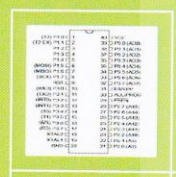
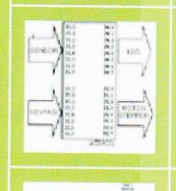
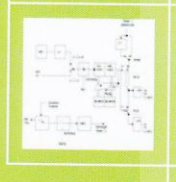





# BELAJAR MIKROKONTROLER ATMEL AT89C51

Buku Ajar ini lebih ditujukan kepada mahasiswa Jurusan Fisika FMIPA UNNES yang sedang mengikuti matakuliah Mikrokontroler dan Aplikasinya. Oleh karena itu beberapa contoh kasus dan latihan yang ada pada Buku Ajar ini sebagian besar diambil dari dunia mahasiswa.



Dr. Sujarwata, Drs., M.T.

## UU No 19 Tahun 2002 Tentang Hak Cipta

Fungsi dan Sifat hak Cipta Pasal 2

1. Hak Cipta merupakan hak eksklusif bagi pencipta atau pemegang Hak Cipta untuk mengumumkan atau memperbanyak ciptaannya, yang timbul secara otomatis setelah suatu ciptaan dilahirkan tanpa mengurangi pembatasan menurut peraturan perundang-undangan yang berlaku.

Hak Terkait Pasal 49

1. Pelaku memiliki hak eksklusif untuk memberikan izin atau melarang pihak lain yang tanpa persetujuannya membuat, memperbanyak, atau menyiarkan rekaman suara dan/atau gambar pertunjukannya.

Sanksi Pelanggaran Pasal 72

1. Barangsiapa dengan sengaja dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam pasal 2 ayat (1) atau pasal 49 ayat (2) dipidana dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp 1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 5.000.000.000,00 (lima miliar rupiah).
2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah)

deepublish| publisher

Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp/Faks: (0274) 4533427
Website: www.deepublish.co.id
www.penerbitdeepublish.com
E-mail: deepublish@ymail.com

**Katalog Dalam Terbitan (KDT)**

**SUJARWATA**

Belajar Mikrokontroler Atmel AT89C51/oleh Sujarwata.--Ed.1, Cet. 1--Yogyakarta: Deepublish, September 2016.

ix, 125 hlm.; Uk:17.5x25 cm

ISBN **978-602-401-481-0**

1. Elektronika I. Judul
621.381



Desain cover : Herlambang Rahmadhani
Penata letak : Cinthia Morris Sartono

**PENERBIT DEEPUBLISH**
**(Grup Penerbitan CV BUDI UTAMA)**
Anggota IKAPI (076/DIY/2012)

# BELAJAR MIKROKONTROLER ATMEL AT89C51




Dr. Sujarwata, Drs., M.T.

# Kata Pengantar

Assalamualaikum Wr. Wb.

Alhamdulillahi robbil 'alamin. Segala puji bagi Allah SWT atas segala nikmat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan buku berjudul **Belajar *Mikrokontroler ATMEL AT89C51***. Dalam penyusunannya buku, penulis memperoleh banyak bantuan berbagai pihak, karena itu ucapan terimakasih yang sebesar-besarnya kepada: segenap dosen fisika FMPIA UNNES dan teman-teman Program Pascasarjana Teknik Elektro dan Ilmu Fisika UGM.

Isi buku ini disajikan secara praktis dan lengkap sehingga membantu siswa, mahasiswa, dosen, guru, dan praktisi industri. Penekanan dan cakupan bidang yang dibahas buku ini sangat membantu dan berperan sebagai sumbangsih pemikiran dalam mendukung pemecahan permasalahan yang muncul pada mikrokontroler ATMEL AT89C51, karakterisasi dan aplikasi dalam bidang kendali elektronik. Oleh karena itu, buku ini disusun secara integratif antar disiplin ilmu, yaitu pengetahuan mikrokontroler, elektronika analog, elektronika daya, dan pemrograman, sehingga skill yang diperlukan terkait satu dengan lainnya.

Tiada gading yang tak retak, begitu pula buku ini masih banyak kekurangan dan jauh dari kesempurnaan. Untuk itu melalui kata pengantar penulis sangat terbuka menerima kritik dan saran membangun sehingga secara bertahap penulis dapat memperbaikinya. Saya ucapkan selamat membaca dan memahami Mikrokontroler ATMEL AT89C51 yang dijelaskan dalam buku ini. Semoga bermanfaat dan bisa memberikan pencerahan bagi pembaca untuk memahami mikrokontroler sebagai kendali elektronik. Amiin Ya Robbal Alamin .Wassalamualaikum Wr.Wb

Semarang, September 2016

Sujarwata

# Daftar Isi


**1 PENDAHULUAN** **1**

1.1 Manfaat Mikrokontroler . . . . . 2
1.2 Macam-macam Mikrokontroler . . . . . 3
1.3 Mikrokontroler Atmel AT89C51 . . . . . 4
1.4 Mode Pengalamatan AT89C51 . . . . . 5
1.4.1 Pengalamatan byte langsung . . . . . 5
1.4.2 Pengalamatan bit langsung . . . . . 5
1.4.3 Pengalamatan Register Basis dan Register Index tidak langsung . . . . . 6
1.4.4 Pengalamatan register tak langsung . . . . . 6
1.4.5 Pengalamatan segera . . . . . 6
1.5 Instruksi Dalam Mikrokontroler AT89C51 . . . . . 6
1.5.1 Operasi Aritmatika . . . . . 7
1.5.2 Operasi logika . . . . . 7
1.5.3 Operasi transfer data . . . . . 7
1.5.4 Operasi manipulasi variable Bolean . . . . . 7
1.5.5 Pencabangan . . . . . 7
1.6 Instruksi Mikrokontroler AT89C51 . . . . . 8
1.7 Timer dan Counter AT89C51/52 . . . . . 9
1.8 Sarana (*Timer/Counter*) AT89C51/52 . . . . . 10
1.9 Mode kerja Timer 0 dan Timer 1 . . . . . 11
1.9.1 Mode 0: Pencacah Biner 13 bit . . . . . 11
1.9.2 Mode 1: Pewaktu/pencacah 16 bit . . . . . 12

1.9.3 Mode 2 : Pencacah Biner 8 bit dengan isi ulang . 13
1.9.4 Mode 3 : Gabungan Pencacah Biner 16 bit dan 8 bit 14

**2 STRUKTUR INTERUPSI 16**
2.1 Pengaktifan Interupsi . . . . . 16
2.2 Prioritas Interupsi . . . . . 16
2.3 Simulasi Tingkat Prioritas Ketiga Dalam Perangkat Lunak 18
2.4 Reset . . . . . 19

**3 PEMROGRAMAN 20**
3.1 Tata Cara Membuat Program Mikrokontroler AT89C51 . 20
3.2 Program Mikrokontroler AT89C51 (TS Controls tata cara mensimulasikan Emulator 8051) . . . . . 21
3.3 Langkah-langkah Percobaan . . . . . 22
3.4 *Flow Chart* . . . . . 23
3.5 Masalah dan pemecahannya . . . . . 25
3.6 Program aplikasi untuk membuat kelompok 4 LED mati-hidup secara bergantian *(flip-flop)* . . . . . 27

**4 INSTRUKSI BAHASA ASSEMBLY AT89C51 31**
4.1 ADD (*Add Accumulator*) dan ADDC (*Add Accumulator With Carry*) . . . . . 31
4.2 JUMP (Lompat) . . . . . 32
4.2.1 Pengujian Nilai Bolean . . . . . 32
4.2.2 Mengatur alur program . . . . . 32
4.2.3 Instruksi JUMP bersyarat yang fungsinya memantau nilai Akumulator A . . . . . 34
4.3 CALL . . . . . 35
4.4 RET *(Return )* . . . . . 35
4.5 RETI . . . . . 35
4.6 ACALL *(Absolute Call)* . . . . . 36
4.7 LCALL *(Long Call)* . . . . . 36

4.8 Instruksi Transfer Data yang mengakses Ruang Memori Data Internal . . . 36
4.9 Instruksi Transfer Data yang mengakses Memori Data External . . . 37
4.10 Tabel-Tengok *(Lookup Table)* . . . 37
4.11 Instruksi SUBB *(Substract From Accumulator With Borrow)* 38
4.12 Instruksi DA A *(Decimal Adjust)* . . . 39
4.13 Instruksi MUL AB . . . 39
4.14 Instruksi DIV AB . . . 39
4.15 Instruksi DEC dan INC . . . 40
4.16 Instruksi INC DPTR . . . 41
4.17 SETB (*Set Bit*) . . . 41
4.18 CLR (*Clear Bit*) . . . 41
4.19 Instruksi ANL (AND Logical), ORL (OR Logical), CPL (Complement Bit) dan EX-OR (Exlusive-OR Logical) . . 42
4.20 ICJNE*(Compare and Jump if Not Equal )* . . . 43

**5 APLIKASI PORT SERIAL 44**

5.1 Antarmuka Serial . . . 44
5.1.1 Mode 0 . . . 45
5.1.2 Mode 1 . . . 45
5.1.3 Mode 2 . . . 45
5.1.4 Mode 3 . . . 46
5.2 Register Kontrol Port Serial . . . 46
5.2.1 Bit SM0 dan bit SM1 . . . 46
5.2.2 Bit REN . . . 47
5.2.3 Mode 2 dan mode 3 . . . 47
5.2.4 Bit TI (bit 1) . . . 47
5.2.5 Bit RI (bit 0) . . . 48
5.3 Baut Rate . . . 49
5.4 Pengunaan Timer 1 Menghasilkan Baut Rate . . . 49

5.5 Menggunakan Timer 2 Untuk Menghasilkan Baut Rate Pada AT89C51 . . . . . 51

**6 APLIKASI PENGGUNAAN PORT PARALEL 54**

6.1 Penggunaan Port . . . . . 54
6.2 Program Pertama . . . . . 57
6.3 Program Kedua . . . . . 59
6.4 Program Ketiga . . . . . 61
6.5 Program Keempat . . . . . 64
6.6 Program Kelima . . . . . 66
6.7 Aplikasi Port untuk Penggerak 7-segment . . . . . 68
6.7.1 Display LED 7 Segment . . . . . 68
6.7.2 Penyederhanaan program . . . . . 71
6.7.3 Aplikasi Port Masukan/Keluaran Penggerak LED 72
6.7.4 Program untuk menghidupkan LED melalui push button di P3.0 . . . . . 73
6.7.5 Program sistem *switch* toggle pada P3.0. . . . . 73
6.8 Menyimpan Program Ke Flash Memori Menggunakan *Easy Programer* . . . . . 74

**7 CONTOH APLIKASI PORT SERIAL 76**

7.1 Inisialisasi Port Serial . . . . . 76
7.2 Subrutin Pengirim Karakter . . . . . 77
7.3 Sub-rutin Penerima Karakter . . . . . 78

**8 KOMUNIKKASI SERIAL DENGAN KOMPUTER 80**

8.1 Program Pertama . . . . . 80
8.2 Program Kedua . . . . . 84

**9 KENDALI AT89C51 DENGAN VISUAL BASIC 6.0 88**

9.1 Prinsip Program Pada Mikrokontroler . . . . . 89
9.2 Komunikasi Data Serial . . . . . 92
9.3 Rangkaian Mikrokontroler AT89C51 . . . . . 94

9.4 Rangkaian Sistem Perancangan . . . . . 95
9.5 Perancangan Logika . . . . . 96
9.6 Pengujian Rangkaian Unit Kendali . . . . . 98

**10 AT89S52 UNTUK KENDALI SUHU 101**
10.1 Rancangan Untuk Pengering . . . . . 103
10.1.1 Perancangan mekanik pengering . . . . . 104
10.2 Perancangan Perangkat Keras . . . . . 105
10.3 Perancangan Perangkat Lunak . . . . . 107
10.4 Perangkat Keras . . . . . 108
10.4.1 Sensor Suhu . . . . . 108
10.4.2 *Zero Crossing Detector* . . . . . 109
10.4.3 Driver . . . . . 110
10.4.4 Pengujian Alat . . . . . 110

**11 CONTOH SENSOR YANG DAPAT DIGUNAKAN 112**
11.1 Sensor Gelombang Ultrasonik . . . . . 112
11.2 Sensor Infra merah . . . . . 112
11.3 Sensor *UV-Tron* . . . . . 113
11.4 Sensor Kompas . . . . . 115

**12 PENUTUP 116**

**DAFTAR PUSTAKA 118**

**INDEKS 120**

**GLOSARIUM 123**

# Daftar Tabel


2.1 Fungsi Register IE-Interrupt Enable Pada AT89C51 . . . 17
2.2 Fungsi Register IP Interrupt Priority Pada AT89C51 . . 18

3.1 Data untuk pemrograman . . . . . 23
3.2 Data untuk pemrograman . . . . . 29

4.1 Instruksi Transfer Data yang mengakses Ruang Memori Data Internal . . . . . 36
4.2 Instruksi transfer data mengakses memori data external . 37
4.3 Instruksi membaca Tabel-Tengok . . . . . 38

5.1 Penentuan Mode Kerja Port Serial . . . . . 46
5.2 Baut Rate Sering Digunakan yang Dihasilkan Timer 1 . . 50
5.3 Ringkasan baut rate untuk Timer 1 sebagai Generator Baut Rate . . . . . 51

7.1 **Port Serial** . . . . . 76

9.1 Tabel fungsi pengganti Port 3 . . . . . 91

# Daftar Gambar


1.1 Konsep dasar *Timer/Counter* Pada AT89C51/52 . . . . 10
1.2 Mode 0: Pencacah Biner 13 Bit . . . . . 12
1.3 Mode : Pencacah Biner 16 Bit . . . . . 13
1.4 Mode 2: Pencacah Biner 8 bit dengan isi Ulang Otomatis 13
1.5 Mode 3 : Gabungan Pencacah Biner 16 Bit dan 8 Bit . . 14

2.1 Register IE-Interrupt Enable pada AT89C51 . . . . . 17
2.2 Register IP Interrupt Priority pada AT89C51 . . . . . 17

3.1 *Flow Chart* bepergian dengan naik bis . . . . . 26
3.2 Diagram Alir *Sub-rutin Delay* . . . . . 30

5.1 Susunan Bit dalam SCON . . . . . 46
5.2 Bit SMOD dalam Register PCON . . . . . 48
5.3 Timer 2 sebagai generator baut rate . . . . . 53

6.1 Aplikasi Port pada AT89C51 . . . . . 55
6.2 Rangkaian Aplikasi LED . . . . . 56
6.3 Aplikasi Menghidupkan LED Melalui Kanal Paralel . . . 57
6.4 Rangkaian Aplikasi 7 Segment . . . . . 68
6.5 Rangkaian Aplikasi 7 Segment . . . . . 72

9.1 Blok diagram fungsional AT89C51 . . . . . 90
9.2 Susunan pin AT89C51 . . . . . 92
9.3 Konektor serial DB-9 . . . . . 93
9.4 Konektor serial DB-25 . . . . . 94

9.5 Rangkaian Mikrokontroler . . . . . 95
9.6 Blok Diagram . . . . . 96
9.7 Flowchart program . . . . . 98
9.8 Pengujian Mikrokontroler AT89C51 . . . . . 99

10.1 Simbol ADC 0804 . . . . . 102
10.2 Simbol AT89S52 . . . . . 104
10.3 Mekanik pengering . . . . . 105
10.4 Diagram blok sistem . . . . . 106
10.5 Blok diagram detektor persilangan nol . . . . . 107
10.6 Diagram kotak penggerak pemanas . . . . . 107
10.7 Alur pelayanan penekanan tombol . . . . . 108
10.8 Grafik tegangan keluaran sensor suhu LM35 terhadap suhu 109
10.9 Skema prinsip kerja oven konvensional . . . . . 111

11.1 Sensor Gelombang Ultrasonik . . . . . 113
11.2 Sensor Inframerah . . . . . 113
11.3 Sensor UVTron . . . . . 114
11.4 Sensor Kompas . . . . . 115

# Bab 1

# PENDAHULUAN

Sebelum mempelajari Mikrokontroler terlebih dahulu untuk mengetahui perbedaan anatara Mikroprosesor, Mikrokomputer dan Mikrokontroler. Mikroprosesor adalah bagian CPU*(Central Processing Unit)* dari suatu komputer, tanpa adanya memori, I/O dan periferal yang dibutuhkan oleh sebuah system lengkap. Misalnya, 8088 dan 80X86 adalah sebuah mikroprosesor. Untuk dapat bekerja mikroprosesor membutuhkan perangkat pendukung yang berupa RAM, ROM dan I/O. Berikut adalah karakteristik penting dari mikroprosesor:

(a) Ukuran bus data internal *(internal data bus size)*: Jumlah saluran yang terdapat dalam mikroprosesor yang menyatakan jumlah bit yang dapat ditransfer antar komponen di dalam mikroprosesor.

(b) Ukuran bus data eksternal *(external data bus size)*: Jumlah saluran yang digunakan untuk transfer data antar komponen mikroprosesor dan komponen-komponen di luar mikroprosesor.

(c) Ukuran alamat memori *(memory address size)*: Jumlah alamat memori yang dapat dialamati oleh mikroprosesor secara langsung.

(d) Kecepatan clock*(clock speed)*: Rate atau kecepatan clock untuk menuntun kerja mikroprosesor.

(e) Fitur-fitur spesial *(special features)*: Fitur khusus untuk mendukung aplikasi tertentu seperti fasilitas pemrosesan *floating point*, multimedia dan sebagainya.

Jika sebuah mikroprosesor dikombinasikan dengan suatu I/O *(Input/Output)* dan memori ( RAM/ROM ), maka akan dihasilkan sebuah mikrokomputer. Pada kenyataannya mengkombinasikan CPU dengan memori dan I/O dilakukan level *chip*, yang menghasilkan *Single Chip Microcomputer* (SCM). Perbedaan yang menonjol antara mikrokomputer dibanding SCM adalah pada penggunaan perangkat masukan/keluaran (I/O ) dan media penyimpanan programnya.

Mikrokomputer (IBM PC) menggunakan disket atau *tape* seabagai penyimpanan suatu program, sedangkan CSM menggunakan EPROM *(Eraseable Programmable Read Only Memory)* sebagai penyimpanan program. Secara sederhana pengertian mikrokontroler adalah untai terpadu (IC ) atau disebut juga *chip* yang bekerja dengan program.

Mikrokontroler adalah suatu alat elektronika digital yang mempunyai masukan dan keluaran serta kendali dengan program yang bisa ditulis dan dihapus dengan cara khusus. Sederhananya, cara kerja mikrokontroler sebenarnya hanya membaca dan menulis data. Mikrokontroler merupakan komputer didalam *chip* yang digunakan untuk mengendalikan peralatan elektronik. Mikrokontroler disebut sebagai "pengendali kecil" dimana sebuah sistem elektronik yang sebelumnya memerlukan komponen-komponen pendukung seperti (IC TTL dan CMOS) dapat direduksi dan dikendalikan.

## 1.1 Manfaat Mikrokontroler

Dengan menguasainya mikrokontroler, kita bisa menerapkannya kedalam kehidupan sehari-hari, seperti mengendalikan suatu perangkat elektronik

dari berbagai sensor dan kondisi. Mikrokontroler banyak diaplikasikan pada sistem kendali atau monitoring, misalnya : sebagai alat kontrol penampil tulisan,sistem pengukuran jarak jauh (telemetri),pengendali pada robot, berbagai macam mainan anak, mesin, cuci, microwave dan sistem elektronika yang lainya. Di bidang pertanian, mikrokontroler dapat digunakan sebagai kendali kelembaban untuk budidaya jamur, bibit tanaman dan sebagainya. Bahkan dapat dimanfaatkan untuk membuat *SMS Gateway*, radio militer frekuensi *hopping* (radio komunikasi anti sadap), sistem monitoring cuaca dengan balon udara,*automatic vehicel locator*.dan sebagainya.

Dengan menggunaan mikrokontroler, kita mndapatkan banyak sekali manfaat yang dapat diperoleh, antara lain:

a) Sistem elektronik akan menjadi efisien, efektif dan fleksibel penggunaannya.

b) Mikrokontroler tersusun dalam satu *chip*, dimana prosesor, memori, dan I/O terintegrasi menjadi satu kesatuan kontrol sistem sehingga mikrokontroler dapat dikatakan sebagai komputer mini yang dapat bekerja secara inovatif sesuai dengan kebutuhan sistem.

c) Tingkat keamanan dan akurasi yang lebih baik.

d) Merancang sistem elektronik akan lebih cepat dan mudah, sebagian besar merupakan perangkat lunak yang mudah dimodifikasi.

e) Kesalahan dari sistem elektronik akan lebih mudah ditelusuri karena sistemnya yang kompak.

## 1.2 Macam-macam Mikrokontroler

Secara teknis hanya ada 2 macam mikrokontroler, yaitu RISC dan CISC yang masing-masing mempunyai keluarga sendiri-sendiri.

1) RISC kependekan dari *Reduced Instruction Set Computer*: instruksi terbatas tapi memiliki fasilitas yang lebih banyak.

2) CISC kependekan dari *Complex Instruction Set Computer*: instruksi bisa dikatakan lebih lengkap tapi dengan fasilitas secukupnya.

Mikrokontroler mempunyai banyak jenisnya, seperti: Motorola dengan seri 68xx, keluarga MCS51, Philip, Dallas, keluarga PIC dari *Microchip*. Setiap keluarga masih terbagi lagi dalam beberapa tipe. Sebagai contoh mikrokontroler yang sering digunakan untuk aplikasikan dengan spesifikasi yang berbeda(sesuai dengan pengalaman penulis). misalnya:

## 1.3 Mikrokontroler Atmel AT89C51

Mikrokontroler ini kompitabel dengan keluarga yang diproduksi oleh Intel Inc USA. Keluarga MCS-51 terdiri dari 4 macam versi, yaitu: 8013, 8051 , 8751 dan 8951. Untuk tipe 89C51 merupakan versi dengan EEPROM. Kode C menyatakan mikrokontroler dibuat mengunakan teknologi CMOS. Mikrokontroler ini juga terdapat pewaktu/pencacah yang digunakan untuk pengukuran interval waktu, pengukuran lembar pulsa, penghitung kejadian, sumber interupsi secara periodik dan pembangkit pulsa data serial. Pewaktu/pencacah ini dapat berfungsi sebagai pencacah, jika isyarat berasal dari luar dan berfungsi sebagai pewaktu jika isyarat bersal dari osilator dalam mikrokontroler.

Dalam buku ini, mikrokontroler yang akan dibahas secara mendalam adalah mikrokontroler AT89C51/ AT89C52 buatan Atmel Icn, sedangkan untuk mikrokontroler IC PICI6C57 akan dibahas pada buku berikutnya.

Mikrokontroler dari Atmel AT89C51 mempunyai spesifikasi, sebagai berikut:

a) Mempunyai 40 kaki atau pin.

b) 4 Kbyte EPROM untuk memori program.

c) 128 byte internal RAM untuk memori data.

d) Memiliki 4 port I/O, masing-masing terdiri dari 8 bit, sifatnya dua arah dan setiap bit dapat dialamati.

e) Mempunyai 2 *timer(counter)* 16 bit dan 6 interupsi dan satu *port serial full duplex*, yaitu port 3.

f) Bekerja pada frekuensi 0 - 24 MHz dengan osilator internal.

g) Mempunyai fasilitas penguncian program untuk menghindari terjadinya pembajakan program dan memiliki mode operasi daya rendah.

## 1.4 Mode Pengalamatan AT89C51

Arsitektur mikrokontroler AT89C51 membedakan memori data dan memori program, sehingga mode pengalamatan keduanya juga berbeda. Terdapat 5 mode pengalamatan umum, yaitu empat melakukan operasi dalam *byte*, dan satu melakukan operasi dalam *byte* maupun bit. Mode pengalamatan tersebut adalah sebagai berikut:

### 1.4.1 Pengalamatan byte langsung

Pengalamatan ini menentukan lokasi byte di dalam RAM ataupun Register Fungsi Khusus (SFR ).
Contoh :*ADD A, alamat*
Artinya, menambahkan isi alamat dengan isi akumulator A, hasilnya ditempatkan di akumulator A. Alamat adalah alamat lokasi memori bite.

### 1.4.2 Pengalamatan bit langsung

Pengalamatan ini untuk melakukan manipulasi atau tes pada 128 bit bendera yang telah didefinisikan melalui perangkat lunak dan 128 bit

pada SFR.
Contoh : *MOV C, P1.0*
Artinya : isi carry dengan kondisi masukan

### 1.4.3 Pengalamatan Register Basis dan Register Index tidak langsung

Pengalamatan ini untuk menyederhanakan pengaksesan *look-up table* (LUT) yang terletak di dalam memori program.
Contoh : *JMP @A + DPTR*
Artinya : menambah data pointer 16 bit dengan isi akumulator A 8 bit dan hasilnya ditempatkan di program pointer.

### 1.4.4 Pengalamatan register tak langsung

Pengalamatan ini untuk mengakses suatu lokasi RAM yang alamatnya ditentukan oleh isi R0 dan R1. Dalam bahasa Assembly, pengalamatan tak langsung ini dituliskan dengan tanda @ di depan R0 dan R1.
Contoh : *ADD A,@R0*. Artinya: menambah isi alamat yang ditunjuk R0 dengan isi akumulator A, hasilnya ditempatkan pada akumulator A.

### 1.4.5 Pengalamatan segera

Pengalamatan ini untuk memindahkan data secara langsung ke register atau memori. Mode pengalamatan ini digunakan tanda # di depan nilai.
Contoh : *ADD A,# data*. Artinya : menambahkan data 8 bit dengan isi akumulator A, hasilnya ditempatkan di akumulator A.

## 1.5 Instruksi Dalam Mikrokontroler AT89C51

Instruksi-instruksi dalam mikrokontroler AT89C51 terdiri atas 51 operasi dasar, yang dibagi menjadi 5 kelompok fungsional , yaitu sebagai berikut:

### 1.5.1 Operasi Aritmatika

Terdiri atas penjumlahan, penambahan satui step increment, pengurangan satu step (*decrement*), penjumlahan format decimal *(decimal add adjust)*, pengurangan , perkalian dan pembagian.
Contoh : *INC, DEC, SUBB, ADD, MUL, DIV, DA*

### 1.5.2 Operasi logika

Operasi ini terdiri dari *AND, OR, dan Exclusive OR* pada register A oleh operan kedua yang dapat berupa sebuah data,register, RAM data internal yang dialamati langsung atau tidak langsung, maupun SFR.
Contoh : *ANL, ORL, XRL, CPL, RL,RRC.*

### 1.5.3 Operasi transfer data

Dapat dilakukan antara dua operan yang dapat berupa register pada bank register, RAM data internal, akumulator dan SFR.
Contoh : *MOV, MOVC, MOVX, PUSH, POP, XCH.*

### 1.5.4 Operasi manipulasi variable Bolean

Prosesor Boolean dapat melakukan operasi transfer data pada logika Boolean (1 bit).Transfer data dapat dilakukan sejumlah 256 bit yang dapat dialamati dari atau ke register carry menggunakan pengamalatan langsung.
Contoh: *CLR,SETB,CPL,JC,JB,JNB,JBC.*



### 1.5.5 Pencabangan

Urutan pelaksanaan program dapat dikendalikan oleh instruksi pencabangan bersyarat maupun tidak bersyarat. Pencabangan bersyarat berhubungan dengan isi akumulator, register, memori, data, maupun bit carry. Sedangkan pencabangan tidak bersyarat memiliki lebar alamat 11

dan 8 bit untuk mengefisienkan penggunaan memori program.
Contoh : *ACALL, LCALL, RET, RETI, AJMP, JMP, JZ, JNZ, CJNE, NOP*

## 1.6 Instruksi Mikrokontroler AT89C51

Instruksi-instruksi dalam mikrokontroler AT89C51 terdiri atas 51 operasi dasar, yang dibagi menjadi 5 kelompok fungsional, yaitu sebagai berikut:

A. Operasi Aritmatika.
   Terdiri atas penjumlahan, penambahan satui step (*increment*), pengurangan satu step (*decrement*), penjumlahan format decimal (*decimal add adjust*), pengurangan , perkalian dan pembagian.
   Contoh : *INC, DEC, SUBB, ADD, MUL, DIV, DA*

B. Operasi logika.
   Operasi ini terdiri dari AND, OR, dan *Exclusive OR* pada register A oleh operan kedua yang dapat berupa sebuah data, register, RAM data internal yang dialamati langsung atau tidak langsung, maupun SFR.
   Contoh : *ANL, ORL, XRL, CPL, RL,RRC.*

C. Operasi transfer data.
   Dapat dilakukan antara dua operan yang dapat berupa register pada *bank register*, RAM data internal, akumulator dan SFR.
   Contoh : *MOV, MOVC, MOVX, PUSH, POP, XCH.*

D. Operasi manipulasi variable Bolean.
   Prosesor Boolean dapat melakukan operasi transfer data pada logika Boolean (1 bit).Transfer data dapat dilakukan sejumlah 256 bit yang dialamati dari atau ke register carry menggunakan pengamalatan langsung.
   Contoh : *CLR, SETB, CPL, JC, JB, JNB, JBC.*

E. Pencabangan.
   Urutan pelaksanaan program dikendalikan oleh instruksi pencabangan bersyarat maupun tidak bersyarat. Pencabangan bersyarat berhubungan

dengan isi akumulator, register, memori, data, dan bit carry. Sedangkan pencabangan tidak bersyarat memiliki lebar alamat 11 dan 8 bit untuk mengefisienkan penggunaan memori program. Contoh :*ACALL, LCALL, RET, RETI, AJMP, JMP, JZ, JNZ, CJNE, NOP.*

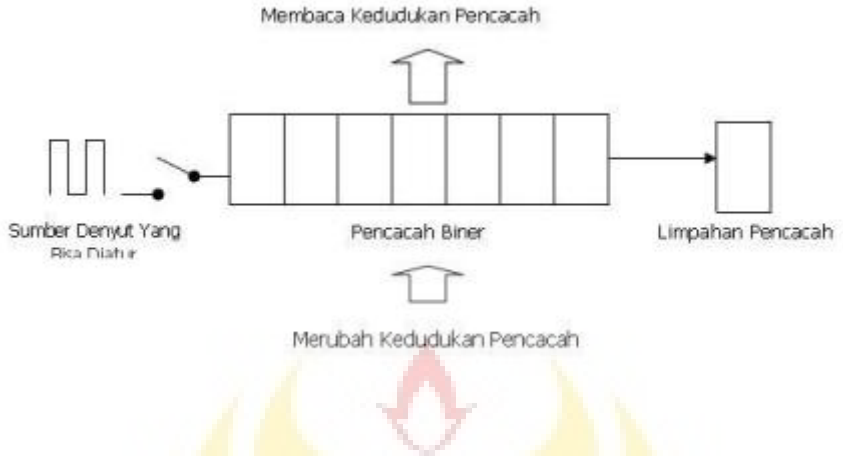
## 1.7 Timer dan Counter AT89C51/52

Mikrokontroler AT89C51 terdapat pewaktu (*timer*)/ pencacah (*counter*) dengan lebar 16 bit independen. Pewaktu digunakan untuk pengukuran interval waktu, pengukuran lebar pulsa, penghitung kejadian, sumber interupsi secara periodik dan pembangkit pulsa data serial. Pada dasarnya sarana masukan yang satu ini merupakan seperangkat pencacah biner*(binary counter)* yang terhubung langsung ke saluran data mikrokontroler, sehingga dapat membaca kondisi pencacah dan bila diperlukan mikrokontroler dapat pula merubah kondisi pencacah tersebut.

Seperti layaknya pencacah biner, saat sinyal detak(*clock*) yang diberikan sudah melebihi kapasitas pencacah , maka pencacah akan memberikan sinyal *overflow* atau limpahan, sinyal ini merupakan hal yang penting dalam pemakian pencacah dan terjadinya limpahan pencacah ini dicatat dalam register. Selain itu, sinyal detak yang diberikan ke pencacah bisa dikendalikan dengan mudah. Pada (Gambar 3.1.), selanjutnya ditunjukkan konsep dasar dari pada *timer/counter* pada AT89C51/52.

Sinyal detak yang diberikan ke pencacah dibedakan menjadi dua macam, yang pertama adalah sinyal detak dengan frekuensi tetap yang sudah diketahui besarnya dan kedua adalah sinyal detak dengan frekuensi yang bisa bervariasi. Jika sebuah pencacah bekerja dengan frekuensi tetap , dikatakan pencacah tersebut bekerja sebagai timer atau pewaktu , karena kondisi pencacah tersebut setara dengan waktu yang bisa ditentukan secara pasti. Jika sebuah pencacah bekerja dengan frekuensi yang bervariasi, dikatakan pencacah tersebut bekerja sebagai counter atau pencacah , kondisi pencach tersebut menyatakan banyaknya pulsa

detak yang sudah diterima. Untai pencacah biner tersebut merupakan pencacah biner naik (*count up binary counter*).




Gambar 1.1: Konsep dasar *Timer/Counter* Pada AT89C51/52

## 1.8 Sarana (*Timer/Counter*) AT89C51/52

Keluarga Mikrokontroler AT89C51, misalnya AT89C51 dan AT89C51, dilengkapi dengan dua perangkat *timer/counter*, masing-masing dinamakan sebagai timer 0 dan timer 1. Sedangkan untuk jenis lainnya, yaitu AT89C52/55 mempunyai tambahan satu perangkan *Timer/Counter* lagi yang dinamakan sebagai timer 2.

Perangkat (*timer/counter*) merupakan perangkat keras yang terpadu dalam mikrokontroler AT89C51, untuk mengaksesnya digunakan register khusus yang tersimpan dalam SFR. Pencacah biner timer 0 diakses melalui register TLO (*Timer 0 Low Byte*, memori data internal alamat 6Ah) dan register TH0 (*Timer 0 High Byte*, memori data internal alamat 6Ch ).Pencacah biner timer 1 diakses melalui register TL1 (*Timer 1 Low Byte*, memori data internal 6Bh) dan register TH1*(Timer 1 High Byte* , memori-data internal alamat 6D h). Pencacah biner *Timer/Counter* AT89C51/52 merupakan pencacah biner 16 bit naik (*count up binary counter*) yang mencacah dari 0000h sampai FFFF h, saat kondisi pencacah

berubah dari FFFF h kembali ke 0000 h akan timbul sinyal limpahan (*overflow*).

Untuk mengatur kerja timer/counter digunakan 2 register tambahan yang dipakai bersama oleh timer 0 dan timer 1. Register tambahan tersebut adalah register TCON (*Timer Control Register*, memori-data internal alamat 88 h,bisa dialamati per bit) dan register TMOD (*Timer Mode Register*, memori-data internal alamat 89 h, tidak bisa dialamati per bit). TL0, TH0, TL1 dan TH1 merupakan SFR (Special Function Register) yang dipakai untuk membentuk pencacah biner timer 0 dan timer 1 .Kapasitas keempat register tersebut masing-masing 8 bit, bisa disusun menjadi 4 macam mode pencacah biner.

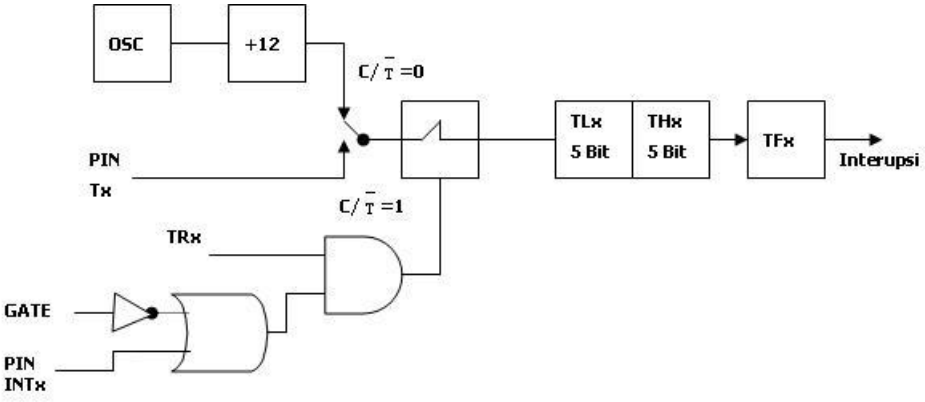
## 1.9 Mode kerja Timer 0 dan Timer 1

Pada mode 0,1 dan 2, timer 0 dan timer 1 masing-masing bekerja sendiri, artinya bisa dibuat timer 0 bekerja pada mode 1 dan timer 1 bekerja pada mode 2 atau kombinasi lainnya sesuai dengan keperluan. Sedangkan pada mode 3 , *TL0, TH0, TL1 dan TH1* dipakai bersama-sama untuk menyusun sistem timer yang terpadu (khusus).

### 1.9.1 Mode 0: Pencacah Biner 13 bit

Pada (Gambar 3.2) ditunjukkan diagram fungsional timer x pada Mode 0. Pencacah biner dibentuk dengan TLx ( bisa TL0 atau TL1) sebagai pencacah biner 5 bit (meskipun sesungguhnya 8 bit), limpahan dari pencacah biner 5 bit ini dihubungkan ke THx ( bisa TH0 atau TH1) membentuk sebuah untui pencacah biner 13 bit. Limpahan dari pencacah 13 bit ini ditampung di TFx (bisa TF0 atau TF1) yang berada di dalam register TCON.

Pada saat terjadi limpahan (dari 1FFF h ke 0000 h), maka *flag interupsi Timer* (TF1) akan diset(=1). Masukan ke pencacah (baik dari eksternal( Tx) maupun internal (1/2 Fosc)) diaktifkan jika TRx =1 dan




Gambar 1.2: Mode 0: Pencacah Biner 13 Bit

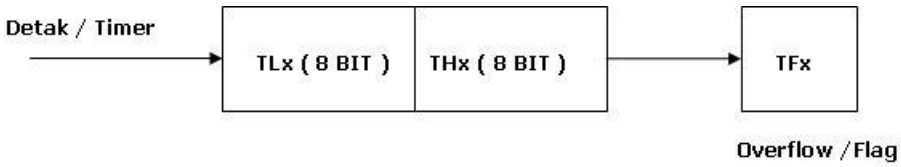
Gate = 0 atau INTx =1, maka keluaran gerbang OR menjadi selalu 1 dan akibatnya hasil gerbang AND juga 1. Jika gate =1 , maka timer sepenuhnya dikendalikan oleh masukan eksternal INTx dan bisa digunakan dalam pengukuran lebar pulsa(*pulse-width* ). TRx merupakan bit control dalam register TCON.

Karena THx dan TLx digunakan hanya untuk membentuk pencacah biner 13 bit maka 3 bit atas TLx tidak menentu dan harus diabaikan. Men-set TRx tidak akan secara otomatis menghaous isi *register timer x*. Mode ini meneruskan sarana timer yang ada pada mikrokontroler MCS48 (mikrokontroler pendahulu AT89C51),dengan maksud rancangan alat yang dibuat dengan MCS48 dapat dengan mudah diadaptasikan ke AT89C51( menjaga kompatibilitas ). Mode ini tidak banyak dipakai lagi unutk saat ini.

### 1.9.2 Mode 1: Pewaktu/pencacah 16 bit

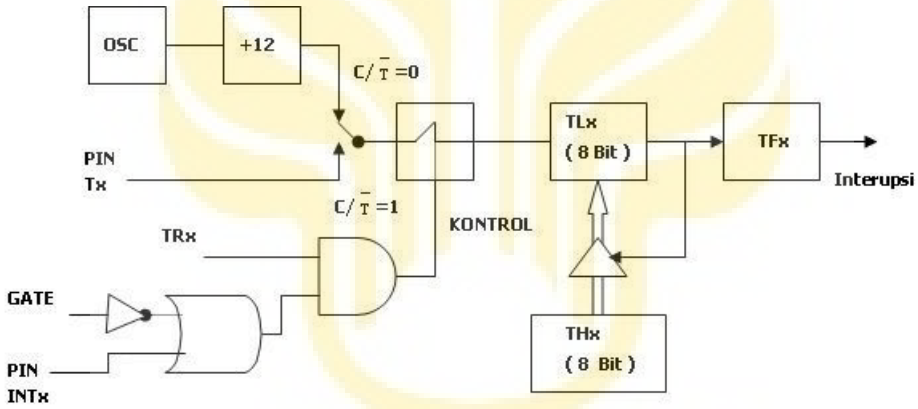
Mode ini sama dengan Mode 0, hanya saja register TLx dipakai sepenuhnya sebagai pencacah biner 8 bit, sehingga kapasitas pencacah biner yang terbentuk adalah 16 bit. Seiring dengan sinyal detak, kondisi pencacah biner 16 bit ini dimulai dari 0000 h,0001 h, 0002 h sampai FFFF h,

kemudian kembali menjadi 0000h (pada saat itu terjadi sinyal limpahan atau over flow pada TFx).




Gambar 1.3: Mode : Pencacah Biner 16 Bit

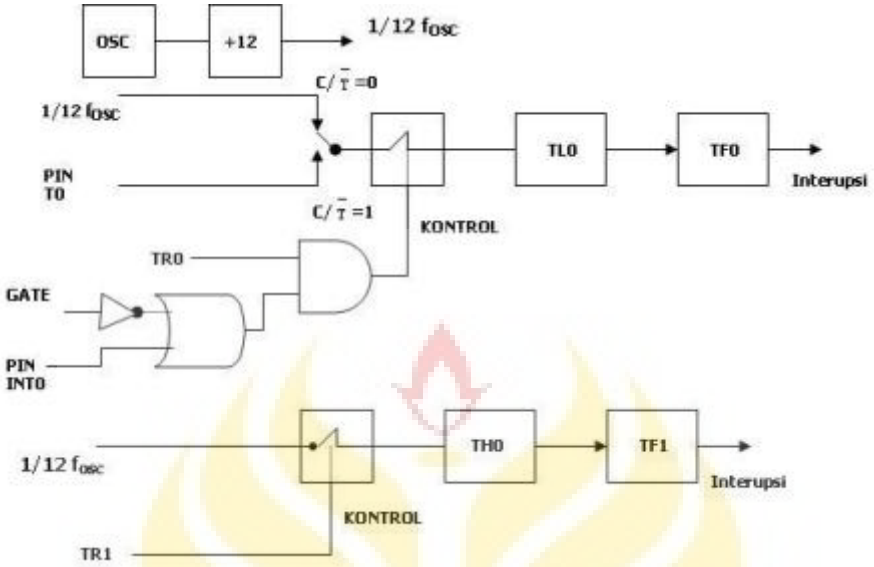
### 1.9.3 Mode 2 : Pencacah Biner 8 bit dengan isi ulang




Gambar 1.4: Mode 2: Pencacah Biner 8 bit dengan isi Ulang Otomatis

Pada (Gambar 1.4) ditunjukkan diagram fungsional timer/counter pada mode 2 TLx dipakai sebagai pencacah biner 8 bit,sedangkan THx dipakai untuk menyimpan nilai yang diisikan ulang ke TLx setiap kali kondisi TLx melimpah (overflow) atau berubah dari FF h menjadi 00 h.Dengan cara ini bisa diperoleh sinyal overflow yang frekuensinya bisa ditentukan oleh nilai yang disimpan dalam THx.

### 1.9.4 Mode 3 : Gabungan Pencacah Biner 16 bit dan 8 bit




Gambar 1.5: Mode 3 : Gabungan Pencacah Biner 16 Bit dan 8 Bit

Pada Mode 3:*TLO, THO, TL1, dan TH1* dipakai untuk membentuk 3 rangka pencacah, yang pertama adalah untai pencacah biner 16 bit tanpa fasilitas pemantau sinyal limpahan atau *overflow* yang dibentuk dengan TL1 dan TH1 (Gambar 1.5).

Kedua adalah TLO yang dipakai sebagai pencacah biner 8 bit dengan TFO sebagai sarana pemantau limpahan.Pencacah biner ketiga adalah THO yang dipakai sebagai pencacah biner 8 bit dengan TF1 sebagai sarana pemantau loimpahan,dengan demikian THO-lah yang mengendalikan interupsi Timer 1 (TFI ).

Mode 3 digunakan pada aplikasi yang membutuhkan sebuah timer atau pencacah 8 bit tambahan. Timer 0 pada Mode 3,AT89C51 memiliki 3 buah timer/pencacah (untuk AT89C52 seakan-akan memiliki 4 buah).

Pada Mode 3 ini,timer 1 dapat dihidupkan dan dimatikan menggunakan M1 dan MO pada register TMOD. Pada kasus seperti ini,timer1 masih dapat digunakan oleh port serial untuk menghasilakan baud rate atau aplikasi apa saja yang tidak membutuhkan interupsi.

# Bab 2

# STRUKTUR INTERUPSI

AT89C51 menyediakan 5 sumber interupsi , yaitu 2 intrupsi eksternal , 2 interupsi pewaktu dan sebuah interupsi serial. Mikrokontroler-51(AT89C52) dari Atmel lainnya memiliki vektor dan sumber interupsi tambahan .

## 2.1 Pengaktifan Interupsi

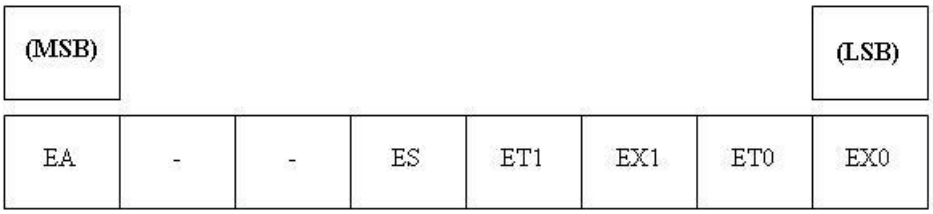
Masing-masing sumber interupsi dapat diaktifkan dan dimatikan secara individu atau dengan me-nol-kan bit-bit IE (*Interrupt Enable*) dalam SFR. Register IE ini juga untuk mengaktifkan atau mematikan interupsi secara keseluruhan (*global*). Pada (Gambar 2.1) dan (Tabel 2.1) ditunjukkan register IE pada AT89C51/52.

Jika isinya = 1, artinya bit aktif *(enable)*

Jika isinya = 0, artinya bit pasif (*disable*)

## 2.2 Prioritas Interupsi

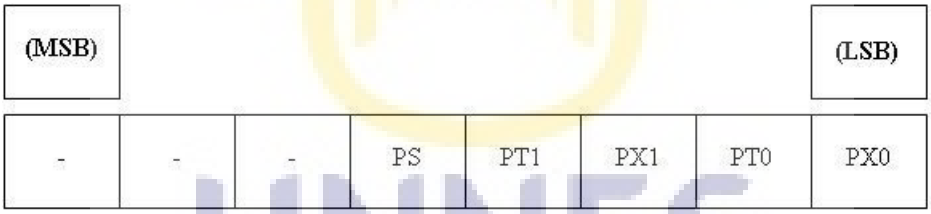
Masing-masing sumber interupsi dapat diprogram secara sendiri-sendiri ke salah satu dari dua tingkat prioritas dengan men-set (1) atau meng-clear (0) bit IP (*Interrupt Priority*) yang bersangkutan dalam SFR, perhatikan (Gambar 2.1) dan (Tabel .2.1).




Gambar 2.1: Register IE-Interrupt Enable pada AT89C51

Tabel 2.1: Fungsi Register IE-Interrupt Enable Pada AT89C51

| Simbol | Posisi | Fungsi |
|---|---|---|
| EA | IE.7 | Untuk menghidupkan (IE=1) dan mematikan (IE=0) seluruh interupsi secara serentak |
| - | IE.6 | Cadangan (*digunakan pada Mikrokontroler Atmel lainnya*) |
| - | IE.5 | Cadangan (*digunakan pada Mikrokontroler Atmel lainnya*) |
| ES | IE.4 | Bit aktivasi interupsi Port serial |
| ET1 | IE.3 | Bit aktivasi interupsi Timer 1 overflow |
| EX1 | IE.2 | Bit aktivasi interupsi external 1 |
| ET0 | IE.1 | Bit aktivasi interupsi Timer 0 overflow |
| EX0 | IE.0 | Bit aktivasi interupsi external 0 |




Gambar 2.2: Register IP-Interrupt Priority pada AT89C51

Jika isinya = 1 artinya bit prioritas tinggi (*high priority*)

Jika isinya = 0 artinya bit prioritas rendah (*low priority*)

Sebuah interupsi dengan prioritas-rendah dapat diinterupsi oleh berprioritas lebih tinggi, tetapi tidak untuk yang prioritasnya sama (sama

rendahnya). Sedangkan interupsi dengan tingkat prioritas tertinggi tidak dapat diinterupsi oleh sumber interupsi yang lain.Jika dua pemintaan interupsi berbeda prioritas muncul pada saat bersamaan, maka layanan ditujukan ke sumber interupsi memiliki tingkat prioritas yang lebih tinggi. Namun jika dua interupsi dengan tingkat prioritas yang sama muncul bersaman, maka metode polling digunakan sebagai penentu struktur prioritas kedua.

Tabel 2.2: Fungsi Register IP Interrupt Priority Pada AT89C51

| Simbol | Posisi | Fungsi |
|---|---|---|
| - | IP.7 | Cadangan (*digunakan pada Mikrokontroler Atmel lainnya*) |
| - | IP.6 | Cadangan (*digunakan pada Mikrokontroler Atmel lainnya*) |
| - | IP.5 | Cadangan (*digunakan pada Mikrokontroler Atmel lainnya*) |
| PS | IP.4 | Bit prioritas untuk interupsi Port serial |
| PT1 | IP.3 | Bit prioritas untuk interupsi Timer 1 |
| PX1 | IP.2 | Bit prioritas untuk interupsi external 1 |
| PT0 | IP.1 | Bit prioritas untuk interupsi Timer 0 |
| PX0 | IP.0 | Bit prioritas untuk interupsi external 0 |

## 2.3 Simulasi Tingkat Prioritas Ketiga Dalam Perangkat Lunak

Beberapa aplikasi sering membutuhkan lebih dari 2 tingkat prioritas interupsi yang telah disediakan oleh mikrokontroler. Pada kasus semacam ini, suatu prosedur sederhana dapat dituliskan dalam program yang efeknya seperti tingkat prioritas ketiga. Pertama, interupsi-interupsi yang membutuhkan prioritas lebih tinggi dari 1 diberikan prioritas 1 dalam register IP. Rutin-rutin layanan untuk interupsi-interupsi prioritas 1 yang

seharusnya dapat dinterupsi dengan prioritas 2 dituliskan dengan menambahkan kode-kode program sebagai berikut:

*PUST IE*
*MOV IE, # mask*
*CALL LABEL*
*.....*
*(eksekusi rutin layanan)*
*.....*
*POP IE*
*RET*
*LABEL : RETI*

Segera setelah sembarang interupsi prioritas 1 diterima dan dijawab, register IE didefinisi ulang untuk mematikan semua interupsi kecuali interupsi dengan prioritas 2. Kemudian instruksi CALL LABEL mengerjakan instruksi RETI, yang menon-aktifkan flip-flop interupsi yang sedang berjalan prioritas 1. Pada saat ini, sembarang interupsi dengan prioritas 1 dapat dilayani, tetapi hanya interupsi 2 saja yang diaktifkan.

Instruksi POP IE digunakan untuk mengembalikan byte asli dari IE, kemudian diikuti dengan RET ( bukan RETI ), yang digunakan untuk menghentikan rutin layanan. Kode-kode tambahan tersebut hanya menambah sekita 10 md (jika digunakan frekuensi 12 MHz) ke interupsi-interupsi dengan prioritas 1.

## 2.4 Reset

Merupakan interupsi istimewa, karena program tidak mengabaikan aksi tegangan dari rendah ke tinggi pada kaki RST, dimana perintah harus melompat ke alamat 0000 h.

# Bab 3

# PEMROGRAMAN

Pada pasal ini akan dijelaskan bagaimana cara menjalankan perangkat lunak dasar untuk membuat suatu program aplikasi mikrokontroler, meliputi:

A. Compiler ASM51.

B. Mengubah berkas objek ke heksa dengan 0H.

C. Menjalankan *emulator TS Controls Emulator 8051* untuk pengujian atau simulasi program.

## 3.1 Tata Cara Membuat Program Mikrokontroler AT89C51

(a) Program dibuat dalam bahasa assembler mikrokontroler yang bersangkutan , dalam hal ini bahasa dari keluarga AT89C51/52 diketik menggunakan sembarang editor teks( misalnya program EDIT pada MSDOS Prompt ), serta disimpan dengan ekstensi *.asm ;

(b) Melakukan kompilasi program yang telah diketik dengan perintah: Asm51 $< nama_file.asm >$

(c) Jika terjadi kesalahan akan ditunjukkan dan dapat diperbaikinya sebelum meneruskan ke tahap berikutnya. Jika ada kesulitan dalam menemukan

letak kesalahan tesebut, cobalah buka berkas dengan nama yang sama, hanya saja yang berektensi *.lst ; . Bila tidak terjadi kesalahan maka akan dihasilkan berkas objek yang kemudian dapat diubah ke format heksa pada langkah berikutnya.

(d) Program yang telah dikompilasi kemudian diubah ke format heksa dengan perintah : oh $< nama_file.obj >$

(e) Sampai di sini, sudah diperoleh berkas dengan ekstensi *.hex yang bisa dimanfaatkan dalam simulator TS Controls Emulator 8051 atau alat pemrograman AT89C51 (yang mengenal format heksa ), jika diperlukan berkas heksa ini bisa diubah menjadi format biner ( misalnya, intel binary) dengan perintah : h $< nama_file.hex >< nama_file.bin >$ i o.

(f) Untuk program Easy Programmer yang dibutuhkan adalah berkas dalam format intel heksa, sehingga cukup melakukan langkah hanya sampai berkas format heksa saja. Untuk pemrograman tipe lain ada yang membutuhkan berkas dengan format biner, sehingga harus melakukan proses hingga langkah yang terakhir sebagaimana dijelaskan sebelumnya.

## 3.2 Program Mikrokontroler AT89C51 (TS Controls tata cara mensimulasikan Emulator 8051)

. Setelah melakukan beberapa langkah tersebut, dapat menggunakan fasilitas lain untuk mempelajari instruksi-instruksi dari program mikrokontroler, misalnya dengan menggunakan suatu simulator perangkat keras (Emulator).Dengan emulator tersebut dapat diketahui jalannya program yang telah dibuat, selain itu juga dapat ditemukan beberapa tampilan untuk mengetahui isi, seperti:

a) Accumulator item Program counter

b) stack pointer

c) Register

d) Register-register lain yang digunakan dalam program.

Dalam hal ini kita akan mencoba menggunakan simulator AT89C51 yaitu *TS Controls Emulator 8051*.Langkah-langkah dalam menggunakan TS Controls Emulator 8051, sebagai berikut:

(1) Untuk membaca kode, pilih menu*Load Hex File* pada menu utama File. Kemudian akan ditampilkan jendela dialog , pastikan pada file contains yang dipilih adalah code. Pada*Boks Bank*, tuliskan nomor bank dimana kode-kode akan disimpan, ingat bahwa nomor bank harus ditulis dalam format heksa(nomor bank tidak diperlukan jika model memorinya adalah*tiny* atau *Tsmall*. Pada mode memori small, kode-kode defaultnya disimpan pada bank1. Pada *TBoks* File Name, pilih atau ketik nama-nama berkas kode. Berkas kode harus dalam format *Intel-Hex atau S-Record*.

(2) Kode-kode program akan ditampilkan pada jendela *Disassembled Code*.

(3) Jika listing program juga ingin ditampilkan pada jendela *Source Listing*, maka pilih menu *Load Source Listing File ( Ctrl+Shift + O )* pada menu File, kemudian pilihlah nama berkas yang sama dengan ekstensi *( *.lst) atau ( *.1)*.

(4) Untuk mengetahui jalannya program langkah-demi-langkah ( step-by-step) bisa dilakukan dengan menekan tombol ( F11 ) berulang-ulang. Perubahan yang terjadi bisa diikuti melalui jendela register atau jendela SF *(Special Function) Registers* atau jendela *Internal/ External RAM*.

(5) Jika ingin mengulangi jalannya program dari awal, lakukan reset dengan memilih menu *Run → Reset (Ctrl+Shift+F5)*.

## 3.3 Langkah-langkah Percobaan

1) Tuliskan contoh program berikut ini sesuai dengan tata-cara penulisan program !

- MOV A, P1
- Mula: Inc A
- MOV R0, A
- MOV P2, A
- SJMP Mulai
- End

2) Lakukan kompilasi serta ubah berkas objek ke dalam format heksa, kemudian panggil melalui Emulator, amati perubahan yang terjadi pada emulator tersebut saat anda menjalankan program *step by step* (dengan menekan F11 berulang-ulang ).

3) Catat hasilnya untuk 32 penekanan F11 dengan (Tabel.1.1) di bawah ini (dalam format heksadesimal).

Tabel 3.1: Data untuk pemrograman

| Langkah pemrograman | R0 | A | P1 | P2 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| ... | | | | |
| ... | | | | |
| 32 | | | | |

4) Kesimpulan program.

## 3.4 *Flow Chart*

Komputer melaksanakan perintah-perintah dalam urut-urutan logik yang sangat tepat. Hal inilah yang perlu dipertimbangkan dalam membuat suatu program.Bayangkan urutan kegiatan apa yang anda lakukan bila anda berkunjung ke suatu perpustakaan untuk meminjam buku. Untuk meminjam buku, pertama-tama anda harus pergi ke perpustakaan, tetapi sebelum itu

anda meninggalkan rumah, dan sebelum itu anda membuat keputusan untuk pergi ke perpustakaan dan mungkin harus membawa buku-buku untuk dikembalikan ke perpustakaan.

Pada kalimat-kalimat di atas , kegiatan-kegiatan dinyatakan dalam urutan yang salah. Tetapi anda pasti akan dapat memilih dan melakukan kegiatan mana yang seharusnya dilakukan terlebih dahulu.Hal ini anda lakukan berdasarkan intuisi yang terbentuk karena kebiasaan dan pengalaman-pengalaman masa lalu yang cukup panjang. Komputer tidaklah sepandai itu. Komputer harus diberi perintah-perintah yang urutannya haruslah benar-benar tepat, kalau tidak akan membuat kesalahan.

*Flow Chart* dapat membantu kita dalam membuat urutan perintah. Anda dapat memotong kegiatan yang harus dilaksanakan komputer. Anda akan membuat langkah-langkah logik sederhana dan menyusun dalam urutan yang tepat. Bayangkan kegiatan yang anda lakukan , bila anda ingin bepergian naik bis. Anda tahu jam berapa bis akan berangkat, jadi anda harus menentukan jam berapa anda harus meninggalkan rumah. Jika anda telah sampai di pemberhentian bis, anda harus menanti bis tersebut dan mengangkat tangan sebagai tanda akan naik bis, bila bis yang dimaksud telah datang.

Setelah anda berada dalam bis, anda mungkin harus menanyakan berapa harga karcis untuk perjalanan anda, kemudian andapun membayarnya. Setelah itu, anda perlu berjaga-jaga untuk turun di tempat tujuan. Terakhir anda turun meninggalkan bis tersebut bila telah sampai di tempat tujuan yang anda maksud. *Flow Chart* di bawah ini akan menggambarkan urut-urutan kejadian itu dalam bentuk diagram. Bentuk kotak berbeda-beda disesuaikan dengan tujuan atau fungsinya.

Macam-macam bentuk kotak dan tujuannya:

1) Lingkaran

   Lingkaran digunakan untuk menyatakan mulai dan akhir suatu proses. Simbol lingkaran berisi kata mulai, sering berisi nama suatu proses.

2) Segi empat

   Segi empat digunakan untuk pernyataan-pernyataan *(statement)* dimana perintah harus dilaksanakan.

3) Wajik

Bentuk kotak wajik ini digunakan untuk menyatakan suatu keputusan, apakah suatu tindakan harus dilaksanakan atau tidak. Proses naik bis sampai di tempat tujuan dinyatakan dinyatakan langkah demi langkah. Keputusan pertama yang harus diambil ialah apakah sudah waktunya untuk meninggalkan rumah. Jika belum, anda harus menunggu sampai tiba waktunya untuk meninggalkan rumah. Hal ini dinyatakan sebagai suatu loop pada *flow chart* kita, yaitu suatu lingkaran kegiatan yang harus dilakukan berulang-ulang sampai suatu syarat tertentu dipenuhi.

Dalam keputusan pertama *flow chart* kita ini, syarat tersebut adalah waktu. Bila waktu untuk meninggalkan rumah telah tercapai, anda pergi ke stasiun bis dan masuk ke proses *loop* berikutnya, yaitu proses menunggu apakah bis yang dimaksud telah datang atau belum. Proses-proses dalam flow chart haruslah jelas, hampir hampir tidak perlu keterangan lebih lanjut, dan dalam beberapa hal bentuk ini adalah bentuk yang paling baik untuk menjelaskan ide-ide baru termasuk penggunaan komputer.

## 3.5 Masalah dan pemecahannya

Langkah-langkah yang perlu diperhatikan dalam memecahkan masalah yang berkaitan dengan pemrograman, sebagai berikut:

1) Tentukan outputnya, yaitu jawaban hasil akhir apa yang anda kehendaki. Setelah itu kumpulkan input(informasi-informasi) yang tersedia. Bila anda telah menentukan input yang tersedia, kemudian tentukan apa yang anda kehendaki dari komputer(output).

2) Bayangkah langkah-langkah apa yang akan anda gunakan untuk menghasilkan jawaban yang anda kehendaki.

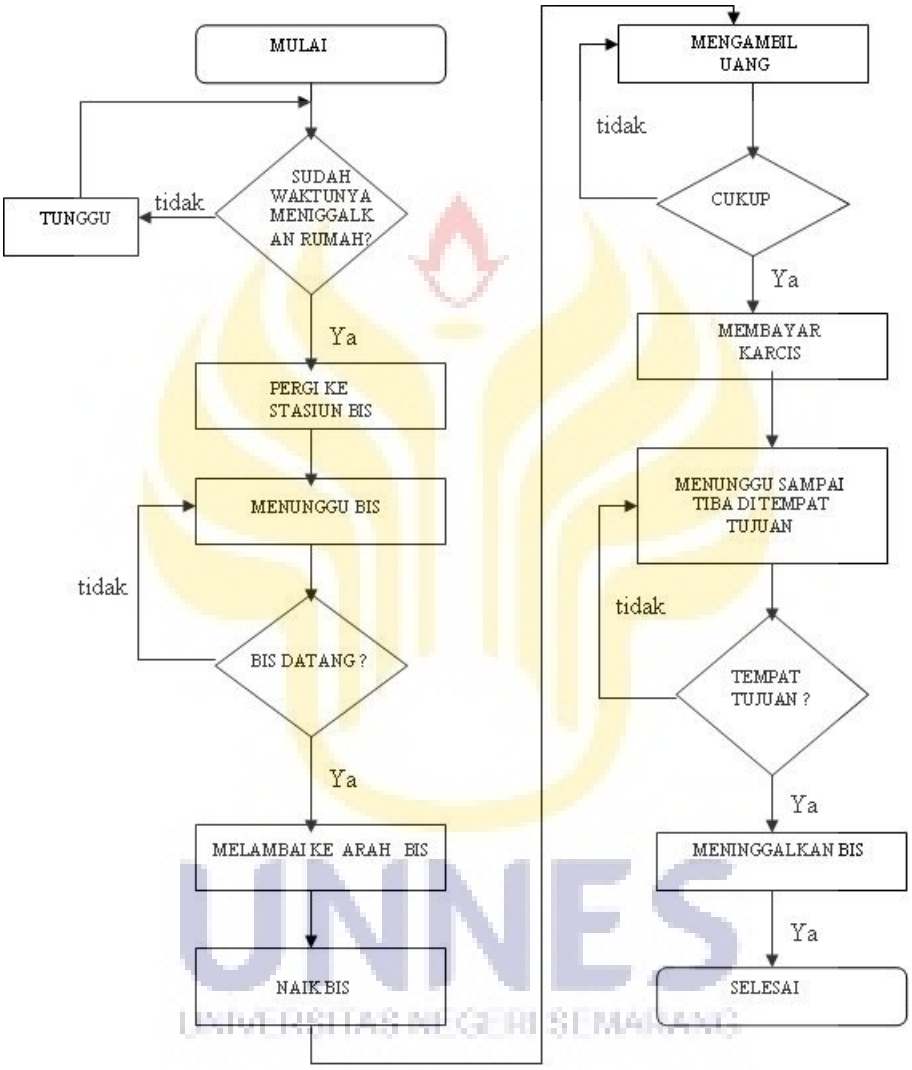
3) Tulis langkah-langkah tersebut dalam bentuk flow chart atau urutan-urutan logik

4) Konversikan ke dalam bahasa komputer

5) Masukkan hasil konversi ke dalam computer dan jalankan (RUN ) program tersebut.

6) Periksa (debug) dan betulkan bila ada kesalahan-kesalahan.

7) Simpan program untuk penggunaan berikutnya.




Gambar 3.1: *Flow Chart* bepergian dengan naik bis

Dengan menggunakan *flow chart* pada (Gambar 1.1), kita dapat menulis program dengan mudah yang berkaitan dengan bepergian naik bis secara . Hal

demikian hanya sekedar contoh, namun dapat diaplikasikan dalam bidang lain. Dengan demikian dapat di ambil kesimpulan bahwa dalam menyusun program seharusnya menentukan flow chart terlebih dahulu.

## 3.6 Program aplikasi untuk membuat kelompok 4 LED mati-hidup secara bergantian *(flip-flop)*

Program aplikasi untuk membuat kelompok 4 LED mati-hidup secara bergantian *(flip-flop)*, sebagai berikut:

```
1: ..... ;
2:; Lampu flip-flop pada port 3 File name Bab3.1.ASM
3: ..... ;
4: ORG 0H

5: MULAI : MOV P3, # 00001111B ..... ;LED P3.4 s/d P3.7 nyala
6: ACALL DELAY ..... ; Panggil subrutin Delay
7: MOV P3,# 11110000B ..... ; LED P3.0 s/d P3.3 nyala
8: ACALL DELAY ..... ; Panggil subrutin Delay
9: SJMP MULAI
10: ..... ;
11: ..... ; Sub-rutin delay
12: ..... ;
13: DELAY : MOV R0,#5 ..... ; Isi Register R0 dengan 5
14: DELAY 1 :MOV R1,#0FF ..... ; Isi Register R1 dengan FF(hex)
15: DELAY 2 :MOV R2,#0

16: DJNZ R2,$
17: DJNZ R1,DELAY 2 ./.... ; Kurangi R1 dengan 1,bila hasilnya belum
18: ..... ; belum = 0 maka lompat ke delay 2
```

*19: DJNZ R0,DELAY 1 ..... ; Kurangi R1 dengan 1,bila hasilnya belum*

*20: ..... ; belum = 0 maka lompat ke delay 1*

*21: RET ..... ; Kembali ke alamat setelah perintah*

*22: ..... ; Acall Delay*

*23: END*

Dari program di atas (4 LED mati-hidup secara bergantian) dapat dijelaskan, sebagai berikut:

1). Baris 1 s/d 3 merupakan komentar sekaligus berisi keterangan program.

2). Baris ke 4 digunakan agar instruksi dituliskan mulai alamat 0h.

3). Baris ke 5 mengirimkan data 0000 1111 B (biner) ke port 1 agar lampu LED 4 s/d LED 7 (berkaitan dengan P3.4 P3.7) menyala, kemudian diikuti dengan *ACALL DELAY.*

4). Mengerjakan tundaan ( baris ke 6 ), yaitu pada *sub-rutin Delay.*

5). Baris ke 7. mengirimkan data 1111 0000 B (biner) ke port 1 agar lampu LED 0 s/d LED 3 ( berkaitan dengan P3.0 P3.3 ) menyala, kemudian diikuti baris 5 dan mengerjakan tundaan lagi ( baris ke 8 ), yaitu pada sub-rutin delay.

6). Baris ke 9 proses diulang dengan instruksi SJMP.

7). Baris ke 13 s/d 21 merupakan *sub-rutin delay* , yang prosesnya digambarkan melalui flow chart yang ditunjukkan pada (gambar 1.1)

Perhitungan waktu dari sub-rutin program di atas: Pada baris ke-4 dikerjakan sebanyak 326.400 kali, karena instruksi tersebut selama 2 siklus, maka waktu totalnya 326.400 x 2 = 652.800 siklus, masih ditambah dengan pengulangan kedua (255 x 3 siklus ) = 765 siklus dan pengulangan yang ke-tiga, delay 1

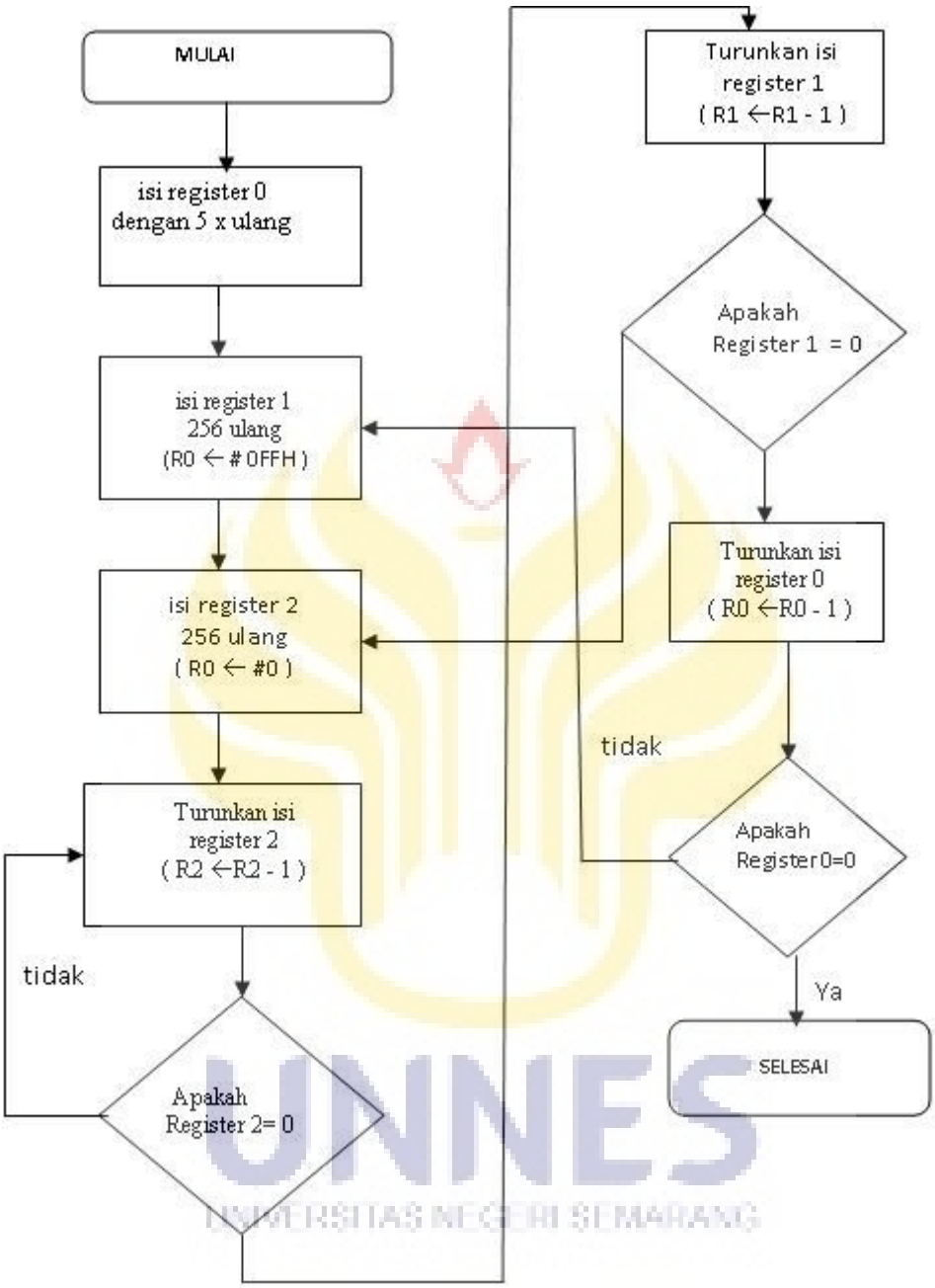
Tabel 3.2: Data untuk pemrograman

| | | | | dikerjakan | cycle |
|---|---|---|---|---|---|
| 1: | DELAY : | MOV | R0,#5 | 1 x | 1 |
| 2: | DELAY 1 : | MOV | R1,#0FF | 1 x | 1 |
| 3: | DELAY 2 : | MOV | R2,#0 | 1 x | 1 |
| 4: | | DJNZ | R2,$ | 256x255x5=326.400 x | 2 |
| 5: | | DJNZ | R1, DELAY 2 | 255 x | 2 |
| 6: | | DJNZ | R0,DELAY 1 | 5 x | 2 |
| 7: | | RET | | | |

sebesar 5 x 3 siklus = 15 siklus, sehingga total siklus = 652.800 + 765 + 15 = 653.580 silkus.

Jika menggunakan frekuensi kristal 12 MHz waktu yang dibutuhkan untuk menyelesaikan sub-rutin adalah 653.580 siklus x 1 md = 653.580 md = 0,65 detik untuk pewaktu yang lebih akurat bisa digunakan *timer*.
Keterangan dari program di atas, sebagai berikut:

- Delay adalah waktu tunda untuk mengeksekusi program selanjutnya.
- MOV*(move memory)*
  adalah suatu instruksi untuk menyimpan suatu konstanta ke dalam akumulator.
- DJNZ*(Decrement and Jump if Not Zero)*,
  merupakan instruksi yang akan mengurangi 1 nilai register serbaguna ( R0, R1,R2, R3, R4, R5, R6, R7 ) atau memori-data dan akan lompat ke memori program yang dituju, jika ternyata setelah pengurangan 1 tersebut hasilnya tidak sama dengan nol.
- RET *(Return from Subroutine)*
  adalah instruksi untuk kembali ke alamat setelah perintah.




Gambar 3.2: Diagram Alir *Sub-rutin Delay*

# Bab 4

# INSTRUKSI BAHASA ASSEMBLY AT89C51

## 4.1 ADD (*Add Accumulator*) dan ADDC (*Add Accumulator With Carry*)

Kedua instruksi ini selalu melibatkan Akumulator. Isi akumulator ditambah dengan suatu bilangan (dalam ukuran byte), maka hasil penjumlahan akan ditampung kembali ke akumulator. Dalam operasi ini *bit carry* (C flag dalam PSW berfungsi sebagai limpahan *(overflow)* hasil penjumlahan. Bilangan 1 byte yang ditambahkan ke akumulator, bisa berasal dari bilangan konstanta, register serbaguna, memori data yang lokasi memorinya disebut secara langsung maupun tidak langsung, seperti terlihat dalam contoh berikut:

ADD A, R0 *; melalui register serba guna R0.*

ADD A, #23h *; langsung dengan suatu bilangan (23h).*

ADD A,@R0 *; dengan isi lokasi memori yang ditunjuk R0.*

ADD A,P1 *; dengan isi register P1 (dari port 1).*

## 4.2 JUMP (Lompat)

Instruksi JUMP dapat digunakan atau dipakai untuk pengujian nilai Bolean, mengatur alur program dan memantau nilai akumulator A.

### 4.2.1 Pengujian Nilai Bolean

Pengujian ini dilakukan dengan instruksi JUMP bersyarat, ada 5 instruksi yang digunakan,yakni:

1) JB *(JUMP if Bit Set)* Instruksi akan lompat ke suatu lokasi memori-program, jika bit yang diperiksa bernilai 1 . Contoh : P1.1,$
   Instruksi ini memantau/ memeriksa Port 1 bit 1, jika P1.1 bernilai 1 akan mengulang instruksi ini. Tanda $ mempunyai arti , jika syarta terpenuhi kerjakan lagi instruksi yang bersangkutan.

2) JNB *(JUMP if Bit Not Set)* Instruksi akan lompat ke suatu lokasi memori-program, apabila bit yang diperiksa hasilnya bernilai 0 . Contoh : P1.1, $ Instruksi ini akan memantau atau memeriksa Port 1 bit 1, apabila P1.1 mempunyai nilai 0 akan mengulang instruksi ini. Tanda $ mempunyai arti , jika syarat terpenuhi kerjakan lagi instruksi yang bersangkutan.

3) JC *(JUMP if Carry Bit Set)* Instruksi akan lompat ke suatu memori proram jika ternyata bit carry bernilai 1 (perlu diingat bahwa bit carry = PSW.7).

4) JNC *(JUMP if Carry Bit Not Set)* Instruksi akan lompat ke suatu memori-proram jika ternyata bit carry mempunyai nilai 0 (per diingat bahwa bit carry= PSW.7).

5) JBC *(JUMP if Bit Set and Clear Bit).* Instruksi JBC sama dengan JB, hanya saja jika ternyata bit yang yang diperiksa bernilai 1 , selain MC51 akan lompat (JUMP) ke instruksi lain yang dikehendaki, MC51 akan me-nol-kan bit yang baru saja diperiksa tersebut.

### 4.2.2 Mengatur alur program

Secara umum kelompok instruksi program yang dipakai untuk mengatur alur program terdiri atas instruksi JUMP , ada 3 instruksi yang digunakan, yaitu :

1) LJMP (Long Jump) Pemakain instruksi LJMP dapat dipelajari dari potongan program sebagai beikut:

```
LJMP        TugasBaru
.....
ORG         2000h
TugasBaru :
MOV A, P3.1
```

ORG adalah instruksi agar program disimpan pada lokasi yang disebut dibelakang ORG (dalam hal ini lokasi 2000 h). TugasBaru disebut sebagai label, dengan demikian memori-program lokasi 2000 h diberi nama TugasBaru, atau juga dapat dikatakan TugasBaru bernilai 2000 h. Label ditulis minimal satu huruf lebih kiri dari instruksi. Dengan demikian instruksi LJMP TuigasBaru tersebut sama artinya LJMP 2000h yang oleh assembler akan diterjemahkan menjadi 02 20 00 (heksadesimal).

2) AJMP *(Absolute Jump)* Instruksi AJMP terdiri atas 2 byte, byte pertama merupakan kode untuk instruksi AJMP (00001b ) yang digabung dengan lokasi memori-program bit lokasi 8 sampai dengan bit lokasi 10, byte kedua dipakai untuk menyatakan lokasi memori program bit lokasi 0 smpai dengan bit lokasi 7.
Pemakain instruksi LJMP dapat dipelajari dari potongan program sebagai berikut :

```
ORG         800h
AJMP        DaerahIni
AJMP        DaerahLain
ORG         900h
DaerahIni:
.....
ORG         1000h
DaerahLain :
.....
```

Potongan program dimulai dari memori-program lokasi 800h, dengan demikian instruksi AJMP Daerah Ini bisa dipakai, karena lokasi memori 800h (tempat instruksi AJMP Daerah Ini) dan Label DaerahIni terletak di dalam satu daerah memori program 2 Kilo Byte yang sama. Sedangkan instruksi SJMP DaerahLain tidak bisa dipakai, karena memori program terletak di daerah memori-program 2 Kilo Byte yang lain. Hal ini disebabkan karena jangkauan instruksi AJMP hanya 2 Kilo Byte.

3) SJMP *(Short Jump)* Meskipun jangkauan instruksi SJMP hanya -128 sampai + 127, tapi instruksi ini tidak dibatasi dengan pengertian daerah memori-program 2 Kilo Byte yang membatasi instruksi AJMP. Pemakain instruksi SJMP dapat dipelajari dari potongan program sebagai berikut:

```
ORG         0F8h
SJMP        DaerahLain
ORG         1000h
DaerahLain:
```

Dalam potongan program di atas, memori-program 0F80h tidak terletak dalam daerah memori program 2 Kilo Byte yang sama dengan 1000h, tapi instruksi SJMP DaerahLain tetap bisa dipakai, asalkan jarak antara instruksi itu dengan label DaerahLain tidak lebih dari 127 byte.

### 4.2.3 Instruksi JUMP bersyarat yang fungsinya memantau nilai Akumulator A

Instruksi JUMP bersyarat yang fungsinya memantau nilai Akumulator A,ada dua macam, yaitu:

a) JZ *(JUMP if Zero)* Instuksi akan dijalankan , jika nilai Akumulator A = 0

b) JNZ *(JUMP if Not Zero)* Instruksi akan dijalankan , jika nilai Akumulator A tidak sema dengan nol, karenaa syaratnya selalu dipenuhi. Perhatikan contoh dari potongan program berikut ini :

*MOV A,# 0*
*JNZ BukanNol*
*JZ Nol*
. . .
*BukanNol :*
. . .
*Nol :*
. . .

Dalam contoh di atas, *MOV A, #0* menjadikan akumulator A berisi nol. Hal ini mengakibatkan instruksi JNZ BukanNol tidak pernah dikerjakan, sedang saat itu nilai akumulator = 0 selalu dikerjakan , karena syaratnya selalu dipenuhi.

## 4.3 CALL

CALL adalah mnemonik untuk instruksi call subroutine ( panggil sub-rutin). Setiap instruksi CALL harus dilengkapi alamat-awal dari sub-rutin yang dikehendaki.

## 4.4 RET *(Return )*

RET adalah mnemonik untuk instruksi return ( kembali), ini digunakan pada akhir setiap sub-rutin , agar kembali ke program semula.

## 4.5 RETI

RETI suatu instruksi dipakai untuk mengakhiri Rutin Layanan Interupsi *(Interupsi Service Routine)*, yaitu semacam program sub-rutin yang dijalankan mikrokontroler pada saat menerima sinyal permintaan interupsi. Sub-rutin merupakan sekumpulan instruksi yang karena berbagai pertimbangan dipisahkan dari program utama.

## 4.6 ACALL *(Absolute Call)*

Instruksi ACALL dipakai untuk memanggil program sub-rutin dalam daerah memori-program 2 KiloByte yang sama, setara dengan instruksi AJMP.

## 4.7 LCALL *(Long Call)*

Instruksi LCALL setara dengan instruksi LJMP, yang bisa menjangkau seluruh memori-progam sebanyak 64 KiloByte. (Tapi tidak ada instruksi *SCALL* yang setara dengan instruksi *SJMP*). Program untuk AT89C1051 dan AT89C2051 tidak perlu menggunakan instruksin *LCALL*.

## 4.8 Instruksi Transfer Data yang mengakses Ruang Memori Data Internal

Tabel 4.1: Instruksi Transfer Data yang mengakses Ruang Memori Data Internal

| Mnemonik | Operasi |
|---|---|
| MOV A,< *sumber* > | A = < *sumber* > |
| MOV < *tujuan* >, A | < *tujuan* >= A |
| MOV < *tujuan* >,< *sumber* > | < *tujuan* >= sumber |
| MOV DPTR,#data16 | DPTR = konstanta 16-bit |
| PUSH < *sumber* > | INC SP<br>MOV @SP,< *sumber* > |
| POP < *tujuan* > | MOV< *tujuan* >, @SP<br>EC SP |
| XCH A,< *byte* > | Tukar data byte<br>antara ACC dan < *byte* > |
| XCHD A,@Ri | Tukar data nilai rendah<br>antara ACC dan @Ri |

Keterangan:

(1) Instruksi MOV mengirimkan data dari register ke register.

(2) Instruksi PUSH pertama kali akan menaikkan isi SP (*Stack Pointer*) kemudian menyalin data byte yang terkait ke lokasi yang ditunjuk SP.

(3) Instruksi POP Pertama kali menyalin data dalam lokasi yang ditujunk SP ke data byte yang terkait, kemudian akan mengurangi SP.

(4) Instruksi XCH A,¡byte¿ digunakan untuk menukar data yang tersimpan dalam akumulator dengan data di lokasinnya yang ditunjuk oleh $< byte >$.

(5) Instruksi XCHD A,@Ri fungsinya sama dengan XCH, namun hanya melakukan penukaran data nible (4-bit) rendah saja, antara akumulator dan lokasi memori yang ditunjuk ole Ri.

## 4.9 Instruksi Transfer Data yang mengakses Memori Data External

Semua instruksi transfer data yang mengakses memori data external dikerjakan dalam waktu 2 mikrodetik dengan asumsi frekuensi clocknya 12 MHz. Perlu dicatat bahwa semau akses data eksternal, akumulator selalu digunakan baik sebagai tujuan maupun sumber. Sinyal atau tanda baca dan tulis ke RAM eksternal diaktifkan selama eksekusi instruksi MOVX.

Tabel 4.2: Instruksi transfer data mengakses memori data external

| Lebar alamat | Instruksi | *Fungsi* |
|---|---|---|
| 8 bit | MOVX A, @Ri | *Baca RAM Eksternal lokasi Ri* |
| 8 bit | MOVX @Ri, A | *Tulis RAM Eksternal lokasi Ri* |
| 16 bit | MOVX A, @DPTR | *Baca RAM Eksternal lokasi DPTR* |
| 16 bit | MOVX @DPTR, A | *Tulias RAM Eksternal lokasi DPTR* |



## 4.10 Tabel-Tengok *(Lookup Table)*

Pada (Tabel 2.3) ditunjukkan dua instrukssi yang dapat digunakan untuk membaca tabel-tengok yang tersimpan dalam memori program. Karena kedua

instruksi ini hanya mengakses memori program saja , maka tabel-tengok hanya bisa dibaca saja, tidak mungkin diremajakan atau update. Jika akses tabelnya ke memori program eksternal, maka tanda bacanya melalui kaki $\overline{PSEN}$.

Tabel 4.3: Instruksi membaca Tabel-Tengok

| Instruksi | Fungsi | Waktu eksekusi |
|---|---|---|
| MOVC A,@A+DPTR | *Baca memori program* di lokasi (A+DPTR) | 2 $\mu$ s |
| MOVC A,@A+PC | *Baca memori Program* di lokasi (A+PC) | 2 $\mu$ s |

Keterangan:

- Instruksi MOVC A,@A+DPTR digunakan untuk menyalin entri yang dikehendaki ke dalam akumulator.
- Instruksi MOVC A,@A+DPTR digunakan untuk menyalin entri yang dikehendaki ke dalam akumulator, hanya saja menggunakan PC sebagai alamat dasarnya (bukan DPTR).

## 4.11 Instruksi SUBB *(Substract From Accumulator With Borrow)*

Isi akumulator A dikurangi dengan bilangan (1 byte) beserta dengan nilai bit Carry, hasil pengurangan akan ditampung kembali dalam akumulator. Dalam operasi ini Carry juga berfungsi sebagai penampung limpahan hasil pengurangan. Jika hasil pengurangan melimpah (nilainya kurang dari 0),maka bit Carry akan bernilai 1, jika tidak, maka bit carry berisi 0 . Contoh:

*SUBB A, R0 ; A = A R0 C*

*SUBB A, #23h ; A = A 23h*

*SUBB A, @R1 ; A = A [R1] (artinya [R1] isi dari R1)*

*SUBB A, P0 ; A = A P0*

## 4.12 Instruksi DA A *(Decimal Adjust)*

Instruksi DA A digunakan setelah instruksi ADD, ADDC atau SUBB untuk merubah nilai biner 8 bit yang tersimpan dalam Akumulator menjadi 2 digit decimal dalam format BCD*(Binary Coded Decimal)*.

## 4.13 Instruksi MUL AB

Bilangan biner 8 bit dalam akumulator A dikalikan dengan biner 8 bit dalam register B. Hasil perkalian berupa bilangan biner 16 bit, 8 bit yang bagian atas *(high byte)* disimpan di register B, sedangkan b bit lainnya *(low byte)* disimpan di akumulator A.
Bit OV dalam PSW ( Program Status Word ) dipakai untuk menandai nilai hasil perkalian yang ada dalam register B. Bit OV akan bernilai 0 jika register B bernilai 00h, jika tidak, maka bit OV selalu bernilai 1 . Contoh :

*MOV A, #10*

*MOV B, # 20*

*MUL AB*

*OV : Overflag, akan aktif bila terjadi overflow pada operasi terkait.*



## 4.14 Instruksi DIV AB

Bilangan biner 8 bit akumulator A dibagi dengan bilangan 8 bit dalam register B. Hasil pembagian berupa bilangan biner 8 bit ditampung di akumulator A, sedang sisa pembagian berupa bilangan biner 8 bit ditampung di register B.

Bit OV dalam PSW *(Program Status Word)* dipakai untuk menandai nilai pembagian yang ada dalam register B. Bit OV akan bernilai $'1'$, jika register asalnya bernilai $'0'$.

## 4.15 Instruksi DEC dan INC

Instruksi DEC ( Decrement Register ) digunakan untuk menurunkan nilai ( 1 byte ), yang tersimpan dalam salah satu dari 4 macam: akumulator, register, nilai langsung atau tak langsung melalui register, sebesar 1. Jika nilai awal 00h , maka setelah dilaksanakan instruksi ini, hasilnya adalah FFh. Tidak *flag* yang terpengaruh.

Sedangakan Instruksi INC *(Increment Register)* digunakan untuk menaikkan nilai (1 byte) sebesar 1, jika nilai awal FFh maka hasilnya adalah 00h. Jika instruksi ini digunakan untuk mengubah keluaran Port, maka nilai awalnya dari *Port* yang bersangkutan berasal dari pengancing data keluaran, bukan pin-pin masukan. Contoh dalam 4 kemungkinan instruksi DEC dan INC:

*DEC A ; artinya A = A 1*

*DEC Ri ; artinya Ri = Ri -1 ( Ri bisa R0 s/d R7 )*

*DEC #50h ; hasilnya 4Fh*

*DEC @Ri ; artinya [Ri] = [Ri] 1 ( Ri bisa R0 s/d R7 )*

*INC A ; artinya A = A + 1*

*INC Ri ; artinya Ri = Ri + 1 ( Ri bisa R0 s/d R7 )*

*INC #50 ; hasilnya 51 h*

*INC @Ri ; artinya [Ri] = [Ri] + 1 ( Ri bisa R0 s/d R7 )*

## 4.16 Instruksi INC DPTR

Agak berbeda dengan instruksi (INC dan DEC), instrukksi INC DPTR adalah satu-satunya instruksi penaikan *( increment )* yang bekerja pada data 16 bit, yaitu DPTR. Hal ini berarti menaikkan data sebesar 1. Suatu limpahan pada bit rendah *(low order)* dari DPTR atau DPL akan menaikkan byte tinggi ( high order ), yaitu yang tersimpan di DPH sebesar 1. Tidak *flag* terpengaruh. Misalnya: DPH = 12h dan DPL = FEh, maka instruksi ini:

*INC DPTR*

*INC DPTR*

*INC DPTR*

Akan menghasilkan DPH = 13 h dan DPL = 01 h

DPTR: (DPH dan DPL) *pointer* data untuk mengakses data eksternal memori sebagai *pointernya.*

## 4.17 SETB (*Set Bit*)

Instruksi ini dipakai untuk memberi nilai $'1'$ pada data 1 bit.

## 4.18 CLR (*Clear Bit*)

Instruksi ini dipakai untuk memberi nilai $'0'$ pada data 1 bit.
Contoh pemakaian instruksi SETB dan CLR, sebagai berikut :

*SETB ACC,0*

*SETB E0H*

*CLR P1.1*

*CLR $ 90*

Instruksi SETB ACC, 0 membuat bit 0 dari akumulator(ACC,0) bernilai $'1'$, akan tetapi mengingat lokasi dari bit 0 akumulator adalah E0 h, maka hasil kerja kedua instruksi SETB tersebut adalah sama. Demikian pula dengan kedua instruksi CLR berikutnya, instruksi-instruksi ini akan mengakibatkan P1.1 bernilai $'0'$.

## 4.19 Instruksi ANL (AND Logical), ORL (OR Logical), CPL (Complement Bit) dan EX-OR (Exlusive-OR Logical)

Untuk MC51 hanya bisa melaksanakan tiga jenis operasi logika yang ada, yakni : ANL, ORL dan CPL.Bit Carry pada PSW diperlukan sebagai *akumulator bit*, dengan demikian operasi AND dan operasi OR dilakukan antara bit yang tersimpan pada bit Carry dengan salah satu dari 256 bit data. Contoh dari instruksi- instruks, sebagai berikut: CPL akan membalik nilai biner dalam bit

| Instruksi | Fungsi Instruksi |
|---|---|
| ANL C,P1.1 | *Meng-AND-kan nilai pada bit Carry dengan nilai Port 1 bit 1(P1.1).* |
| ANL C, /P1.2 | *Sebelum meng-AND-kan nilai pada bit Carry dengan nilai Port 1 bit 1(P1.2), terlebih dahulu nilai dari P1.2 di-NOT-kan.* |
| ORL C,P1.1 | *Meng-OR-kan nilai pada bit Carry dengan nilai Port 1 bit 1.* |
| CPL P1.0 | *P1.0 di-NOT-kan.* |

Carry (jangan lupa bit Carry merupakan salah satu bit yang ada dalam 256 bit 0), yakni: bit lokasi E7 h atau PSW 7.

## 4.20 ICJNE*(Compare and Jump if Not Equal )*

Instruksi CJNE membandingkan dua nilai yang disebut dan MCS akan lompat ke memori program yang dituju, jika kedua nilai tersebut tidak sama. Contoh dalam aplikasinya:

*MOV A, P1*
*CJNE A, # 0Ah, Tidaksama*
*.....*
*SJMP EXIT*
*;*
*Tidaksama:*
*.....*

Instruksi MOV A, P1 membaca nilai masukan dari Port 1. Instruksi CJNE A, #oAh, *Tidak sama* memeriksa apakah nilai Port 1 sudah disimpan A sama dengan 0Ah atau tidak, jika tidak maka lompat ke *Tidak sama*.

# Bab 5

# APLIKASI PORT SERIAL

Ada 2 macam cara pengiriman (transmisi) data secara serial. Kedua cara dibedakan oleh sinyal detak yang digunakan untuk mendorong data serial , kalau detak dikirim bersama-sama dengan data serial, cara tersebut dikatakan sebagai transmisi data serial secara sinkron. Sedangkan dalam transmisi data serial secara asinkron, detak tidak dikirim bersama-sama data serial, rangkaian penerima data harus membangkitkan sendiri detak pendorong data serial.

## 5.1 Antarmuka Serial

Port Serial pada *AT89C51/52* bersifat dupleks-penuh atau full-duplex, artinya serial biasa menerima dan mengirim secara bersamaan. Selain itu juga memiliki penyangga penerima, artinya port serial mulai bisa menerima byte yang kedua sebelum byte yang pertama dibaca oleh register penerima. Catatan : Jika sampai byte yang kedua selesai diterima sedangkan byte pertama belum juga dibaca, maka salah satu byte akan hilang. Penerimaan dan pengiriman data port serial melalui register SBUF.

Penulisan ke SBUF berarti mengisi register pengiriman SBUF, sedangkan pembacaan dari SBUF berarti membaca register penerimaan SBUF. Anatara pengiriman dan penerimaan SBUF memang terpisah secra fisik(secara perangkat lunak namanya menjadi satu, yaitu SBUF). Port Serial pada *AT89C51/52* bisa digunakan dalam 4 mode kerja yang berbeda. Dari 4 mode tersebut, 1

mode diantaranya bekerja secara sinkron dan 3 mode lainnya bekerja secara asinkron.

### 5.1.1 Mode 0

Mode ini bekerja secara sinkron, data serial dikirim dan diterima melalui kaki *P3.0 (Rx D)*, sedangkan kaki*P3.1(Tx D)* digunakan untuk menyalurkan detak pendorong data serial yang dibangkitkan oleh AT89C51. Data dikirim/ diterima 8 bit sekaligus, dimulai dari bit yang bobotnya paling kecil atau *LSB (bit 0)* dan diakhiri dengan bit yang bobotnya paling besar atau*MSB (bit 7)*. Kecepatan pengiriman data *(baut rate)* adalah 1/12 frekuensi kristal yang digunakan.

### 5.1.2 Mode 1

Pada mode ini tetap, yaitu data dikirim melalui kaki *P3.1 (TxD)* dan diterima melaui kaki *P3.0 (RxD)*, secara asinkron (juga pada mode 2 dan mode 3). Mode 1 data dikirim/ diterima 10 bit sekaligus, diawali dengan bit start, disusul dengan 8 bit data yang dimulai dari bit yang bobotnya paling kecil *(bit 0)* , dan diakhiri dengan bit stop. Pada AT89C51/52 yang berfungsi sebagai penerima bit stop adalah RB8 dalam register SCON. Kecepatan pengiriman data *( baut rate )* bisa diatur sesuai dengan keperluan. Mode inilah (termasuk mode 2 dan mode 3) yang umum dikenal sebagai UART*(Universal Asynchronous Receiver/Transmitter)*.

### 5.1.3 Mode 2

Data dikirm/diterima 11 bit sekligus, diawali dengan bit start, disusul 8 bit data yang dimulai dari bit yang bobotnya paling kecil *(bit 0 )*, kemudian bit ke 9 yang bisa diatur lebih lanjut, diakhiri dengan *bit stop*. Pada AT89C51/52 yang berfungsi sebagai pengirim, *bit 9* tersebut berasal dari *bit TB 8* dalam register SCON. Pada AT89C51/52 yang berfungsi sebagai penerima , bit 9 ditampung pada *bit RB8* dalam register SCON. Sedangkan *bit stop* diabaikan tidak ditampung. Kecepatan pengiriman data *(baut rate)* bisa dipilih antara *1/32* atau *1/64* frekuensi kristal yang digunakan.

### 5.1.4 Mode 3

Mode ini sama dengan mode 2, hanya saja kecepatan pengiriman data *(baut rate)* bisa diatur sesuai dengan keperluan, seperti halnya mode 1.

## 5.2 Register Kontrol Port Serial

Registrasi control dan status untuk *Port Serial* berada dalam SCON (perhatikan gambar 1.1.). Register ini mengandung bit-bit pemilihan mode kerja *port serial*, bit data ke-9 pengiriman dan penerimaan *(TB8 dan RB8)* serta *bit-bit interupsi port serial (TI dan RI )*

*SCON Serial Port Control Register 98#*


Gambar 5.1: Susunan Bit dalam SCON

Keterangan :

### 5.2.1 Bit SM0 dan bit SM1

Bit *SM0 dan bit SM1 (bit 7 dan bit 8 pada register SCON)* dipakai untuk menentukan mode kerja *port serial*. Setelah reset kedua bit ini bernilai 0 dan penentuan mode kerja port serial mengikuti tabel 8.

Tabel 5.1: Penentuan Mode Kerja Port Serial

| SM0 | SM1 | Mode | Keterangan | Baut Rate |
|---|---|---|---|---|
| 0 | 0 | 0 | Register geser | Tetap (fosc/12) |
| 0 | 1 | 1 | UART 8-bit | Bisa diubah (dengan Timer) |
| 1 | 0 | 2 | UART 9-bit | Tetap (fosc/64 atau fosc/32) |
| 1 | 1 | 3 | UART 9-bit | Bisa diubah (dengan Timer) |

### 5.2.2 Bit REN

*Bit REN ( bit 4 )* dipakai untuk mengaktifkan kemampuan port serial untuk menerima data. Pada mode 0 kaki *RxD (P3.0)* dipakai untuk mengirim data serial dan juga untuk menerima data serial. Sifat ini terbawa pula pada saat port serial bekerja pada mode 1, mode 2 dan mode 3. Meskipun pada mode-mode tersebut kaki RxD hanya dipakai untuk mengirim data. Agar kaki RxD bisa dipakai untuk menerima data, terlebih dahulu harus dibuat $REN = 1$ . Setelah reset *bit REN* bernilai 0 .

### 5.2.3 Mode 2 dan mode 3

Pada mode 1, *RB8 dipakai untuk menampung bit stop yang diterima*, dengan demikian apabila RB8 bernilai 1 , maka data diterima dengan benar, sebaliknya apabila *RB8 bernilai 0* berarti terjadi kesalahan frame *(framing error)*. Kalau *bit SM2 (bit 5) bernilai 1* pada mode 1, jika terjadi kesalahan frame. RI tidak akan menjadi 1 (aktif) meskipun SBUF sudah berisi data dari *port serial ( bit stop diterima dengan benar)*.

Bit ke-9 ini disa digunakan sebagai bit paritas, hanya saja bit paritas yang dikirim harus ditentukan sendiri dengan program dan diletakkan pada *TB8 dan bit paritas* yang diterima pada *RB8* dipakai untuk menentukan integritas data secara program pula. Tidak seperti dalam *UART standart*, semuanya dikerjakan olah perangkat keras dalam *IC UART*.

### 5.2.4 Bit TI (bit 1)

Bit TI (bit 1) merupakan sinyal yang setara dengan sinyal THRE(Transmitter Holding Register Empty) yang umum dijumpai pada UART standart. Setelah port serial selesai mengirim data yang terssimpan dalam SBUF, bit TI akan bernilai 1 dengan sendirinya, kemudian bit ini harus di-nol-kan dengan program agar bisa dipakai untuk memantau keadaan SBUF dalam pengiriman data berikutnya.

Contoh : *Sub-rutin SerialOut* berikut digunakan untuk mengirim data serial (untuk semua mode port serial). Baris 2 menunggu *TI menjadi 1 (aktif)* untuk memastikan pengiriman data yang sebelumnya sudah selesai. Data yang

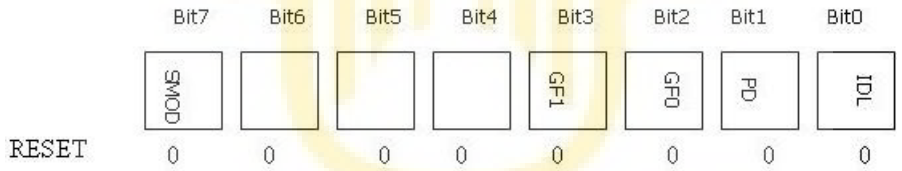
akan dikirim sebelumnya disimpan di A pada baris 3, data tersebut dikirim malalui port serial, menggunakan cara penyalinan ke *SBUF*. Agar TI bisa dipakai untuk memantau keadaan SBUF pada pengiriman data berikutnya, maka pada *baris 4, TI di-nol-kan.*

```
1 : Serial Out :
2 : JNB TI,$          ; tunggu data sebelumnya selesai dikirim.
3 : MOV SBUF,A        ; kirim data baru'
4 : CLC TI            ; sinyal ada pengiriman baru.
5 : RET
```

### 5.2.5 Bit RI (bit 0)

Bit RI (bit 0) merupakan sinyal yang setara dengan sinyal RDA *(Receiver Data Available)* yang umum dijumpai pada *UART standard*. Setelah SBUF menerima data dari *port serial*, bit RI akan bernilai 1 dengan sendirinya, bit ini harus di-nol-kan dengan program agar bisa dipakai untuk memantau keadaan SBUF dalam peberimaan data berikutnya.

*PCON Power Control Register 87H*




Gambar 5.2: Bit SMOD dalam Register PCON

Contoh : *Sub-rutin SerialIN* berikut dipakai untuk menerima data serial, (untuk semua mode port serial). Baris 2 menunggu RI menjadi 1 (aktif), untuk memastikan sudah ada data baru yang diterima pada SBUF. Pada baris 3 data pada SBUF diambil ke A. Agar RI bisa dipakai untuk memantau keadaan SBUF pada pengiriman data berikutnya, *pada baris 4 RI, di-nol-kan.*

```
01 : SerialIn :
02 : JNB RI,$          ; tunggu SBUF berisi data baru
```

*03 : MOV A, SBUF ; ambil data*
*04 : CLR RI ; pertanda data sudah diambil*
*05 : RET*

## 5.3 Baut Rate

Baut Rate untuk mode 0 nilainya tetap dan mengikuti persamaan :

$$BautRate = \frac{(FrekuensiKristal)}{12}$$

Baut rate untuk mode 2 bergantung pada nilai bit SMOD pada register PCON.

Jika SMOD = 0, maka

$$bautratenya = \frac{1}{64}(FrekuensiKristal)$$

Jika SMOD=1, maka

$$BautRate = \frac{1}{32}(FrekuensiKristal)$$

Baut rate untuk Mode 2 mengikuti persamaan sebagai berikut :

$$BautRate = \frac{2^{SMOD}}{64}(FrekuensiKristal)$$

Mikrokontroler AT89C51, laju limpahan Timer 1 menentukan baud rate Mode 1 dan Mode 3. Sedangkan AT89C52, baut rate ini dapat ditentukan menggunakan Timer 1, Timer 2 atau keduanya *(satu untuk pengiriman dan satunya untuk penerima).*

## 5.4 Pengunaan Timer 1 Menghasilkan Baut Rate

Pengunaan Timer 1 untuk Menghasilkan Baut Rate. Timer 1 digunakan sebagai generator baut rate, maka baut rate Mode 1 dan Mode 3 ditentukan

berdasarkan laju limpahan Timer 1 dan nilai SMOD dengan persamaan :

$$BautRate1dan3 = \frac{2^{SMOD}}{64}\frac{(FrekuensiKristal)}{12x[256-(TH_1)]}$$

Bisa diusahakan baut rate yang sangat rendah (lambat) dengan cara membiarkan interupsi Timer 1 tetap aktif dan dikonfigurasikan sebagai pewaktu 16-bit ( nible atas dari TMOD = 0001B) dan menggunakan Interupsi Timer1 itu sendiri untuk mengisi-ulang pewaktu 16-bit tersebut. Pada (Tabel 5.2.) ditunjukkan baut rate yang sering digunakan dengan menggunakan Timer 1.

Tabel 5.2: Baut Rate Sering Digunakan yang Dihasilkan Timer 1

| | | | Timer | | |
|---|---|---|---|---|---|
| Baur Rate | Fosc | SMOD | C/ T | Mode | Isi-ulang |
| Mode 0 Max:1MHz | 12 MHz | X | X | X | X |
| Mode 2 Max: 375k | 12 MHz | 1 | X | X | X |
| Mode 1 dan 3:62,5k | 12 MHz | 1 | 0 | 2 | FFH |
| 19,2K | 11,059 MHz | 1 | 0 | 2 | FDH |
| 9,6K | 11,059 MHz | 0 | 0 | 2 | FDH |
| 4,8K | 11,059 MHz | 0 | 0 | 2 | FAH |
| 2,4K | 11,059 MHz | 0 | 0 | 2 | F4H |
| 1,2K | 11,059 MHz | 0 | 0 | 2 | E8H |
| 137,5 | 11,059 MHz | 0 | 0 | 2 | 1DH |
| 110 | 6 MHz | 0 | 0 | 2 | 72H |
| 110 | 12 MHz | 0 | 0 | 1 | FEEBH |

Misal dengan $SMOD = 0$, digunakan baut rate 1200, maka limpahan timer 1 dihitung, sebagai berikut:

$$1200 = \frac{2^o}{32}(LajulimpahanTimer1)$$

Laju limpahan Timer1 = 32 X 1.2000 = 38.400 Hz. Jika digunakan kristal 12 MHz, maka timer 1 di detak dengan laju 1 MHz atau 1.000 KHz, maka limpahan harus terjadi pada tiap :

$$\frac{1000}{38,4} = 26,04(detik)$$

26,04 (dibulatkan menjadi 26).

Karena merupakan pencacah naik dan limpahan terjadi pada saat transisi FFh ke 00h, maka nilai awalnya adalah 26 kurang dari nol yang merupakan nilai isi-ulang pada TH1, dituliskan dengan instruksi :

```
MOV     TH1, #-26 atau
MOV     TH1,#0E6h
```

Karena dilakukan pembulatan, ada sedikit ralat pada *baut rate*-nya. Pada umumnya, ralat 5 % bisa ditolerir dalam komunikasi tak-sinkron *(asynchronous).Baut rate* yang tepat bisa diperoleh jika menggunakan kristal 11,059 MHz, selanjutnya perhatikan (Tabel 5.3).

Tabel 5.3: Ringkasan baut rate untuk Timer 1 sebagai Generator Baut Rate

| Baut Rate | Frekuensi Krista | SMOD | Nilai isi-ulang TH1 | Baut Rate aktual | Ralat |
|---|---|---|---|---|---|
| 9600 | 12,000 MHz | 1 | -7(F9H) | 8923 | 7 % |
| 2400 | 12,000 MHz | 0 | -13(F3H) | 2404 | 0,16 % |
| 1200 | 12,000 MHz | 0 | -26(F6H) | 1202 | 0,16% |
| 19200 | 11,059 MHz | 1 | -3(FDH) | 19200 | 0 |
| 9600 | 11,059 MHz | 0 | -3(FDH) | 9600 | 0 |
| 2400 | 11,059 MHz | 0 | -12(F4H) | 2400 | 0 |
| 1200 | 11,059 MHz | 0 | -24(E8H) | 1200 | 0 |

## 5.5 Menggunakan Timer 2 Untuk Menghasilkan Baut Rate Pada AT89C51

Menggunakan Timer 2 Untuk Menghasilkan Baut Rate Pada AT89C51, menset TCLK = 1 dan/atau RCLK = 1 dalam register T2CON mengakibatkan Timer 2 sebagai generator baut rate. Pada kondisi seperti ini , baut rate pengiriman dan penerimaan dapat berbeda. Struktur Timer 2 sebagai generator baut rate.

Mode generator baut rate sama seperti mode isi-ulang otomatis, pada mode ini limpahan dalam TH2 akan mengakibatkan pengisian-ulang register

Timer 2 dengan nilai 16-bit dari register RCAP2H dan RCAP2L (yang sebelumnya diinisialisasi melalui program). Pada kasus ini , baut rate Mode 1 dan 3 ditentukan oleh laju limpahan Timer 2 dengan menggunakan persamaan :

$$BautrateMode1dan2 = \frac{(LajuLimpahanTimer2)}{16}$$

Timer 2 dapat dikonfigurasikan baik sebagai pencacah atau pewaktu. Pada umumnya aplikasi Timer 2 dikonfigurasikan sebagai pewaktu (C/T2 = 0 ). Biasanya, kenaikkan pewaktu setiap siklus mesin (atau 1/12 frekuensi kristal ), tetapi kerja pewaktu berbeda untuk hal ini, kenaikkan pewaktu (Timer 2) tiap waktu kondisi (atau frekuensi kristal). Baut Rate ditentukan dengan persamaan :

$$BautrateMode1dan2 = \frac{(FrekuensiKristal)}{32x[6553 - (RCH2H, RCAP2L)]}$$

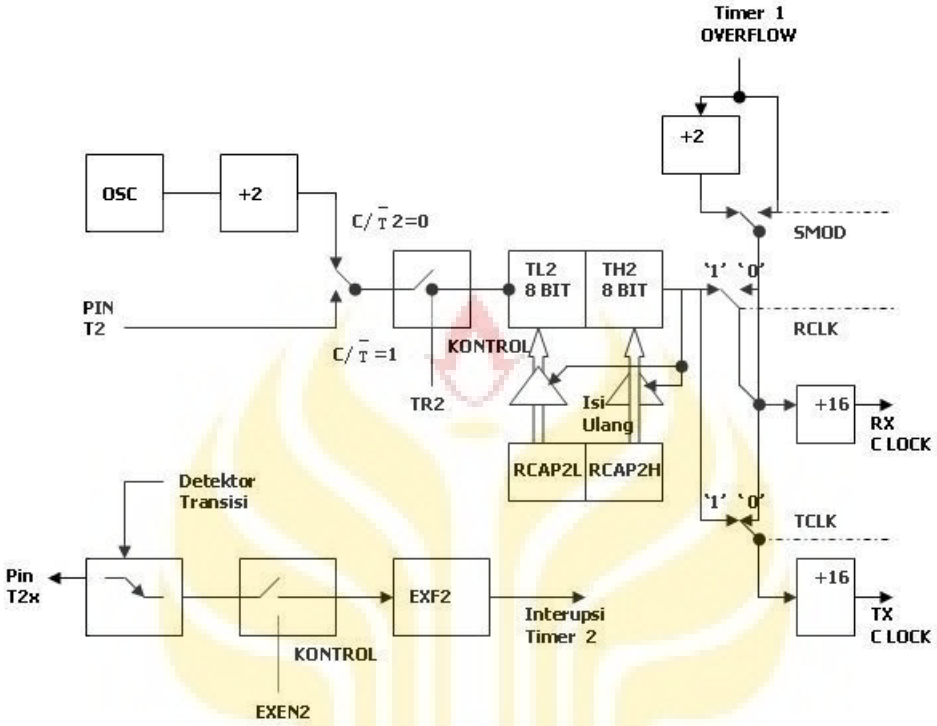
Dengan ( RACP2H, RCAP2L) adalah RCAP2 dianggap sebagai bilangan bulat tak bertanda 16-bit. (Gambar 5.3.) tersebut berlaku hanya pada saat *RCLK dan /atau TCLK = 1* pada register T2CON. Limpahan pada TH2 tidak akan menyebabakan TF2 aktif dan juga tidak menghasilkan interupsi. Dengan demikian, interupsi Timer 2 tidak perlu dimatikan(jika Timer 2 digunakan sebagai generator baut rate).

EXEN = 1, maka transisi 1- ke-0 pada pin T2EX mengaktifkan tetapi tidak menyebabkan pengisian ulang pada *TL2 dan TH2*. Dengan demikian , saat timer 2 digunakan sebagai generator *baut rate*, T2EX dapat digunakan sebagai sebuah interupsi eksternal tambahan.



Pada saat Timer dua bekerja, maka (TR = 1) sebagai pewaktu dalam mode generator baut rate, sebaiknya program jangan membaca maupun menulis *TH2 atau TL2*. Hal ini disebabkan, pada saat itu pewaktu berubah (naik) tiap waktu kondisi bukan tiap siklus , sehingga pembacaan dan penulisan menjadi tidak akurat. Register RCAP2 bisa dibaca, tetapi jangan ditulisi,

karena menyebabkan tumpang tindih denagn nilai isi-ulang sebelumnya. Jika ingin mengakses isi *RCAP2 atau Timer 2*, maka matikan dulu *Timer 2 (TR2 = 0 )*.




Gambar 5.3: Timer 2 sebagai generator baut rate

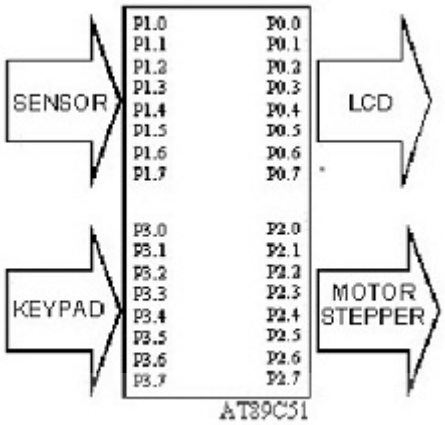
# Bab 6

# APLIKASI PENGGUNAAN PORT PARALEL

## 6.1 Penggunaan Port

Sistem mikroprosesor (mikrokontroler) akan berguna jika pada sistem tersebut telah dihubungkan dengan komponen-komponen luar. Komponen-komponen tersebut dihubungkan dengan mikroprosesor melalui port (bandar). Sebagaimana telah disebutkan pada sebelumnya, bahwa mikrokontroler AT89C51 memiliki 4 (empat) buah port, yaitu Port 0, Port 1, Port 2 dan Port 3.

Port 0 merupakan port dua fungsi yang berada pada pin 32-39 dari IC AT 89C51. Merupakan port I/O 8 bit dua arah yang serba guna port ini dapat digunakan sebagai multlipleks bus data dan bus alamat rendah untuk pengaksesan memori eksternal. Port 1 merupakan port I/O yang berada pada pin 1-8. Port ini dapat bekerja dengan baik untuk operasi bit maupun byte, tergantung dari pengaturan pada software. Port 2 merupakan port I/O serba guna yang berada pada pin 21- 28, port ini dapat juga digunakan sebagai bus alamat byte tinggi untuk rancangan yang melibatkan pengaksesan memori eksternal. Port 3 merupakan port I/O yang memiliki dua fungsi yang berada pada pin 10-17, port ini mempunyai multi fungsi. Masing-masing port tersebut memiliki 8 bit aliran data, seperti yang tampak pada Gambar 6.1 di bawah ini.

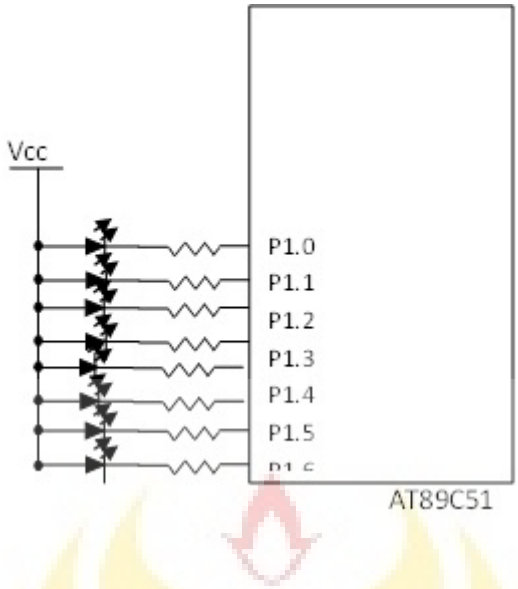



Gambar 6.1: Aplikasi Port pada AT89C51

Pada AT98C51, 8 bit aliran data tersebut dapat difungsikan sebagai ataupun sebagai output. Beberapa peralatan yang dapat digunakan sebagai input, antara lain adalah: saklar, push button, keypad, keyboard atau sensor. Sedangkan peralatan yang dapat digunakan sebagai outpu adalah: LED, Seven Segment, LCD, Speaker (Buzzer), motor-motor dan lain-lain. Beberapa aplikasi Input/Output pada port akan dibahas secara bertahap pada buku ini dan buku-buku tentang mikrokontroler yang berikutnya.

Beberapa aplikasi LED, mengihidupkan sekaligus mengendalikannya, akan dibahas pada bagian ini. Aplikasi LED ini akan memanfaatkan port paralel AT89C51 dengan berbagai kombinasinya. Rangkaian minimum untuk menghidupkan 8 LED melalui Port 1 ditunjukkan pada Gambar 6.2. Yang perlu diperhatikan adalah konfigurasi rangkaian LED itu sendiri, yaitu harus dirangkai dengan aturan Common Anode (CA). Pada konfigurasi ini, untuk menghidupkan LED pada port yang bersangkutan harus dikirimkan data 0, dan sebaliknya.

Dalam pasal ini akan dijelaskan bagaimana menghidupkan (sekaligus mengendalikan) beberapa LED melaului *Port Paralel AT89C51* dengan berbagai macam kombinasi. Yang perlu diperhatikan adalah konfigurasi rangkaian LED

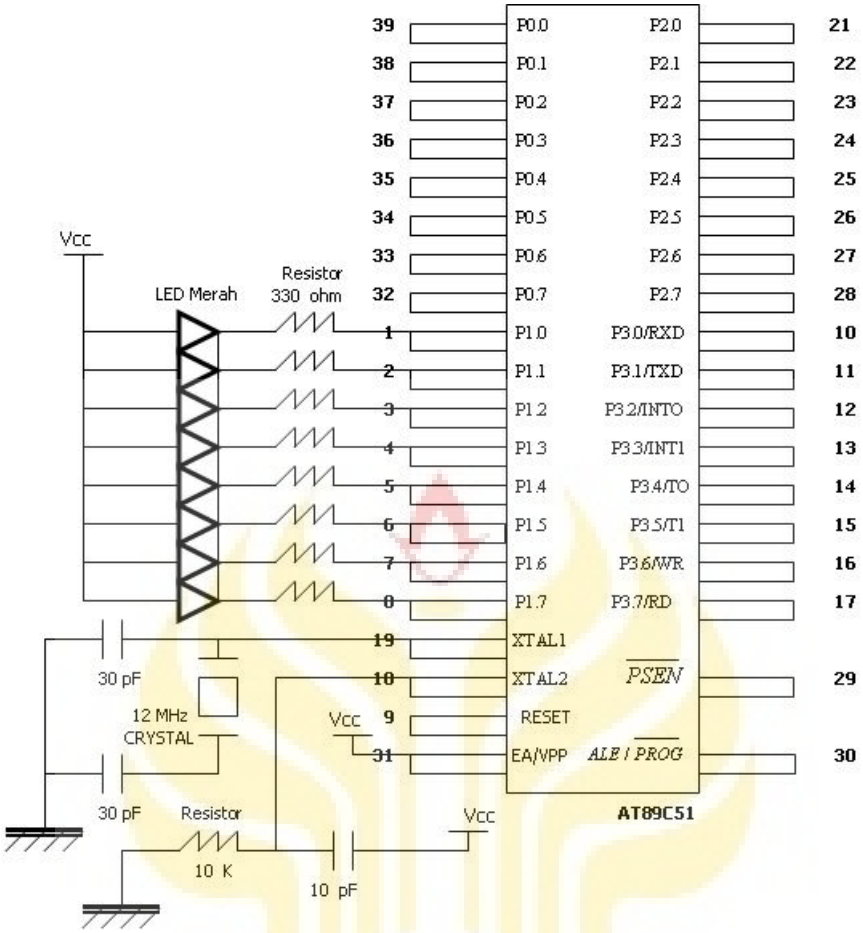



Gambar 6.2: Rangkaian Aplikasi LED

itu sendiri, yaitu *Common Anode* (CA), artinya untuk menghidupkan*LED port 1* yang bersangkutan harus dikirim atau dituliskan logika 0 . Jika nilai 0 dituliskan ke port 0, maka keluaran dari*latch* akan menghidupkan LED, sehingga baik kaki port 0 maupun resistor pullup internal akan di *pulled-low* (secara internal juga), sehingga LED yang dihubungkan secara *Common Chatode* (CC) bisa menyala.

Sedang jika menggunakan rangkaian *Common Chatode* (CC), maka untuk menyalakan LED butuh penulisan nilai 1 , namun penulisan 1 ini menyebahkan port 0 menjadi masukan berimpedansi (karena adanya *resistor pullup internal* ) dan hanya cocok untuk masukan bukan keluaran (arus dari pullup internal tidak kuat untuk menyalakan LED karena ordenya $\mu$A, sedangkan keluarannya bisa mencapai sekitar 3,8 mA).Hal ini berlaku juga untuk Port 2 dan Port 3.

Penggunaan resistor sebesar 330 ohm sebagai pembatas arus, dengan tegangan Vcc adalah 5 volt, maka arusnya sekitar 15 mA cukup untuk menghidupkan LED (biasanya sekitar 10 mA ). Rangkaian minimum untuk menghidupkan 8 LED melalui Port 1 ditunjukkan pada (Gambar 6.3.)




Gambar 6.3: Aplikasi Menghidupkan LED Melalui Kanal Paralel

## 6.2 Program Pertama

Program ini merupakan aplikasi untuk membuat kelompok empat LED *mati-hidup* secara bergantian.

```
1 : ; ————————————————————————————————
2 : ; Lampu flip-flop pada Port 3 empat LED hidup  mati bergantian
3 : ; ————————————————————————————————-
```

*4 : ORG OH*

*5 : MULAI : MOV P3, # 00001111B ; LED P3.4 s/d P3.7 nyala*

*6 : ACALL DELAY ; Panggil sub-rutin Delay*

*7: MOV P3, #11110000B ; LED P3.0 S/D P3.3 nyala*

*8 : ACALL DELAY ; Panggil sub-rutin Delay*

*9 : SJMP MULAI*

*10: ; ————————————————————————————*

*11: ; Sub-rutin delay*

*12: ; ————————————————————————————-*

*13: DELAY : MOV R0,#5 ; Isi R0 dengan 5*

*14: DELAY 1: MOV R1,#0FFH ; Isi R1 dengan FF (hex) 5*

*15: DELAY2 MOV R2, # 0*

*16: DJNZ R2, #$*

*17: DJNZ R1, DELAY2 ; Kurangi R1 denagn 1, bila*

*18: ; hasli belum sama dengan*

*19: ; 0 maka lompat ke DELAY2*

*20: DNJZ R0, DELAY1 ; Kurangi R0 dengan 1, jika*

*21: ; hasilnyan belum sama*

*22: ; dengan 0 maka lompat ke*

*23: ; DELAY1*

*24: RET ; Kembali ke alamat setalah*

*25: ; perintah*

*26: END ; ACALL DELAY*

*27: ; ————————————————————————*

Keterangan Program:

1 *Baris 1 s/d 3 merupakan komentar sekaligus keterangan program.*

2 *Baris 4 s/d 9 merupakan program utama. Baris 4 digunakan agar instruksi dituliskan mulai dari alamat 0h. Baris 5 mengirimkan data 00001111B (biner) ke Port 1 agar lampu LED 4 s/d LED 7 ( berkaitan dengan P3.4 s/d P3.7) menyala, kemudian diikuti dengan mengerjakan tundaan (baris 6) yaitu pada sub-rutin DELAY.*

3 *Baris 7 ( berkaitan dengan P3.0 s/d P3.3 ) LED 0 s/d LED 3 menyala secara bergantian dengan LED 4 s/d LED 7. Kemudian (baris 5 ) tundaan lagi (baris 8 ). Proses ini berjalan secara berulang-ulang dari awal dengan instruksi SJMP mulai pada baris 9.*

4 *Baris 13 s/d 24 merupakan program sub-rutin delay.*

## 6.3 Program Kedua

Program ini , kita akan memodifikasi program pertama , sedemikian hingga bisa untuk menghidupkan lampu LED *(sebanyak 4)* dalam kelompok genap

dan ganjil secara bergantian sebagai bagaimana program sebagai berikut:

```
1 : ; ————————————————————————————————
2 : ; Lampu flip-flop genap dan ganjil pada Port 3 menyala bergantian
3 : ; ————————————————————————————————
4 : ORG OH
5 : MULAI : MOV P3, #01010101B ; LED P3.1 s/d P3.7 nyala
6 : ACALL DELAY ; Panggil sub-rutin Delay
7: MOV P3, #10101010B ; LED P3.0 S/D P3.6 nyala
8 : ACALL DELAY ; Panggil sub-rutin Delay
9 : SJMP MULAI
10: ; ————————————————————————-
11: ; Sub-rutin delay
12: ; ————————————————————————
13: DELAY : MOV R0,#5 ; Isi R0 dengan 5
14: DELAY 1: MOV R1,#0FFH ; Isi R1 dengan FF (hex) 5
15: DELAY2 MOV R2, # 0
16: DJNZ R2, #$
```

*17: DJNZ R1, DELAY2 ; Kurangi R1 dengan 1, bila*

*18: ; hasli belum sama dengan*

*19: ; 0 maka lompat ke DELAY2*

*20: DNJZ R0, DELAY1 ; Kurangi R0 dengan 1, jika*

*21: ; hasilnyan belum sama*

*22: ; dengan 0 maka lompat ke*

*23: ; DELAY1*

*24: RET ; Kembali ke alamat setalah*

*25: ; perintah*

*26: END ; ACALL DELAY*

*27: ; ————————————————————————————-*

Keterangan:

Lampu flip-flop (genap dan ganjil) menyala secara bergantian, dalam hal ini ada dua kelompok, yaitu genap dan ganjil. Kelompok genap ( P3.0., P3.2., P3.4. dan P3.6.), sedangkan kelompok ganjil ( P3.1., P3.3., P3.5. dan P3.7. )



## 6.4 Program Ketiga

Modifikasi program lainnya adalah untuk menghidupkan lampu LED (sebanyak 2) dimulai dari LED pada posisi P3.3 dan P3.4, P3.2 dan P3.5, P3.1 dan P3.6, P3.0 dan P3.7. Kemudian menyalakan LED agar bergerak ke tengah

atau kembali ke posisi awal dari lampu LED yang menyala. Susunan program ditunjukkan sebagai berikut :

```
1 : ; ————————————————————————
2 : ; Lampu LED menyala dari tengah pada Port 3
3 : ; ————————————————————————-
4 : ORG OH
5 : MULAI : MOV P3, #11100111B ; LED P3.4 s/d P3.7 nyala
6 : ACALL DELAY ; Panggil sub-rutin Delay
7: MOV P3, #11011011B ; LED P3.0 S/D P3.3 nyala
8 : ACALL DELAY
9 : MOV P3, #10111101B
10: ACALL DELAY
11: MOV P3, #01111110B
12: ACALL DELAY
13: MOV P3, #10111101B
14: ACALL DELAY
15: MOV P3, #11011011B
```

*16: ACALL DELAY*

*17: SJMP MULAI*

*18: ; ————————————————————————————*

*19: ; Sub-rutin delay*

*20: ; ————————————————————————————*

*21: DELAY : MOV R0,#5 ; Isi R0 dengan 5*

*22: DELAY 1: MOV R1,#0FFH ; Isi R1 dengan FF (hex)*

*23: DELAY2 MOV R2, # 0*

*24: DJNZ R2, #$*

*25: DJNZ R1, DELAY2 ; Kurangi R1 denagn 1, bila*

*26: ; hasli belum sama dengan*

*27: ; 0 maka lompat ke DELAY2*

*28: DNJZ R0, DELAY1 ; Kurangi R0 dengan 1, jika*

*29: ; hasilnyan belum sama 0*

*30: ; maka lompat ke Delay1*

*31: RET ; Kembali ke alamat setelah*

*32: ; perintah ACALL DELAY*

*33: END*

## 6.5 Program Keempat

Program berikut ini merupakan aplikasi untuk membuat LED menyala secara bergantian (bergeser) dari P3.0 ke P3.7 kemudian kembali ke awal lagi secara berulangulang.

*1 : ; ————————————————————————————-*

*2 : ; Lampu LED nyala bergeser di Port 3*

*3 : ; ————————————————————————————-*

*4 : ORG OH*

*5 : MOV A,# 11111110B ; Simpan data 11111110B*

*6 : ; ke Akumulator*

*7 : MULAI :*

*8 : MOV P3, A ; Keluarkan isi Akumulator*

*9 : ; ke Port 3*

*10 : ACALL DELAY ; Panggil sub-rutin Delay*

*11: RL A ; Putar isi Akumulator ke*

*12: ; kiri 1 bit*

*13: SJMP MULAI ; Lompat ke alamat*

*14: ; dengan label MULAI*

*15: ; ————————————————————————*

*16: ; Sub-rutin delay*

*17: ; ————————————————————————-*

*18: DELAY : MOV R0,#0 ; Isi R0 dengan 0*

*19: DELAY 1: MOV R1,#0 ; Isi R1 dengan 0*

*20: DELAY2 DJNZ R1, DELAY2*

*21: DJNZ R0, DELAY 1*

*22: RET ; Kembali ke alamat setalah*

*23: ; perintah*

*24: END ; ACALL DELAY*

*25: ; ————————————————————————-*



Program di atas (program keempat) dapat dimodifikasi agar lampu LED menyala bergantian dengan arah berlawanan ( dari P3.7 ke P3.0) , dengan mengganti pada baris 5 dengan instruksi, sebagai berikut:

MOV A, #01111111B ; simpan data 01111111B (=7Fh)

; ke Akumulator

Baris 10 diganti dengan instruksi

RR A ; putar isi akumulator ke kanan 1 bit

## 6.6 Program Kelima

Berikut ini diberikan gabungan dari gerakan LED menyala kanan-kiri dan kiri kanan , sehingga nampak seperti ping-pong.

*1 : ; ————————————————————————-*

*2 : ; Lampu nyala ping-pong di Port 3*

*3 : ; ————————————————————————*

*4 : ORG OH*

*5 : MOV A,# 11111110B ; Simpan data 11111110B*

*6 : ; ke Akumulator*

*7 : MULAI :*

*8 : MOV P3, A ; Keluarkan isi Akumulator*

*9 : ; ke Port 1*

*10 : ACALL DELAY ; Panggil sub-rutin Delay*

*11 : RL A ; Putar isi Akumulator ke*

*12 : ; kiri 1 bit*

*13 : CJNE A,#7FH, MULAI ; apakah A=01111111B ?*

*14 : ; jika tidak ulangi !*

*15 : MULAI1 : MOV P3,A ; kirim data ke port 3*

*16 : ACALL DELAY ; Tunda*

*17 : RR A ; Putar isi akumulator ke*

*18 ; kanan 1 bit*

*18 : ; ————————————————————*
*20 : ; Sub-rutin delay*
*21 : ; ————————————————————-*
*22 : DELAY : MOV R0,#0 ; Isi R0 dengan 0*

*23 : DELAY 1: MOV R1,#0 ; Isi R1 dengan 0*

*24 : DELAY2 DJNZ R1, DELAY2*

*25 : DJNZ R0, DELAY 1*

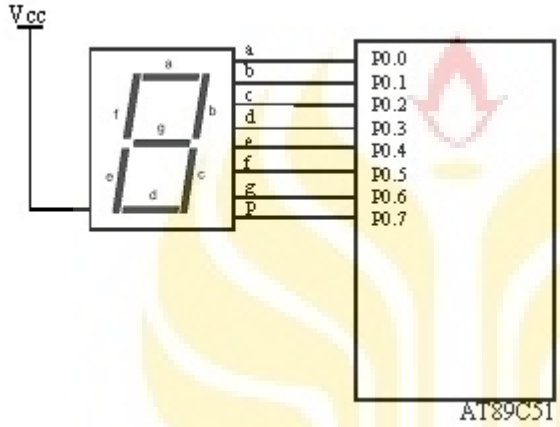
*26 : RET ; Kembali ke alamat setalah*

*27 : ; perintah*
*28 : END ; ACALL DELAY*

*29 : ; ————————————————————*

## 6.7 Aplikasi Port untuk Penggerak 7-segment

7 Segment adalah kumpulan 7 buah LED yang dirangkai sedemikian rupa sehingga dapat menampilkan (awalnya) tampilan numerik (angka) dari 0 hingga 9. Sehingga, dengan tampilan tersebut, kita dapat menunjukan sebuah hasil perhitungan, misalnya. Rangkaian penggerak 7 Segment ditampilkan pada Gambar 5.3. Port yang digunakan adalah Port 0. Sebagaimana LED, pada aplikasi ini, 7 Segment juga dirangkai menggunakan prinsip *Common Anode*.




Gambar 6.4: Rangkaian Aplikasi 7 Segment

### 6.7.1 Display LED 7 Segment

Program ini menghidupkan display LED 7 Segment untuk menampilkan angka 0 sampai dengan 9. Cara yang dipakai pada contoh program ini adalah dengan mengeluarkan data langsung ke port yang bersangkutan.

*Line I Addr Code Source*

*1: $MOD51*

*2:*

*3: N 0000 ORG 0H*

*4: 0000 75 80 C0 MULAI:MOV P0, #0C0H ; keluarkan kode angka 0 ke port 0*

*5: 0003 11 34 ACALL DELAY ; panggil subrutin DELAY*

*6: 0005 75 80 F9 MOV P0, #0F9H ; keluarkan kode angka 1 ke port 0*

*7: 0008 11 34 ACALL DELAY ; panggil subrutin DELAY*

*8: 000A 75 80 A4 MOV P0, #0A4H ; keluarkan kode angka 2 ke port 0*

*9: 000D 11 34 ACALL DELAY ; panggil subrutin DELAY*

*10: 000F 75 80 B0 MOV P0, #0B0H ; keluarkan kode angka 3 ke port 0*

*11: 0012 11 34 ACALL DELAY ; panggil subrutin DELAY*

*12: 0014 75 80 99 MOV P0, #099H ; keluarkan kode angka 4 ke port 0*

*13: 0017 11 34 ACALL DELAY ; panggil subrutin DELAY*

*14: 0019 75 80 92 MOV P0, #092H ; keluarkan kode angka 5 ke port 0*

*15: 001C 11 34 ACALL DELAY ; panggil subrutin DELAY*

*16: 001E 75 80 82 MOV P0, #082H ; keluarkan kode angka 6 ke port 0*

*17: 0021 11 34 ACALL DELAY ; panggil subrutin DELAY*

*18: 0023 75 80 F8 MOV P0, #0F8H ; keluarkan kode angka 7 ke port 0*

*19: 0026 11 34 ACALL DELAY ; panggil subrutin DELAY*

*20: 0028 75 80 80 MOV P0, #080H ; keluarkan kode angka 8 ke port 0*

*21: 002B 11 34 ACALL DELAY ; panggil subrutin DELAY*

*22: 002D 75 80 90 MOV P0, #090H ; keluarkan kode angka 9 ke port 0*

*23: 0030 11 34 ACALL DELAY ; panggil subrutin DELAY*

*24: 0032 80 CC SJMP MULAI*

*25:*

*26: 0034 78 05 DELAY: MOV R0,#5 ; Isi Register R0 dengan 5*

*27: 0036 79 FF DELAY1: MOV R1,#00FFH ; Isi Register R1 dengan FFH*

*28: 0038 7A 00 DELAY2: MOV R2,#0 ; Isi Register R2 dengan 0*

*29: 003A DA FE DJNZ R2,$ ; Terjadi 3 kali loop, untuk*

*30: 003C D9 FA DJNZ R1,DELAY2 ; mendapatkan, waktu tunda*

*31: 003E D8 F6 DJNZ R0,DELAY1 ; yang diharapkan*

*32: 0040 22 RET ;*

*33:*

*34: END*

### 6.7.2 Penyederhanaan program

Program diatas memang dapat menampilkan angka 0 sampai dengan 9 pada 7 Segment. Namun, program yang dituliskan cukup panjang. Bagian berikut mencoba untuk menyederhanakan program yang sudah ada dengan memanfaatkan pengambilan data melalui array.

*Line I Addr Code Source*

*1: $MOD51*

*2:*

*3: N 0000 ORG 0H*

*4: 0000 7B 0A MULAI: MOV R3,#10 ; jumlah angka numerik*

*5: 0002 90 00 1D MOV DPTR,#NUMERIK ; address array ROM NUMERIK*

*6: 0005 NEXTDATA:*

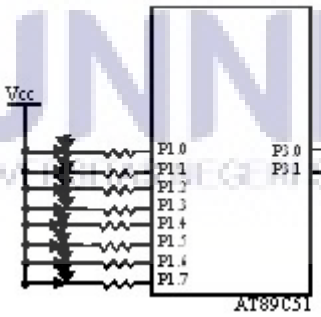
*7: 0005 E4 CLR A ; A = 0*

*8: 0006 93 MOVC A,@A+DPTR ; ambil data dari array ROM sesuai A*

*9: 0007 F5 80 MOV P0,A ; keluarkan data A ke port 0*

```
10: 0009    A3      INC DPTR ; naikan harga DPTR
11: 000A    11 10     ACALL DELAY ; panggil subrutin DELAY
12: 000C    DB F7      DJNZ R3,NEXTDATA ; kurangi R3 dengan 1,
hingga habis
13: 000E    80 F0     SJMP MULAI
14: 0010    78 05          DELAY: MOV R0,#5 ; Isi Register R0
dengan 5
15: 0012    79 FF          DELAY1: MOV R1,#00FFH ; Isi Register
R1 dengan FFH
16: 0014    7A 00          DELAY2: MOV R2,#0 ; Isi Register R2
dengan 0
17: 0016    DA FE     DJNZ R2,$ ; Terjadi 3 kali loop, untuk
18: 0018    D9 FA     DJNZ R1,DELAY2 ; mendapatkan, waktu tunda
19: 001A    D8 F6     DJNZ R0,DELAY1 ; yang diharapkan
20: 001C    22        RET ;
21:
22:     001D C0 F9 A4 B0 NUMERIK:DB 0C0H,0F9H,0A4H,0B0H,099H,
        092H,082H,0F8H,080H,090H 0021 99 92 82 F8 0025 80 90
23:     END
```

### 6.7.3 Aplikasi Port Masukan/Keluaran Penggerak LED




Gambar 6.5: Rangkaian Aplikasi 7 Segment

Bagian ini menggabungkan aplikasi masukan berupa 2 buah push button yang dihubungkan dengan Port 3 dengan aplikasi LED. Rangkaian untuk LED sama dengan rangkaian sebelumnya, serta dihubungkan dengan Port 1. Sedangkan push button dipasangkan dengan P3.0 dan P3.1 yang dihubungkan dengan ground. Rangkaian lengkapnya diperlihatkan pada Gambar 6.5.

### 6.7.4 Program untuk menghidupkan LED melalui push button di P3.0

```
Line I      Addr      Code      Source
1:          $MOD51
2:
3: N 0000 ORG 0H
4: 0000     E5 B0 MULAI:MOV A,P3 ; Baca Port 3, masukan ke A
5: 0002     B4 FE 03 CJNE A,#0FEH,TERUS ; A=FED?, Jika tidak sama, TERUS
6: 0005     75 90 00 MOV P1,#0 ; keluarkan 0 ke Port 1, LED nyala
7:
8: 0008     B4 FD F5 TERUS: CJNE A,#0FDH,MULAI ; A=FDH?, jika tidak MULAI
9: 000B     75 90 FF MOV P1,#0FFH ; keluarkan 1 ke Port 1, LED padam
10: 000E    80 F0 SJMP MULAI ; loncat ke MULAI
11:
12:                  END
```

Program ini menghidupkan LED melalui push button di P3.0 dan memadamkan LED melalui P3.1. Dengan demikian akan dilakukan proses polling pada Port 3 untuk mengecek tombol yang ditekan.

### 6.7.5 Program sistem *switch* toggle pada P3.0.

Program berikut ini adalah modifikasi program sebelumnya. Pada program ini, digunakan sistem switch toggle pada P3.0. Sistem ini menghidupkan dan

memadamkan LED secara bergantian. Oleh karena itu, diperlukan suatu register (dalam hal ini mengguanakn $R_0$) untuk menyimpan status LED, apakah sedang menyala ($R_0 = 1$) atau sedang padam ($R_0 = 0$). Dengan kata lain, register ini berfungsi sebagai mata bagi prosesor.

```
Line I Addr Code Source
1: $MOD51
2:
3: N 0000          ORG 0H
4: 0000        E5 B0          MULAI: MOV A,P3 ; Baca Port 3
5: 0002    B4 FE FB     CJNE A,#0FEH,MULAI ; A=FED?, Jika tidak sama, MULAI
6: 0005       B8 00 0C       CJNE R0,#0,TERUS ; R0=0?, jika tidak sama, TERUS
7: 0008       78 01          MOV R0,#1 ; R0 = 1
8: 000A       75 90 00       MOV P1,#0 ; keluarkan 0 ke Port 1, LED nyala
9:
10: 000D      E5 B0        TUNGGU:MOV A, P3; tunggu hingga push button dilepas
11: 000F       B4 FF FB       CJNE A,#0FFH,TUNGGU ;
12: 0012        80 EC         SJMP MULAI
13:
14: 0014    78 00         TERUS:  MOV R0, #0 ; R0 = 0
15: 0016    75 90 FF     MOV P1,#0FFH ; keluarkan 1 ke Port 1, LED padam
16: 0019    80 E5           SJMP MULAI ; loncat ke MULAI
17:
18: END
```

## 6.8 Menyimpan Program Ke Flash Memori Menggunakan *Easy Programer*

(1) Setelah program ditulis pada editor, lakukanlah kompilasi program hingga menghasilkan berkas dalam format heksa atau biner, kemudian lakukan

download program tersebut ke AT89C51 menggunakan pemrograman*Easy-Programer program EZ31.EXE.*

(2) Pilih kanal serial *COM1 atau COM2* sesuai letak kabel seial Easy-Programer program-nya , tunggu hingga muncul pesan yang menyatakan bahwa programmer sudah siap digunakan (tanda *OFF LINE* akan berganti dengan identifikasi mikrokontroler yang terpasang pada *programmer* yang bersangkutan).

(3) Pilih tombol send untuk mengirimkan berkas heksa yang akan di-downloadkan ke mikrokontroler (anda akan ditanya berkas mana yang akan digunakan pada jendela dialog yang dimunculkan), jika proses pengisian program sudah selesai, matikan sumber daya *programmer* sebelum mencabut IC mikrokontroler .Sekarang mikrokontroler sudah siap untuk dipindahkan ke papan percobaan *(breadboard, PCB, dan lain-lainnya).*

(4) Sebagai percobaan , buatlah rangkaian sebagaimana pada gambar 15 pada papan *breadboard*, atau PCB), tulis program, *download-kan* ke mikrokontroler dan amati hasilnya.

# Bab 7

# CONTOH APLIKASI PORT SERIAL

## 7.1 Inisialisasi Port Serial

Port Serial akan digunakan dengan konfiguarasi 8 bit UART dengan baut rate 2.400 menggunakan Timer 1 untuk menghasilkan detak baut rate. Pada kasus ini, empat buah register harus diinisialisasi , yaitu: *SMOD, TMOD, TCON dan TH1.* Maka nilai-nilai yang harus dipersipkan sebagai berikut :

Tabel 7.1: **Port Serial**

| SCON: | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI1 | RI |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| TMOD: | GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| TCON: | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| TH1 : | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

Dengan mengatur *SM0* dan *SM1* menjadikan port serial bekerja dalam Mode UART 8 bit. Pengaturan REN = 1 akan mengaktifkan port serial agar mulai dapat menerima karakter, dengan mengatur TI = 1 membolehkan segera mengirimkan karakter yang pertama ( TI = 1 berarti penyangga pengirim

*(SBUF))* dalam kondisi kosong). Dengan membuat M1 = 1 dan M0 = 0 ( pada register *TMOD*), maka Timer 1 bekerja sebagai pewaktu dengan isi-ulang otomatis. Sedangkan mengatur TR1 = 1 akan mengaktifkan *Timer 1* itu sendiri. Bit-bit lainnya diisi 0, karena bit-bit tersebut hanya mengatur *fitur* atau kode yang tidak digunakan pada contoh ini.

TH1 diisi ulang yang sesuai dengan kecepatan baut rate yang diinginkan, yaitu 2400 baut, dengan demikian laju limpahannya adalah 2400/32 = 76,8 kHz, jika dalam contoh ini digunakan kristal dengan frekuensi 12 MHz *(Timer 1 terdetak dengan laju 1 MHz atau 1000 kHz)*, maka laju , limpahannya harus terjadi tiap 1000/76,8 = 13,02 detak atau dibulatkan menjadi 13 detak yang diisikan ke TH1 adalah -13 atau F3h.
Program inisialisasi, selengkapnya sebagai berikut:

```
1 : ORG 0h
2 : INIT : MOV SCON,#52h ; port serial, mode 1
3 : MOV TMOD, #20h ; timer 1, mode 2
4 : MOV TH1, # -13 ; nilai isi-ulang untuk 2400
5 : ; baut
6 : SET TR1 : aktifkan timer 1
7 : ( . . . program selanjutnya . . . )
```

## 7.2 Subrutin Pengirim Karakter

Kali akan dibuat sebuah sub-rutin yang digunakan untuk mengirimkan kode ASCII 7-bit yang tersimpan dalam akumulator ke port serial AT89C51/52, dengan tambahan sebuah but paritas ganjil sebagai bit ke-8. Sebelum dan sesudah pemanggilam sub-rutin, isi akumulator tidak boleh berubah. Perhatikan sub-rutin OUTCHR berikut:

```
1 : OUTCHR: MOV C, P ; simpan bit paritas di C
```

2 : CPL C *; ubah ke paritas ganjil*

3 : MOV ACC.7,C *; tambahkan ke kode karakter*

4 : LAGI JNB TI, LAGI *; SBUF-Tx kosong? tidak cek lagi*

5 : CLR TI *; ya : kosongkan tanda dan*

6 : MOV SBUF, A *; kirim karakter*

7 : CLR ACC,7 *; kosongkan bit paritas*

8 : RET *; selesai sub-rutin*

Keterangan :

- Instruksi baris 1,2 dan 3 menempatkan paritas ganjil ke bit 7 pada akumulator. Karena bit P pada PSW merupakan bit paritas genap akumulator, maka sebelum disimpan dikomplemen (CPL) terlebih dahulu.
- Instruksi JNB (baris 4) digunakan untuk menunggu apakah karakter ( plus bit paritas ganjil ) yang sebelumnya, telah selesai dikirim (selesai dengan tanda TI = 1 ).
- Jika sudah selesai , maka TI di-nol-kan secara manual.( baris 5 ) dan isi akumulator akan dikirim ke SBUF (akan mengirimkan karakter keluar melalui port serial).

Cara penggunaan sub-rutin ini sangat mudah, siapkan data karakter ASCII yang akan dikirimkan ke dalam akumulator kemudian panggil sub-rutin ini , misalnya :

- MOV A, # G *; siapkan karakter G*
- CALL OUTCHR *; kirimkan karakter !*

## 7.3 Sub-rutin Penerima Karakter

Akan membuat sub-rutin untuk menerima karakter yang dikirimkan ke AT89C51/52, dengan bit-7 pada akumulator berisi bit paritas ganjil, jika terjadi kesalahan

paritas, maka set tanda *carry* ( C ).

Perhatikan sub-rutin INCHAR berikut ini :

```
1 : INCHAR : JNB RI, $        ; tunggu karakter yang dating
2 : CLR RI                    ; kosongkan tanda carry
3 : MOV A, SBUF               ; baca karakter di A
4 : MOV C, P                  ; untuk paritas ganjil di A, P harus
5 :                           ; di-set
6 : CPL C                     ; lakukan komplemen di C, agar C
7 :                           ; sebagai indikasi kesalahan
8 : CLR ACC, 7                ; kosongkan paritas di A
9 : RET                       ; selesai sub-rutin
```

Keterangan sub-rutin INCHAR :

a : Pada baris 1 sub-rutin dimulai dengan menunggu tanda flag interupsi penerimaan atau R1 = 1,yang menandakan bahwa ada sebuah karakter berada di SBUF siap dibaca.

b Pada baris 2 dan 3 , ketika RI = 1, maka instruksi berikutnya , RI di-nol-kan dank ode dalam SBUF dibaca ke akumulator.

c Pada baris 4 , bit P dalam PSW merupakan paritas genapnya akumulator, maka P = 1 jika akumulator, dengan sendirinya, berisi cacah logika 1-nya ganjil, karena C harus sama dengan 0 jika tidak terjadi kesalahan, maka setelah bit P disalin ke C.

d Pada baris 6, C harus dikomplemen.

e Pada baris 7, ACC dikosongkan untuk menyakinkan bahwa hanya kode 7 bit saja yang dikembalikan setelah selesainya sub-rutin.

# Bab 8

# KOMUNIKKASI SERIAL DENGAN KOMPUTER

Dalam bagian ini akan dijelaskan bagaimana cara inisialisasi dan memulai komunikasi serial dengan menggunakan komputer (Program Pertama). Pada program pertama ini untuk aplikasi mikrokontroler dengan PC melalui saluran serial. Mikrokontroler akan menunggu karakter yang dikirim dari komputer (melalui program *Hyper-terminal* atau lainnya yang sejenis), kemudian membuat jadi huruf besar , kemudian mengirimkan kembali ke komputer(melalui program *Hyper-terminal* akan ditampilkan lagi hasilnya). Sedangkan pada program kedua diaplikasikan untuk mengatur 8 LED yang terpasang pada mikrokontroler melaui Port 1.

## 8.1 Program Pertama

```
1 : ; ---------------------------------------------------------------
2 : ; Semua menggunakan interupsi, tidak ada program utama, kecuali inisialisasi baut rate
3 : ; dan interupsi.
4 :; Program akan menerima karakter (huruf), kemudian mengirimkannya kembali ke
5 : ; pengirim berupa huruf besar ( kode ASCII - 32 ).
```

```
6 : ; ————————————————————————————————————
7 : ORG 00H
8 : SJMP MAIN ; lompat ke program utama
9 : ; ————————————————————————————————————-
10: isi vektor interupsi port serial dengan RLI
11: ;
12: ORG 23H ; lokasi vektor interupsi port serial
13: SJMP SERINT ; karena panjang lompat SERINT
14: ; ————————————————————————————————————
15: ;
16: ORG 030H ; program utama mulai di sini
17: MAIN : MOV TMOD,#20H ; Timer 1 mode 2 ( 8 - bit, isi-ulang )
18: MOV TH1, #0F3H ; 2400 baut rate
19: MOV SCON, #50H ; MODE Serial : 8  bit UART
20: SETB TR1 ; Star Timer 1
21: MOV SP, #10H ; DEFINISIKAN lokasi stack
22: ; ————————————————————————————————————
23: ; Inisialisasi inteupsi
24: ;
25: SETB ES
26: SETB EA
27: ; ————————————————————————————————————
28:  ; Interupsi sekarang aktif
29: ;
30: DEAD : SJMP DEAD ; PERULANGAN TERUS MENERUS
31: ; ————————————————————————————————————-
32: ; Penanganan interupsi
33: ; saat pengguna menekan tombol di computer program terminal akan
34: ; mengirimkan karakter tombol yang bersangkutan ke msc-51
35: ;
36: SERINT :
37: ; apa yang menyebabakna interupsi
38: ;
```

```
39: JB RI, RCV_chG ; apakah RI = 1 (set, data diterima) ?
40: ; Ya  ada karakter di SBUF
41: CLR TI ; tidak - harus TI
42: RETI ; abaikan interupsi pengirim
43: ; ---------------------------------------------------------------
44: ; penanganan interupsi penerima
45: ;
46: RCV_ch PUSH PSW ; simpan register-register
47: PUSH ACC
48: MOV A, SBUF ; baca karakter yang diterima
49: CLR RI ; kosongkan RI (agar bisa diterima lagi)
50: SUBB A, #32 ; konversi ke huruf besar ( ASCII  32 )
51: ;
52: MOV SBUF,A ; kirimkan kembali
53: ;
54: EXIT : POP ACC ; ambil ACC kembali (dari stack)
55: POP PSW ; dan PSW
56: RETI ; kembali ke ???
57: END
58: ; ---------------------------------------------------------------
```

Program diawali dengan lompatan (baris 8) ke program MAIN (baris 17, lokasi memori 0030h), karena lokasi berikutnya akan digunakan untuk meletakkan Rutin Layanan Interupsi(RLI) Port Serial(baris 12).Karena RLI Port Serial panjangnya lebih dari 8 byte, RLI tersebut perlu diletakkan ke lokasi yang lain(baris 37) dan pada alamat vektor interupsi Port Serial tersebut dilakukan lompatan ke RLI yang sesungguhnya tersebut (baris 13).

Program utama (MAIN), diawali dengan inisialisasi Timer 1 sebagai Mode 2 (8-bit,isi ulang). Nilai isi-ulang ke TH1 adalah F4h ( untuk 2400 baut rate dengan frekuensi kristal 11,059 MHz atau jika menggunakan 12 MHz isi dengan F3h). Anda bisa mengganti dengan kecepatan berapapun juga, karena di computer tinggal menyesuaikan saja. Kemudian inisialisasi port serial dengan mode UART 8-bit tanpa paritas (baris 19), diikuti aktivasi Timer 1 (baris

20) dan inisialisasi awal stack(baris 21). Akhirnya interupsi diaktifkan ( baris 25 dan baris 26), kemudian dilakukan pengulangan tanpa-kerja *(do-nothing loop)*,(baris 30).

Selanjutnya mikrokontroler akan menunggu datangnya data (sebenarnya juga selesainya pengiriman data). Jika mikrokontroler menerima data dari computer dan proses pengirimannya sudah selesai, RI akan set sehingga menyebabkan terjadinya interupsi port serial atau RLI SERINT dijalankan. Mikrokontroler hanya tahu terjadi interupsi port serial , tetapi tidak tahu apakah interupsi penerimaan data (RI = 1) atau pengiriman data (TI = 1 ), dengan demikian di awal RLI SERINT diawali dengan memeriksa interupsi apa yang terjadi (baris 39). Jika terjadi interupsi pengiriman data, abaikan saja, artinya tidak ada yang harus dilakukan dan selesai(baris 41 dan baris 42).

Namun, jika yang terjadi adalah interupsi penerimaan data ( RI = 1), maka lakukan pemrosen lebih lanjut. Pada awalnya, simpan terlebih dahulu isi akumulator dan register PSW (baris 46 dan baris 47), kemudian baca isi SBUF, untuk mengetahui kode ASCII yang diterima(baris 48), selanjutnya jangan lupa untuk membuat port serial dapat menerima data lagi ( dengan me-nol-kan RI secara manual, baris 49).

Selisih antara kode ASCII untuk huruf besar dan kecil adalah 32, dengan demikian data yang diterima harus dikurangi 32 (baris 50, jika huruf yang terkirim sudah huruf besar, maka hasilnya berupa karakter lain). Sebelum keluar dari RLI SERINT jangan lupa untuk mengambil kembali data PSW dan akumulator dari stack.

Cara Menjalankan Program :

1) Aktifkan rangkaian minimal pada gambar 16.

2) Melalui computer, jalankan program Hyperterminal ( Start $--> $ Programs $--> $ Accessories $--> $ Communication $--> $ Hyperterminal ) , sehingga muncul jendela *Connection Description.*

3) Pada jendela Connection Description, tuliskan nama koneksinya, kemudia klik OK, sehingga akan dilanjutkan dengan jendela Connect To. Pada jendela Connection To isikan Connect Using dengan sambungan port serial

yang digunakan ( bisa COM1, COM2 dan seterusnya ).

4) Kemudian dilanjutkan dengan jendela COMx Properrties, pada saat ini, masukkan data parameter komunikasi yang diinginkan, yaitu :

   - Bits per second : 2400
   - Data Bits : 8
   - Parity : None
   - Stop bits : 1
   - Flow control : none

5) Setelah itu Hyperterminal siap untuk berkomunikasi dengan mikrokontroler.

## 8.2 Program Kedua

```
1 : ; ---------------------------------------
2 : ; Semua menggunakan interupsi, tidak ada program utama, kecuali inisialisasi baut rate
3 : ; dan interupsi.
4 : ; Program akan menerima karakter (huruf), kemudian memeriksanya , jika
5 : ; isinya  a  --> lampu LED 1, 3, 5, dan 7 menyala
6 : ; isinya  b  --> lampu LED 0, 2, 4, dan 6 menyala
7 : ; isinya  s  --> mematikan semau lampu LED
8 : ; isi selain di atas --> lampu LED hidup semua
9 : ; ---------------------------------------
10: ORG 00H
11: SJMP MAIN ; lompat ke program utama
12: ; ---------------------------------------
13: ; isi vektor interupsi port serial dengan RLI
14: :
15 : ORG 23H ; lokasi vektor interupsi port serial
16: SJMP SERINT ; karena panjang lompat SERINT
```

```
17: ; ----------------------------------------------------------
18: ;
19: ORG 030H ; program utama mulai di sini
20: MAIN : MOV TMOD,#20H ; Timer 1 mode 2 ( 8 - bit, isi-ulang )
21: MOV TH1, #0F4H ; 2400 baut rate
22: MOV SCON, #50H ; MODE Serial : 8  bit UART
23: SETB TR1 ; Star Timer 1
24: MOV R0, #60H ; penunjuk tampilan
25: MOV SP, #10H ; definisi lokasi stacj
26: ; ----------------------------------------------------------
27: ; Inisialisasi interupsi agar aktif
28: ; 29: SETB ES
30: ;
31: ; ----------------------------------------------------------
32: : sekarang kirimkan kalimat  Selamat dating
33: ;
34: MOV DPTR, #DATANYA ; DPTR menunjuk awal data
35: MOV R2, #20 ; ada 20 karakter (termasuk 13 dan 10)
36: KIRIM : CLR A ; kosongkan akumulator
37: MOVC A,@A+DPTR ; baca karakter
38: MOV SBUF,A ; kirimkan
39: JNB TI,$ ; tunggu hingga selesai mengirim
40: CLR TI ; nol-kan TI
41: INC DPTR ; karakter berikutnya
42: DJNZ R2,KIRIM
43: DEAD : SJMP DEAD ; perulangan terus-menerus
44: ; ----------------------------------------------------------
45: ; penanganan interupsi
46: ; saat pengguna menekan tombol di computer program terminal akan
47: ; mengirimkan karakter tombol yang bersangkutan ke mcs-51
48: ;
49: SERINT :
50: ; apa yang menyebabkan interupsi
```

```
51: ;
52: JB RI,RCV_ch ; apakah RI = 1 (set, data diterima)?
53: ; ya, maka ada karakter di SBUF
54: CLR TI ; Tidak, harus TI
55: RETI ; abaikan interupsi pengiriman
56: ; ———————————————————————————
57: ; penanganan interupsi penerima
58: ;
59: RCV_ch : PUSH PSW ; simpan register-register
60: PUSH ACC
61: MOV A,SBUF ; baca karakter yang diterima
62: CLR RI ; kosongkan RI (agar bisa terima lagi)
63: CJNZ A,#  a , LED1
64: MOV P1,#01010101B
65: SJMP EXIT
66: LED1 : CJNZ A,#  B ,LED2
67: MOV P1, #10101010B
68: SJMP EXIT
69: LED2 : CJNZ A, #  s ,LED3
70: MOV P1,# 11111111B
71: SJMP EXIT
72: LED3 : MOV P1, #00000000B
73: ;
74: EXIT : MOV SBUF,A ; kirimkan kembali
75:;
76: POP ACC ; ambil ACC kembali (dari stack)
77: POP PSW ; dan PSW
78: RETI ; kembali ke ???
79: ;
80: DATANYA :
81: DB  Selamat dating13,10
82: END
```

Program kedua ini sama seperti program pertama, hanya saja melalui terminal computer, kita bisa mengendalikan menyalanya lampu LED yang terpasang pada port 1 dan saat terjadi koneksi dengan computer, mikrokontroler akan mengirimkan kalimat **Selamat Datang** ke terminal komputer. Pada baris 34 hingga 42 digunakan untuk mengirimkan kalimat Selamat dating ke computer. Data kalimat ini disimpan menjadi satu dengan program (berada di memori program, baris 79 dan baris 80), dengan demikian rutin ini diawali dengan menyimpan lokasi data ke DPTR ( baris 34). Karena ada 20 karakter, maka diperlukan pencacah, dalam hal ini digunakan R2 (baris 35).

Proses pengiriman kalimat tersebut dilakukan huruf demi huruf, diawli dengan mengosongkan akumulator (sebagai penyimpanan sementara data yang akan dikirimkan , baris 36), kemudian data dibaca (baris 37), kemudian dikirimkan ke port serial (baris 38). Sebelum proses dilanjutkan tunggu hingga pengiriman selesai (baris 39 dan baris 40), naikkan nilai DPTR, turunkan nilai pencacah dan proses diulangi lagi hingga semua huruf/karakter terkirim(baris 41, baris 42 dan baris 43).

Proses yang kedua adalah sama seperti progam pertama, yaitu menunggu datangnya data dari computer (terjadi interupsi penerimaan). Jika ada data yang diterima, maka diperiksa, karakter atau huruf apakahn itu ( baris 63 ). Jika merupakan huruf a , maka hidupka lampu LED 1,3,5 dan 7( baris 63 s/d baris 65 ). Jika bukan huruf a , maka periksa lagi, apakah huruf b (baris 66), jika memang benar huruf b , hidupkan lampu LED 0, 2, 4 dan 6 (baris 67 dan 68 ), jika tidak periksa lagi, apakah huruf s (baris 69), jika memang benar huruf s , hidupk semua lampu LED (baris 70). Akhirnya jika bukan data a, b atau s, matikan semua LED (baris 72).

# Bab 9

# KENDALI AT89C51 DENGAN VISUAL BASIC 6.0

Perkembangan dibidang mikro-elektronika mengakibatkan banyaknya peralatan yang dikendalikan oleh komputer , baik itu berbentuk PC maupun dalam bentuk mikrokomputer (biasa dikenal dengan mikrokontroler). Kelebihan dari mikrokontroler yaitu setiap perubahan terjadi cara kerja sistem dapat dilakukan dengan merubah program yang terdapat pada memorinya.

Rangkaian mikrokontroler sebagai pusat kendali sistem tidak perlu mengalami perubahan yang banyak, karena karakteristik dari rangkaian sistem minimum mikrokontroler pada dasarnya hampir sama tergantung pemrogram/ perancangan rangkaian. Sistem kendali perangkat elektronik dirancang menggunakan mikrokontroler AT89C51 sebagai pengendali utama dalam menjalankan instruksi. Perangkat lunak dibuat dalam bahasa assembly ASM51 sehingga pemrograman mikrokontroler dilakukan dengan mudah. Realisasi sistem diawali dilakukan dalam model *project board* untuk setiap rangkaian yang kemudian dibuat dalam PCB (*printed circuit board*) sederhana sesuai dengan rangkaian yang dibuat.

Seiring dengan perkembangan zaman, aktifitas manusia semakin meningkat sehingga menyebabkan manusia sering meninggalkan rumah. Dengan ke-

sibukan dalam beraktifitas tersebut, seseorang akan mengalami kesulitan berkomunikasi atau berinteraksi dangan peralatan elektronik yang ada dirumah. Misalkan saja bila seseorang berpergian jauh dan pulang larut malam, tentunya ia sebelumnya harus mempersiapkan terlebih dahulu beberapa hal selama bepergian. Hal tersebut tentunya akan menyita waktu dan akan membuang energi elektronik dengan sia-sia, maka diperlukan program yang dapat digunakan untuk mengendalikan peralatan elektronik, baik yang ada dirumah, perkantoran, ruko, mall, maupun apartemen dengan komputer sebagai pusat kendali yang memadukan *hardware* dan *software* untuk membuat suatu sistem kendali peralatan elektronik dengan mikrokontroler dan bahasa pemrograman *visual basic*.

Mikrokontroler AT89C51 merupakan salah satu keluarga dari MCS-51 keluaran Atmel. Jenis mikrokontroler ini pada prinsipnya dapat digunakan untuk mengolah data per bit ataupun data 8 bit secara bersamaan. Gambar 9.1 menunjukkan blok diagram fungsional mikrokontroler AT89C51.

Sebuah mikrokontroler dapat bekerja bila dalam mikrokontroler tersebut terdapat sebuah program yang berisi instruksi-instruksi yang akan digunakan untuk menjalankan system mikrokontroler tersebut. Instruksi-instruksi dari sebuah program pada tiap jenis mikrokontroler mempunyai beberapa perbedaan, misalkan saja instruksi pada mikrokontroler Atmel berbeda dengan instruksi pada mikrokontroler Motorola.

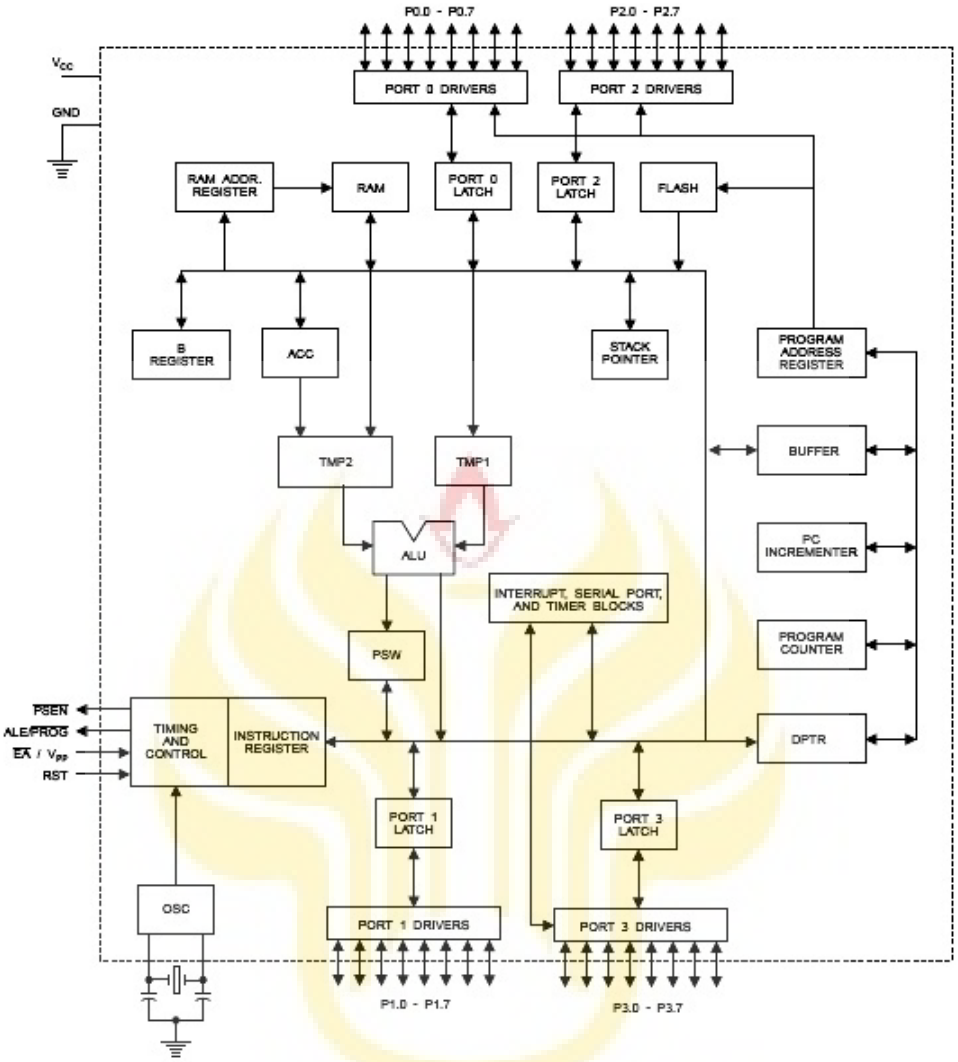
## 9.1 Prinsip Program Pada Mikrokontroler

Pada prinsipnya program pada mikrokontroler dijalankan secara bertahap, jadi pada program itu sendiri terdapat beberapa set instruksi dan tiap instruksi itu dijalankan secar bertahap atau berurutan. Beberapa fasilitas yang dimilki oleh mikrokontroler AT89C51 adalah sebagai berikut:

1) Sebuah *Central Processing Unit 8 bit*

2) Osilator internal dan rangkaian pewaktu

3) RAM internal 128 byte




Gambar 9.1: Blok diagram fungsional AT89C51

4) Flash memori 4 byte

5) Lima buah jalur interupsi (dua buah interupsi eksternal dan tiga buah interupsi internal).

6) Empat buah programmable port I/O yang masing-masing terdiri dari delapan buah jalur I/O

7) Sebuah port serial dengan control serial *full duplex UART*.

8) Kemampuan untuk melaksanakan operasi aritmetika dan operasi logika.

9) Kecepatan dalam melaksanakan instruksi per siklus 1 mikrodetik pada frekuensi 12 MHz.

10) Vcc digunakan sebagai catu daya.

11) GND digunakan sebagai ground.

12) Port 0 merupakan port pararel 8 bit dua arah. Posisi *Low Significant Bit* (LSB) terletak pada pin 39 dan Most Significant Bit (MSB) terletak pada pin 32

13) Port 1 merupakan port pararel 8 bit dua arah. Posisi LSB terletak pada pin 1dan MSB terletak pada pin 8

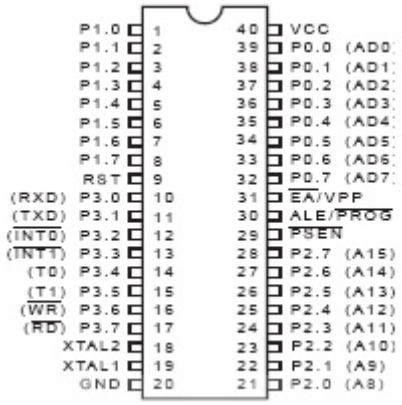
14) Port 2 merupakan port pararel 8 bit dua arah. *Port* ini mengirim byte alamat bila dilakukan pengaksesan memori eksternal pada pin 21 dan MSB terletak pada pin 28

15) Port 3 merupakan port pararel 8 bit dua arah. LSB terletak pada pin 10 dan MSB terletak pada pin 17. port ini mempunyai beberapa fungsi khusus seperti pada Tabel 9.1.

Tabel 9.1: Tabel fungsi pengganti Port 3

| Pin-pin pada port 3 | Fungsi Pengganti |
|---|---|
| P3.0 | RXD (port input serial) |
| P3.1 | TXD (port output serial) |
| P3.2 | INT0 (interrupt eksternal 0) |
| P3.3 | INT1 (interrupt eksternal 1) |
| P3.4 | T0 (input eksternal timer 0) |
| P3.5 | T1 (input eksternal timer 1) |
| P3.6 | WR (perintah write pada memori eksternal) |
| P3.7 | RD (perintah read pada memori eksternal) |

1) RST (reset) pada kondisi high akan aktif selama dua siklus




Gambar 9.2: Susunan pin AT89C51

2) ALE/PROG di gunakan untuk menahan alamat memory eksternal selama pelaksanaan instruksi

3) PSEN (Program Store Enable) merupakan strobe pembacaan ke memori eksternal

4) Jika EA/Vpp pada produksi low maka mikrokontroler menjalankan instruksi-instruksi yang ada pada memori internal

5) XTAL 1 sebagai masukan dari rangkaian osilator

6) XTAL 2 sebagai keluaran dari rangkaian osilator

## 9.2 Komunikasi Data Serial

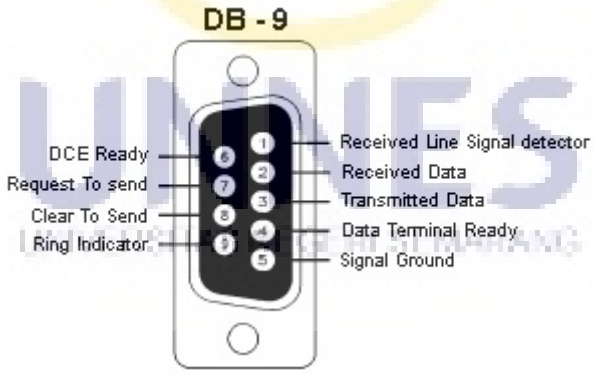
Komunikasi data pada umumnya dapat dilakukan dengan dua cara, yaitu secara serial dan secara pararel. Komunikasi data serial dilakukan dengan mengirim dan menerima data 8 bit secara satu per satu, sedangkan komunikasi data pararel dilakukan dengan mengirimkan dan menerima data 8 bit secara bersamaan atau sekaligus. RS232 *(recomende standard number 232*) merupakan seperangkat dari alat yang diciptakan oleh *electrical industry association* yang berfungsi anatar muka dalam mentransfer data dengan komputer yang mana pengiriman data dilakukan dengan kode biner.pada seperangkat komputer

biasanya tersedia *comunication port* atau sering disebut dengan COM. Biasanya terdapat dua buah *Communication port*, yaitu COM1 dan COM2. port tersebut biasanya digunakan untuk *mouse*.

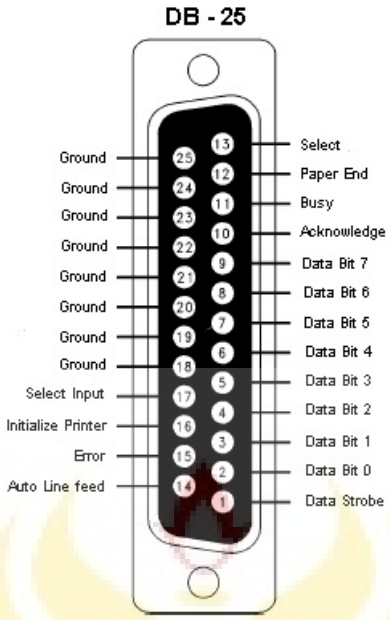
Ada dua jenis komunikasi data serial, yaitu komunikasi data serial sinkron dimana pengiriman *clock* dilakukan secara bersamaan dengan data serial. Komunikasi data asinkron, dimana pengiriman *clock* dilakukan secara 2 tahap, yaitu pada saat data dikirimkan dan diterima. RS232 pada komputer mempunyai 2 jenis konektor, yaitu konektor dengan 25 pin atau sering disebut *DB-25 Connector* dan konektor dengan 9 pin atau disebut *DB-9 Connector*. Pada dasarnya 3 pin yang terpakai, yaitu pin pengiriman, penerima, dan *ground*. Pengiriman data serial semakin jauh jarak kirim maka *noise* semakin besar.

Proses transfer data serial, RS232 memerlukan sebuah Data Terminal Equipment (DTE) dan *Data Communication Equipment* (DCE) pada masing-masing terminal. Pengiriman data dilakukan secara bit per bit, misalnya jika ingin mengirim karakter 'A'format ASCII adalah 41 heksa atau 1000001 biner maka data dikirim mulai bit pertama, kedua, sampai bit terakhir.

Kecepatan transfer data harus sama penerima dan pengirim, jika kecepatannya tidak sama akan terjadi *overflow*. Kecepatan transmisi sering disebut *baud rate*. *Baud rate* yang sering dipakai diantaranya adalah 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, dan 921600. panjang data bit yang sering digunakan di antaranya adalah 4, 5, 6, 7, dan 8 bit.




Gambar 9.3: Konektor serial DB-9



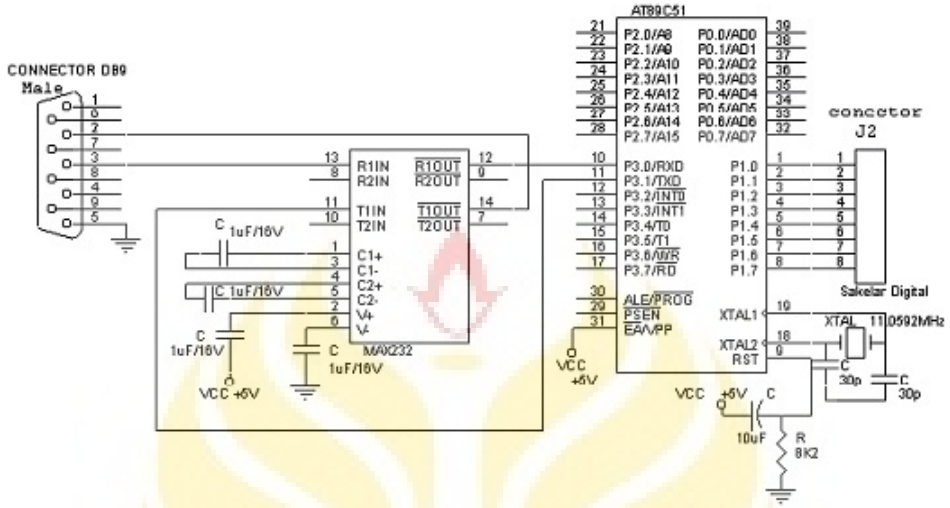

Gambar 9.4: Konektor serial DB-25

Pada komunikasi data serial pada dasarnya yang dikirimkan adalah tegangan dan kemudian dibaca dalam data bit. Besar level tegangannya adalah antara -25 volt sampai +25 volt. Untuk bit dengan logika 1 maka besar level tegangannya adalah antara -3 volt sampai -25 volt, sedangkan untuk bit dengan logika 0 maka besar level tegangannya adalah antara +3 volt sampai +25 volt. Ada beberapa besar level tegangan yang tidak mempunyai logika, yaitu antara -3 volt sampai +3 volt, lebih kecil dari -25 volt, dan lebih besar dari +25 volt.

## 9.3 Rangkaian Mikrokontroler AT89C51

Rangkaian mikrokontroler terdapat 2 port yang digunakan menampung input atau input data yang terhubung langsung oleh rangkaian dari alat pengendali. Rangkaian ini tersusun atas osilator kristal 11.0592 MHz yang berfungsi membangkitkan pulsa internal dan 2 buah kapasitor 30 pF yang berfungsi menstabilkan frekuensi. Kapasitor 10 $\mu$F dan resistor 8 k ohm berfungsi un-

tuk rangkaian reset sebelum program yang terdapat pada mikrokontroler dijalankan. Selain itu terdapat IC MAX 232 yang berfungsi untuk mensinkronkan komunikasi serial antara mikrokontroler dengan komputer. Konektor J2 pada port 1 terhubung ke rangkaian sakelar digital.




Gambar 9.5: Rangkaian Mikrokontroler

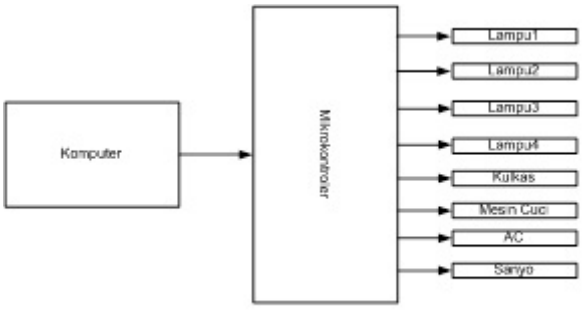
## 9.4 Rangkaian Sistem Perancangan

Pembahasan tentang perancangan sistem termasuk perangkat keras dan perangkat lunak akan dimulai dengan uraian singkat tentang proses sistem kendali perangkat elektronik.

Keterangan :

1) Komputer berfungsi sebagai pusat kendali dan bekerja dengan memerintahkan mikrokontroler untuk menyalakan atau mematikan peralatan elektronik dengan cara mengirimkan data delapan bit ke mikrokontroler.

2) Mikrokontroler berfungsi untuk menyalakan dan mematikan peralatan elektronik. Mikrokontroler akan bekerja bila menerima data dari komputer,




Gambar 9.6: Blok Diagram

tapi jika tidak menerima data maka mikrokontroler tersebut tidak akan bekerja.

3) Seperangkat peralatan elektronik yang terdiri dari lampu1, lampu2, lampu3, lampu4, kulkas, mesin cuci, AC, dan sanyo (pompa air). Untuk sakelar digital 5 sampai sakelar digital 8, bebannya diganti dengan lampu5, lampu6, lampu7, lampu8.

Sistem dirancang untuk melakukan instruksi pada data sakelar digital yang dikendalikan oleh mikrokontroler langsung diambil dari data yang dikirim oleh komputer. Data yang dikirim oleh mikrokontroler adalah sebesar 8 bit. Saat program aplikasi pada Visual Basic menyalakan lampu 1 maka data yang dikirim ke mikrokontroler adalah XXXXXXX1, sedangkan untuk mematikan lampu 1 maka data yang dikirim ke mikrokontroler adalah XXXXXXX0. data yang dikirim tersebut dalam bentuk Bit. Jika ada dalam bentuk Hexa, data tersebut adalah X1H dan X0H. Data X diatas dapat berupa bit 0 dan bit 1 atau sering disebut dengan data bebas, istilah tersebut jika dalam ilmu teknik disebut sebagai Dont care, maksudnya data dapat bernilai 1 atau 0.

## 9.5 Perancangan Logika

Rangkaian dalam mikrokontroler ini merupakan pusat dari pengolahan data dan pusat pengendali alat. Dengan 4 tombol saklar pada software dan output

8 saklar pada hardware, yang dimana tiap 1 saklar mempunyai output 2 saklar. Didalam rangkaian hardware sakelar 1 sampai sakelar 8 ini berfungsi untuk mengendalikan beban yang dapat digunakan lampu atau peralatan elektronik. Rangkaian sakelar ini dikendalikan langsung oleh mikrokontroler.

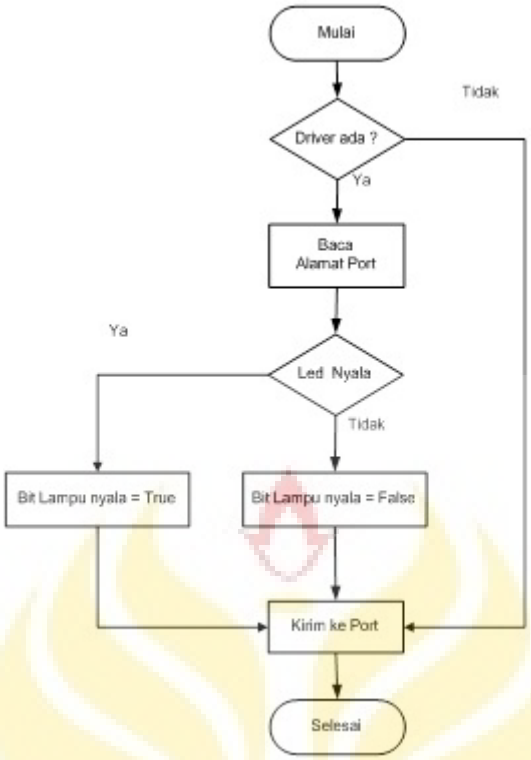
Rangkaian dan komponen pada sakelar 1 sampai sakelar 8 pada dasarnya sama, perbedaannya terdapat pada port pengendali ke mikrokontroler. Sakelar 1 dan sakelar 2 terhubung ke alamat P1.0, sedangkan sakelar 3 dan sakelar 4 terhubung pada alamat P1.1, sakelar 5 dan sakelar 6 terhubung pada alamat P1.2, sakelar 7 dan sakelar 8 terhubung pada alamat P1.3.

Dalam Gambar 9.7 *flowchart* ini diberikan tentang asumsi-asumsi dan ketentuan-ketentuan sebagai berikut :

- True = 1
- False = 0
- Bit lampu nyala berfungsi sebagai pengendali nyala atau matinya lampu.

Pada Gambar 9.7 merupakan *flowchart* program utama perangkat lunak mengatur interface pengendali lampu berjalan sebagai berikut :

1) Mulai
2) Pemeriksaan ada tidaknya driver dari interface yang digunakan. Jika ada kemudian membaca alamat dari interface atau port, jika tidak ditemukan maka Flowchart langsung selesai
3) Pemeriksaan apakah akan menyalakan beberapa lampu, jika ya maka lakukan procedure nyala, jika tidak bit lampu nyala di set False dan lakukan pemeriksaan selanjutnya.
4) Periksa apakah akan menyalakan semua lampu, jika ya lakukan procedure semua jika tidak bit lampu nyala di set False.
5) Bit lampu nyala dikirim ke port.
6) Selesai



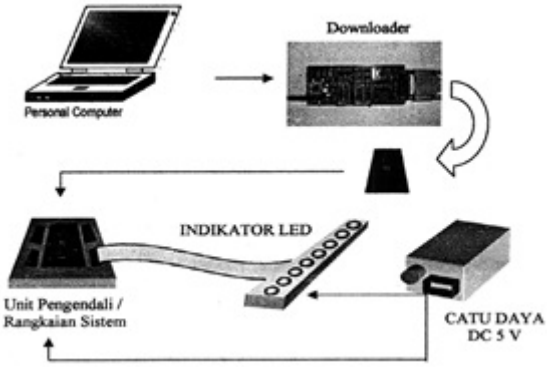

Gambar 9.7: Flowchart program

## 9.6 Pengujian Rangkaian Unit Kendali

Pengujian rangkaian unit pengendali utama atau mikrokontroler AT89C51 dilakukan untuk mengetahui apakah unit kendali yang digunakan bekerja dengan baik atau tidak. Pengujian pada unit kendali ini memerlukan catu daya 5 volt, konektor DB9 female, kabel data (RS232) yang terhubung dengan indikator berupa susunan 8 buah led untuk mengetahui keluaran dari port mikrokontroler yang digunakan yaitu port 1 dan port 3 serta *donwloader*.

Pada rangkaian indikator bagian anoda LED dihubungkan ke pin keluaran mikrokontroler, sedangkan bagian katodanya dihubungkan ke tegangan 0 Volt DC (*ground*). Kombinasi ini ditujukan untuk menguji jalannya program dan keluaran yang dihasilkan bila tak diberikan logika khusus oleh kita (menurut *data book*, keluaran mikrokontroler akan selalu berlogika tinggi/*high*).

Setiap port yang digunakan dalam sistem ini mempunyai fungsi masing-




Gambar 9.8: Pengujian Mikrokontroler AT89C51

masing sehingga perlu dilakukan pengujian agar sistem dapat berjalan dengan baik. Pengujian dilakukan dengan cara sederhana yaitu keluaran setiap port mikrokontroler mempunyai 4 masukan P1.0, P1.1, P1.2, P1.3 yang dihubungkan dengan 8 unit led sebagai keluarannya untuk mengetahui port bekerja dengan baik atau tidak. Kondisi 8 unit led tersebut disesuaikan dengan program yang dibuat, program tersebut dibuat melalui aplikasi ASM51 untuk memperoleh file dengan ekstention *.hex. Sedangkan untuk memasukkan program tersebut ke dalam memori program IC mikrokontroler AT89C51 diperlukan rangkaian *donwloader*.

Dibawah ini akan diuraikan tahapan-tahapan cara penggunaan alat sebagai berikut :

- Sambungkan Lampu ke sakelar pada konektor alat kendali. Lampu jangan di hubungkan terlebih dahulu ke listrik.
- Sambungkan Alat kendali dengan komputer melalui kabel serial DB9.
- Sambungkan Alat kendali dengan *power supply* / catu daya.
- Sambungkan lampu ke listrik.
- Jalankan program visual basic kendali, lalu klik icon lampu yang akan di kendalikan.

Berdasarkan hasil pengujian dan analisa terhadap sistem, maka dapat diperoleh bahwa kendali elektronik dengan memerintahkan mikrokontroler

AT89C51 dapat beroperasi untuk menyalakan atau mematikan peralatan elektronik dengan cara mengirimkan data delapan bit ke mikrokontroler. Dengan menggunakan instruksi *Visual Basic* dan mikrokontroler maka sistem pengaturan peralatan elektronik dapat bekerja sesuai yang diharapkan dan kendali elektronik ini mempunyai fungsi sebagai pengganti saklar.

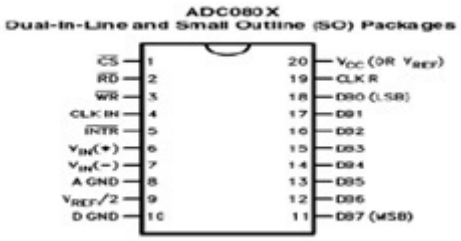
# Bab 10

# AT89S52 UNTUK KENDALI SUHU

Suhu merupakan keadaan tingkat panas atau dingin pada benda padat, cair ataupun gas. Tingkatan suhu pada suatu ruang dapat diukur dengan menggunakan sensor suhu yang terpasang pada ruang tersebut. Besaran suhu dinyatakan dengan derajat, dan untuk satuan yang lazim digunakan di Indonesia adalah dalam satuan derajat celcius. Besaran suhu tidak bisa langsung diterima oleh komponen elektronik, sehingga perlu perantara pengubah keadaan suhu menjadi besaran elektronik. Piranti atau sensor ini, adalah LM35D, karakteristik LM35D antara lain :

a. Kalibrasi output linier terhadap satuan celcius
b. Mempunyai output yang linier, yaitu 10.0 mV/ $^{o}$C
c. Mempunyai jangkah pengukuran -55 $^{o}$C sampai dengan +150 $^{o}$C
d. Mempunyai impedansi keluaran (*output*) yang cukup rendah, yaitu 0,1 ohm untuk kuat arus 1 mA beban

Dalam dunia komputer, semua nilai tegangan dijadikan dalam bentuk digital, dan menggunakan sistem bilangan biner. Untuk itu dalam sistem ini, karena *output* dari sensor suhu berupa tegangan analog, maka diperlukan pengubah tegangan analog ke digital.




Gambar 10.1: Simbol ADC 0804

IC ADC0804 adalah IC ADC 8-bit dengan diagram pewaktuan yang kompatibel dengan mikrokontroler. Fitur yang disajikan dalam IC ADC 0804 antara lain:

- Mempunyai timing bus yang kompatibel dengan mikroprosesor dan mikrokontroler.
- Mempunyai internal Clock Generator
- Resolusi 8-bit
- Mempunyai waktu konversi 100S (maksimal).

Dalam pembuatan sistem pengaturan suhu digunakan beberapa komponen elektronika sebagai berikut:

1. Aktuator
   Pada sistem pengaturan suhu ini, digunakan satu aktuator yaitu, pemanas berupa lampu pijar.
2. Triac
   Untuk dapat bekerja dengan sumber tegangan bolak balik, komponen yang biasa digunakan adalah Thirystor. Salah satunya adalah Triac, Triac merupakan saklar elektronik yang sangat ideal untuk mengatur daya arus bolak-balik. Kombinasi Triac dan mikrokontroler menghasilkan sistem pengaturan daya yang sangat fleksibel dan akurat.
3. Transistor
   Transistor merupakan komponen aktif dimana arus, tegangan atau daya

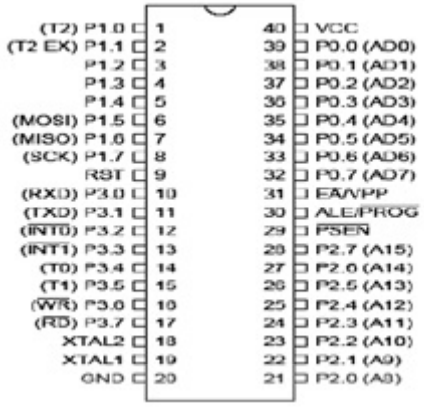
keluarannya dikendalikan oleh arus masukan. Di dalam sistem komunikasi, transistor digunakan untuk menguatkan sinyal. Di dalam untai elektronis komputer transistor digunakan untuk saklar elektronis laju tinggi. Transistor terdiri dari 3 terminal yaitu basis, kolektor dan emitor. Ada dua transistor yaitu PNP dan NPN.

Mikrokontroler merupakan piranti terprogram, yang sudah diengkapi dengan unit pengolah pusat (CPU), Memory (RAM, ROM), dan juga Unit I/O. Mikrokontroler yang digunakan pada sistem ini adalah mikrokontroler kelurga Intel 80xx, dengan pabrikasi dari Atmel. Fitur yang disajikan dalam mikrokontroler ini sebagai berikut:

- Mempunyai 8-KB *memory program* (PEROM)
- In Sistem Programming
- Frekuensi osilator sampai dengan 33 MHz
- Tiga tingkat *memory lock*
- 256 X 8 internal RAM
- 3 buah *Timer/Counter*
- Dual Data Pointer
- 32 Jalu I/O
- Dilengkapi dengan *Watchdog timer*

## 10.1 Rancangan Untuk Pengering

Rancangan penelitian ini terdiri dari tiga bagian, yaitu perancangan mekanik pengeringan, perancangan elektronik, dan perancangan perangkat lunak program pengendalian.




Gambar 10.2: Simbol AT89S52

### 10.1.1 Perancangan mekanik pengering

Bagian ruang oven pengering yang akan digunaan sebagai berikut:

1. Elemen pemanas (*Heating Coils*)
   Fungsi elemen pemanas adalah memasok energi thermal ke dalam ruang oven sehingga udara dalam ruang akan menjadi panas juga.

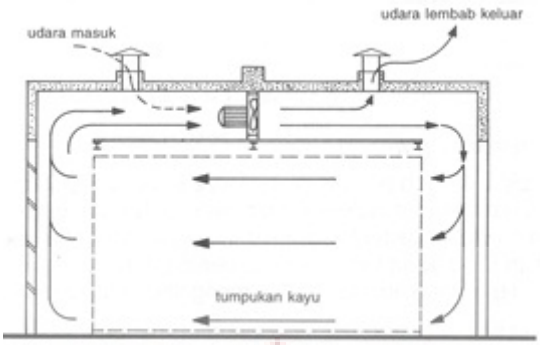
2. Kipas sirkulasi (*Fans Implller*)
   Kipas merupakan alat penggerak utama sirkulasi udara dalam ruang oven agar udara panas dapat merata ke seluruh ruangan. Udara yang bergerak dapat ditekan masuk di antara celah-celah pada tumpukan.

3. Plafon antara digunakan untuk mengarahkan sirkulasi udara dalam ruang oven sehingga dapat terbentuk suatu tekanan gerak udara yang kuat, merata dan tidak menyebar.

4. Cerobong buang (*Dumper*)
   Cerobong buang digunakan untuk membuang udara lembab dari dalam ruang oven dan menghisap udara luar masuk ke dalam oven.

5. Sensor suhu
   Sensor suhu ini berupa LM35D yang dihubungkan ke panel elktronik

Secara garis besar blok diagram perancangan perangkat keras alat dapat dilihat pada Gambar 10.3. Mesin pengatur suhu yang dibuat berbentuk kotak




Gambar 10.3: Mekanik pengering

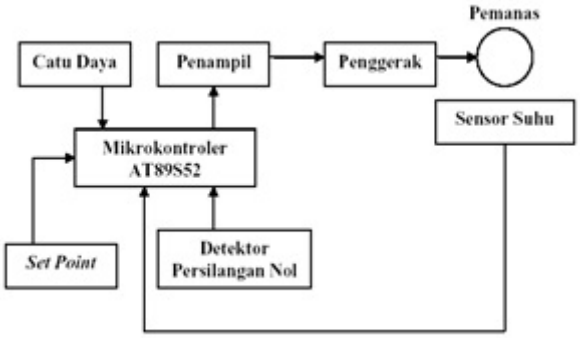
berdimensi (180 180 150) cm dengan susunan rak rangkap 3. Mesin terdiri atas elemen pemanas 100 W, 220 V sebanyak 4 buah. Untuk pendinginan, mesin dilengkapi 2 kipas angin 12 volt DC yang diletakkan di atas kotak. Panel kendali diletakkan di bagian atas kotak. Otomatisi mengandalkan sensor suhu yang selanjutnya sinyal analog diubah ke sinyal digital. Energi berasal dari listrik, sebesar 300 W saat kerja yang hanya berfungsi saat cuaca dingin. Sumber energi ini dikondisikan agar sesuai dengan kebutuhan, yaitu untuk menghidupkan elemen pengendali. Mesin ini berventilasi untuk memberikan kelembaban udara yang diinginkan.

## 10.2 Perancangan Perangkat Keras

Secara garis besar sistem diagram blok perancangan perangkat keras alat dapat ditunjukkan pada Gambar 10.4.

1. Set Point

   Set Point merupakan bagian masukan sistem, dalam hal ini user bisa menggganti nilai set point dengan menekan tombol. Tombol terdiri dari keypad yang terhubung dengan mikrokontroler. Keypad menggunakan sistem active low, artinya saat terjadi penekanan tombol maka keypad akan mengirim




Gambar 10.4: Diagram blok sistem

sinyal low ke mikrokontroler. Jika tidak ada penekanan tombol maka logika high akan diberikan ke mikrokontroler karena adanya pull-up dengan VCC melalui R1. Nilai resistor yang biasa digunakan sesuai dengan data sheet adalah 10K.

2. Penampil

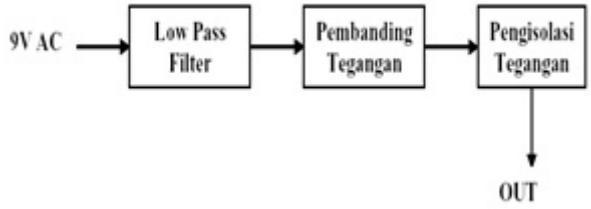
   Untuk menampilkan informasi suhu dan informasi nilai set point diperlukan media berupa display. Penampil yang digunakan adalah LCD dengan tipe dot matrik 16x2 keluaran seico.

3. Sensor Suhu

   Sensor suhu yang digunakan dalam alat ini adalah IC LM35DZ keluaran National Semiconduktor. Alasan digunakannya IC ini adalah sifat linier-nya terhadap perubahan suhu. IC ini bekerja dengan skala celcius, yaitu mempunyai resolusi 10mV tiap derajat celcius. Artinya, setiap kenaikan satu derajat celcius, maka tegangan output sensor akan naik 10mV. Sebagai perumpamaan ketika suhu 10 C mempunyai tegangan output 100 mV, maka saat suhu 11 $^{o}$C tegangan outputnya 110mV, dan saat suhu 20 $^{o}$C maka output tegangannya 200 mV.

4. Detektor Persilangan Nol

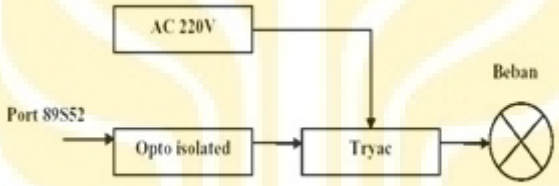
   Detaktor persilangan nol diperlukan untuk mengetahui saat tegangan PLN mencapai titik nol. Hal ini diperlukan sebagai acuan pemberian sudut picu kepada Thirystor.




Gambar 10.5: Blok diagram detektor persilangan nol

5. Penggerak Pemanas

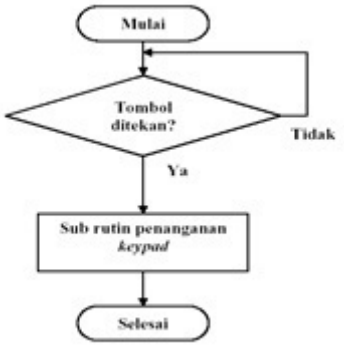
   Pada sistem pengaturan suhu ruang, pemanas yang digunakan berupa lampu pijar AC 220V 100W. Untuk besarnya pemanasan kita menggunakan pengaturan intensitas lampu dengan Phase Power Control, Diagram kotak perantaraan mikrokontroler ke pemanas disajikan pada Gambar 10.6.




Gambar 10.6: Diagram kotak penggerak pemanas

## 10.3 Perancangan Perangkat Lunak

Perangkat lunak yang dibuat mengacu pada perangkat keras yang terhubung pada komputer. Fungsi perangkat lunak untuk mengendalikan sistem kerja perangkat keras agar sesuai dengan yang diinginkan. Set Point Adapun subrutin dasar untuk penanganan keypad kita menggunakan teknik polling biasa tanpa interupsi, yaitu dengan menunggu adanya kondisi low pada pin mikrokontroler. Sebagai gambaran dasar penanganan keypad, dapat dilakukan dengan alur seperti Gambar 10.7.




Gambar 10.7: Alur pelayanan penekanan tombol

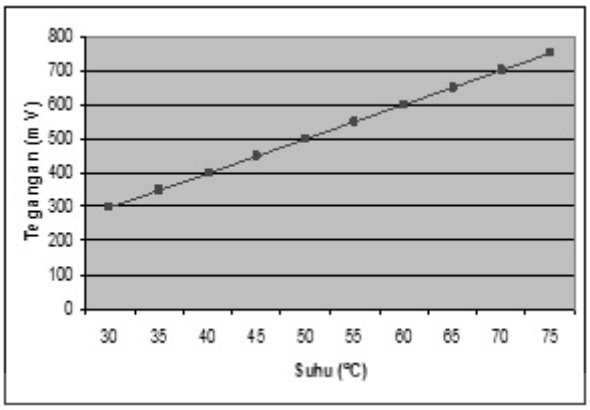
## 10.4 Perangkat Keras

Pada bagian ini akan dibahas prinsip kerja dan data pengamatan yang telah dilakukan sebelumnya. Prinsip kerja yang dibahas adalah prinsip kerja per blok, yaitu : sensor suhu, pengubah tegangan ke frekuensi, *keypad*, *zero crossing detector*, penggerak/*driver* dan rangkaian daya.

### 10.4.1 Sensor Suhu

Sensor suhu LM35 ini mengubah besaran suhu ($^o$C) sekitar sensor menjadi besaran tegangan dengan keluaran 0 mV saat suhu 0 $^o$C dan 1000mV DC saat suhu 100 $^o$C. Dengan demikian setiap kenaikan suhu 1 $^o$C sensor suhu LM35 memberikan kenaikan tegangan keluaran 10 mV DC. Berikut data hasil pengamatan keluaran IC LM35 pada suhu yang berbeda-beda. Semakin tinggi suhu yang dibaca sensor, maka semakin tinggi tingkat kesalahan yang diberikan, hal ini menunjukkan bahwa sensor suhu LM35 kurang linier. Grafik hubungan antara suhu yang dibaca dan tegangan keluaran sensor suhu LM35 ditunjukkan pada Gambar 8.10.

Bagian *keypad* terdiri dari tiga tombol yauitu tombol Set/Ok, Up dan Down yang berfungsi untuk memasukan data seting suhu pada alat. Tombol Set/Ok berfungsi untuk memulai dan mengakhiri proses memasukan data seting, tombol Up berfungsi untuk menaikan data seting dan tombol Down berfungsi untuk menurunkan data setting. Pada dasarnya semua tombol ter-




Gambar 10.8: Grafik tegangan keluaran sensor suhu LM35 terhadap suhu

sebut bekerja dengan memberikan logika rendah ke mikrokontroler. Berikut data hasil pengamatan bagian keypad.

### 10.4.2 *Zero Crossing Detector*

Pada *zero crossing detector* ini sinyal dari jala-jala PLN akan diubah ke dalam bentuk pulsa dengan menggunakan komparator, masukan dari komparator ini diambil dari sekunder transformator catu daya yang memiliki tegangan 12 volt yang sudah dilewatkan resistor secara seri sebesar 100 K ohm. Keluaran dari komparator yang berupa pulsa yang memiliki frekuensi sama dengan frekuensi jala-jala PLN ini kemudian diubah oleh differensiator menjadi pulsa yang sangat sempit, dan setelah diubah menjadi pulsa yang sempit pulsa ini disempurnakan oleh schmitt trigger agar menjadi pulsa yang kotak.

Pada *zero crossing detector* keluaran komparator IC 4558 mempunyai gelombang kotak dimana lereng dari pulsa sama dengan saat sinyal dari jala-jala PLN melintas titik nol volt (frekuensi dari pulsa keluaran komparator sama dengan frekuensi dari jala-jala PLN) dengan amplitudo sama dengan amplitudo dari catu daya komparator tersebut yaitu sebesar 8.98 volt.. Keluaran dari differensiator berupa pulsa yang mempunyai amplitudo maksimum 5 volt, hal ini disebabkan karena adanya diode bypass 1N4002 yang dipasang dari keluaran differensiator ke +Vcc. Keluaran dari differensiator ini akan masuk ke

IC 74LS132 yang berupa gerbang NAND yang juga sebagai schmitt trigger. Bentuk keluaran dari differensiator yang masuk ke schmitt trigger akan dimantapkan menjadi gelombang kotak. Pulsa yang dihasilkan dari keluaran schmitt trigger mempunyai lebar sebesar 200 $\mu$S. Dengan pulsa sebesar 200 $\mu$S ini, maka mikrokontroler sudah dapat diinterupsi.

### 10.4.3 Driver

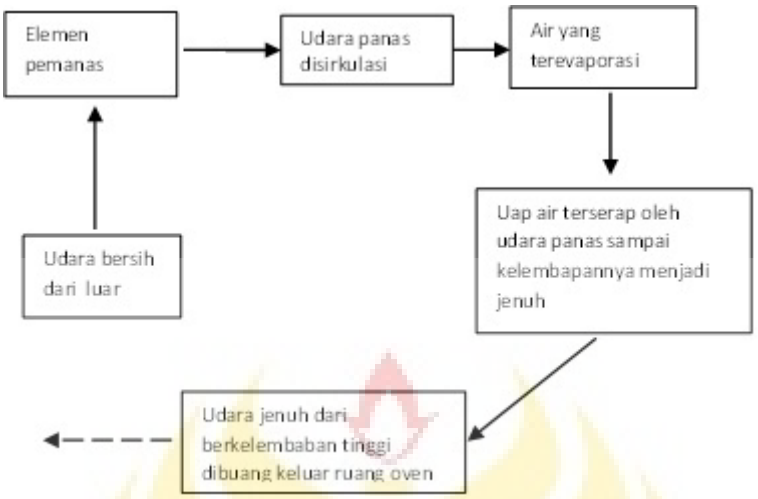
Rangkaian *driver*/penggerak ini digunakan untuk memicu *gate* dari *triac*, sehingga *triac* akan terhubung. Driver yang digunakan adalah MOC 3021 yang merupakan komponen yang terdiri dari diode infra merah dan *photo triac*. Sinyal penyulutan pada photo triac dilakukan dengan memberikan cahaya yang akan dilakukan melalui diode infra merah. Diode infra merah akan on jika dialiri arus yang sesuai dengan karakteristik dari optoisolator (pada optoisolator ini arus yang mengalir kurang lebih sebesar 9.48 mA (tidak dilakukan pengamatan)). Besarnya arus yang mengalir ke diode infra merah tidak diamati karena pulsa yang diberikan hanya 5 $\mu$S sehingga tidak bisa diamati. Pengamatan yang dilakukan adalah sudut pemicuan untuk menyulut *triac*. Pengamatan dilakukan dengan mengamati bentuk gelombang pada masukan driver (diode infra merah) dan dibandingkan dengan bentuk gelombang pada sekunder transformator.

Perbandingan antara sudut picu perancangan dengan sudut picu pengamatan terjadi perbedaan, hal ini disebabkan karena adanya faktor kesalahan yang diakibatkan salah membaca osiloskop dan dikarenakan waktu satu cycle tidak 1 $\mu$S (kurang lebih 1.18 $\mu$S), dan adanya waktu untuk mengeksekusi instruksi program

### 10.4.4 Pengujian Alat

Sistem ini dengan elemen pemanas dalam ruang oven yang menyebabkan udara dalam ruang terinduksi panas. Kemudian udara panas disirkulasikan oleh kipas-kipas dan diarahkan dengan menggunakan plafon antara (sub-celling). Bila udara panas ini sudah jenuh dengan uap air yang dievaporasi dari ikan, maka udara itu akan dibuang melalui cerobong pembuangan damper dan pada

saat yang sama di masukkan udara bersih ke dalam ruang. Skema prinsip kerja oven sistem konvensional ditunjukkan pada Gambar 10.9.




Gambar 10.9: Skema prinsip kerja oven konvensional

Proses pengendalian suhu ruangan dilakukan dengan membandingkan nilai seting dengan suhu ruangan tersebut untuk menyalakan atau mematikan lampu pemanas. Proses pembacaan suhu dilakukan oleh sensor suhu LM35 kedalam nilai tegangan DC tertentu sesuai nilai suhu yang dideteksi. Untuk mengendalikan rangkaian daya dengan sumber catu yang berbeda dengan catu mikrokontroler (tegangan AC), mikrokontroler membutuhkan antarmuka dan driver (opto isolator). Untuk mempercepat proses pemanasan suhu alat penetas telor dapat dilakukan dengan memperbesar ukuran daya lampu. Penambahan beban (lampu pemanas) dapat dilakukan dengan memasang lampu secara paralel. Apabila menginginkan susunan tombol menu diubah urutannya dapat dilakukan dengan mengganti inisialisai *port* pada program utama.

# Bab 11

# CONTOH SENSOR YANG DAPAT DIGUNAKAN

## 11.1 Sensor Gelombang Ultrasonik

Sensor ultrasonik merupakan sensor yang bekerja dengan cara memancarkan gelombang dan kemudian menghitung waktu pantulan gelombang tersebut. Secara umum sensor ultrasonik digunakan untuk menghitung jarak dari suatu objek yang berada didepan sensor tersebut. Sehingga dengan fungsinya tersebut, sensor ultrasonik biasa digunakan pada perangkat yang membutuhkan perhitungan jarak. Contoh : smart robot, Kapal laut, kapal selam, dan lain-lainnya.

## 11.2 Sensor Infra merah

Sistem sensor ini pada dasarnya menggunakan inframerah sebagai media komunikasi yang menghubungkan antara dua perangkat. Penerapan sistem sensor infra ini sangat bermanfaat sebagai operator jarak jauh, alarm keamanan dan otomatisasi pada sistem. Adapun transmisi pada sistem ini terdiri atas sebuah LED (*Light emitting diode*) infra merah yang telah dilengkapi dengan jaringan yang mampu membangkitkan data untuk dikirimkan melalui sinar

Gambar 11.1: Sensor Gelombang Ultrasonik

inframerah, sedangkan pada bagian penerima biasanya terdapat foto transistor, foto dioda, atau modulasi infra merah yang berfungsi untuk menerima sinar inframerah yang dikirimkan oleh pemancar.

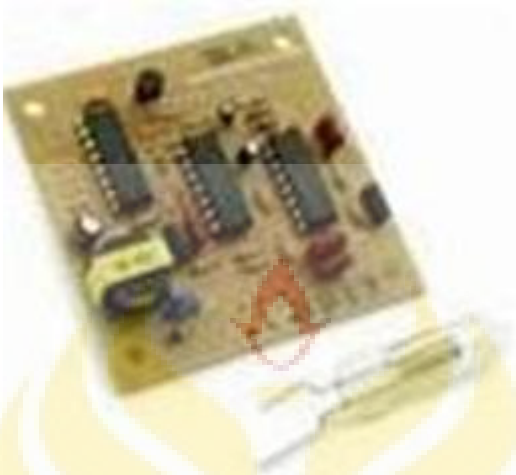
Gambar 11.2: Sensor Inframerah



## 11.3 Sensor *UV-Tron*

Sensor api UV-Tron adalah sebuah sensor yang mendeteksi adanya nyala api yang memancarkan sinar *ultraviolet*. Pancaran cahaya *ultraviolet* dari sebuah nyala lilin berjarak 5 meter dapat dideteksi oleh sensor ini. Sensor api *UV-Tron* biasanya digunakan pada lomba robot, seperti KRCI (kontes robot cerdas indonesia) yang berfungsi mendeteksi keberadaan lilin yang akan dipadamkan

oleh sirobot. Untuk lebih mudah jika anda berniat untuk membeli sensor *UV-Tron* sebaiknya langsung membeli lengkap dengan drivernya alias membeli yang sudah jadi modul yang siap dipakai.



Gambar 11.3: Sensor UVTron

Sensor UV-Tron akan mengeluarkan logika *high* (1) jika ia mendeteksi keberadaan api dan sebaliknya sensor UV-Tron akan mengeluarkan logika *low* (0) jika ia tidak mendeteksi api, anda bisa mengecek keluarannya dengan multimeter analog. Perlu diketahui, *output* yang dikeluarkan adalah sinyal kotak dengan frekuensi yang bergantung pada kapasitor yang digunakan pada driver. Pemilihan kapasitor driver harus disesuaikan dengan kebutuhan, jika kita ingin mendapatkan output dengan sampling yang lebih cepat maka gunakan kapasitor dengan kapasitansi yang lebih kecil (biasanya 0.01 $\mu$ F), sebaliknya jika ingin sampling yang lebih lambat gunakan kapasitansi kapasitor yang lebih besar (misal 1 F). Biasanya nilai kapasitansi 0.01 $\mu$ F memiliki periode sampling 0.01 s begitupun untuk 1 $\mu$ F memiliki periode sampling 1s kira-kira begitu.

## 11.4 Sensor Kompas

Sensor kompas adalah sensor yangmampu mendeteksi arah secara horisontalterhadap medan magnet bumi. Sensor inimemanfaatkan efek magnetoresistive untuk mengetahui arah medan magnet yang melewatisensor. Efek magnetoresistive adalah perubahannilai hambatan pada suatu penghantar akibatdari perubahan arah medan magnet yangmelewatinya.

Gambar 11.4: Sensor Kompas

# Bab 12

# PENUTUP

Mikrokontroler memiliki unit memori sendiri, unit I/O (*Input/Output*) yang bisa dikoneksikan langsung dengan sensor atau aktuator. Program disimpan dalam memori yang tidak hilang bila catu daya padam, biasanya dalam bentuk ROM, PROM atau EPROM diluar mikrokontroler, atau beberapa seri atau varian memiliki ROM di dalam mikrokontroler itu sendiri. Cara mengisi program dengan suatu alat pemrogram, yang biasanya berhubungan dengan PC. Untuk mempelajari dan mengaplikasikan mikrokontroler diperlukan perangkat pengembang, literatur dan forum-forum diskusi.

Salah satu aplikasi dalam dunia elektonika dan komputer adalah mikrokontroler. Mikrokontroler merupakan pengontrol mikro atau disebut juga *Single Chip Microcomputer* (SCM). Mikrokontroler dapat digunakan sebagai sistem pengendali suatu aplikasi tanpa menggunakan bantuan dari komputer, dengan kata lain mikrokontroler ini bersifat stand alone. Penggunaan mikrokontroler sebagai pengontrol micro sangatlah tepat. Sebagai contoh aplikasinya, seperti pengontrol temperatur, penampil display *Liquid Crystal Display* (LCD), pemroses sinyal digital, pemroses dan pengontrol mesin-mesin industri, dan sebagainya. Tetapi yang paling banyak digunakan dan yang paling cepat berkembang yaitu dalam dunia robot. Sebagai objek yang penting dalam dunia teknologi, robot dapat memberi sumbangan yang sangat berarti dalam memperbaiki efisiensi dan efektivitas suatu industri. Keberadaannya sangat penting dan pemakaiannnya yang berkembang dari waktu ke waktu.

Buku ini bertujuan memberikan alternatif untuk memahami mikrokontroler yang dapat kita lakukan dalam kondisi keterbatasan alat. Memang terkadang didapatkan hasil riset yang kurang eksak, tetapi cukup memberikan informasi tentang mikrokontroler sebagai kendali elektronik yang sedang kita pelajari. Jelas, hal ini lebih baik dari pada kita diam saja tidak melakukan apa-apa dan mengeluh karena keterbatasan peralatan yang ada. Penulis merencanakan untuk menyusun kelanjutan dari buku ini yang berkaitan dengan mikroktroler *Basic Stamp* dengan bahasa pemrograman *Pbasic* yang relatif sederhana, bahkan dapat menghindari algoritma yang sangat rumit.

# DAFTAR PUSTAKA


[1] Agfianto.2002. *Belajar mikrokontroler AT89C51/52/53 (Teori dan Aplikasinya)*. Penerbit Gava Media.Yogyakarta.

[2] Anonim.2000. *Basic Stamp (Programming Manual Version 1.9)*. Parralax, Inc .California.

[3] Atmel. 2003. *AT89 Series Hard ware Description*. USA: Atmel Inc (http://www.atmel.com)

[4] M. Ahsan Muslim.2004 *Kumpulan subrutin untuk berbagai Chip Interface Berbasis AT89C51*. Laporan Kerja Praktek II, ITENAS.

[5] Motorola Semiconductor. 1995. *MOC3021 6 Pin DIP Random Phase Optoisolator Triac Driver output*. Hong Kong : Motorola Semiconductor Inc. (http://www.Motorola-DesignNET.com)

[6] National Semiconductor Corporation. (November. 2003). *LM78XX Series Voltage Regulators*. USA : National Semiconductor Inc (http://www.national.com)

[7] Retna Prasetia dan Catur Edi Widodo.2004 *Teori dan Praktek Interfacing Port Pararel dan Port Serial Komputer dengan Visual Basic 6.0*. ANDI, Yogyakarta.

[8] Sutrrisno. 1987. *Elektronika Lanjutan. Dikat kuliah Fisika ITB*. Bandung.kedua. Alih bahasa Sutisna. Penerbt Erlangga. Jakarta.

[9] Suhata, ST.2005. *Aplikasi Mikrrokontroler Sebagai Peralatan Elektronik*. Elex Media Komputindo, Jakarta.

[10] Suhata, ST.2005. *VB Sebagai Pusat Kendali Peralatan Elektronik*. Elex Media Komputindo, Jakarta.

[11] Teccor Electronics. 2003. *Thyristor Catalog 0,8A-35A*. USA : Teccor Electronics Thyristor Product Catalog. (http://www.teccor.com)

[12] Wasito S. 1997. *Data Sheet Book 1*. PT Elek Media Komputindo, Jakarta

[13] http://www. *parallaxinc.com*

[14] http://www. *Stampinclass.com.*

[15] http://www. *iptek.net.id/ind/?ch*=jsti&id=263,

[16] http://www. *teccor.com*

# Indeks


Register IE-Interrupt Enable , 18
*baut rate*, 52
*(decimal add adjust)*, 7
*(statement)*, 27
*Central Processing Unit*, 1
*Common Anode*, 68
*Internal/ External RAM*, 23
*Interrupt Enable*, 120
*Interrupt Priority*, 18
*Light emitting diode*, 113
*SMS Gateway*, 3
*Short Jump*, 34
*Single Chip Microcomputer*, 2
*Source Listing*, 23
*Stack Pointer*, 37
*TL2 dan TH2*, 52
*Timer Control Register*, 11
*Timer/Counter*, 10
*Zero Crossing Detector*, 110
*automatic vehicel locator*, 3
*chip*, 1–3
*count up binary counter*, 10
*decrement*, 7
*flow chart*, 27
*hopping*, 3
*look-up table*, 6
*output*, 115
*overflow*, 11
*photo triac*, 110
*port serial full duplex*, 5
*port*, 112
*timer(counter)*, 5
*triac*, 110
*ultraviolet*, 114

bit carry, 8
diode bypass 1N4002, 110
EXEN = 1, 52
IC 74LS132, 110
LED, 68
RAM, 1
*Absolute Jump*, 34

akumulator, 37
amplitudo, 110
AT89C51, 6, 7
AT89C51/52, 10
AT89C52, 15
Atmel AT89C51, 4

biner, 11
bit, 8, 11
bit carry, 9
bit IP, 18
byte, 6, 19, 34, 37

CMOS, 4

debug, 27
driver, 110

EEPROM, 4
Emulator, 23
EPROM, 2, 5

flip-flop, 19, 68
flow chart, 27
format heksadesimal, 23
foto dioda, 113

Gelombang Ultrasonik, 112

I/O, 1–3, 5
IC, 2
IC TTL dan CMOS, 2
Instruksi MOV, 37
instruksi AJMP, 34
instruksi CALL LABEL, 19
Instruksi JUMP, 34
Instruksi POP, 37
interupsi, 15
Interupsi Timer, 52

Keluarga MCS-51, 4

LED, 27, 114
LJMP, 34
logika *high*, 115
logika *low*, 115
loop, 27

magnetoresistive, 115
Memori Data External, 37
memori program, 8, 9
mikrokontroler, 2–4, 6, 112
mikrokontroler AT89C51, 7, 9, 10

Operasi logika, 9
opto isolato, 112
optoisolator, 110
osilator, 5
oven, 114

pencacah biner 8 bit, 15
pencacah biner, 15
Pencacah Biner 16 Bit dan 8 Bit, 15
Pengaktifan Interupsi, 120
pengendali kecil, 2
penggerak 7 Segment, 68
Pewaktu/pencacah 16 bit, 15
port 3, 5
Prioritas Interupsi, 18

register, 8
register IE, 19
register IE AT89C51/52, 120
register IP, 19
register TCON, 11
register TMOD, 15
Reset, 19
RETI, 19
ROM, 1

Sensor Infra merah, 113, 114
Sensor Kompas, 115
Sensor *UV-Tron*, 114
SFR, 7, 18
simulator AT89C51, 23

smart robot, 112, 113
Sub-rutin delay, 68

TCLK = 1, 52
telemetri, 3
TFx, 11
tipe 89C51, 4
TMOD, 15
tombol Set/Ok, 110
TS Controls Emulator 8051, 23

# GLOSARIUM

**Bit** : Merupakan singkatan dari Binary Digit. Sebuah bit atau digit biner adalah sebuah sinyal yang masing-masing berada dalam dua kondisi yaitu 0 dan 1.

**Byte** : Dalam system computer byte bilangan biner sebanyak 8 bit dan 1 karakter dapat disebut dengan satu byte. Pemakaian yang luas dalam komunikasi data dalam system computer mikro disebut American Standar For Information Interchange (ASCII).

**Clock** : Pulsa lonceng yang berbentuk gelombang kotak yang tidak mempunyai kondisi setimbang untuk mengontrol serialisasi dan deserialisasi.

**CISC kependekan dari** *Complex Instruction Set Computer* : Instruksi bisa dikatakan lebih lengkap tapi dengan fasilitas secukupnya.

**Driver** : Penggerak yang digunakan untuk memicu *gate* dari *triac*, sehingga *triac* akan terhubung.

**EPROM** *(Eraseable Programmable Read Only Memory)* : Untuk penyimpanan program.

**Fitur-fitur spesial** *(special features)* : Fitur khusus untuk mendukung aplikasi tertentu seperti fasilitas pemrosesan *floating point*, multimedia dan sebagainya.

**Karakteristik mikroprosesor** : *chip*, *Central Processing Unit*, RAM, ROM dan I/O.

**Kecepatan clock** *(clock speed)* : Rate atau kecepatan clock untuk menuntun kerja mikroprosesor.

**Mikroprosesor** : bagian CPU*(Central Processing Unit)* dari suatu komputer, tanpa adanya memori, I/O dan periferal yang dibutuhkan oleh sebuah system lengkap.

**Mikrokontroler** : Alat elektronika digital yang mempunyai masukan dan keluaran serta kendali dengan program yang bisa ditulis dan dihapus dengan cara khusus.

**Mikrokontroler** : Pengendali kecil.

**Mikrokomputer** : Kombinasi mikroprosesor dengan suatu I/O *(Input/Output)* dan memori ( RAM/ROM ) menghasilkan sebuah mikrokomputer.

**Pengalamatan register tak langsung** : Pengalamatan ini untuk mengakses suatu lokasi RAM yang alamatnya ditentukan oleh isi R0 dan R1

**Pengalamatan bit langsung** : Pengalamatan ini untuk melakukan manipulasi atau tes pada 128 bit bendera yang telah didefinisikan melalui perangkat lunak dan 128 bit pada SFR.

**Pengalamatan Register Basis dan Register Index tidak langsung** : Pengalamatan ini untuk menyederhanakan pengaksesan *look-up table* (LUT) yang terletak di dalam memori program.

**Pengalamatan segera** : Pengalamatan ini untuk memindahkan data secara langsung ke dalam register atau memori. Pada mode pengalamatan ini digunakan tanda # di depan nilai yang digunakan.

**Rangkaian 7 Segment** : Kumpulan 7 buah LED yang dirangkai sedemikian rupa sehingga dapat menampilkan (awalnya) tampilan numerik (angka) dari 0 hingga 9'.

**RISC kependekan dari** *Reduced Instruction Set Computer* : Instruksi terbatas tapi memiliki fasilitas yang lebih banyak.

**Sensor api UV-Tron** : Sebuah sensor yang mendeteksi adanya nyala api yang memancarkan sinar *ultraviolet.*

**Sensor kompas** : Sensor yangmampu mendeteksi arah secara horisontalterhadap medan magnet bumi.

**Sensor Infra merah** : Sistem sensor ini pada dasarnya menggunakan inframerah sebagai media komunikasi yang menghubungkan antara dua perangkat.

**Sensor gelombang ultrasonik** : Sensor yang bekerja dengan cara memancarkan suatu gelombang dan kemudian menghitung waktu pantulan gelombang tersebut.

**Telemetri** : Sistem pengukuran jarak jauh.

**Ukuran alamat memori** *(memory address size)* :Jumlah alamat memori yang dapat dialamati oleh mikroprosesor secara langsung.

**Ukuran bus data eksternal** *(external data bus size)* : Jumlah saluran yang digunakan untuk transfer data antar komponen antara mikroprosesor dan komponen-komponen di luar mikroprosesor.