_data_a
                cjne    r1,#04h,macam_a
                mov     r1,#01h
macam_a:        cjne    r1,#01h,a_minor
a_mayor:        call    atas
                mov     dptr,#data_a
                call    tampil
                inc     r6
                cjne    r6,#04fh,a_mayor
                mov     r6,#00h
                ljmp    keypad
a_minor:        cjne    r1,#02h,a_aug
a_min1:                 call    atas
                mov     dptr,#data_ami
                call    tampil
                inc     r6
                cjne    r6,#04fh,a_min1
                mov     r6,#00h
                ljmp    keypad
a_aug:          cjne    r1,#03h,awal_a
a_aug1:                 call    atas
                mov     dptr,#data_aaug
                call    tampil
                inc     r6
                cjne    r6,#04fh,a_aug1
                mov     r6,#00h
```

```
awal_a:          mov   key_data_a,#00h
                 ljmp  keypad
;-------------------------------------------------------
kunci_ais:       jb    p2.6,kunci_b
                 djnz  bouncing,kunci_ais
                 jnb   p2.6,$
                 inc   key_data_akres
                 mov   r1,key_data_akres
                 cjne  r1,#04h,macam_akres
                 mov   r1,#01h
macam_akres:     cjne  r1,#01h,akres_minor
akres_mayor:     call  atas
                 mov   dptr,#data_ais
                 call  tampil
                 inc   r6
                 cjne  r6,#04fh,akres_mayor
                 mov   r6,#00h
                 ljmp  keypad
akres_minor:     cjne  r1,#02h,akres_aug
akres_min1:      call  atas
                 mov   dptr,#data_aismi
                 call  tampil
                 inc   r6
                 cjne  r6,#04fh,akres_min1
                 mov   r6,#00h
                 ljmp  keypad
akres_aug:       cjne  r1,#03h,awal_akres
akres_aug1:      call  atas
                 mov   dptr,#data_aisaug
                 call  tampil
                 inc   r6
                 cjne  r6,#04fh,akres_aug1
                 mov   r6,#00h
awal_akres:      mov   key_data_akres,#00h
                 ljmp  keypad
;-------------------------------------------------------
kunci_b:         jb    p2.7,demo
                 djnz  bouncing,kunci_b
                 jnb   p2.7,$
                 inc   key_data_b
                 mov   r1,key_data_b
                 cjne  r1,#04h,macam_b
                 mov   r1,#01h
macam_b:         cjne  r1,#01h,b_minor
b_mayor:         call  atas
                 mov   dptr,#data_b
```

```
                call    tampil
                inc     r6
                cjne    r6,#04fh,b_mayor
                mov     r6,#00h
                ljmp    keypad
b_minor:        cjne    r1,#02h,b_aug
b_min1:                 call    atas
                mov     dptr,#data_bmi
                call    tampil
                inc     r6
                cjne    r6,#04fh,b_min1
                mov     r6,#00h
                ljmp    keypad
b_aug:          cjne    r1,#03h,awal_b
b_aug1:                 call    atas
                mov     dptr,#data_baug
                call    tampil
                inc     r6
                cjne    r6,#04fh,b_aug1
                mov     r6,#00h
awal_b:                 mov     key_data_b,#00h
                ljmp    keypad
;-----------------------------------------------
demo:           mov     p2,#0f7h
                jb      p2.4,kosong1
                djnz    bouncing,demo
                jnb     p2.7,$
                call    demo1
                ljmp    keypad
kosong1:        jb      p2.5,kosong2
                jmp     nokey
kosong2:        jb      p2.6,kosong3
                jmp     nokey
kosong3:        jb      p2.7,nokey
                jmp     nokey
nokey:          ljmp    keypad
;===============================================
tampil:         clr     a
                call    lcd_kord
tampila:        clr     a
                movc    a,@a+dptr
                inc     dptr
                mov     p3,a
                mov     p1,r3
                call    delay
                mov     a,r3
```

```
                rl      a
                mov     r3,a
                inc     r0
                cjne    r0,#07h,tampila
                mov     r0,#00h
                mov     r3,#0feh
                ret
;-----------------------------------------------
delay:  mov     r4,#0ffh
delay1:mov      r5,#06h
                djnz    r5,$
                djnz    r4,delay1
                ret
;-----------------------------------------------
tamtam:                 clr     a
                movc    a,@a+dptr
                inc     dptr
                mov     p3,a
                mov     p1,r3
                call    delay
                mov     a,r3
                rl      a
                mov     r3,a
                inc     r0
                cjne    r0,#07h,tamtam
                mov     r0,#00h
                mov     r3,#0feh
                ret
;-----------------------------------------------
demo1:                  call    atas_demo
demo1a:                 mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#03ch,demo1a
                mov     r6,#00h
demo1b:                 mov     dptr,#data_demoa
                call    tamtam
                inc     r6
                cjne    r6,#03ch,demo1b
                mov     r6,#00h
demo1c:                 mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#03ch,demo1c
                mov     r6,#00h
demo1d:                 mov     dptr,#data_demoa
```

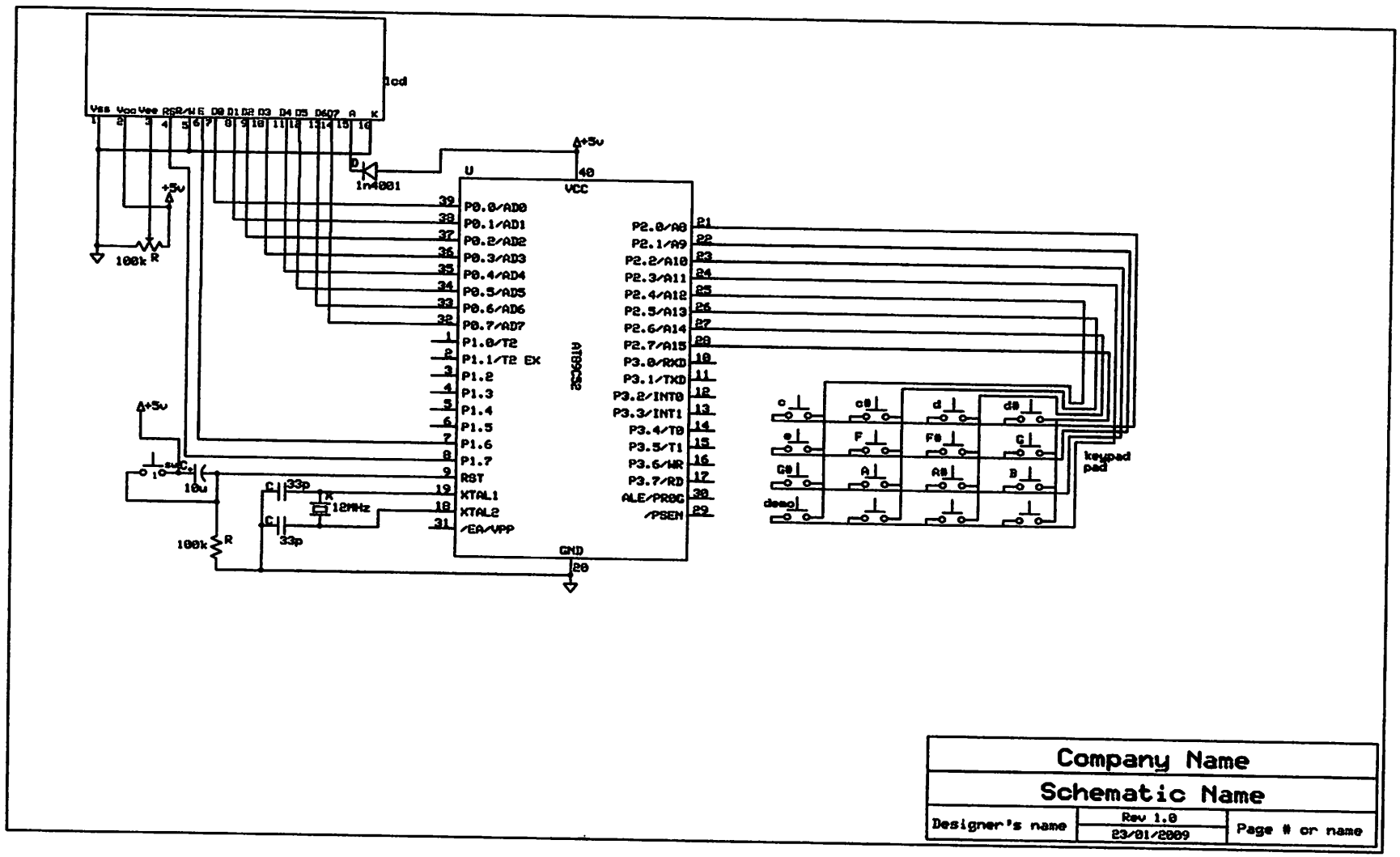
```
                call    tamtam
                inc     r6
                cjne    r6,#03ch,demo1d
                mov     r6,#00h
demo1e:         mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#03ch,demo1e
                mov     r6,#00h
demo1f:         mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#0f0h,demo1f
                mov     r6,#00h
demo1g:         mov     dptr,#data_demoa
                call    tamtam
                inc     r6
                cjne    r6,#01eh,demo1g
                mov     r6,#00h
demo1h:         mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#0f0h,demo1h
                mov     r6,#00h
demo1i:         mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#04bh,demo1i
                mov     r6,#00h
demo1j:         mov     dptr,#data_demoa
                call    tamtam
                inc     r6
                cjne    r6,#01eh,demo1j
                mov     r6,#00h
demo1k:         mov     dptr,#data_demoe
                call    tamtam
                inc     r6
                cjne    r6,#069h,demo1k
                mov     r6,#00h
demo1l:         mov     dptr,#data_demoa
                call    tamtam
                inc     r6
                cjne    r6,#087h,demo1l
                mov     r6,#00h
demo1m:         mov     dptr,#data_demod
                call    tamtam
```

```
            inc     r6
            cjne    r6,#0fh,demo1m
            mov     r6,#00h
demo1n:             mov   dptr,#data_demoa
            call    tamtam
            inc     r6
            cjne    r6,#02dh,demo1n
            mov     r6,#00h
demo1o:             mov   dptr,#data_demoe
            call    tamtam
            inc     r6
            cjne    r6,#069h,demo1o
            mov     r6,#00h
demo1p:             mov   dptr,#data_demoa
            call    tamtam
            inc     r6
            cjne    r6,#04bh,demo1p
            mov     r6,#00h
demo1q:             mov   dptr,#data_demod
            call    tamtam
            inc     r6
            cjne    r6,#0fh,demo1q
demo1r:             mov   dptr,#data_demoa
            call    tamtam
            inc     r6
            cjne    r6,#02dh,demo1r
demo1s:             mov   dptr,#data_demoe
            call    tamtam
            inc     r6
            cjne    r6,#05ah,demo1s
demo1t:             mov   dptr,#data_demod
            call    tamtam
            inc     r6
            cjne    r6,#01eh,demo1t
demo1u:             mov   dptr,#data_demoa
            call    tamtam
            inc     r6
            cjne    r6,#01eh,demo1u
demo1v:             mov   dptr,#data_demoe
            call    tamtam
            inc     r6
            cjne    r6,#02dh,demo1v
demo1w:             mov   dptr,#data_demod
            call    tamtam
            inc     r6
            cjne    r6,#02dh,demo1w
```

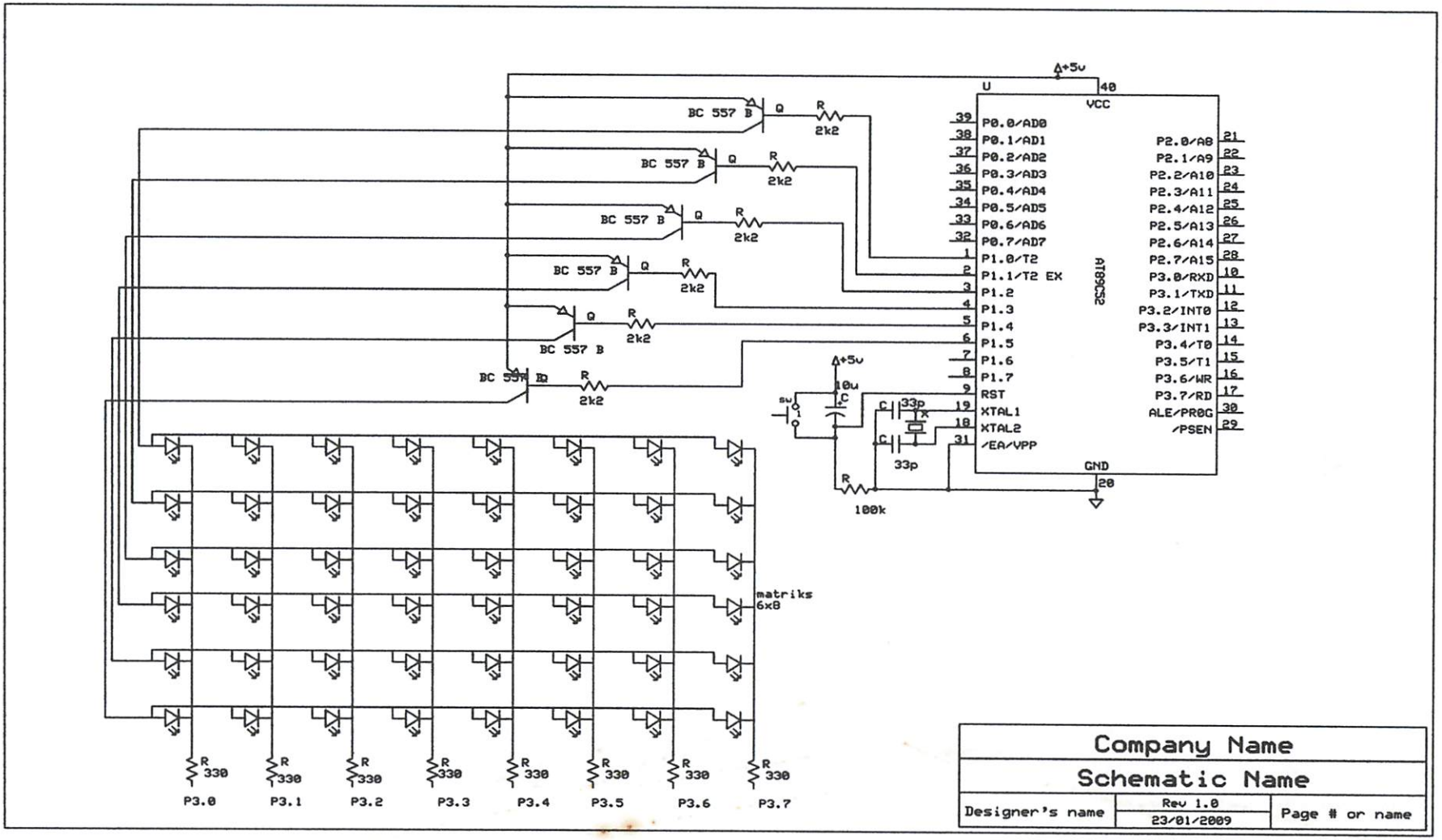
```
demo1x:         mov   dptr,#data_demoa
                call  tamtam
                inc   r6
                cjne  r6,#02dh,demo1x
                ret
;=============================================:
;       SUBROUTINE LCD                        :
;=============================================:
init_lcd:       mov   a,#03fh
                call  instruksi
                mov   a,#0eh
                call  instruksi
                mov   a,#06h
                call  instruksi
                mov   a,#01h
                call  instruksi
                ret
;=============================================:
instruksi:      clr   p1.7   ;rs
                mov   p0,a
                setb  p1.6
                clr   p1.6
                call  delay
                ret
;=============================================:
tulis_data:     setb  p1.7   ;rs
                mov   p0,a
                setb  p1.6
                clr   p1.6
                call  delay
                ret
;=============================================:
;lcd kord
;=============================================:
lcd_kord:       mov   r7,#16
                mov   a,#0c0h
                call  instruksi
tulis2:         clr   a
                movc  a,@a+dptr
                inc   dptr
                call  tulis_data
                djnz  r7,tulis2
                ret
;=============================================:
;lcd atas
;=============================================:
```

```
atas:       mov   dptr,#data_lcd_atas
            mov   r2,#16
            mov   a,#80h
            call  instruksi
tulis1:     clr   a
            movc  a,@a+dptr
            inc   dptr
            call  tulis_data
            djnz  r2,tulis1
            ret
;=============================================:
;lcd atas demo
;=============================================:
atas_demo:  mov   dptr,#data_lcd_atas_demo
            mov   r2,#16
            mov   a,#80h
            call  instruksi
tulis1_demo: clr  a
            movc  a,@a+dptr
            inc   dptr
            call  tulis_data
            djnz  r2,tulis1
            ret
;=============================================:
cursor_home:
      mov   A,#03H
      call  instruksi
      ret
;---------------------------------------------
data_c:     DB    'C MAYOR     ',0ffh,0dfh,0bfh,0ffh,07fh,0ffh,0ffh
data_cmin:  db    'C MINOR     ',0ffh,0dfh,0f7h,0f7h,0efh,0dfh,0ffh
data_caug:  db    'C AUGMEN     ',0ffh,0dfh,0bfh,07fh,07fh,0ffh,0ffh
data_cis:   db    'C# MAYOR     ',0ffh,0efh,0fbh,0fbh,0fbh,0efh,0ffh
data_cismi: db    'C# MINOR     ',0ffh,0efh,0fbh,0fbh,0f7h,0efh,0ffh
data_cisaug: db   'C# AUGMEN     ',0ffh,0ffh,0dfh,0bfh,0bfh,07fh,0ffh
data_d:     db    'D MAYOR     ',0ffh,0ffh,0ffh,0bfh,0dfh,0bfh,0ffh
data_dmi:   db    'D MINOR     ',0ffh,0ffh,0ffh,0bfh,0dfh,07fh,0ffh
data_daug:  db    'D AUGMEN     ',0ffh,0ffh,0efh,0dfh,0dfh,0bfh,0ffh
data_dis:   db    'D# MAYOR     ',0ffh,0fbh,0feh,0feh,0feh,0fbh,0ffh
data_dismi: db    'D# MINOR     ',0ffh,0fbh,0feh,0feh,0fdh,0fbh,0ffh
data_disaug: db   'D# AUGMEN     ',0ffh,0ffh,07fh,0ffh,0ffh,0dfh,0ffh
data_e:     db    'E MAYOR     ',0ffh,0bfh,0bfh,07fh,0ffh,0ffh,0ffh
data_emi:   db    'E MINOR     ',0ffh,0bfh,0bfh,0ffh,0ffh,0ffh,0ffh
data_eaug:  db    'E AUGMEN     ',0ffh,0ffh,0bfh,07fh,07fh,0ffh,0ffh
data_f:     db    'F MAYOR     ',07fh,0dfh,0dfh,0bfh,07fh,07fh,0ffh
data_fmi:   db    'F MINOR     ',07fh,0dfh,0dfh,07fh,07fh,07fh,0ffh
```

```
data_faug:      db      'F AUGMEN      ',0ffh,0ffh,0dfh,0bfh,0bfh,07fh,0ffh
data_fis:       db      'F# MAYOR      ',0bfh,0efh,0efh,0dfh,0bfh,0bfh,0ffh
data_fismi:     db      'F# MINOR     ',0bfh,0efh,0efh,0bfh,0bfh,0bfh,0ffh
data_fisaug:    db      'F# AUGMEN      ',0bfh,0ffh,0efh,0dfh,0dfh,0bfh,0ffh
data_g:         db      'G MAYOR       ',0dfh,0bfh,0ffh,0ffh,0ffh,0dfh,0ffh
data_gmi:       db      'G MINOR      ',0dfh,0f7h,0f7h,0dfh,0dfh,0dfh,0ffh
data_gaug:      db      'G AUGMEN       ',0ffh,0ffh,07fh,0ffh,0ffh,0dfh,0ffh
data_gis:       db      'G# MAYOR      ',0efh,0fbh,0fbh,0f7h,0efh,0efh,0ffh
data_gismi:     db      'G# MINOR     ',0efh,0fbh,0fbh,0efh,0efh,0efh,0ffh
data_gisaug:    db      'G# AUGMEN      ',0ffh,0ffh,0bfh,07fh,07fh,0ffh,0ffh
data_a:         db      'A MAYOR       ',0ffh,0ffh,0bfh,0bfh,0bfh,0ffh,0ffh
data_ami:       db      'A MINOR      ',0ffh,0ffh,0bfh,0bfh,07fh,0ffh,0ffh
data_aaug:      db      'A AUGMEN       ',0ffh,0ffh,0dfh,0bfh,0bfh,07fh,0ffh
data_ais:       db      'A# MAYOR      ',0ffh,07fh,0dfh,0dfh,0dfh,07fh,0ffh
data_aismi:     db      'A# MINOR     ',0ffh,07fh,0dfh,0dfh,0bfh,07fh,0ffh
data_aisaug:    db      'A# AUGMEN      ',0ffh,0ffh,0efh,0dfh,0dfh,0bfh,0ffh
data_b:         db      'B MAYOR       ',0ffh,0bfh,0efh,0efh,0efh,0bfh,0ffh
data_bmi:       db      'B MINOR      ',0ffh,0bfh,0efh,0efh,0dfh,0bfh,0ffh
data_baug:      db      'B AUGMEN       ',0ffh,0bfh,07fh,0ffh,0ffh,0dfh,0ffh
data_lcd_atas: db       'tampilan kord: '
data_lcd_atas_demo: db      'demo ajah      '
data_demoe:     db      0ffh,0bfh,0bfh,07fh,0ffh,0ffh,0ffh
data_demoa:     db      0ffh,0ffh,0bfh,0bfh,0bfh,0ffh,0ffh
data_demod:     db      0ffh,0ffh,0ffh,0bfh,0dfh,0bfh,0ffh
end
```



Company Name		
Schematic Name		
Designer's name	Rev 1.0 23/01/2009	Page # or name


+5v
U
40
VCC
AT89C52
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P1.0/T2
P1.1/T2 EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
XTAL1
XTAL2
/EA/VPP
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
ALE/PROG
/PSEN
GND
20
BC 557 B
Q
R
2k2
sw
10u
C
33p
X
100k
matriks
6x8
330
P3.0
P3.1
P3.2
P3.3
P3.4
P3.5
P3.6
P3.7
Company Name
Schematic Name
Designer's name
Rev 1.0
23/01/2009
Page # or name

Akord

Dari Wikipedia bahasa Indonesia, ensiklopedia bebas

Akord adalah kumpulan tiga nada atau lebih yang bila dimainkan secara bersamaan terdengar harmonis. Akord bisa dimainkan secara terputus-putus ataupun secara bersamaan. Akord ini digunakan untuk mengiringi suatu lagu. Ketika Anda menekan tiga tuts piano C, E dan G secara bersamaan, ini berarti anda sudah memainkan akord. Contoh alat musik lainnya yang bisa memainkan akord adalah gitar (akustik dan listrik), organ, *electone*.

Akord C Mayor (C) pada gitar (dengan tuning standar)

Daftar isi

Macam-macam akord

Akord itu banyak macamnya. Antara lain **akord mayor**, **akord minor**, **akord dominan septim**, **akord *diminished***, **akord *augmented***, **akord minor 6**, **akord mayor 7**, **akord *suspended*** dan masih banyak yang lainnya. Akord yang paling sering dipakai dalam suatu lagu yang sederhana adalah akord mayor, akord minor dan akord dominan septim. Akord lainnya digunakan untuk memperindah atau mengubah kualitas suatu lagu. Penyisipan akord yang berbeda akan memberikan efek rasa yang berbeda dalam iringan suatu lagu.

Akord

Daftar isi

Macam-macam akord

Akord mayor

Akord mayor adalah akord yang interval antara nadanya 2 - 1 1/2

Contoh akord mayor:

- Cb (Cb-Eb-Gb) = B
- C (C-E-G)
- C# (C#-E#-G#) = Db (Db-F-Ab)
- D (D-F#-A)
- D# (D#-G-A#) = Eb (Eb-G-Bb)
- E (E-G#-B) = Fb (Fb-Ab-Cb)
- E# (E#-A-B#) = F (F-A-C)
- F (F-A-C)
- F# (F#-A#-C#) = Gb (Gb-Bb-Db)
- G (G-B-D)
- G# (G#-B#-D#) = As (Ab-C-Eb)
- A (A-C#-E)
- A# (A#-D-E#) = Bb (Bb-D-F)
- B (B-D#-F#) = Cb
- B# (B#-E-G) = C

Akord yang memiliki nama berbeda namun bila dimainkan bersuara sama disebut **Akord Enharmonis**. Contohnya: akord Cb (Ces mayor) dengan B (B mayor).

Akord di atas adalah **akord dasar**. Akord tersebut bisa dibalik-balik urutannya (disebut **balikan pertama** dan **balikan kedua**). Misalnya: C on E(C/E). Ini berarti kita harus memainkan akord dengan urutan E-G-C' bukan C-E-G. C on E adalah balikan pertama dari akord dasar C. Balikan keduanya adalah C on G(C/G) yaitu G-E'-C'.

Notasi: C, CM, Cma

Akord minor

Akord minor adalah akord yang interval antara nadanya 1 1/2 - 2. Apabila anda sudah tahu suatu akord mayor misalnya; **C mayor** maka anda bisa mengetahui pula akord minornya (**C minor**) yaitu dengan cara *menurunkan nada yang ada ditengah sebanyak setengah interval*. Sehingga didapat akord C minor adalah **C-Es(E diturunkan setengah menjadi Es)-G.**

Akord mayor

Akord mayor adalah akord yang interval antara nadanya 2 - 1 1 2

Contoh akord mayor:

- C (C-E-G) = C
- D (D-F#-A)
- E (E-G#-B) = E (E-A-C)
- F (F-A-C)
- G (G-B-D)
- A (A-C#-E)
- B (B-D#-F#) = C

Akord minor

Akord minor adalah akord yang interval antara nadanya 1 1 2 - 2. Apabila anda

Notasi: C7, C^7

Akord augmented

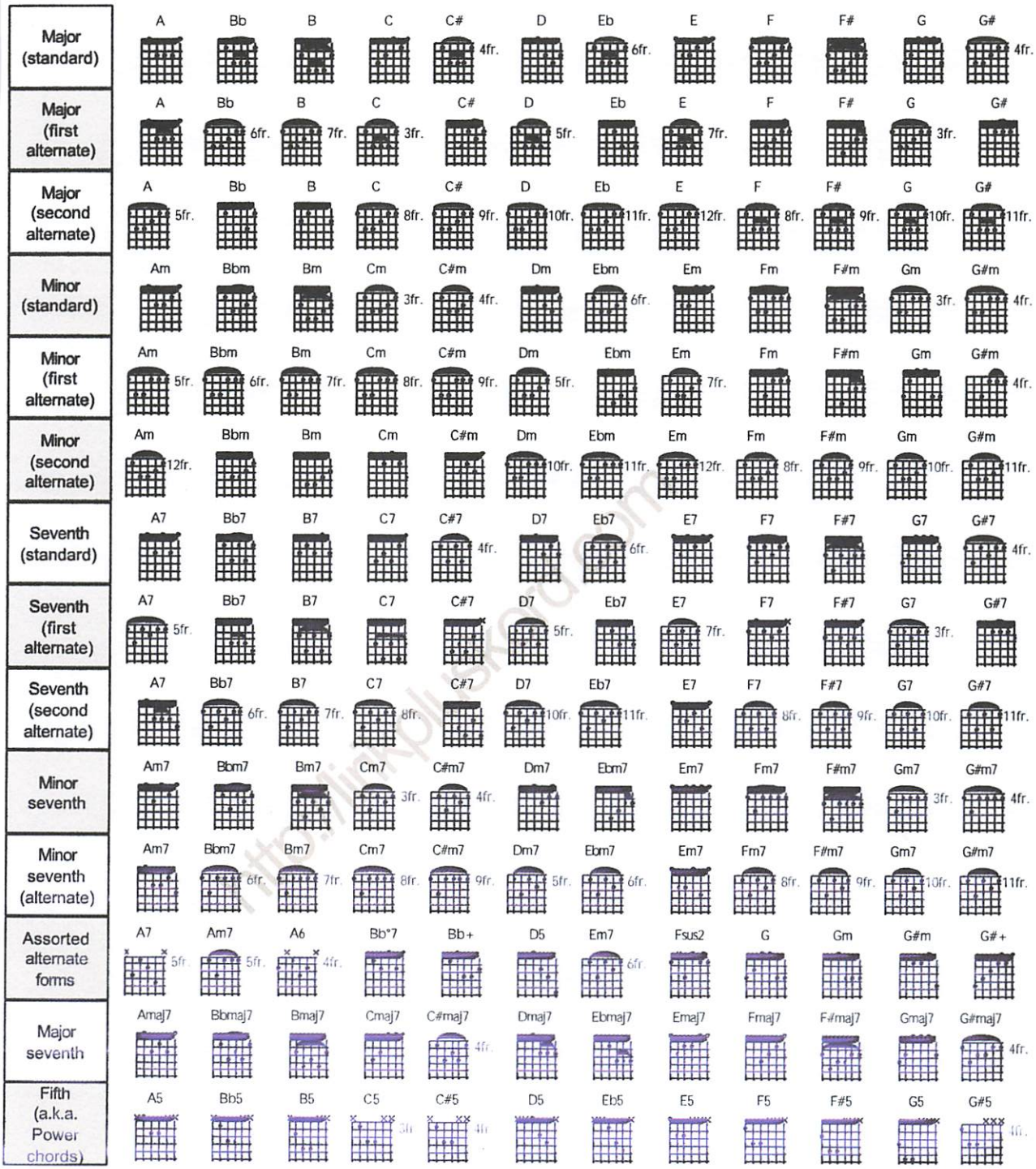
Akord augmented adalah akord yang interval antara nadanya 2 - 2. Notasi : Caug / C+

Akord diminished

Akord diminished adalah akord yang interval antar nadanya adalah 1 1/2 - 1 1/2. Notasi : Cdim.

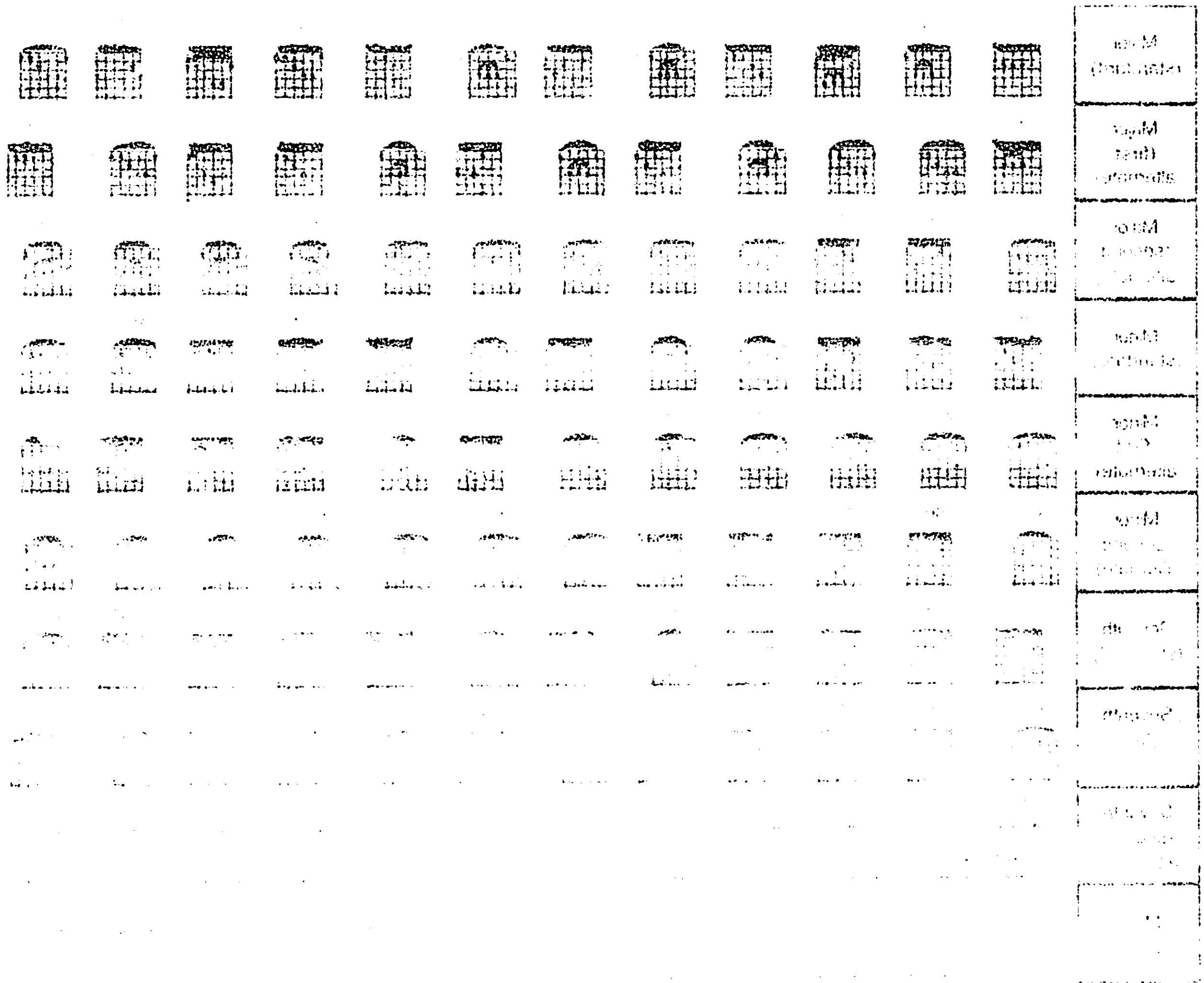
Guitar Chord Chart

From Songbird Chords



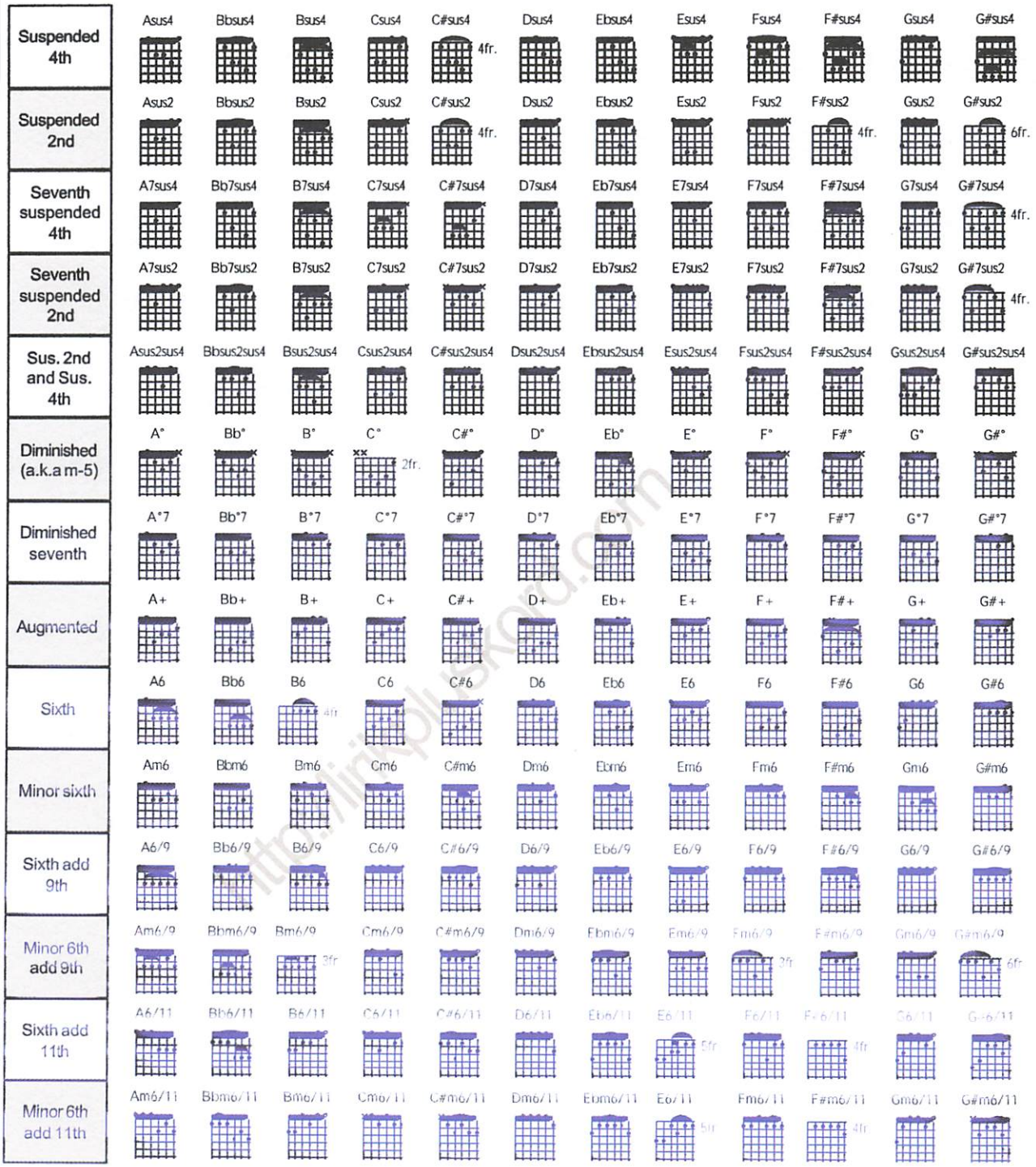
Guitar Chord Chart

From Songbird Chords



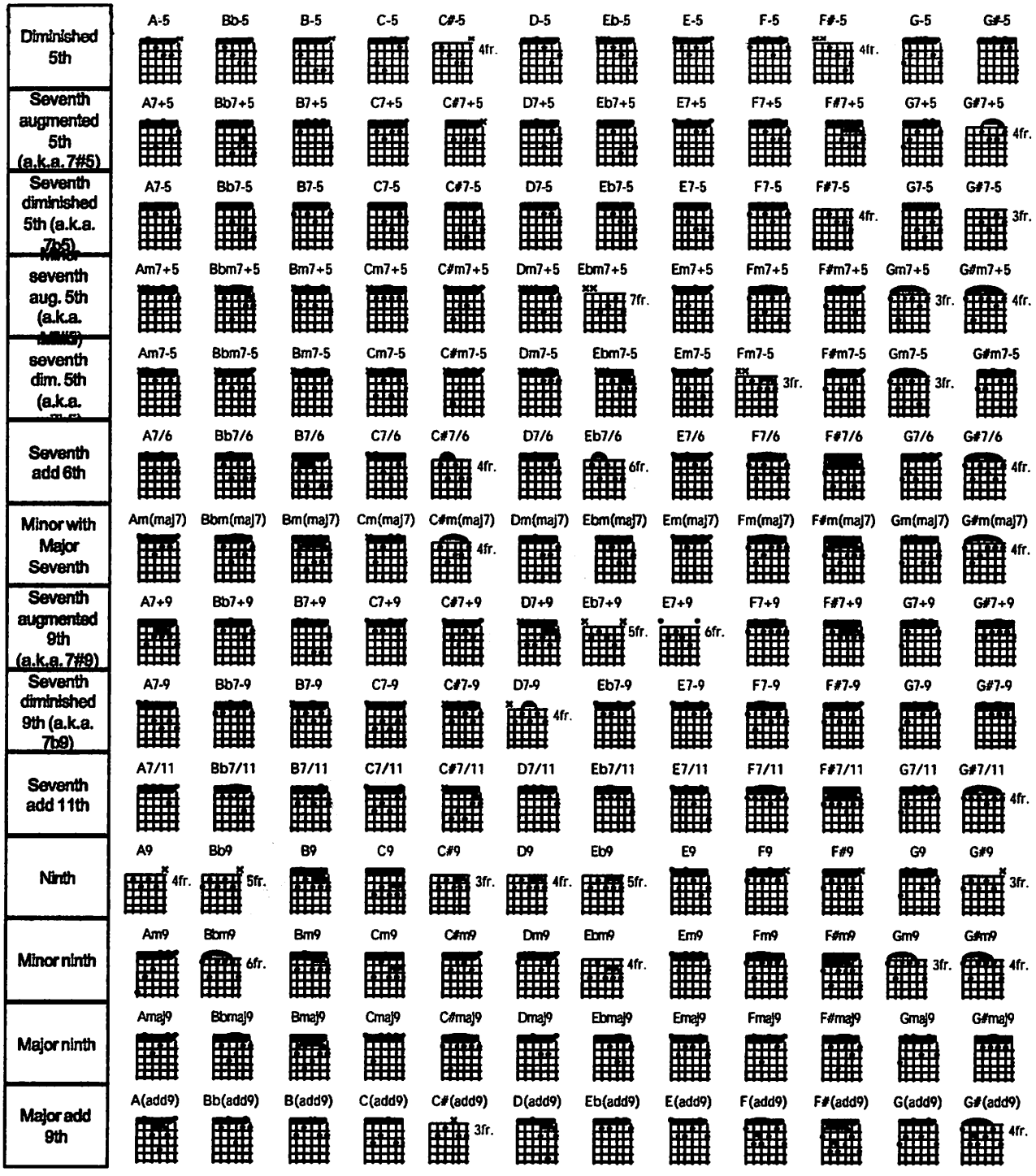
Guitar Chord Chart

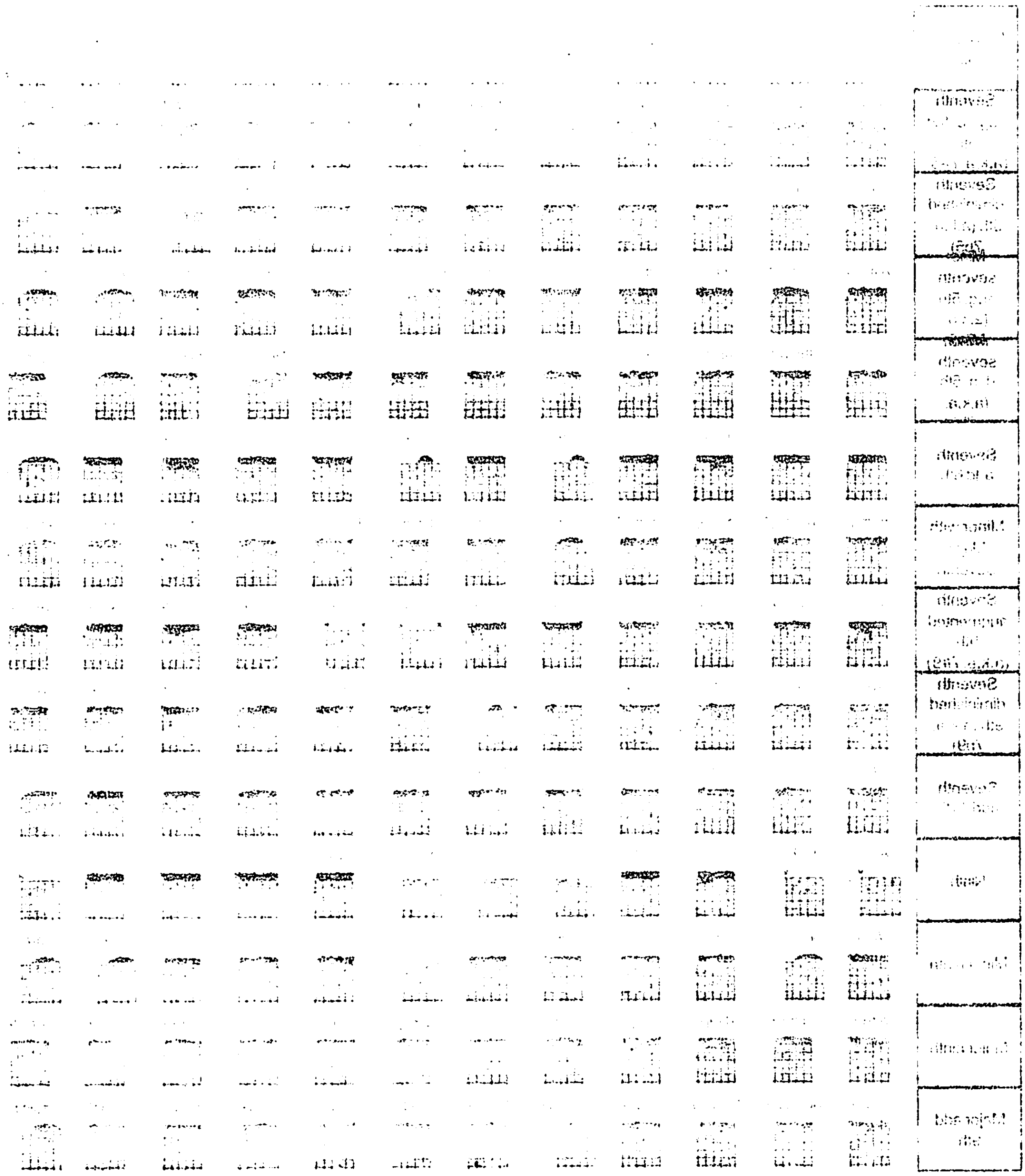
From Songbird Chords



Guitar Chord Chart

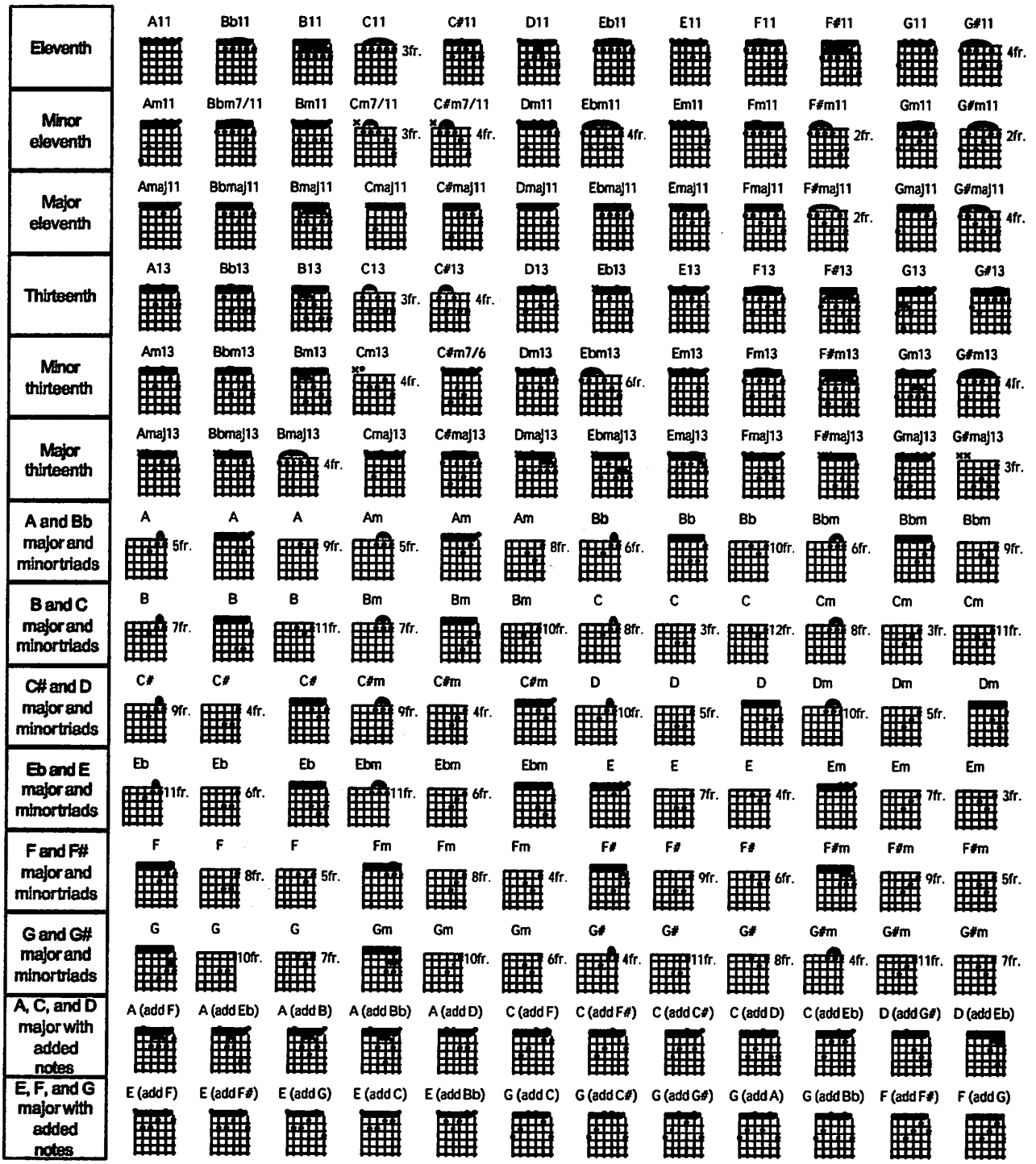
From Songbird Chords





Guitar Chord Chart

From Songbird Chords

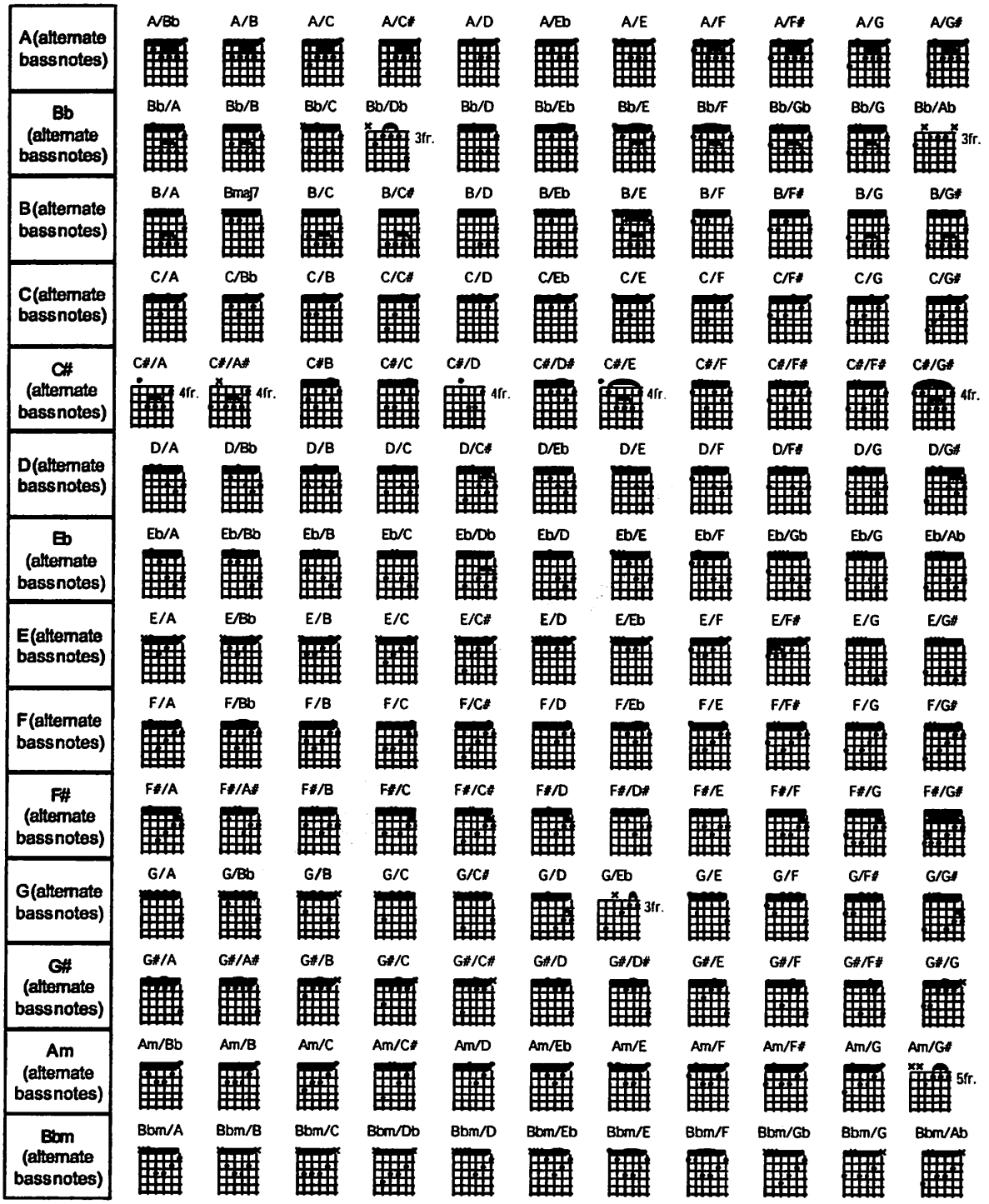


Guitar Chord Chart

From Songbird Words

Guitar Chord Chart

From Songbird Chords

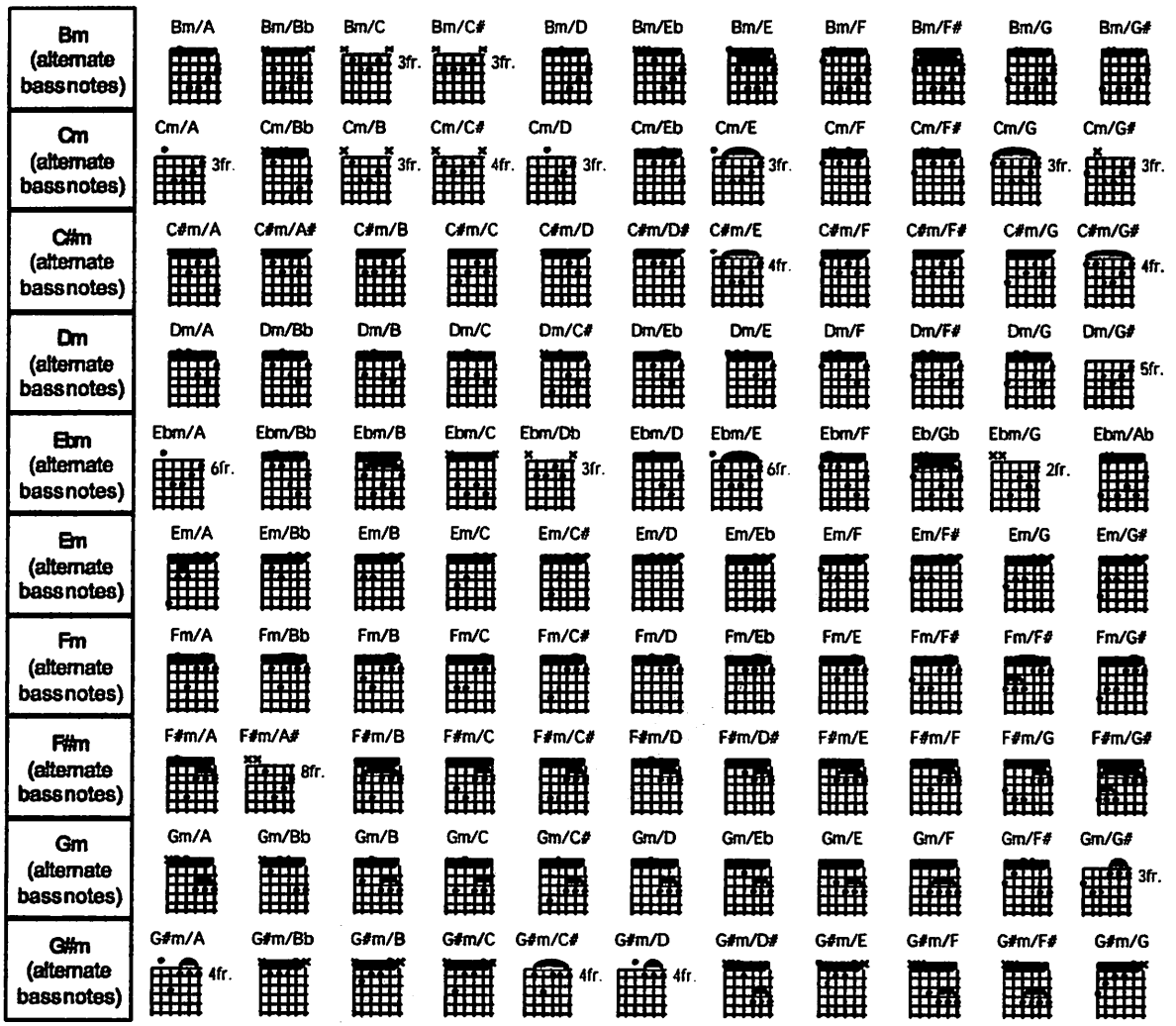


Guitar Chord Chart

From Songbird Chords

Guitar Chord Chart

From Songbird Chords



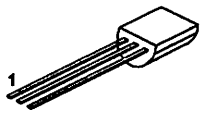
Guitar Chord Chart

From Songbird Chords

UTC BC556/557/558 PNP EPITAXIAL SILICON TRANSISTOR

SWITCHING AND AMPLIFIER APPLICATIONS

FEATURES
* High Voltage: BC556, V_{CEO}=-65V



TO-92

1: COLLECTOR 2: BASE 3: EMITTER

ABSOLUTE MAXIMUM RATINGS (Ta=25°C, unless otherwise specified)

PARAMETER	SYMBOL	RATING	UNIT
Collector-base voltage	V_{CBO}		
: BC556		-80	V
: BC557		-50	V
: BC558		-30	V
Collector-emitter voltage	V_{CEO}		
: BC556		-65	V
: BC557		-45	V
: BC558		-30	V
Emitter-base voltage	V_{EBO}	-5	V
Collector current (DC)	I_C	-100	mA
Collector dissipation	P_C	500	mW
Junction Temperature	T_J	150	°C
Storage Temperature	T_{STG}	-65 ~ +150	°C

ELECTRICAL CHARACTERISTICS (Ta=25°C, unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Collector Cut-off Current	I_{CBO}	V_{CB}=-30V, I_E=0			-15	nA
DC current gain	h_{FE}	V_{CE}=-5V, I_C=2mA	110		800	
Collector-emitter saturation voltage	$V_{CE(sat)}$	I_C=-10mA, I_B=-0.5mA		-90	-300	mV
		I_C=-100mA, I_B=-5mA		-250	-650	mV
Base-emitter saturation voltage	$V_{BE(sat)}$	I_C=-10mA, I_B=-0.5mA		-700		mV
		I_C=-100mA, I_B=-5mA		-900		
Base-emitter on voltage	$V_{BE(on)}$	V_{CE}=-5V, I_C=-2mA	-600	-660	-750	mV
		V_{CE}=-5V, I_C=-10mA			-800	mV
Current gain bandwidth product	f_T	V_{CE}=-5V, I_C=-10mA, f=10MHz		150		MHz
Output Capacitance	Cob	V_{CB}=-10V, I_E=0, f=1MHz			6	pF

SWITCHING AND AMPLIFIER APPLICATIONS

FEATURES

- High Voltage: BC556: -65V

1. COLLECTOR 2. BASE 3. EMITTER

ABSOLUTE MAXIMUM RATINGS (Ta=25°C unless otherwise specified)

PARAMETER	SYMBOL	RATING	UNIT
Collector-base voltage	V_{CBO}		
BC556		-80	V
BC557		-50	V
BC558		-30	V
Collector-emitter voltage	V_{CEO}		
BC556		-65	V
BC557		-45	V
BC558		-30	V
Emitter-base voltage	V_{EBO}	-5	V
Collector current (DC)	I_C	-100	mA
Collector dissipation	P_C	500	mW
Junction Temperature	T_J	150	°C
Storage Temperature	T_{STG}	-65 ~ +150	°C

ELECTRICAL CHARACTERISTICS (Ta=25°C unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Collector Cutoff Current	I_{CBO}	V_{CB}=-30V, I_E=0			-15	nA
		V_{CB}=-30V, I_E=0, Ta=125°C			-4.0	μA
Collector-emitter saturation voltage	$V_{CE(sat)}$	I_C=-10mA, I_B=-0.5mA		-90	-300	mV
		I_C=-100mA, I_B=-5mA		-250	-650	mV
Base-emitter saturation voltage	$V_{BE(sat)}$	I_C=-10mA, I_B=-0.5mA		-700		mV
		I_C=-100mA, I_B=-5mA		-900		mV
Base-emitter on voltage	$V_{BE(on)}$	V_{CE}=-5V, I_C=-2mA	-600	-660	-750	mV
		V_{CE}=-5V, I_C=-10mA			-800	mV
Current gain bandwidth product	f_T	V_{CE}=-5V, I_C=-10mA, f=100MHz		150		MHz
Output Capacitance	C_{ob}	V_{CB}=-10V, I_E=0, f=1MHz		6		pF

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Noise Figure	NF	V_{CE}=-5V, I_C=-200μA, f=1KHz, R_G=2KΩ		2	10	dB

CLASSIFICATION OF h_{FE}

RANK	A	B	C
h_{FE}	110 - 220	200 – 450	420 - 800

TYPICAL CHARACTERISTICS

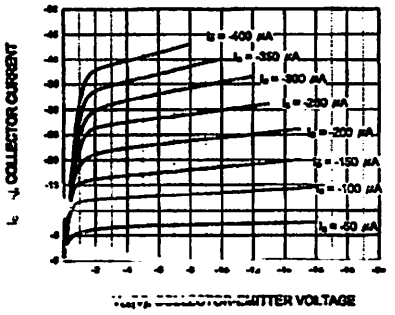


Figure 1. Static Characteristic

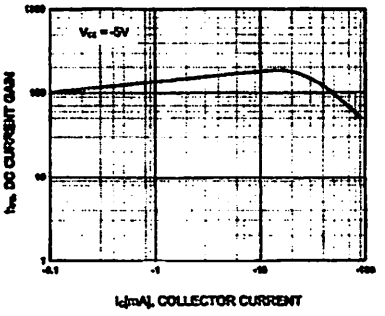


Figure 2. DC current Gain

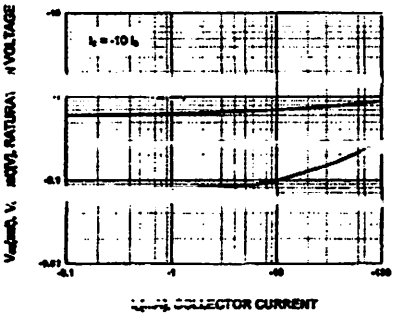


Figure 3. Base-Emitter Saturation Voltage Collector-Emitter Saturation Voltage

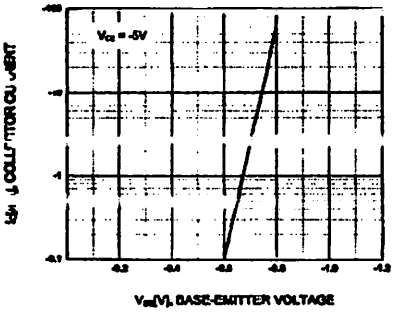


Figure 4. Base-Emitter On Voltage

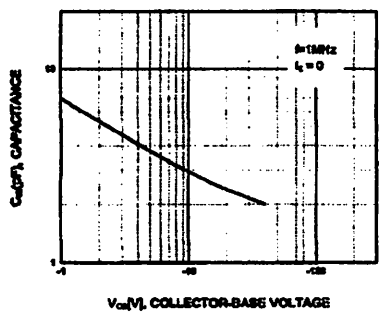


Figure 5. Collector Output Capacitance

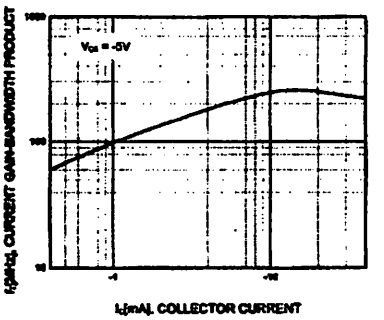


Figure 6. Current Gain Bandwidth Product

Hello. Welcome to Futurlec. The electronic components Superstore. To find the component your looking for, either search by Part Number or visit the relative department.
Need Help.

Transistor - General Purpose

Comparison Table

Technical Data

Electrical Characteristics - BC Series

Type	Package	NPN/PNP	Ic (mA)	Pd (mW)	Vce (max)	Vcb (max)	hfe	@ Ic	FT (MHz)	Complement	Application
BC107	TO-18	NPN	100	300	45	50	110-450	2	300		G.P.
BC108	TO-18	NPN	100	300	20	30	110-800	2	300		G.P.
BC109	TO-18	NPN	100	300	20	30	200-800	2	300		G.P.
BC109C	TO-18	NPN	100	300	20	30	420-800	2	300		G.P.
BC142	TO-39	NPN	1.0A	800	60	80	20-60	200	80	BC143	Audio Driver
BC143	TO-39	PNP	1.0A	800	60	60	20-40	300	-	BC142	Audio Driver
BC177	TO-18	PNP	100	300	45	50	75-260	2	150		G.P.
BC178	TO-18	PNP	100	300	25	30	75-500	2	150		G.P.
BC179	TO-18	PNP	100	300	20	25	125-500	2	150		G.P.
BC182L	TO-92	NPN	100	350	50	60	120-500	2	150	BC212L	G.P.
BC183L	TO-92	NPN	100	350	30	45	120-800	2	150	BC213L	G.P.
BC184L	TO-92	NPN	100	350	30	45	250-800	2	150	BC214L	G.P.
BC212L	TO-92	PNP	100	350	50	60	60	2	280	BC182L	G.P.
BC213L	TO-92	PNP	100	350	30	45	80-400	2	350	BC183L	G.P.
BC214L	TO-92	PNP	100	350	30	45	140-600	2	320	BC184L	G.P.
BC307	TO-92	PNP	100	350	45	50	180-460	2	280		Amplifier
BC327	TO-92	PNP	500	625	45	50	100-600	100	100	BC337	Audio Driver
BC328	TO-92	PNP	500	625	25	30	100-600	100	100		Audio Driver
BC337	TO-92	NPN	500	625	45	50	100-600	100	100		Audio Driver
BC338	TO-18	NPN	500	625	25	30	100-600	100	100		Audio Driver
BC546	TO-92	NPN	100	500	65	80	110-450	2	300		G.P.
BC547	TO-92	NPN	100	500	45	50	110-800	2	300		G.P.
BC548	TO-92	NPN	100	500	30	30	110-800	2	300		G.P.
BC549	TO-92	NPN	100	500	30	30	200-800	2	300		Low Noise
BC556	TO-92	PNP	100	500	65	80	75-475	2	200		G.P.
BC557	TO-92	PNP	100	500	45	50	75-800	2	200		G.P.
BC558	TO-92	PNP	100	500	30	30	75-800	2	200		G.P.
BC559	TO-92	PNP	100	500	30	30	125-800	2	200		G.P.
BC639	TO-92	NPN	1.0A	1.0W	80	100	40-250	150	130		Audio Driver
BC640	TO-92	PNP	1.0A	1.0W	80	100	40-250	150	50		Audio Driver

Electrical Characteristics - C Series

Type	Package	NPN/PNP	Ic (mA)	Pd (mW)	Vce (max)	Vcb (max)	hfe	@ Ic	FT (MHz)	Complement	Application
C1815	TO-92	NPN	400	600	70	70	120	-	200		G.P.
C2233	TO-220	NPN	4.0A	75W	300	300	25	-	4		High Current

C2330	T16	NPN	100	900	300	300	100	-	50		HV Video
C2482	T16	NPN	100	900	300	300	100	-	50		HV Video
C9012	TO-92	PNP	500	625	20	40	120	-	200		G.P.
C9013	TO-92	NPN	500	625	20	40	120	-	200		G.P.
C9014	TO-92	NPN	600	500	40	75	200	-	300		G.P.
C9015	TO-92	PNP	600	500	40	75	200	-	300		G.P.

Electrical Characteristics - 2N Series

Type	Package	NPN/PNP	Ic (mA)	Pd (mW)	Vce (max)	Vcb (max)	hfe	@ Ic	FT (MHz)	Complement	Application
PN2222	TO-39	NPN	800	500	40	75	100-300	150	300		Switching
2N2222	TO-18	NPN	800	500	40	75	100-300	150	300		Switching
2N2905	TO-39	PNP	600	600	40	60	100-300	150	200		Switching
2N3703	TO-92	PNP	200	300	30	50	30-150	50	100	2N3705	General Purpose
2N3704	TO-92	NPN	800	360	30	50	100-300	50	100	2N3702	General Purpose
2N3904	TO-92	NPN	200	350	40	60	100-300	10	300	2N3906	Switching
2N3906	TO-92	PNP	200	350	40	40	100-300	10	250	2N3904	Switching
2N4401	TO-92	NPN	600	350	40	60	100-300	150	250	2N4403	G.P.
2N4403	TO-92	PNP	600	350	40	40	100-300	150	200	2N4401	G.P.
2N5401	TO-92	PNP	600	625	150	160	60-240	10	100		Amplifier

* The information appearing here is for comparison purposes only and to use the devices within their specifications, please consult the full data sheet.

Features

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

Description

The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with in-system programmable Flash on a monolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

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

AT89S52

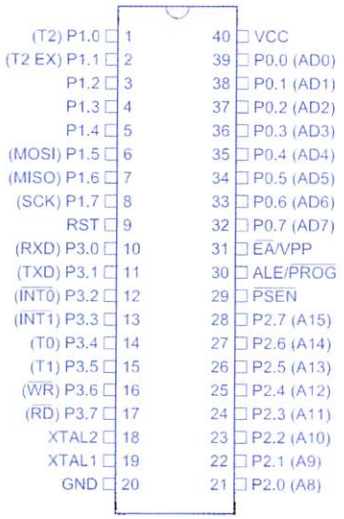
1919B–MICRO–11/03

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

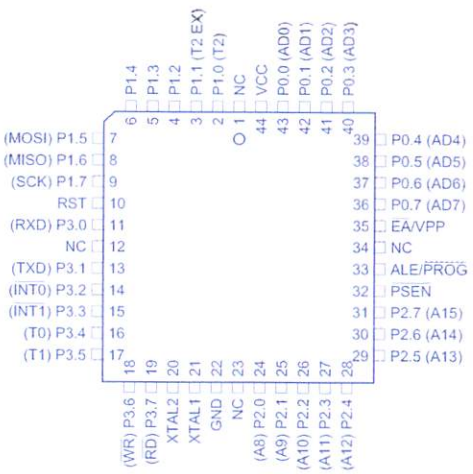
AT89S52

Pin Configurations

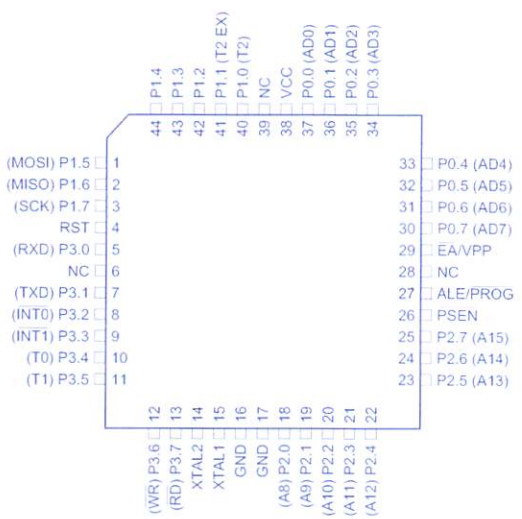
PDIP



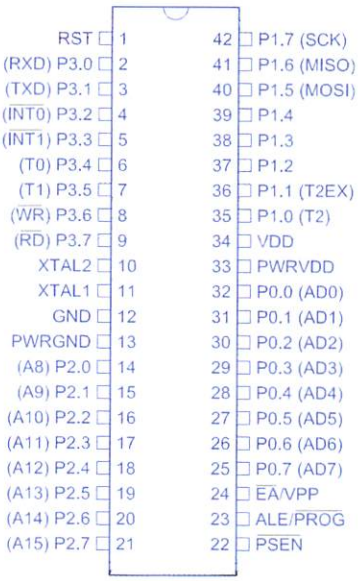
PLCC



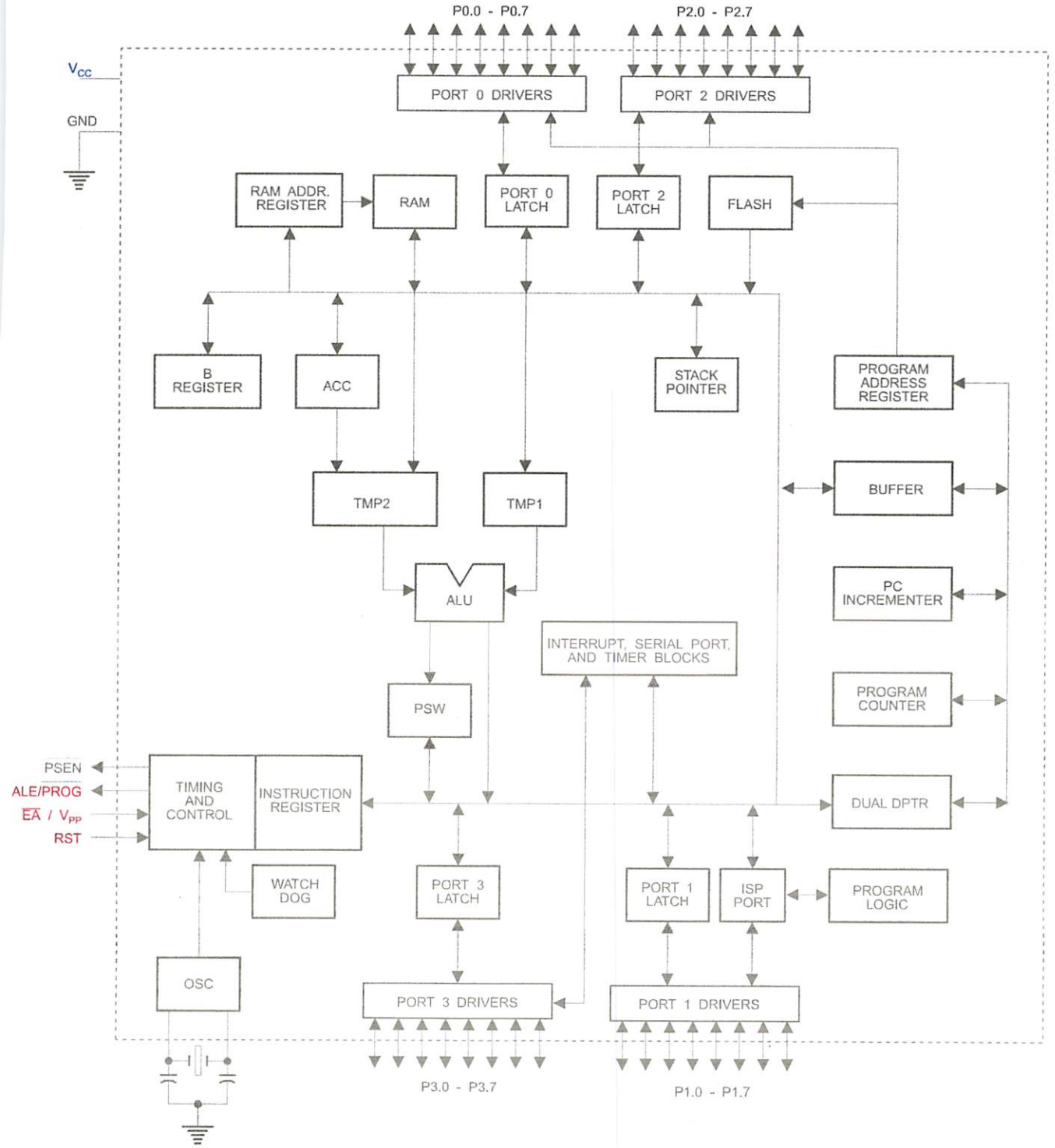
TQFP



PDIP



Block Diagram



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

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

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

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

Port 1

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

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

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

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2

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

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

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

Port 3

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

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

Port 3 also serves the functions of various special features of the AT89S52, as shown in the following table.

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

RST

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

ALE/$\overline{PROG}$

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

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

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

$\overline{PSEN}$

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

When the AT89S52 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP

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

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

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

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

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

Port 3 also receives some additional special features of the AT89C52, as shown in the following table.

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

RST Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives high for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR1 (address A2H) can be used to disable this feature. In the default state of bit DISRTO, RST pin is driven High.

ALE/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 (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.

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

When the AT89C52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP External Access Enable. 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, EA will be internally latched on reset.

EA should be strapped to VCC for internal program executions.

This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming.

XTAL1 Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting oscillator amplifier.

Special Function Registers

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

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

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

Timer 2 Registers: Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 6) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

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

Table 1. AT89S52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								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

Table 2. T2CON – Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

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

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

Table 3. 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 4. 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

Table 3. AUXR0 Auxiliary Register 0

AUXR	Address = [illegible]	Reset Value = XXXXXX0B
	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 bit DPS = 1 selects DP1. The user should [illegible] accessing the respective Data Pointer Register.

Power Off Flag: The Power Off Flag [illegible] SFR PCON. [illegible] up. It can be set and reset under software control and is not affected by reset.

Table 4. AUXR1 Auxiliary Register 1

AUXR1	Address = A2H	Reset Value = XXXXXX0xb
	Not Bit Addressable	

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

–	Reserved for future expansion
DPS	Data Pointer Register Select
	DPS
0	Selects DPTR Registers DP0L, DP0H
1	Selects DPTR Registers DP1L, DP1H

Memory Organization

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

Program Memory

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

On the AT89S52, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

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

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

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

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

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

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

Watchdog Timer (One-time Enabled with Reset-out)

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

Using the WDT

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

WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 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 AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

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

UART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select "Products", then "8051-Architecture Flash Microcontroller", then "Product Overview".

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers" operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select "Products", then "8051-Architecture Flash Microcontroller", then "Product Overview".

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 5. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 5. Timer 2 Operating Modes

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

WDT During Power-down and Idle

UART

Timer 0 and 1

Timer 2

Table 5. Timer 2 Operating Modes

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

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

Capture Mode

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

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 6). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

Figure 1. Timer in Capture Mode

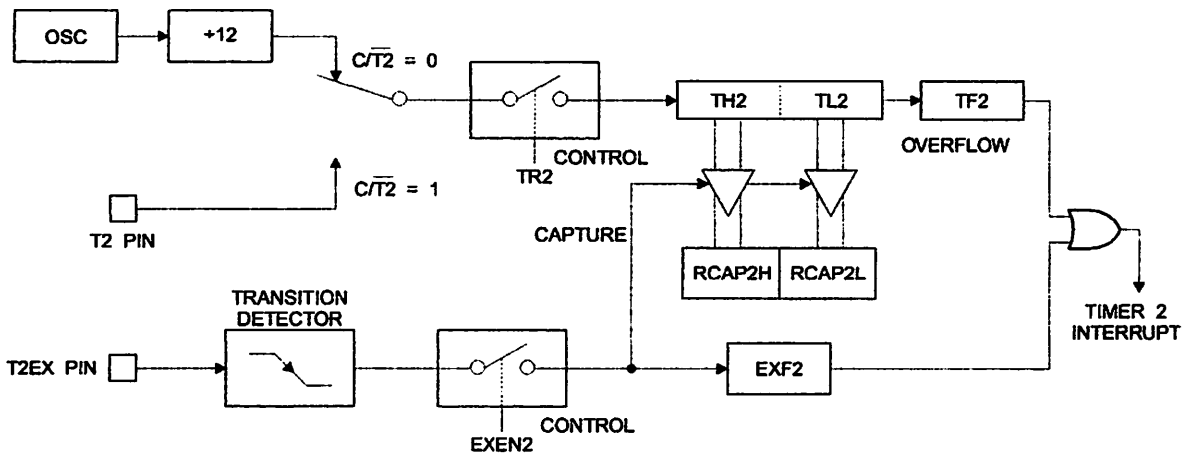


Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

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

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

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

Figure 2. Timer 2 Auto Reload Mode (DCEN = 0)

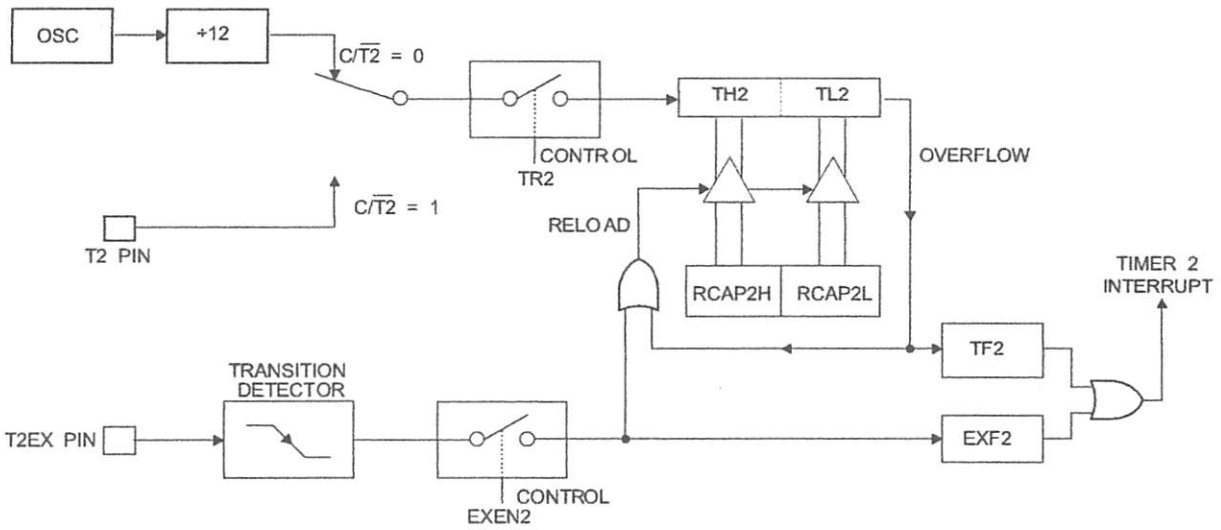


Table 6. T2MOD – Timer 2 Mode Control Register

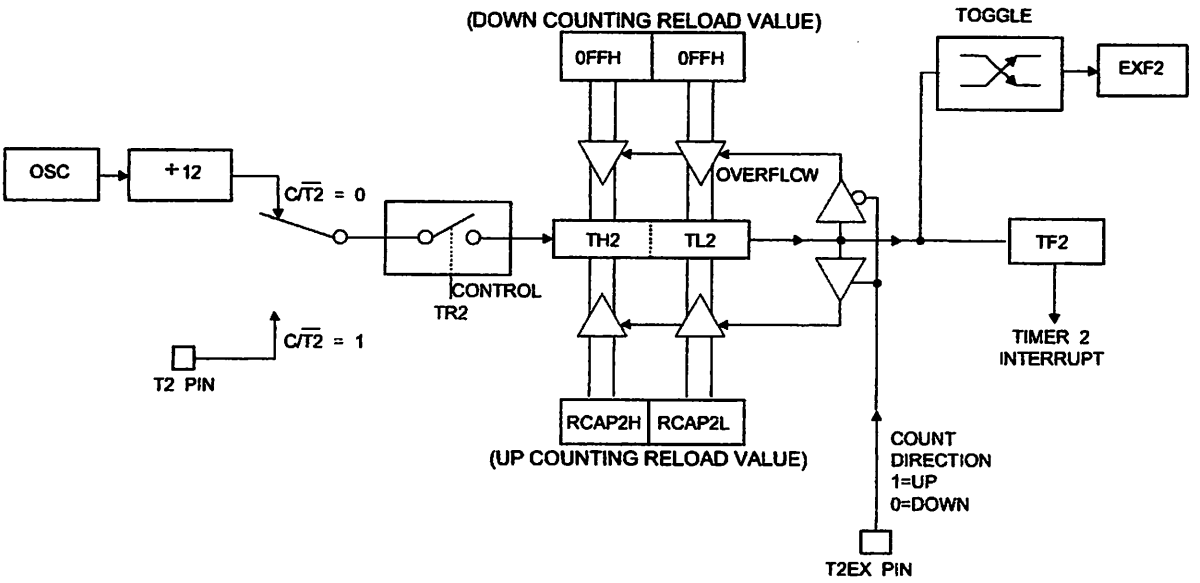
T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

Symbol	Function
–	Not implemented, reserved for future
T2OE	Timer 2 Output Enable bit
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter

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



Baud Rate Generator

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

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

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

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

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

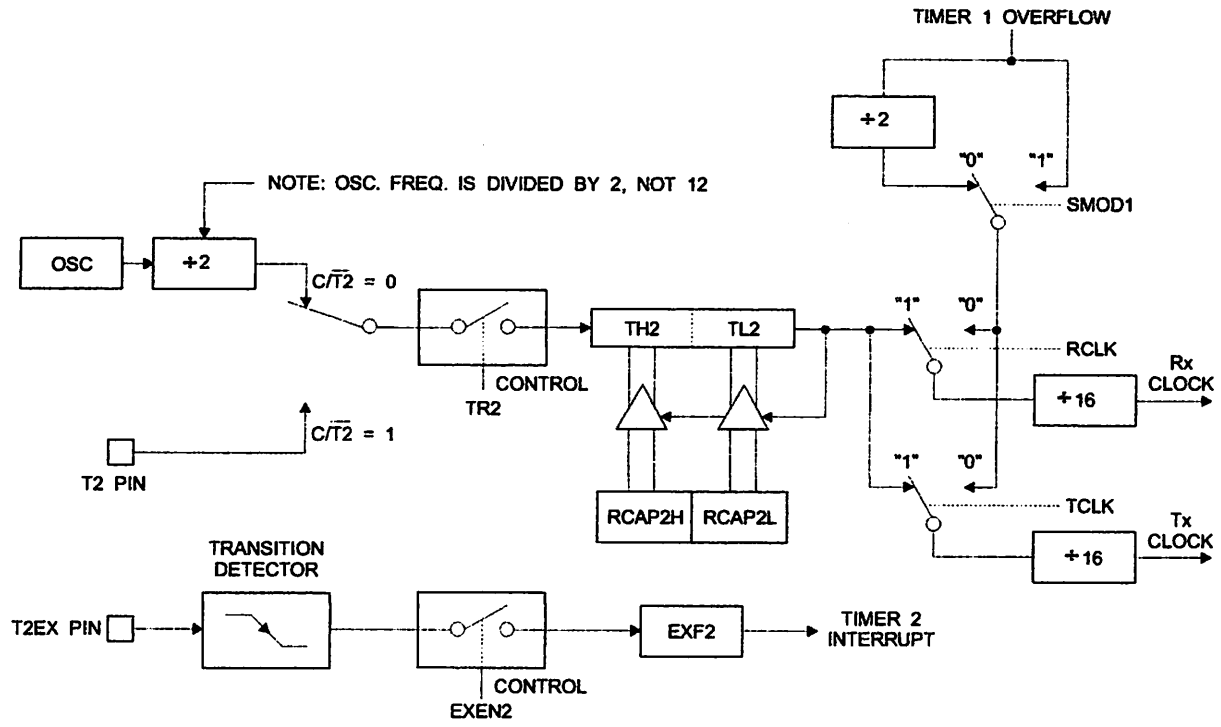
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536\text{-RCAP2H,RCAP2L})]}$$

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

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

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

Figure 4. Timer 2 in Baud Rate Generator Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz (for a 16-MHz operating frequency).

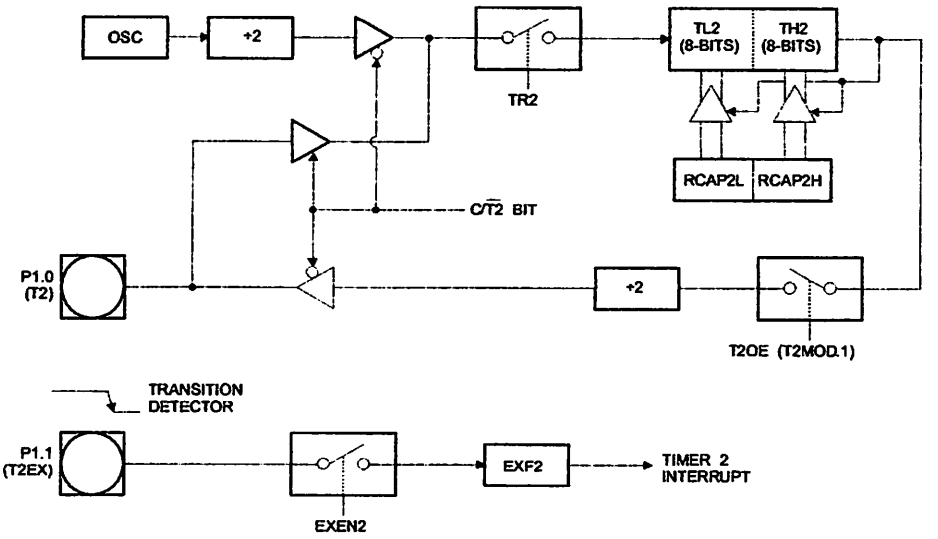
To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536-(\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

Figure 5. Timer 2 in Clock-Out Mode



Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 6.

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 5 shows that bit position IE.6 is unimplemented. User software should not write a 1 to this bit position, since it may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Table 7. Interrupt Enable (IE) Register

(MSB) (LSB)

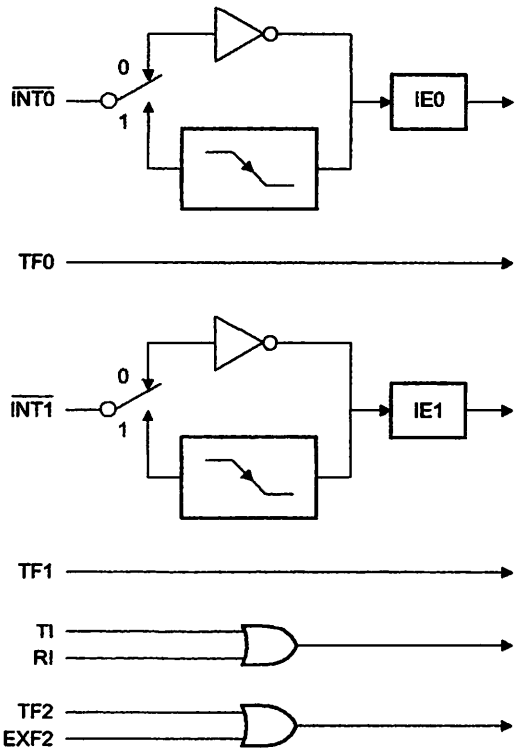
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	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 6. Interrupt Sources



Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 7. 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 8. 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.

Idle Mode

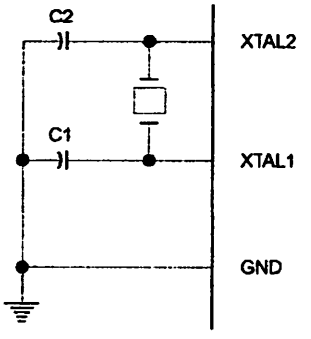
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

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

Power-down Mode

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

Figure 7. Oscillator Connections



Note: 1. C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 8. External Clock Drive Configuration

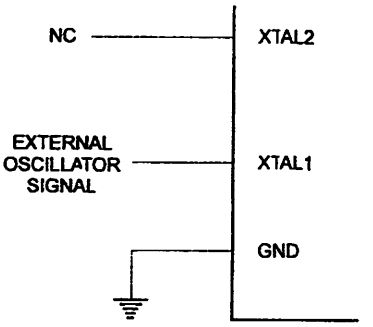


Table 8. Status of External Pins During Idle and Power-down Modes

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

Program Memory Lock Bits

The AT89S52 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 9. Lock Bit Protection Modes

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

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

Programming the Flash – Parallel Mode

The AT89S52 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 AT89S52 code memory array is programmed byte-by-byte.

Programming Algorithm: Before programming the AT89S52, 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 AT89S52, 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 AT89S52 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/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.0 is pulled low after ALE goes high during programming to indicate $\overline{BUSY}$. P3.0 is pulled high again when programming is done to indicate READY.

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

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

(000H) = 1EH indicates manufactured by Atmel
(100H) = 52H indicates AT89S52
(200H) = 06H

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

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

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

Programming the Flash – Serial Mode

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

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

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

Serial Programming Algorithm

To program and verify the AT89S52 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 which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal 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.

Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 11.

Programming Interface – Parallel Mode

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

Most worldwide major programming vendors offer support for the Atmel AT89 microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Table 10. Flash Programming Modes

Mode	V_{CC}	RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.4-0 Address	P1.7-0 Address
Write Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A12-8	A7-0
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A12-8	A7-0
Write Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
Write Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
Write Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	X 0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	52H	X 0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	06H	X 0010	00H

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

Figure 9. Programming the Flash Memory (Parallel Mode)

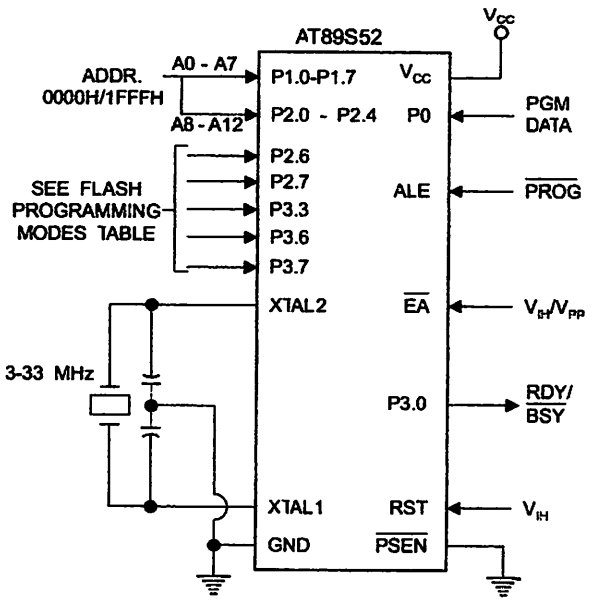
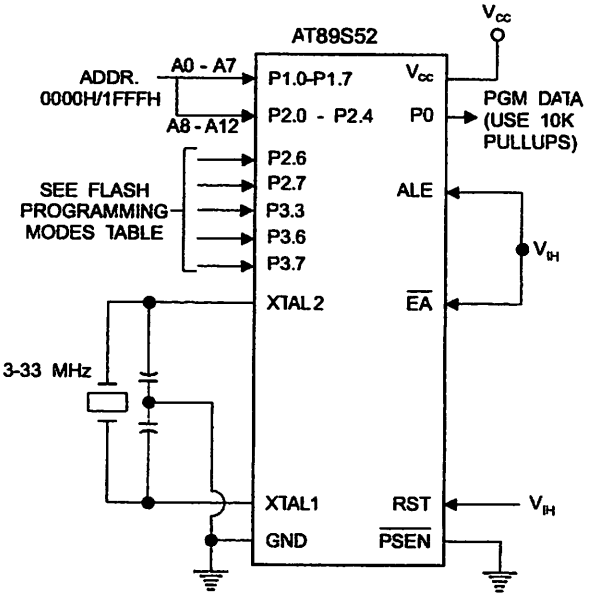


Figure 10. Verifying the Flash Memory (Parallel Mode)



Flash Programming and Verification Characteristics (Parallel Mode)

T_A = 20°C to 30°C, V_{CC} = 4.5 to 5.5V

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

Figure 11. Flash Programming and Verification Waveforms – Parallel Mode

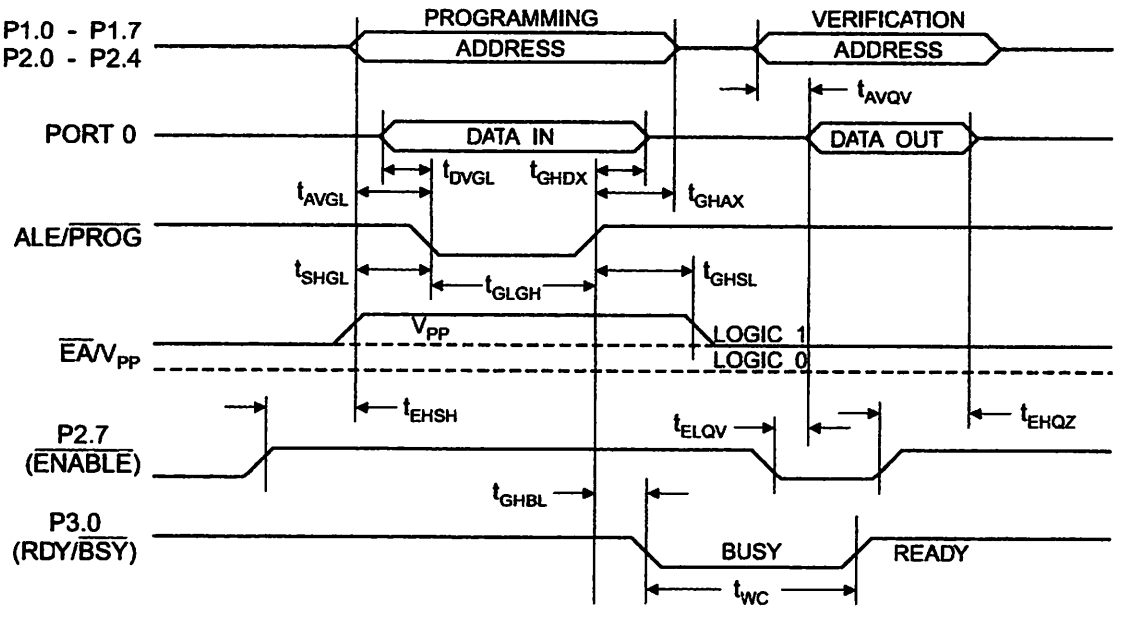
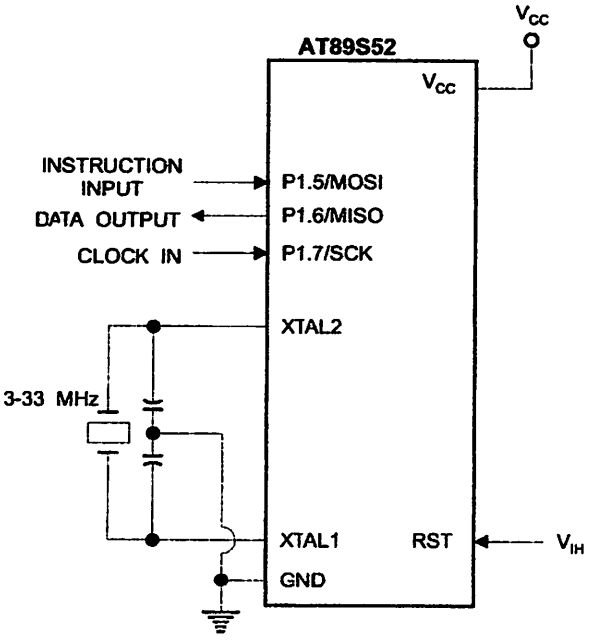


Figure 12. Flash Memory Serial Downloading



Flash Programming and Verification Waveforms – Serial Mode

Figure 13. Serial Programming Waveforms

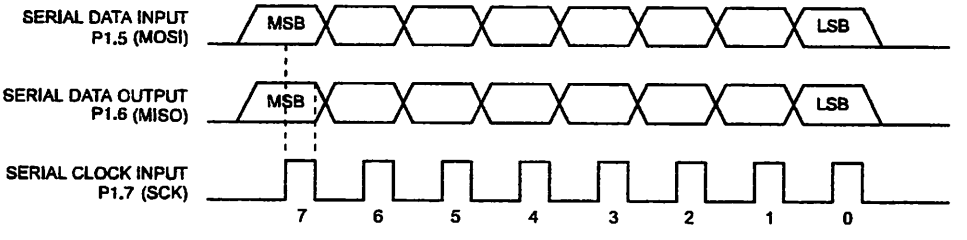


Table 11. Serial Programming Instruction Set

Instruction	Instruction Format Byte 1	Byte 2	Byte 3	Byte 4	Operation
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output on MISO)	Enable Serial Programming while RST is high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
Read Program Memory (Byte Mode)	0010 0000	xxx A12 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
Write Program Memory (Byte Mode)	0100 0000	xxx A12 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
Write Lock Bits[1]	1010 1100	1110 00 B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (1).
Read Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xxx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a "1")
Read Signature Bytes	0010 1000	xxx A12 A11 A10 A9 A8	A7 xxx xxx0	Signature Byte	Read Signature Byte
Read Program Memory (Page Mode)	0011 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Write Program Memory (Page Mode)	0101 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

Note: 1. 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, B2 = 1 —> Mode 4, lock bit 3 activated

} Each of the lock bit modes 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 14. Serial Programming Timing

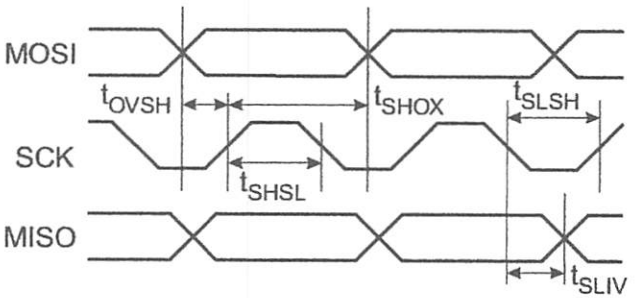


Table 12. Serial Programming Characteristics, T_A = -40° C to 85° C, V_{CC} = 4.0 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	3		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

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC 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.

DC Characteristics

The 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.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
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
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
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ±10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode[1]	V_{CC} = 5.5V		50	µA

Notes:
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.

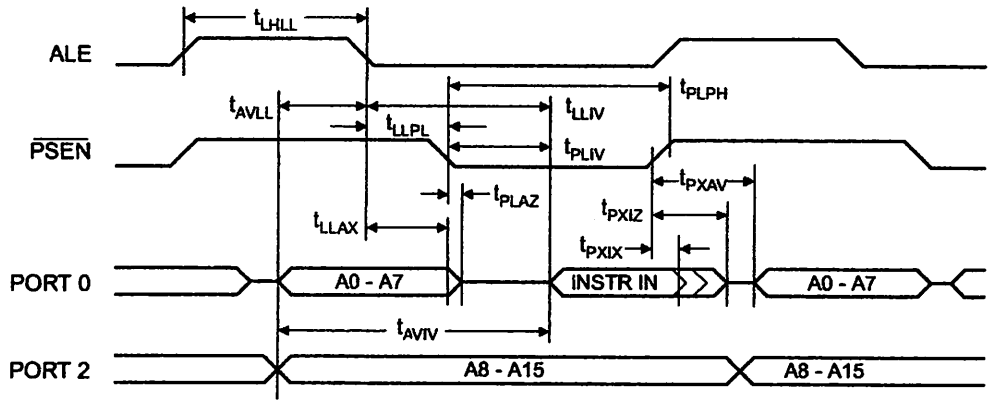
AC Characteristics

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

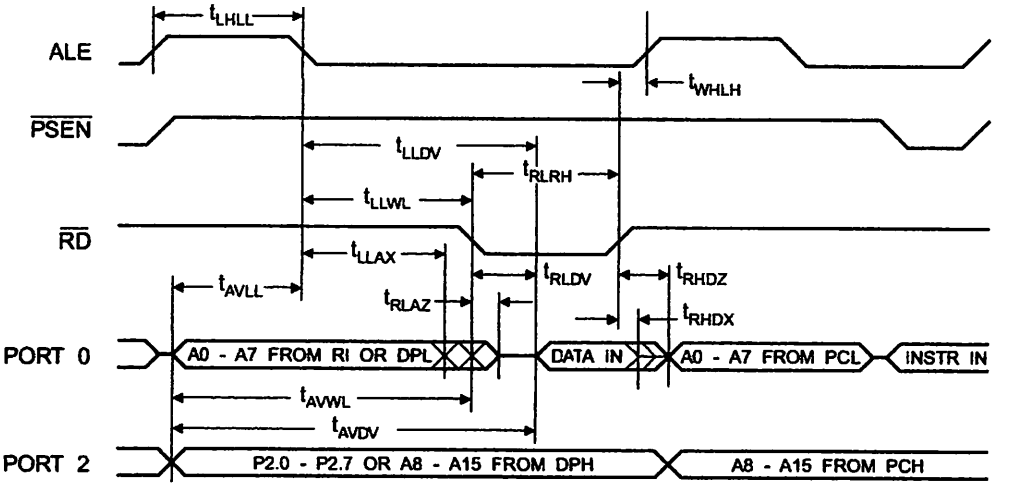
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	33	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
t_{AVLL}	Address Valid to ALE Low	43		t_{CLCL}-25		ns
t_{LLAX}	Address Hold After ALE Low	48		t_{CLCL}-25		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-25		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-45		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-60	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-25	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}$-80	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-30		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-130		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		t_{CLCL}-25		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-25	t_{CLCL}+25	ns

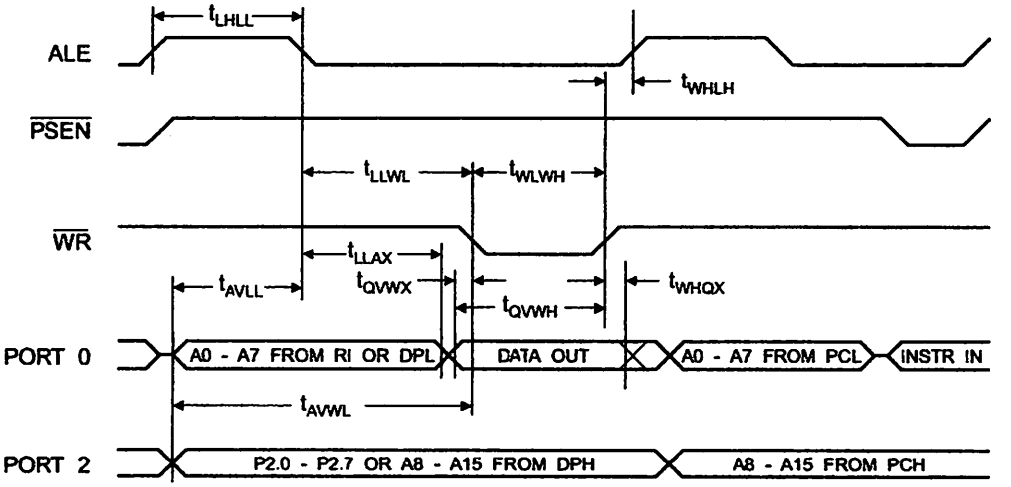
External Program Memory Read Cycle



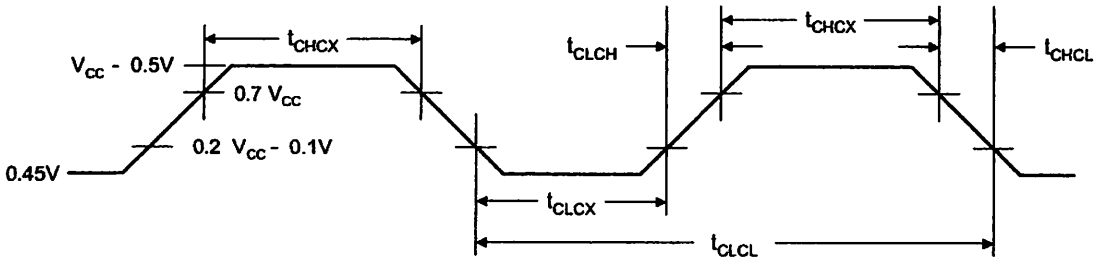
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

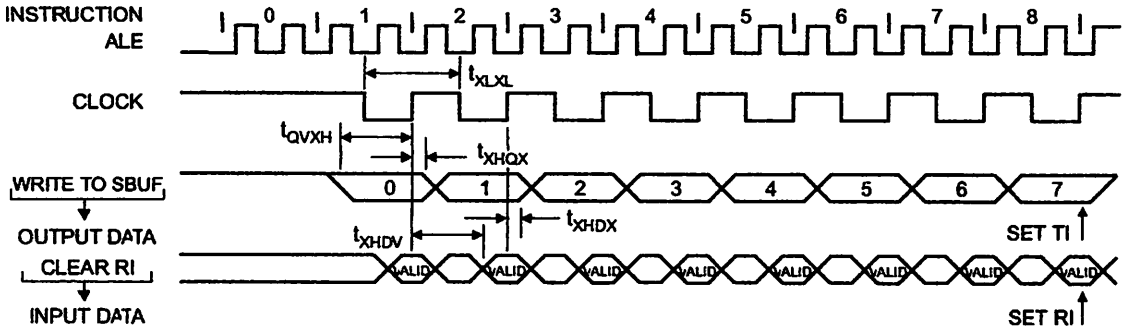
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	33	MHz
t_{CLCL}	Clock Period	30		ns
t_{CHCX}	High Time	12		ns
t_{CLCX}	Low Time	12		ns
t_{CLCH}	Rise Time		5	ns
t_{CHCL}	Fall Time		5	ns

Serial Port Timing: Shift Register Mode Test Conditions

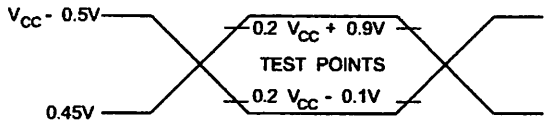
The values in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-80		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

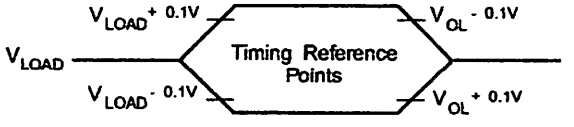


AC Testing Input/Output Waveforms(1)



Note: 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.

Float Waveforms(1)



Note: 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.

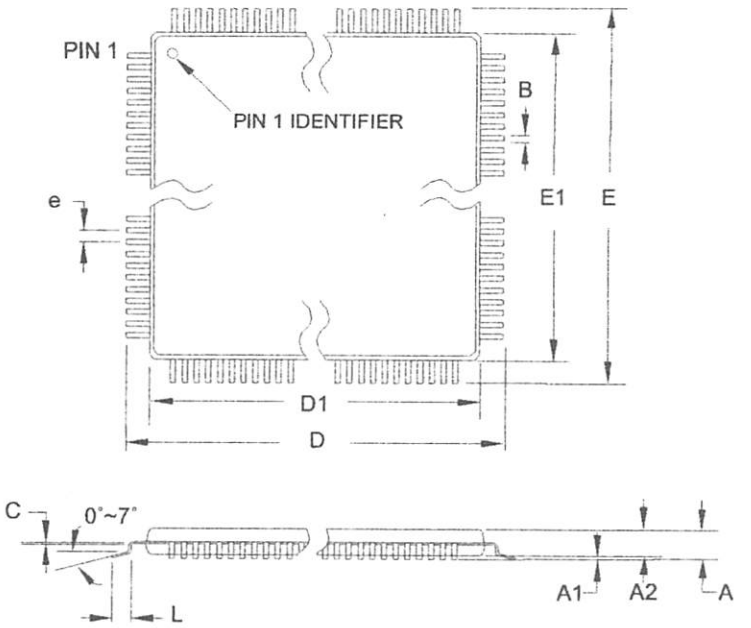
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S52-24AC AT89S52-24JC AT89S52-24PC AT89S52-24SC	44A 44J 40P6 42PS6	Commercial (0° C to 70° C)
		AT89S52-24AI AT89S52-24JI AT89S52-24PI AT89S52-24SI	44A 44J 40P6 42PS6	Industrial (-40° C to 85° C)
33	4.5V to 5.5V	AT89S52-33AC AT89S52-33JC AT89S52-33PC AT89S52-33SC	44A 44J 40P6 42PS6	Commercial (0° C to 70° C)

Package Type	
44A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
42PS6	42-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

Packaging Information

44A – TQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

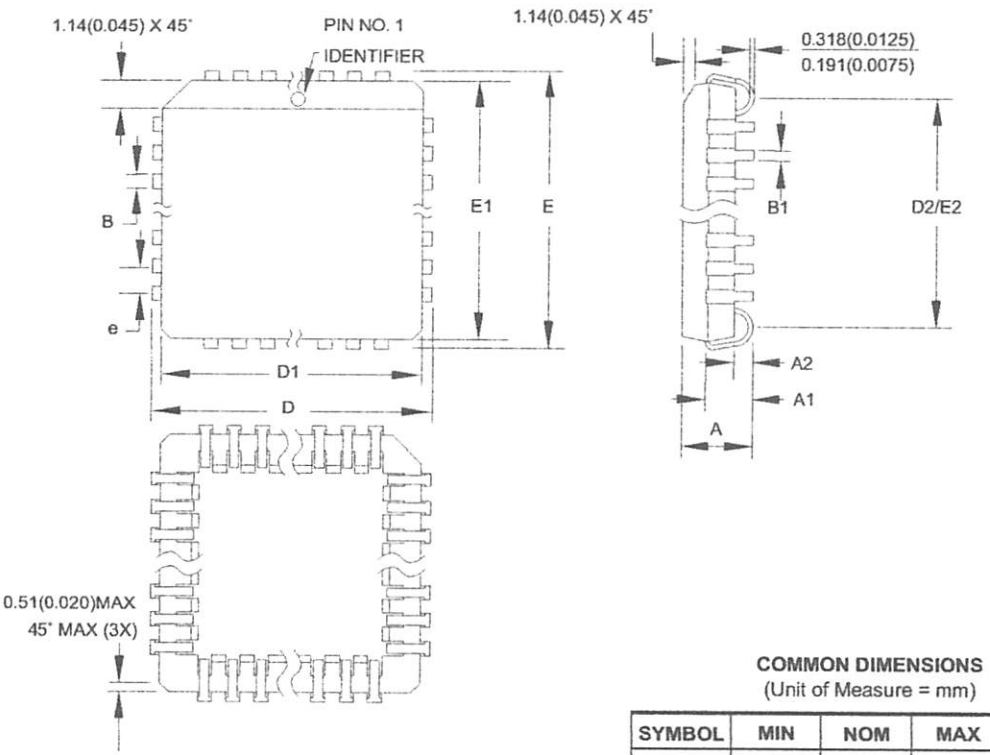
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

44J – PLCC



COMMON DIMENSIONS
(Unit of Measure = mm)

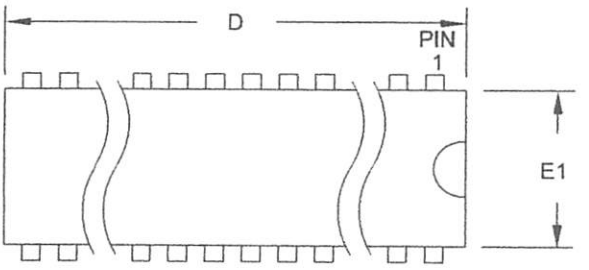
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

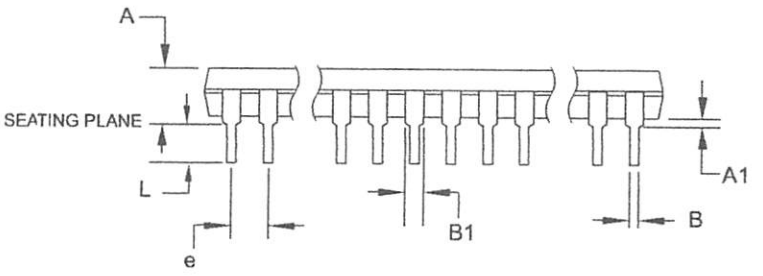
Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

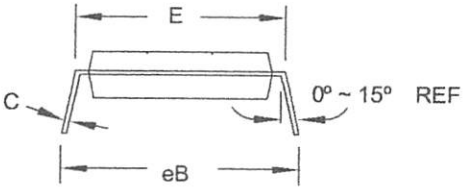
10/04/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B

40P6 – PDIP







COMMON DIMENSIONS
(Unit of Measure = mm)

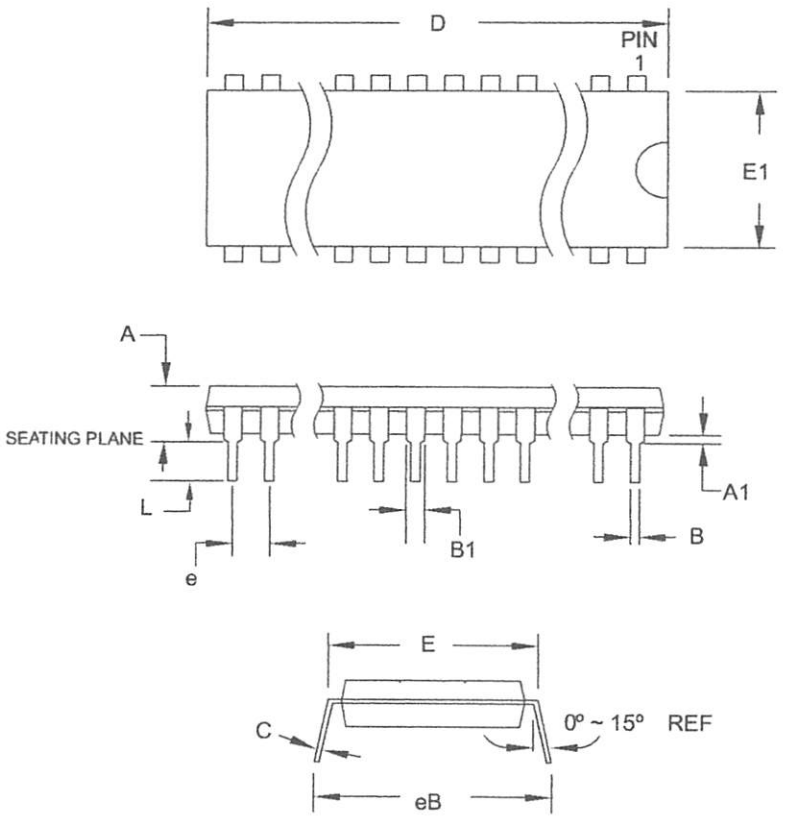
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

42PS6 – PDIP



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.83	
A1	0.51	–	–	
D	36.70	–	36.96	Note 2
E	15.24	–	15.88	
E1	13.46	–	13.97	Note 2
B	0.38	–	0.56	
B1	0.76	–	1.27	
L	3.05	–	3.43	
C	0.20	–	0.30	
eB	–	–	18.55	
e	1.78 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

11/6/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**42PS6**, 42-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	42PS6	A

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

1919B–MICRO–11/03

SPECIFICATION

MODULE NO	NC1602B-serise
VERSION	VER.0
CUSTOMER	
APPROVE	

Sale by	Check by	Prepare by

科創光電股份有限公司

台中市北屯區路 360 號 9 樓之一

NewTec Display Co., Ltd.

9F-1, No360, Beitun Rd., Taichung City 406, Taiwan (R.O.C.)

TEL: 886-4-22474858 FAX: 886-4-22474859

http://www.newtec.com.tw

INDEX

ISSUE RECORD

Module NO.		
DATE	Version	Description
05/03/11	0	Published

1.Numbering system

N	C	1602	B	–	G	H	Y	–	RS	
①	②	③	④		⑤	⑥	⑦		⑧	⑨

1 .Brand Name

N	NEWTEC Display Co., LTD

2. Display Type

T	TAB
B	Graphic
C	Character
O	COG
P	PLED
R	Color-STN
S	Seven-Segment
F	TFT

3. Number of Pixels

Character Module	Characters per line ×Lines
Graphic Module	Row Dots ×Column Dots

4. Series number

A~Z	Series Number

5. LCD Mode:

	TN	STN		FSTN	Color-STN	TFT
Positive	T	G	Gary	F	R	T (Black)
		Y	Yellow/Green			
Negative	N	B	Blue	M		

6. LCD Polarize

	Normal Temperature		Wide Temperature	
	6:00	12:00	6:00	12:00
Reflective	A	D	G	J
Transflective	B	E	H	K
Transmissive	C	F	I	L

7. Backlight

None	N	None
EL	H	White
	U	Blue Green
LED	A	Amber
	B	Blue
	E	Yellow/Green, edge
	G	Green
	R	Red
	W	White
	Y	Yellow/Green
CCFL	C	White

8. IC font (Character)

Cyrillic/English	TS
Chinese/English	C(BIG 5), S(GB)
Japanese/English	PN,PS,PM
European/English	RN,RS,RK

9. Special code

A	Anti-glare
H	Touch panel
M	Negative voltage output and temperature compensation on board
N	With negative voltage output on board
X	Without negative voltage output on board

10. Others

2. Precaution in use of LCD Module

(1) Avoid applying excessive shocks to the module or making any alterations or modifications to it.
(2) Don't make extra holes on the printed circuit board, modify its shape or change the components of LCD module.
(3) Don't disassemble the LCM.
(4) Don't operate it above the absolute maximum rating.
(5) Don't drop, bend or twist LCM.
(6) Soldering: only to the I/O terminals.
(7) Storage: please storage in anti-static electricity container and clean environment.
(8) Don't touch the elastomer connecter, especially insert a backlight panel (EL or CCFL)

3. General Specification

3.1 Mechanical Dimension

Item	Dimension	Unit
Number of Characters	16 characters x 2 Lines	–
Module dimension (L x W x H)	80.0 x 36.0 x 9.4 (Max)-EL B/L or NO B/L 80.0 x 36.0 x 13.2 (Max)- LED B/L	mm
View area	66.0 x 16.0	mm
Active area	56.21 x 11.5	mm
Dot size	0.56 x 0.66	mm
Dot pitch	0.6x 0.7	mm
Character size (L x W)	2.96 x 5.46	mm
Character pitch (L x W)	3.55 x 5.94	mm

3.2 Controller IC: KS0066 (or Equivalent) controller

3.3 Temperature Range

	Normal temperature	Wide temperature
Operation temperature	0℃ ~ +50℃	-20℃ ~ +70℃
Storage temperature	-10℃ ~ + 60℃	-30℃ ~ + 80℃

4. Absolute Maximum Ratings

4.1 Electrical Absolute Maximum Ratings

(Vss=0V, Ta=25℃)

Item	Symbol	Min	Max	Unit
Supply Voltage (Logic)	Vdd- Vss	-0.3	7.0	V
Supply Voltage (LCD driver)	Vdd-Vo	-0.3	13	V
Input Voltage	VI	Vss	Vdd	V
Normal Type	TOP	0	+50	℃
	TSTG	-10	+60	℃
Wide Temperature Type	Top	-20	+70	℃
	Tstg	-30	+80	℃

4.2 Environmental Absolute Maximum Ratings

Item	Operating		Storage		Comment
	(Min.)	(Max.)	(Min.)	(Max.)	
Humidity	Note(2)		Note(2)		Without condensation
Vibration	--	4.9M/S^2	--	19.6M/S^2	XYZ Direction
Shock	--	29.4M/S^2	--	490M/S^2	XYZ Direction

Note (1) Ta = 0℃ : 50Hr Max.

Note (2) Ta ≦40℃ : 90% RH MAX

Ta > 40℃ : Absolute humidity must be lower than the humidity of 90% at 40℃.

5. Electrical Characteristics

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Supply Voltage For Logic	Vdd-Vss	—	2.8	5.0	5.5	V
Supply Voltage For LCD *Wide Temp、Type	Vdd-Vo	—			4.5	-20℃
				4.0		25℃
			3.5			70℃
Input High Vol.	V_{IH}	—	2.2	—	Vdd	V
Input Low Vol.	V_{IL}	—	—	—	0.6	V
Output High Vol.	V_{OH}	—	2.4	—	—	V
Output Low Vol.	V_{OL}	—	—	—	0.4	V
Supply Current(Logic)	Idd	Vdd=5V	—	1.2	—	mA

6. Optical Characteristics

- STN

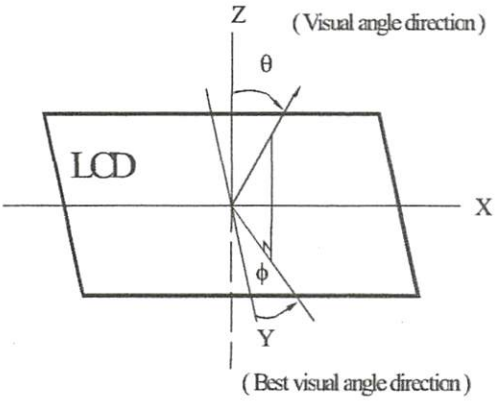
Item	Symbol	Condition	Min.	Typ.	Max.	Unit
View Angle	(V) θ	CR≧2	10		45	deg
	(H) ϕ	CR≧2	-30		30	deg
Contrast Ratio	CR	—		3		—
Response Time 25℃	T rise	—		100	150	ms
	T fall	—		150	200	Ms

- FSTN

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
View Angle	(V) θ	CR≧2	10		50	deg
	(H) ϕ	CR≧2	-45		45	deg
Contrast Ratio	CR	—		5		—
Response Time 25℃	T rise	—		100	150	ms
	T fall	—		150	200	ms

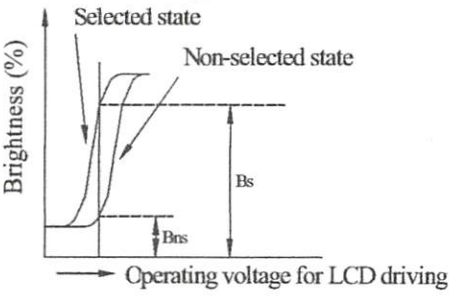
6.1 Definitions

■View Angles

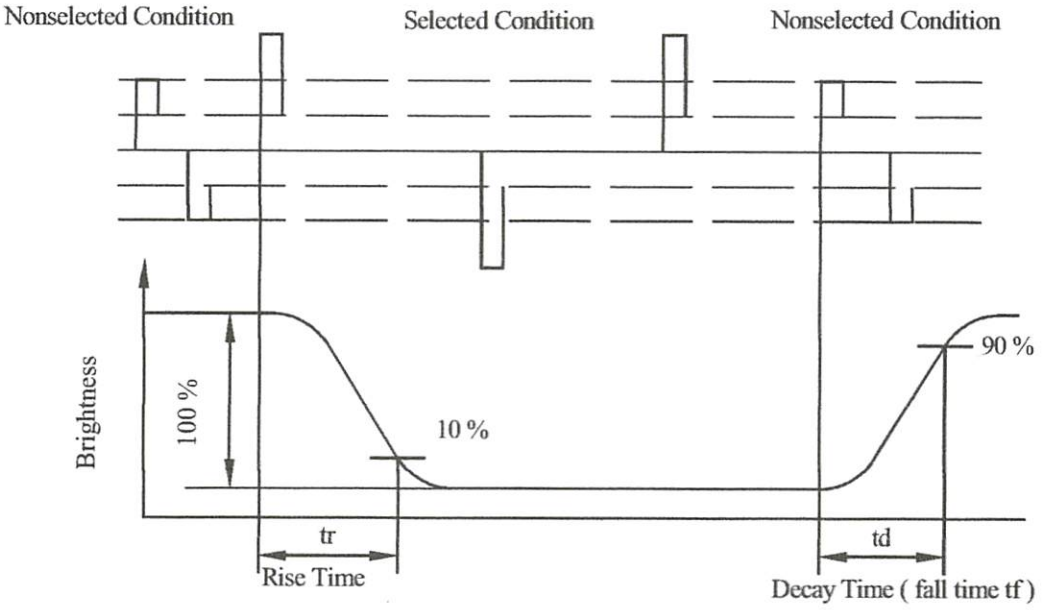


■Contrast Ratio

$$CR = \frac{\text{Brightness at selected state (BS)}}{\text{Brightness at non-selected state (Bns)}}$$



■Response Time

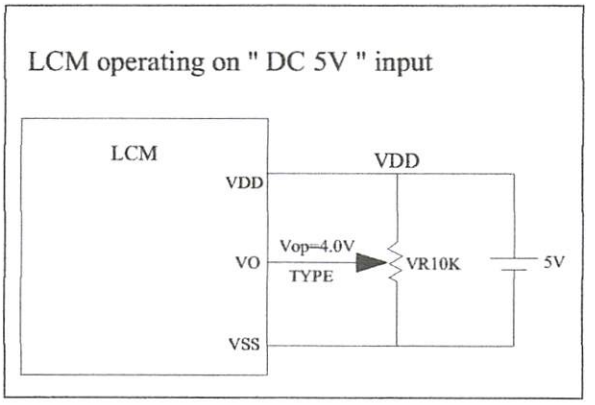


7. Interface Pin Function

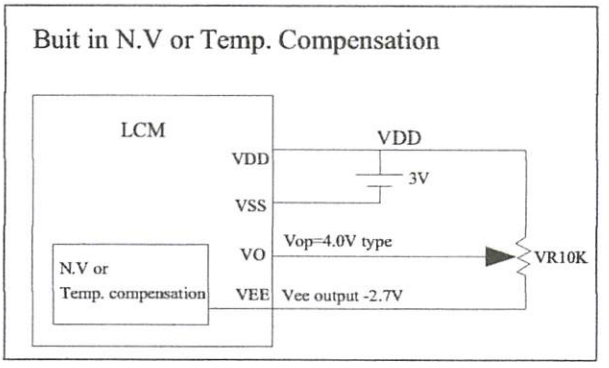
Pin No.	Symbol	Level	Description
1	Vss	0V	Supply Voltage for logic Ground
2	Vdd	5.0V	Supply Voltage for logic and LED backlight
3	Vo	(Variable)	Operating voltage for LCD
4	RS	H/L	H:DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) ; L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A/Vee		Power supply for backlight V+/Negative Voltage Output
16	K		Power supply for backlight V-

8. Power Supply for LCD Module and LCD Operating Voltage Adjustment

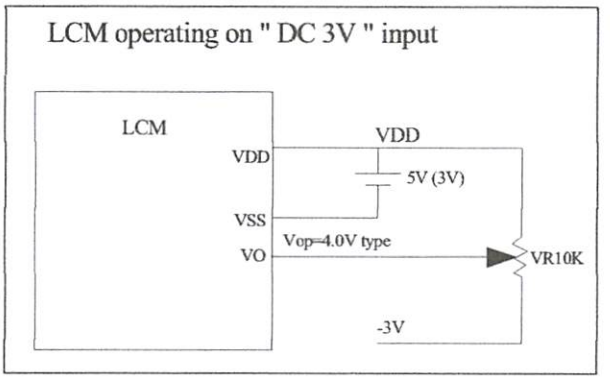
***Stander Type**



* (Option) LCM operating on " DC 3V " input , with Built in negative voltage



* (Option) LCM operating on " DC 3V " input , with external negative voltage



9. Backlight Information

9. 1 Specification

- LED array yellow-green

Parameter	Symbol	Min	Typical	Max	Unit	Test Condition
Supply Current	ILED	–	100	–	mA	V_{LED}=4.2V
Supply Voltage	V	4.0	4.2	4.3	V	–
Reverse Voltage	VR	–	–	8	V	–
Luminous Intensity	IV	100	–	–	cd/m^2	I_{LED}=100mA
Wave Length	λ p	–	575	–	nm	I_{LED}=100mA
Life Time	–	–	100,000	–	Hr.	V≦4.3V
Color	Yellow green					

- LED edge yellow-green

Parameter	Symbol	Min	Typical	Max	Unit	Test Condition
Supply Current	ILED	–	20	–	mA	V_{LED}=4.2V
Supply Voltage	V	4.0	4.2	4.3	V	–
Reverse Voltage	VR	–	–	8	V	–
Luminous Intensity	IV	6	–	–	cd/m^2	I_{LED}=20mA
Wave Length	λ p	–	575	–	nm	I_{LED}=20mA
Life Time	–	–	100,000	–	Hr.	V≦4.3V
Color	Yellow green					

- LED edge white

Parameter	Symbol	Min	Typical	Max	Unit	Test Condition
Supply Current	ILED	–	20	–	mA	V_{LED}=3.2V

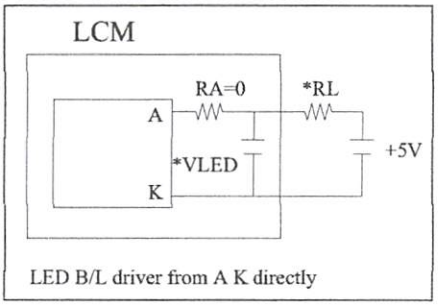
Supply Voltage	V		3.2	3.4	V	—
Reverse Voltage	VR	—	—	5	V	—
Luminous Intensity	IV	80	—	—	cd/m^2	ILED=20mA
Chromaticity	X	—	0.30	—		ILED=20mA
	Y		0.31			
Life Time	—	—	70,000	—	Hr.	V≦3.2V
Color	white					

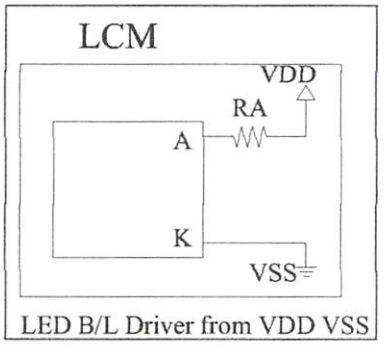
- EL white / blue

Parameter	Symbol	Min	Typ	Max	Unit	Test Condition
Voltage	Vrms	--	110 (AC)		--	
Frequency	HZ	--	400		--	
Brightness*	cd/m^2	48	60		--	110Vrms 400Hz
CIE Chromaticity Diagram	X	--	0.3019(white)		--	
			0.330 (blue)			
	Y	--	0.3929(white)		--	
			0.365 (blue)			
Current Dissipation	mA/cm^2	--	3.63		--	
Power Dissipation	mW/cm^2	--	71.71		--	
Color	Blue , white					

9.2 Backlight driving methods

a. LED B/L drive methods



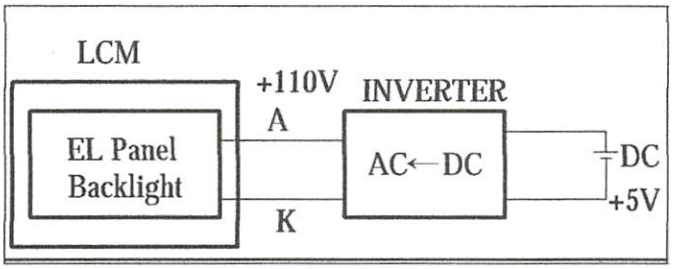


* 1.array (yellow green) LED B/L driver : **VLED=4.2V RL=8Ω**

2. edge (yellow green) LED B/L driver : **VLED=4.2V RL=40Ω**

3. edge (white/blue) LED B/L driver : **VLED=3.2V RL=90Ω**

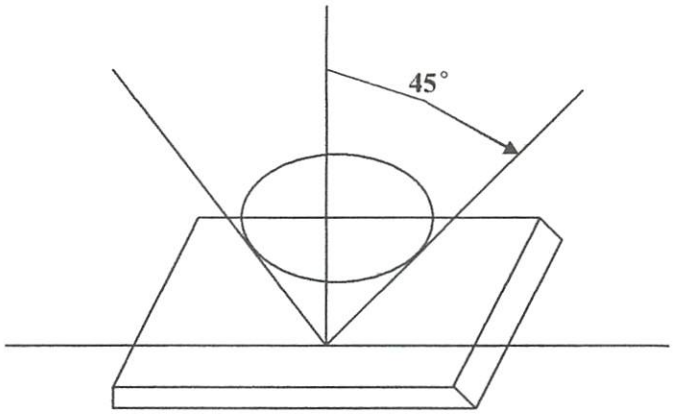
b. E/L B/L driven from A.K cable directly



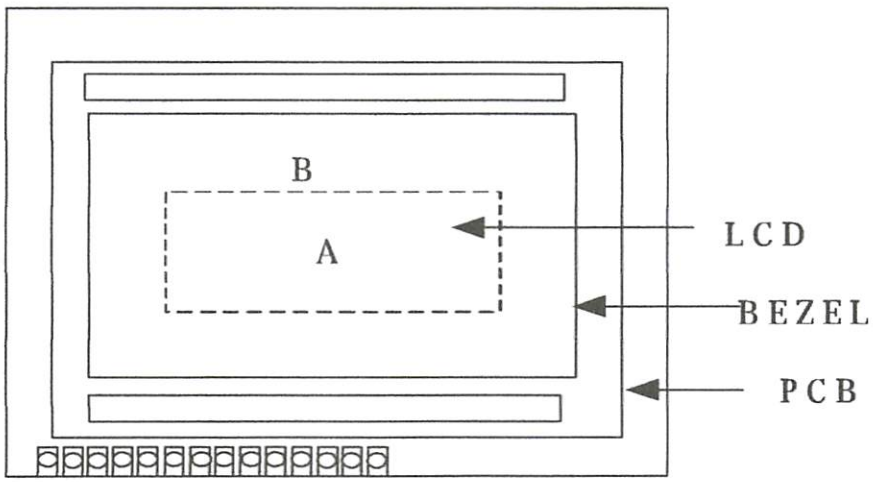
10. Quality Assurance

10.1 Inspection conditions

The LCD shall be inspected under 40W white fluorescent light.



Definition of applicable Zones



A : Display Area

B : Non-Display Area

10.2 Inspection Parameters

<table>
<tr><th>NO.</th><th>Parameter</th><th>Criteria</th></tr>
<tr><td>1</td><td>Black or White spots</td><td>
<table>
<tr><th>Zone / Dimension</th><th colspan="2">Acceptable Number</th><th>Class Of Defects</th><th>Acceptable Level</th></tr>
<tr><td></td><td>A</td><td>B</td><td></td><td></td></tr>
<tr><td>D＜0.15</td><td>*</td><td>*</td><td>Minor</td><td>2.5</td></tr>
<tr><td>0.15≦D≦0.2</td><td>4</td><td>4</td><td></td><td></td></tr>
<tr><td>0.2≦D≦0.25</td><td>2</td><td>2</td><td></td><td></td></tr>
<tr><td>D≦0.3</td><td>0</td><td>1</td><td></td><td></td></tr>
</table>
D=(Long + Short)/2 *: Disregard
</td></tr>
<tr><td>2</td><td>Scratch, Substances</td><td>
<table>
<tr><th colspan="2">Zone / X(mm) Y(mm)</th><th colspan="2">Acceptable Number</th><th>Class Of Defects</th><th>Acceptable Level</th></tr>
<tr><td></td><td></td><td>A</td><td>B</td><td></td><td></td></tr>
<tr><td>*</td><td>0.04≧ W</td><td>*</td><td>*</td><td>Minor</td><td>2.5</td></tr>
<tr><td>3.0≧ L</td><td>0.06≧ W</td><td>4</td><td>4</td><td></td><td></td></tr>
<tr><td>2.0≧ L</td><td>0.08≧ W</td><td>2</td><td>3</td><td></td><td></td></tr>
<tr><td>—</td><td>0.1＜W</td><td>0</td><td>1</td><td></td><td></td></tr>
</table>
X: Length Y: Width *: Disregard
Total defects should not exceed 4/module
</td></tr>
<tr><td>3</td><td>Air Bubbles
(between glass & polarizer)</td><td>
<table>
<tr><th>Zone / Dimension</th><th colspan="2">Acceptable Number</th><th>Class Of Defects</th><th>Acceptable Level</th></tr>
<tr><td></td><td>A</td><td>B</td><td></td><td></td></tr>
<tr><td>D≦0.15</td><td>*</td><td>*</td><td>Minor</td><td>2.5</td></tr>
<tr><td>0.15＜D≦0.25</td><td>2</td><td>*</td><td></td><td></td></tr>
<tr><td>0.25＜D</td><td>0</td><td>1</td><td></td><td></td></tr>
</table>
*: Disregard
Total defects shall not excess 3/module.
</td></tr>
</table>

11. Reliability

11.1 Content of Reliability Test

Environmental Test				
No.	Test Item	Content of Test	Test Condition	Applicable Standard
1	High Temperature storage	Endurance test applying the high storage temperature for a long time.	60℃ / 200hrs	——
2	Low Temperature storage	Endurance test applying the high storage temperature for a long time.	-20℃ 200hrs	——
3	High Temperature Operation	Endurance test applying the electric stress (Voltage & Current) and the thermal stress to the element for a long time.	50℃ 200hrs	——
4	Low Temperature Operation	Endurance test applying the electric stress under low temperature for a long time.	0℃ 200hrs	——
5	High Temperature/ Humidity Storage	Endurance test applying the high temperature and high humidity storage for a long time.	60℃,90%RH 96hrs	——
6	High Temperature/ Humidity Operation	Endurance test applying the electric stress (Voltage & Current) and temperature / humidity stress to the element for a long time.	40℃,90%RH 96hrs	——
7	Temperature Cycle	Endurance test applying the low and high temperature cycle. -20℃ 25℃ 60℃ 30min 5min 30min 1 cycle	-20℃/60℃ 10 cycles	——
Mechanical Test				
8	Vibration test	Endurance test applying the vibration during transportation and using.	10~22Hz→1.5mmp-p 22~500Hz→1.5G Total 0.5hrs	——
9	Shock test	Constructional and mechanical endurance test applying the shock during transportation.	50G Half sign wave 11 msedc 3 times of each direction	——
10	Atmospheric pressure test	Endurance test applying the atmospheric pressure during transportation by air.	115mbar 40hrs	——
Others				
11	Static electricity test	Endurance test applying the electric stress to the terminal.	VS=800V,RS=1.5kΩ CS=100pF 1 time	——

***Supply voltage for logic system=5V. Supply voltage for LCD system =Operating voltage at 25℃

12. Controller data

12.1 Function description

The LCD display Module is built in a LSI controller, the controller has two 8-bit registers, an instruction register (IR) and a data register (DR).

The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator (CGRAM). The IR can only be written from the MPU. The DR temporarily stores data to be written or read from DDRAM or CGRAM. When address information is written into the IR, then data is stored into the DR from DDRAM or CGRAM. By the register selector (RS) signal, these two registers can be selected.

RS	R/W	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB7)
1	0	Write data to DDRAM or CGRAM (DR to DDRAM or CGRAM)
1	1	Read data from DDRAM or CGRAM (DDRAM or CGRAM to DR)

Busy Flag (BF)

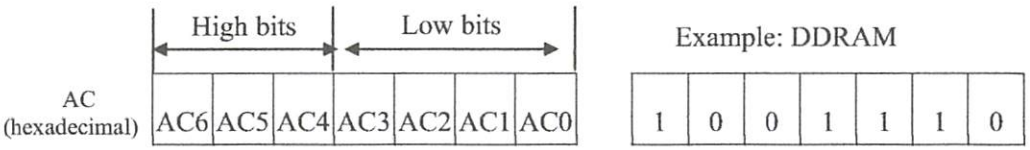
When the busy flag is 1, the controller LSI is in the internal operation mode and the next instruction will not be accepted. When RS=0 and R/W=1, the busy flag is output to DB7. The next instruction must be written after ensuring that the busy flag is 0.

Address Counter (AC)

The address counter (AC) assigns addresses to both DDRAM and CGRAM

Display Data RAM (DDRAM)

This DDRAM is used to store the display data represented in 8-bit character codes. Its extended capacity is 80×8 bits or 80 characters. Below figure is the relationship between DDRAM addresses and positions on the liquid crystal display.



DDRAM Address

Display position DDRAM address

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06									0F
40	41	42	43	44	45	46									4F

Example:2-Line by16-Character Display

Character Generator ROM (CGROM)

The CGROM generate 5×8 dot or 5×10 dot character patterns from 8-bit character codes. See Table 2.

12.2 Character Generator RAM (CGRAM)

In CGRAM, the user can rewrite character by program. For 5×8 dots, eight character patterns can be written, and for 5×10 dots, four character patterns can be written.

Write into DDRAM the character code at the addresses shown as the left column of table 1. To show the character patterns stored in CGRAM.

Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character Patterns (CGRAM Data)

Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character Patterns (CGRAM Data)

For 5 * 8 dot character patterns

Character Codes (DDRAM data)	CGRAM Address	Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0	5 4 3 2 1 0	7 6 5 4 3 2 1 0	
High Low	High Low	High Low	
0 0 0 0 * 0 0 0	0 0 0 0 0 0	* * * ■■■■ 0	Character pattern(1)
	0 0 1	* * * ■ 0 0 0 ■	
	0 1 0	* * * ■ 0 0 0 ■	
	0 1 1	* * * ■■■■ 0	
	1 0 0	* * * ■ 0 ■ 0 0	
	1 0 1	* * * ■ 0 0 ■ 0	
	1 1 0	* * * ■ 0 0 0 ■	
	1 1 1	* * * 0 0 0 0 0	Cursor pattern
0 0 0 0 * 0 0 1	0 0 1 0 0 0	* * * ■ 0 0 0 ■	Character pattern(2)
	0 0 1	* * * 0 ■ 0 ■ 0	
	0 1 0	* * * ■■■■■	
	0 1 1	* * * 0 0 ■ 0 0	
	1 0 0	* * * ■■■■■	
	1 0 1	* * * 0 0 ■ 0 0	
	1 1 0	* * * 0 0 ■ 0 0	
	1 1 1	* * * 0 0 0 0 0	Cursor pattern
	0 0 0	* * *	
	0 0 1		
0 0 0 0 * 1 1 1	1 1 1 1 0 0		
	1 0 1		
	1 1 0		
	1 1 1	* * *	

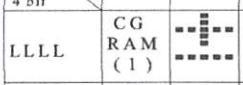
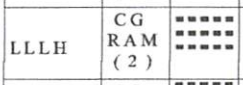
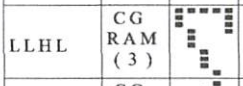
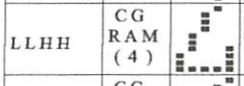
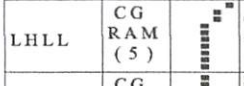
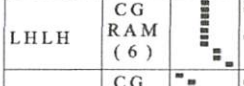
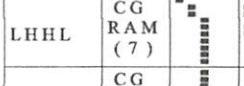
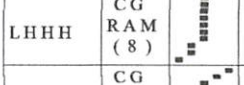
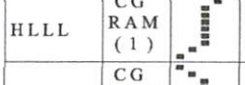
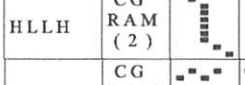
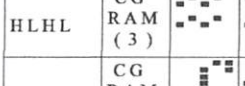
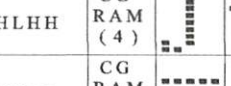
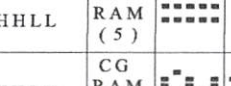
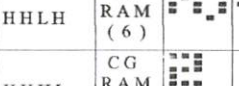
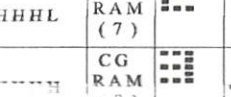
For 5 * 10 dot character patterns

Character Codes (DDRAM data)	CGRAM Address	Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0	5 4 3 2 1 0	7 6 5 4 3 2 1 0	
High Low	High Low	High Low	
0 0 0 0 * 0 0 0	0 0 0 0 0 0	* * * 0 0 0 0 0	Character pattern
	0 0 0 1	* * * 0 0 0 0 0	
	0 0 1 0	* * * ■ 0 ■■■ 0	
	0 0 1 1	* * * ■■■ 0 0 ■	
	0 1 0 0	* * * ■ 0 0 0 ■	
	0 1 0 1	* * * ■ 0 0 0 ■	
	0 1 1 0	* * * ■■■■■ 0	
	0 1 1 1	* * * ■ 0 0 0 0	
	1 0 0 0	* * * ■ 0 0 0 0	
	1 0 0 1	* * * ■ 0 0 0 0	
	1 0 1 0	* * * 0 0 0 0 0	Cursor pattern
	1 1 1 1	* * * * * * * *	

■ : " High "

12.3 C.G ROM table (table 2)

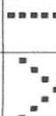
Code RS RN RK: English –European Font

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	CG RAM (2)															
LLHL	CG RAM (3)															
LLHH	CG RAM (4)															
LHLL	CG RAM (5)															
LHLH	CG RAM (6)															
LHHL	CG RAM (7)															
LHHH	CG RAM (8)															
HLLL	CG RAM (1)															
HLLH	CG RAM (2)															
HLHL	CG RAM (3)															
HLHH	CG RAM (4)															
HHLL	CG RAM (5)															
HHLH	CG RAM (6)															
HHHL	CG RAM (7)															
	CG RAM (8)															

C ode PS PN PM: English –Japanese Font

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	(2)															
LLHL	(3)															
LLHH	(4)															
LHLL	(5)															
LHLH	(6)															
LHHL	(7)															
LHHH	(8)															
HLLL	(1)															
HLLH	(2)															
HLHL	(3)															
HLHH	(4)															
HHLL	(5)															
HHLH	(6)															
HHHL	(7)															
HHHH	(8)															

Code TS: English –Cyrillic Font

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	CG RAM (2)															
LLHL	CG RAM (3)															
LLHH	CG RAM (4)															
LHLL	CG RAM (5)															
LHLH	CG RAM (6)															
LHHL	CG RAM (7)															
LHHH	CG RAM (8)															
HLLL	CG RAM (1)															
HLLH	CG RAM (2)															
HLHL	CG RAM (3)															
HLHH	CG RAM (4)															
HHLL	CG RAM (5)															
HHLH	CG RAM (6)															
HHHL	CG RAM (7)															
HHHH	CG RAM (8)															

Code MS MM: English –European Font

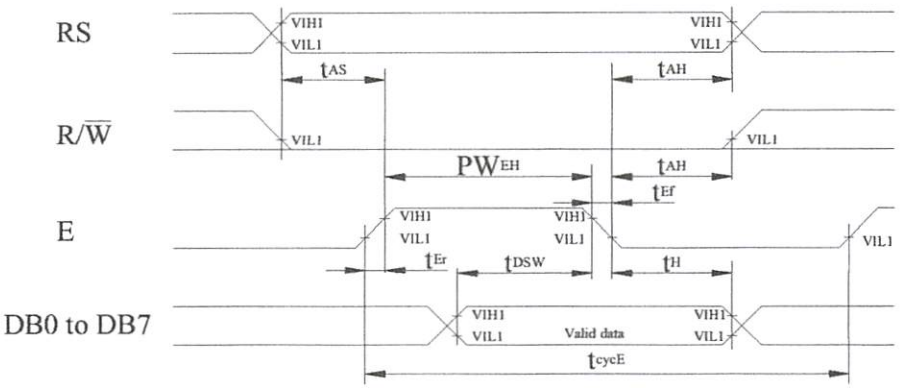
Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	CG RAM (2)															
LLHL	CG RAM (3)															
LLHH	CG RAM (4)															
LHLL	CG RAM (5)															
LHLH	CG RAM (6)															
LHHL	CG RAM (7)															
LHHH	CG RAM (8)															
HLLL	CG RAM (1)															
HLLH	CG RAM (2)															
HLHL	CG RAM (3)															
HLHH	CG RAM (4)															
HHLL	CG RAM (5)															
HHLH	CG RAM (6)															
HHHL	CG RAM (7)															
HHHH	CG RAM (8)															

12.4 Instruction table

Instruction	Instruction Code										Description	Execution time (fosc=270Khz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "00H" to DDRAM and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	—	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 μ s
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blinking of cursor (B) on/off control bit.	39 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	—	—	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 μ s
Function Set	0	0	0	0	1	DL	N	F	—	—	Set interface data length (DL:8-bit/4-bit), numbers of display line (N:2-line/1-line)and, display font type (F:5×11 dots/5×8 dots)	39 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 μ s
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43 μ s
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43 μ s

12.5 Timing characteristics

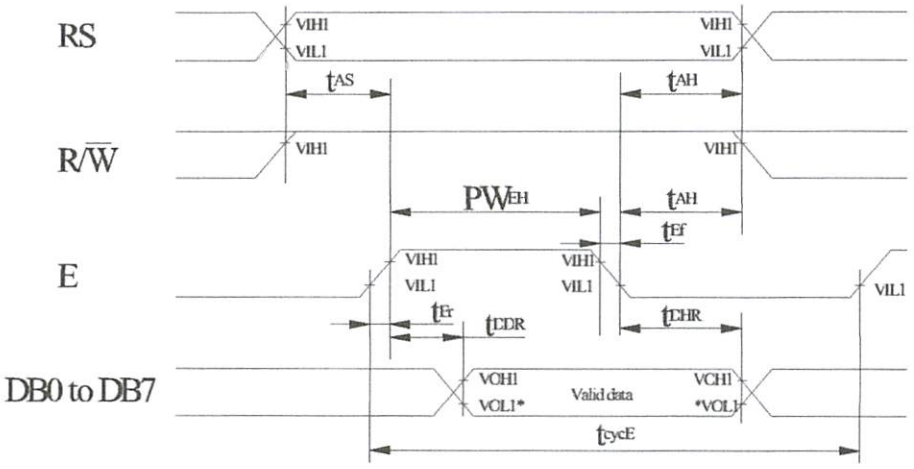
Write Operation



Ta= 25℃,Vdd=5.0±0.5V

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	500	–	–	ns
Enable pulse width (high level)	PW_{EH}	230	–	–	ns
Enable rise/fall time	t_{Er},t_{Ef}	–	–	20	ns
Address set-up time (RS, R/W to E)	t_{AS}	40	–	–	ns
Address hold time	t_{AH}	10	–	–	ns
Data set-up time	t_{DSW}	80	–	–	ns
Data hold time	t_H	10	–	–	ns

Read Operation



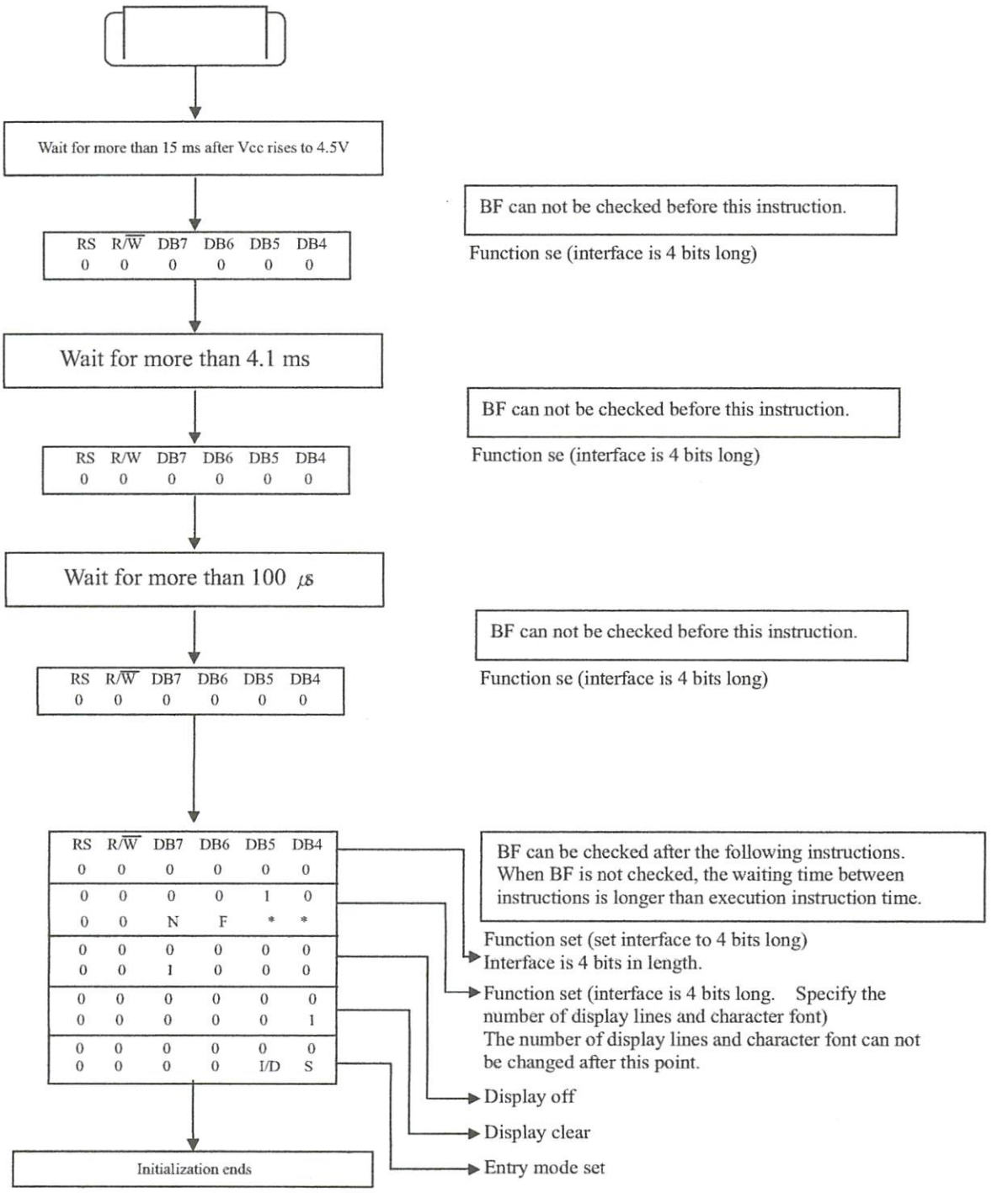
NOTE: *VOL1 is assumed to be 0.8V at 2 MHZ operation.

Ta=25℃,Vdd=5.0±0.5V

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	500	—	—	ns
Enable pulse width (high level)	PW_{EH}	230	—	—	ns
Enable rise/fall time	t_{Er},t_{Ef}	—	—	20	ns
Address set-up time (RS, R/W to E)	t_{AS}	40	—	—	ns
Address hold time	t_{AH}	10	—	—	ns
Data delay time	t_{DDR}	—	—	100	ns
Data hold time	t_{DHR}	5	—	—	ns

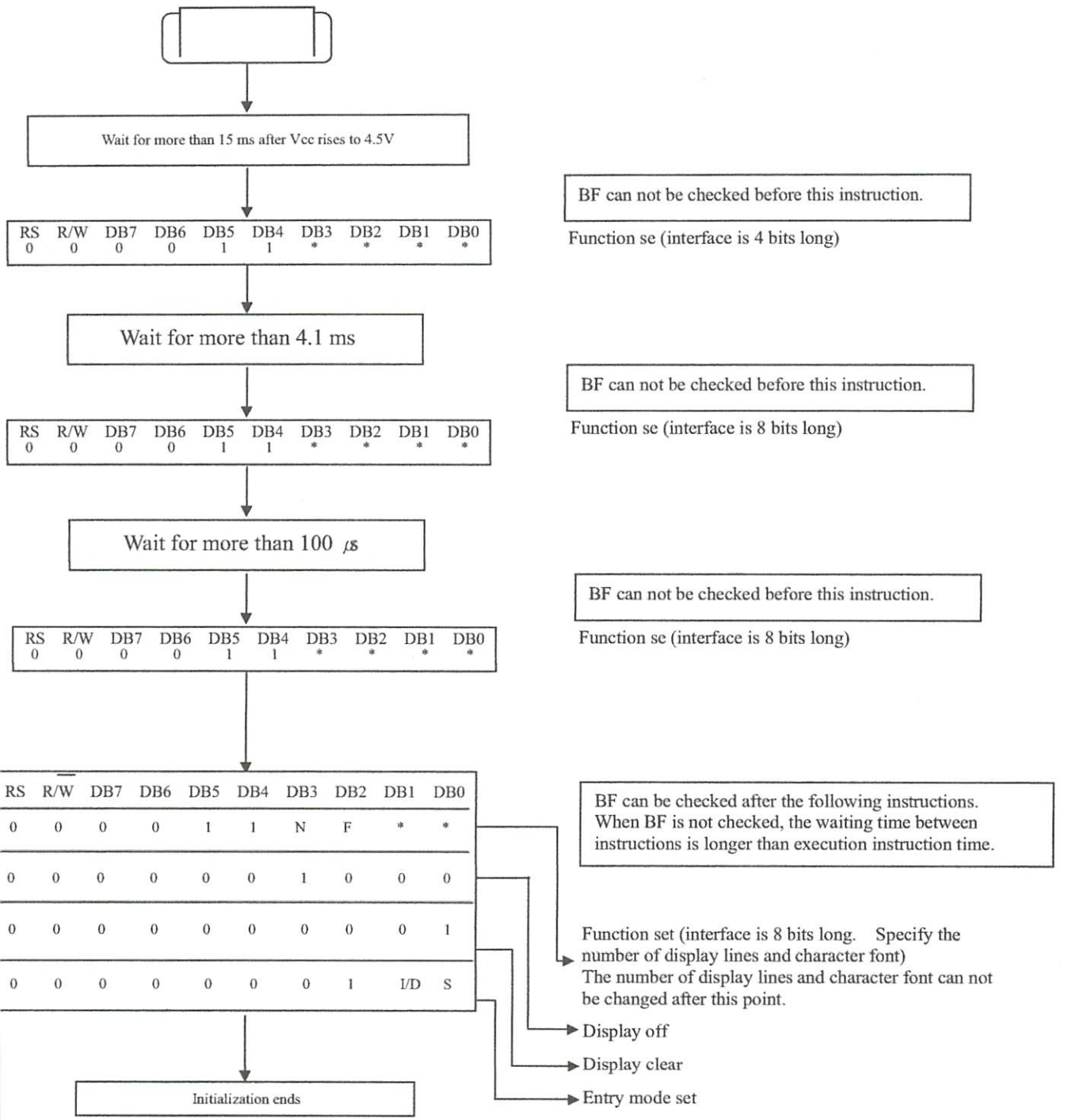
12.6 Initializing soft ware of LCM

4-bit interface



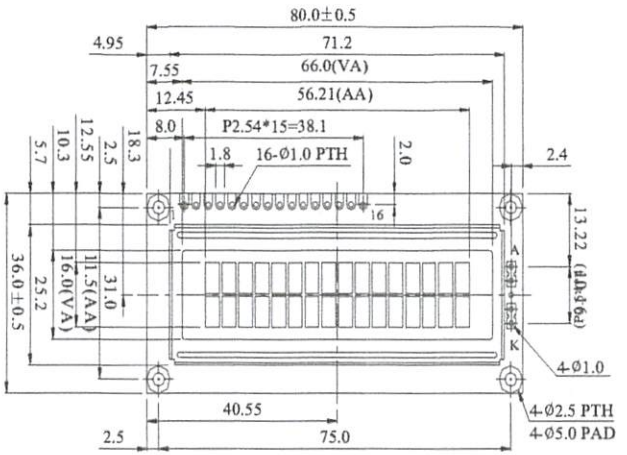
4-Bit interface

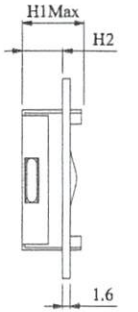
8-bit interface

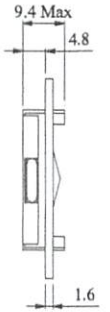


8-Bit interface

13. Outline drawing





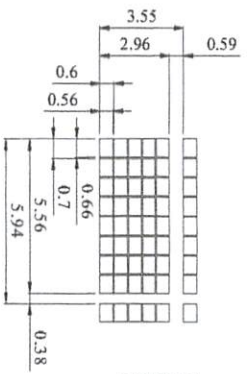


LED H/L B/L

	High	Low
H1	13.2	12.1
H2	8.6	7.5

EL or NO B/L

PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/$\overline{W}$
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K



DOT SIZE
SCALE 5/1

The tolerance of non-specified dimension is ±0.3mm.

科創光電股份有限公司 NEWTEC DISPLAY CO., LTD			REV: 0
			PAGE: 1/1
		UNITS: mm	SCALE: 1/1
APPROVE		TITLE	LCM Drawing
CHECK		MODEL	NC1602B
DRAW	Larry 10/04/04	DWG NO.	MD-931007