PERANCANGAN BEL/ALARM SEKOLAH SECARA OTOMATIS BERBASIS MIKROKONTROLER AT89S51

SKRIPSI

Diajukan untuk memenuhi persyaratan
Memperoleh gelar sarjana Teknik

Disusun Oleh:

M RIZA YANUAR R

NIM 0210633054

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2009

PERANCANGAN BEL/ALARM SEKOLAH SECARA OTOMATIS BERBASIS MIKROKONTROLER AT89S51

SKRIPSI

Diajukan untuk memenuhi persyaratan
Memperoleh gelar sarjana Teknik

Disusun Oleh:

M RIZA YANUAR R

NIM 0210633054

Telah diperiksa dan disetujui
Dosen Pembimbing

Panca Mudjirahardjo, ST., MT	**Ir. Nurussa'adah, MT**
NIP. 19700329 200012 1 001	**NIP. 19680706 199603 1 001**

PERANCANGAN BEL/ALARM SEKOLAH SECARA OTOMATIS BERBASIS MIKROKONTROLER AT89S51

Disusun Oleh:

M RIZA YANUAR R

0210633054-63

Skripsi ini telah diuji dan dinyatakan lulus pada

Tanggal 13 Agustus 2009

DOSEN PENGUJI

Ir. Ponco Siwindarto, M.Eng.Sc
NIP: 19590304 198903 1 001

Moch. Rif'an, ST., MT
NIP: 19710301 200012 1 001

Adharul Muttaqin, ST., MT
NIP: 19760121 200501 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir. Heru Nurwasito, M.kom
NIP: 19650402 199002 1 001

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas terselesaikannya penulisan Tugas Akhir ini, karena hanya atas perkenan dan petunjukn-Nya lah, maka segala sesuatu terjadi. Kendala dan cobaan yang mengiringinya semata adalah jalan yang dibentangkanNya untuk menyadari betapa kerdil dan tidak berdayanya manusia dalam menghadapi takdirNya.

Tugas Akhir ini mengambil judul ***" Perancangan Bel/Alarm Sekolah Secara otomatis Berbasis Mikrokontroler AT89S51".***

Sangat sulit rasanya menyatakan bahwa penulis telah berhasil menyelesaikan Tugas Akhir ini tanpa bantuan banyak pihak. Pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Bapak Heru Nurwasito, Ir., MKom, selaku Ketua Jurusan Teknik Elektro, Bapak Rudy Yuwono, ST., M.Sc selaku Sekretaris Jurusan Teknik Elektro.
2. Bapak M. Julius Ir, St, MS sebagai KKDK Teknik Elektronika yang telah menerima dan mensyahkan diterimanya judul skripsi ini.
3. Bapak Panca Mudjirahardjo, ST., MT sebagai pembimbing I, atas banyak pengarahan dan diskusi untuk penyelesaian Tugas Akhir ini.
4. Ibu Nurussa'adah, Ir sebagai pembimbing II , atas saran dan bimbingannya untuk meneyelesaikan Tugas Akhir ini.
5. Seluruh karyawan dan petugas Laboratorium Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.
6. Dan berbagai pihak yang tidak mungkin saya sebutkan satu persatu. Terima kasih telah memberikan sumbang asih demi terselesainya tugas akhir ini. Semoga kebaikan dan amalnya dibalas oleh Allah SWT, Amiiiin.....

Tugas akhir ini hanyalah upaya untuk mendekati realitas dengan disiplin ilmiah yang didasarkan pada rasionalitas yang serba terbatas. Semoga

Tugas Akhir ini dapat bermanfaat bagi penulis maupun bagi yang sempat menyimak dan membacanya.

Kalaupun ada kekurangan, wajar sebagai makhluk-Nya dan kalaupun ada lebihnya itu semua datangnya dari Allah semata. Penulis sangat berharap saran dan kritik yang membangun demi perbaikan dan penyempurnaan Tugas Akhir ini.

Penulis

M Riza Yanuar R

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

RINGKASAN

M Riza Yanuar R, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Agustus 2009, *Perancangan Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51*, Dosen Pembimbing : Panca Mudjirahardjo, ST., MT Dan Ir. Nurussa'adah

Bel/ alarm yang terdapat di sekolah-sekolah dibunyikan secara manual, yakni menggunakan operator untuk menekan tombol bel/alarm-nya. Hal demikian masih kurang praktis, seiring jaman otomatisasi. Pada Skripsi ini akan dibahas mengenai bel/ alarm sekolah secara otomatis berbasis mikrokontroler AT89S51. Definisi bel/ alarm sekolah secara otomatis adalah suatu penanda untuk memulai dan mengakhiri pelajaran di sekolah yang akan berbunyi pada waktu yang ditentukan tanpa harus menekan tombol. Alat ini menggunakan AT89S51 sebagai pengontrol, Real Time Clock DS1307 sebagai pewaktu. Sebagai input adalah 4 tombol push-button untuk melakukan seting waktu alarm, dan untuk memasukkan input suara kedalam ISD (record). Sedangkan output-nya adalah output waktu pada LCD, output buzzer sebagai alarm, dan output suara yang terekam dalam ISD. Pertama-tama dapat melakukan seting waktu bel/alarm melalui tombol push button menu, up, dan down, dan kemudian memasukkan input suara agar nantinya dihasilkan output suara sebagai alarm. Pada saat waktu telah sesuai dengan input nilai waktu alarm, maka secara otomatis mikrokontroler akan mengeluarkan suara alarm dan output suara sesuai dengan seting alarm yang telah di inputkan.

Kata Kunci : ISD 25120., Mikrokontroler, RTC, LCD

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kemajuan pengetahuan dan teknologi dewasa ini berkembang cepat sekali dan berpengaruh dalam pembuatan alat-alat canggih, yaitu alat yang dapat bekerja secara otomatis dan memiliki ketelitian tinggi dengan bantuan mikrokontroller. Dengan adanya perkembangan teknologi inilah manusia semakin dipermudah dalam segala aspek kehidupan dan dituntut untuk lebih jeli dalam menyikapinya.

Bel/ alarm yang terdapat di sekolah-sekolah dibunyikan secara manual, yakni menggunakan operator untuk menekan tombol bel/alarm-nya. Hal demikian masih kurang praktis, seiring jaman otomatisasi. Pada Skripsi ini akan dibahas mengenai bel/ alarm sekolah secara otomatis berbasis mikrokontroler AT89S51. Definisi bel/ alarm sekolah secara otomatis adalah suatu penanda untuk memulai dan mengakhiri pelajaran di sekolah yang akan berbunyi pada waktu yang ditentukan tanpa harus menekan tombol.

Untuk merealisasikannya, penulis menggunakan AT89S51 sebagai pengontrol, Real Time Clock DS1307 sebagai pengatur waktunya agar waktunya tetap menghitung apabila tidak diberi supply. Sebagai input adalah 4 tombol push-button untuk mengatur waktu dan alarm. Sedangkan output-nya adalah output waktu dan output alarm. Output waktu merupakan tampilan dalam display LCD dan bunyi alarm yang keluar dari buzzer dan output suara dari speaker.

1.2. Rumusan Masalah

Dari uraian tersebut diatas maka timbul beberapa permasalahan diantaranya adalah sebagai berikut:

1) Bagaimana merancang rangkaian mikrokontroler AT89S51?
2) Bagaimana merancang rangkaian RTC DS1307?
3) Bagaimana mengatur waktu alarm?
4) Bagaimana menampilkan output alarm pada Speaker?
5) Bagaimana merancang rangkaian Power Supply?

1.3. Tujuan

Skripsi ini bertujuan untuk merancang dan membuat Bel /Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51 sehingga dapat membantu kelancaran proses belajar/mengajar di sekolah.

1.4. Batasan Masalah

Agar permasalahan tidak meluas maka penulis membatasi permasalahan sebagai berikut:

1) Menggunakan Mikrokontroler AT89S51 sebagai pengendali utama.
2) Menggunakan RTC DS1307 sebagai pewaktu.
3) Menggunakan LCD M1632 sebagai display kerja alat.
4) Menggunakan ISD 25120 untuk merekam suara alarm agar nantinya output suara sebagai alarm dapat sesuai yang kita inginkan.
5) Bel/Alarm ditujukan untuk sekolah.

1.5. Metodologi Penulisan

Metodologi yang dilakukan dalam penulisan Skripsi Perancangan Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51 adalah sebagai berikut:

1) Kajian pustaka, meliputi konsep dasar dan teori yang digunakan
2) Perancangan dan pembuatan aplikasi meliputi perancangan model, proses pembuatan disertai diagram alirnya
3) Pengujian tentang aplikasi yang sedang dibuat
4) Membuat kesimpulan serta saran dari hasil pengujian aplikasi

1.6. Sistematika Penulisan

Untuk mempermudah dan memperjelas pembahasan dari laporan Skripsi ini penulis menerapkan system penulisan seperti dibawah ini :

BAB I : PENDAHULUAN

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

BAB II : DASAR TEORI

Membahas tentang teori dasar rangkaian yang digunakan, mikrokontroler, hardware dan teori dasar alat-alat pendukung lainnya.

BAB III : METODOLOGI

Berisi tentang metodologi yang digunakan meliputi pengumpulan data studi literatur, perancangan sistem, pembuatan perangkat keras *(hardware)* dan perangakat lunak *(software)*, pengujian alat dan anlisis hasil pengujian, serta pengambilan kesimpulan dan saran.

BAB IV : PERENCANAAN DAN PEMBUATAN ALAT

Perancangan Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51.

BAB V : PENGUJIAN ALAT

Berisi tentang uji coba alat yang telah dibuat, pengoperasian dan spesifikasi alat.

BAB VI : PENUTUP

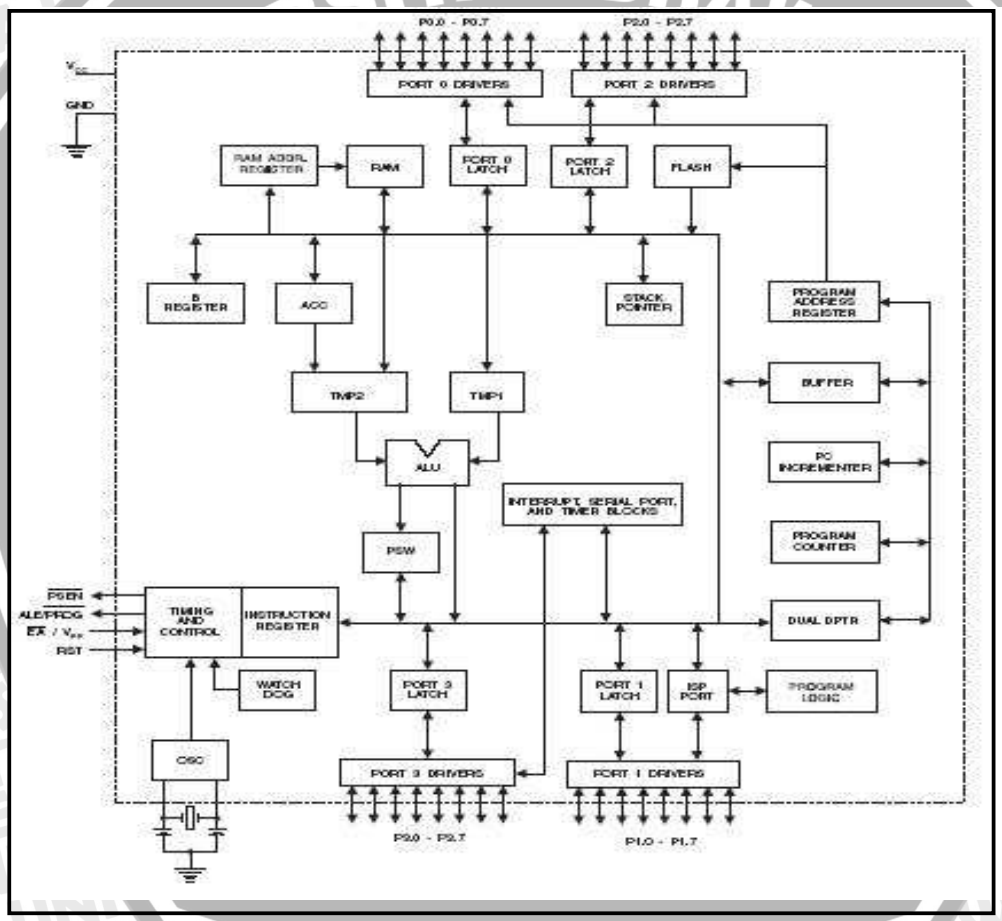
Merupakan kesimpulan dari pembahasan pada bab–bab sebelumnya dan kemungkinan pengembangan alat.

BAB II
TEORI DASAR

Bab ini akan menguraikan tentang dasar-dasar teori dasar yang menunjang dalam perancangan dan pembuatan alat. Uraian teori dalam bab ini meliputi teori IC AT89S51, ISD 25120, LCD M1632 dan perangkat-perangkat pendukung lainnya.

2.1. Mikrokontroler AT89S51

2.1.1. Teori umum.



Gambar 2.1. Diagram Blok AT89S51

AT89S51 merupakan sebuah mikrokontroller 8-bit CMOS, Low Power dengan 4Kb flash Programmable and Erasable Read Only Memory (PEROM). IC

ini dibuat sesuai dengan standart industri konfigurasi pin dan intruction set dari MCS-51.

- 4Kb Flash Memory.
- 128 Byte Internal RAM.
- 32 I/O Lines.
- 2 Timer/Counter 16-Level.
- 1 Serial Port Full Duplex.
- On Chip Osilator.

AT89S51 mempunyai dua buah Power-Saving mode yang dapat diatur melalui software, yaitu: IDE Mode yang akan menghentikan CPU sebagai RAM, Timer/Counter, Serial Port dan Interrupsi system tetap berfungsi. Power Down Mode yang akan menyimpan ini di RAM, tetapi menahan Oscilator untuk tidak mengaktifkan chip yang lain sampai terjadi reset secara hardware.

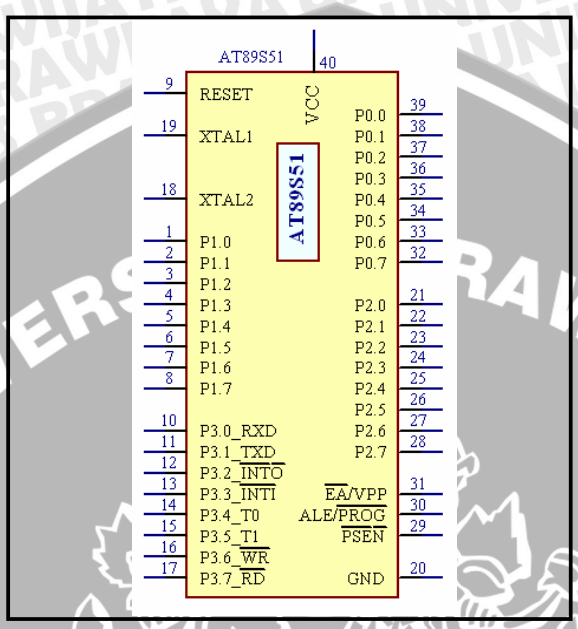
2.1.2. Arsitektur AT89S51

Arsitektur Mikrokontroler AT89S51 adalah sebagai berikut:

- CPU (Central Processing Unit) 8-bit dengan register A (Accumulator) dan B.
- 16-Bit Program Counter (PC) dan data pointer (DPTR).
- 8-Bit Program status word (PSW).
- 4-Bit Stack pointer (SP).
- 4 Kbyte internal EPROM
- 128 byte internal RAM.
- 4 bank register, masing-masing berisi 8 register.
- 16 Byte yang dapat dialamati pada bit level.
- 80 Byte general purpose memory data.
- 32 pin input-output tersusun atas P0-P3 masing-masing 8-bit.
- 2 Buah 16-bit timer counter.
- Receiver Register, yaitu : TCON, TMOP, SCON, IP, dan IE.
- 5 Buah sumber interupt (2 buah sumber interupt eksternal dan 3 buah sumber interupt internal).
- Oscilator dan Clock internal.

2.1.3. Konfigurasi Pin pada Mikrokontroler AT89S51

Konfigurasi kaki-kaki mikrokontroler AT89S51 terdiri dari 40 pin, seperti terlihat dalam Gambar 2.2 sebagai berikut.



Gambar 2.2. Konfigurasi Pin-Pin AT89S51

Fungsi dari tiap-tiap pin adalah sebagai berikut:

1) VCC (Supply tegangan).
2) GND (Ground).
3) Port 0

   Merupakan port input-output dua arah dan dikonfigurasikan sebagai multiplex dua bus alamat rendah (A0 – A7) dan data selama pengaksesan program memory dan data memory internal.
4) Port 1

   Merupakan port input-output dua arah dengan internal pull-up.
5) Port 2

   Merupakan port input-output dua arah dengan internal pull-up. Mengeluarkan address tinggi selama pengambilan (fetch) program memory internal dan selama pengaksesan ke data memory port 2 mengeluarkan isi P2SFR (Special Function Register) menerima addres tinggi dan beberapa sinyal kontrol selama pemograman dan veritifikasi.

6) Port 3.

Merupakan port input-output dua arah dengan internal pull-up. Port 3 juga memiliki fungsi khusus, yaitu:

- RXD (P3.0) : Port input serial.
- TXD (P3.1) : Port out-put serial.
- INTO (P3.2) : Interrup 0 external.
- INTI (P3.3) : Internal 1 external.
- TO (P3.4) : Input external timer 0.
- T1 (P3.5) : Input external timer 1.
- WR (P3.6) : Strobe tulis data memory external.
- RD (P3.7) : Strobe baca data memory external.

7) RST

Input reset.

8) ALE\PROG

Pulsa output ALE digunakan untuk proses “latching” byte addres rendah (A0 - A7) selama pengaksesan ke external memory. Pin ini juga untuk memasukkan pulsa program selama pemrograman.

Pada operasi normal ALE mengeluarkan rate konstant yaitu 16 frekuensi osilasi dan boleh digunakan untuk timing external.

9) PSEN

Merupakan strobe baca ke program memory external.

10) EA\VPP

External addres enable EA digroundkan jika mengakses memory external. Untuk mengakses memory internal maka dihubungkan ke VCC.

11) XTAL 1 dan XTAL 2.

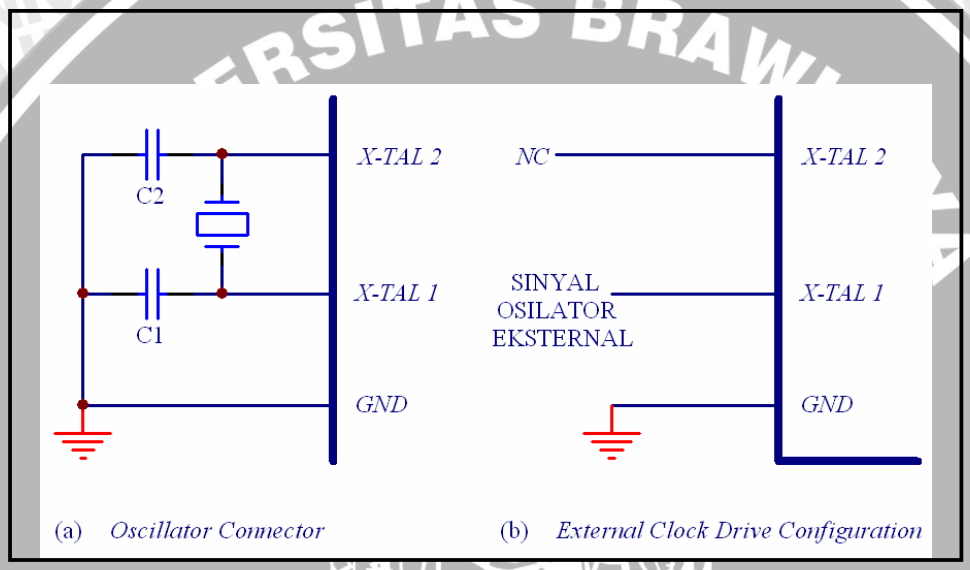
Kaki ini dihubungkan dengan kristal bila menggunakan osilator internal. XTAL 1 merupakan input inverting osilator amplifier sedangkan XTAL 2 merupakan out-put inverting osilator amplifier.

2.1.4. Karakteristik Oscillator Inverting

X-TAL 1 dan X-TAL 2 secara berurutan merupakan input dan output dari sebuah inverting amplifier yang dapat dikonfigurasikan penggunaannya sebagai

on chip oscillator seperti yang ditunjukkan dalam Gambar 2-3(a). X-TAL 1 dan X-TAL 2 ini dapat menggunakan sebuah kristal quartz maupun resonator keramik.

Untuk memberikan AT89S51 dari sumber clock external. Maka pin X-TAL 2 dibiarkan tidak berhubungan dan X-TAL 1 dihubungkan dengan sumber clock external seperti dalam Gambar 2-3(b). Rangkaian ini tidak melakukan duty cycle dari setiap sinyal clock internal, karena input bagi masukan rangkaian clock internal dihubungkan ke flip-flop pembagi dua, tetapi spesifikasi nilai tegangan pada saat tinggi dan rendah, maksimum dan minimumnya harus diberikan.



Gambar 2.3. Karakteristik Oscilator

2.1.5. Organisasi Memory

Didalam AT89S51 memiliki ruangan alamat telah dibedakan untuk program memory dan data memory. Pemisahan memori program dan data tersebut membolehkan memori data diakses dengan alamat 8-bit, sehingga dapat dengan cepat dan mudah disimpan dan dimanipulasi oleh CPU 8-bit. Namun demikian, alamat memori data 16-bit bisa juga dihasilkan melalui register DPTR.

2.1.5.1. Program Memory Internal

AT89S51 memiliki program memory internal sebesar 4Kbyte dengan ruang alamat 0000H-0FFFH. Jika alamat-alamat program lebih tinggi daripada

0FFFH, yang melebihi kapasitas ROM\Fash memory internal menyebabkan AT89S51 secara otomatis mengambil Code Byte dari program memory external. Code Byte juga dapat diambil hanya dari external memory dengan alamat 0000H-FFFFH dengan cara menghubungkan Pin EA ke Ground.

2.1.5.2. Data Memory (RAM) Internal

Ruang alamat bawah memory data (RAM) internal dengan kapasitas 128 byte yaitu 00H-7FH yang terbagi atas 3 daerah, yaitu :

- 4 Bank Register

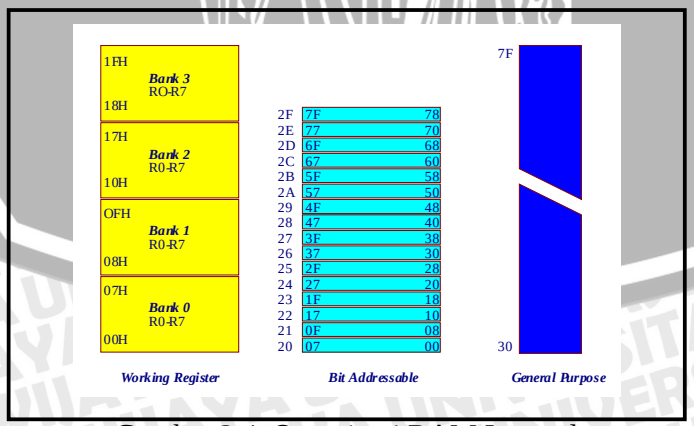
  Setiap bank terdiri dari 8 register (R0-R7). Sehingga jumlah register untuk keempat bank register (bank 0 - bank 3) menjadi 32 buah register yang menempati ruang alamat 00H-1FH. Mengaktifkan salah satu bank register dapat dilakukan dengan mengatur RSO-RSI pada PSW (Program Status Word).

- Bit Addressable.

  Terdiri dari 16 byte yang berada pada alamat 20H-2FH. Masing-masing 128 bit lokasi ini dapat di alamatkan secara langsung.

- General Purpose.

  Terdiri dari 80 byte yang menempati alamat 30H-7FH. yang dapat dialamati secara langsung dan dapat digunakan untuk keperluan umum (General Purpose RAM). Misalkan digunakan untuk lokasi stack. Gambar 2.4 menunjukkan Organisasi RAM.



Gambar 2.4. Organisasi RAM Internal

Tabel 2-1. Pengaturan RS0-RS1 Untuk Select Register Bank

RS1	RSO	Select Register Bank
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

2.1.5.3. SFR (Special Function Register).

Untuk mengoperasikan AT89S51 yang tidak menggunakan alamat internal RAM (00-7FH) dilakukan oleh SFR yang beraddres 80H-FFH, tetapi tidak semua address tersebut digunakan sebagai SFR. Fungsi-fungsi SFR dijelaskan dalam Tabel 2 - 2 sebagai berikut.

Tabel 2-2. 128 Byte Special Function Register

SYMBOL	NAME	ADDRESS
ACC	ACCUMULATOR	0E0H
B	B REGISTER	0F0H
PSW	PROGRAM STATUS WORD	0D0H
SP	STACK POINTER	81H
DPTR	DATA POINTER 2 BYTE	
DPL	LOW BYTE	82H
DPH	HIGH BYTE	83H
P0	PORT 0	80H
P1	PORT 1	90H
P2	PORT 2	0A0H
P3	PORT 3	080H
IP	INTERUPT PERIORITY CONTROL	088H
IE	INTERUPT ENABLE CONTROL	0ABH
TMOD	TIMER / COUNTER MODE CONTROL	89H
TCON	TIMER/COUNTER CONTROL	88H
TH0	TIMER/COUNTER 0 HIGH CONTROL	8CH
TL0	TIMER/COUNTER 0 LOW CONTROL	8DH
TH1	TIMER/COUNTER 1 HIGH CONTROL	8DH

TL1	TIMER/COUNTER 1 LOW CONTROL	8CH
SCON	SERIAL CONTROL	98H
SBUF	SERIAL DATA BUFFER	99H
PCON	POWER CONTROL	87H

2.2. RTC DS1307

DS1307 adalah IC serial *Real Time Clock* (RTC) dimana alamat dan data ditransmisikan secara serial melalui sebuah jalur data dua arah I2C. Karena menggunakan jalur data I2C maka hanya memerlukan dua buah pin saja untuk komunikasi. Yaitu pin untuk data dan pin untuk sinyal clock. Sistem jalur data I2C adalah suatu standar protokol sistem komunikasi data serial yang dikembangkan oleh Philips dan cukup populer dikarenakan penggunaannya cukup mudah.

Pada dasarnya, pada sistem I2C terbagi atas dua bagian, yaitu suatu *device* yang bertindak sebagai pengontrol atau Master dan suatu *device* yang dikontrol atau *Slave*. Master dan *Slave* saling berkomunikasi melalui jalur data bus I2C. Alat yang mengendalikan komunikasi data disebut Master dan alat yang dikendalikan oleh Master dikenal sebagai *Slave*. Dimana pada perancangan ini, penulis menggunakan mikrokontroler AT89S51 sebagai Master.

Pada satu jalur data I2C yang sama dapat terdapat *slave* lebih dari satu oleh karena itu I2CBus harus dikendalikan oleh sebuah Master yang dapat membangkitkan serial clock (SCL), mengontrol sistem komunikasi data (SDA), dan juga dapat menghasilkan kondisi-kondisi “START” dan “STOP”. Pada hal ini DS1307 beroperasi sebagai *slave* pada I2C bus. Contoh bagaimana data ditransfer pada jalur data I2C adalah seperti dalam Gambar 2.5. Pada jalur data bus I2C hanya terdapat 2 buah jalur yang digunakan yaitu Clock (SCL) dan Data (SDA). Terdapat beberapa macam jenis kondisi pada jalur data I2C, jenis kondisi tersebut adalah:

1. ***Bus not busy:*** Jalur data (SDA) dan clock (SCL) berlogika high
2. ***Start data transfer:*** Suatu perubahan kondisi pada jalur data, dari logika *high* ke logika *low*, ketika jalur data sedang berlogika *high*, menandakan kondisi START.

3. ***Stop data transfer:*** Suatu perubahan kondisi pada jalur data, dari logika *low* ke logika *high*, ketika jalur data sedang berlogika *high*, menandakan kondisi STOP
4. ***Data valid:*** Suatu kondisi ketika jalur data menandakan data valid, yaitu ketika setelah kondisi START, jalur data tetap stabil selama periode *high* sinyal clock. Data pada jalur data harus berubah selama periode *low* dari sinyal clock. Terdapat satu pulsa clock untuk setiap bit data. Setiap proses pengiriman data dimulai dengan kondisi START dan diakhiri dengan kondisi STOP. Banyaknya jumlah *byte* data yang ditransfer diantara kondisi START dan STOP tersebut tidak terbatas, dan diatur oleh Master.
5. ***Acknowledge:*** Setiap *device* yang dituju telah menerima data dengan benar akan membangkit kan kondisi *Acknowledge* setiap menerima *byte* data. *Device* yang membangkitkan *Acknowledge* harus membangkitkan logika *low* pada jalur SDA selama sebuah pulsa clock. Untuk mengakhiri suatu proses pengiriman data Master harus memberikan suatu tanda dengan tidak memberi kan tanda *acknowledge* melainkan memberikan tanda STOP pada *slave*.

Pada sistem jalur data I2C, terdapat dua tipe arah proses pengiriman data yaitu:

1. **Data transfer dari master menuju *slave*.** *Byte* pertama yang dikirimkan oleh master menuju *slave* adalah alamat *slave*. Lalu selanjutnya adalah *byte-byte* data. *Slave* membalas dengan bit *acknowledge* setiap berhasil menerima 1 *byte* data.
2. **Data transfer dari *slave* menuju master.** *Byte* pertama (alamat *slave*) dikirimkan oleh master. Kemudian *slave* yang mempunyai alamat yang dituju oleh master membalas dengan bit *acknowledge*. Lalu diikuti dengan proses pengiriman *byte-byte* data dari *slave* menuju master. Master membalas dengan mengirimkan bit *acknowledge* setiap berhasil menerima 1 *Byte*. Untuk mengakhiri proses pengiriman data master membalas dengan mengirimkan bit *"not acknowledge"* kepada *slave*.

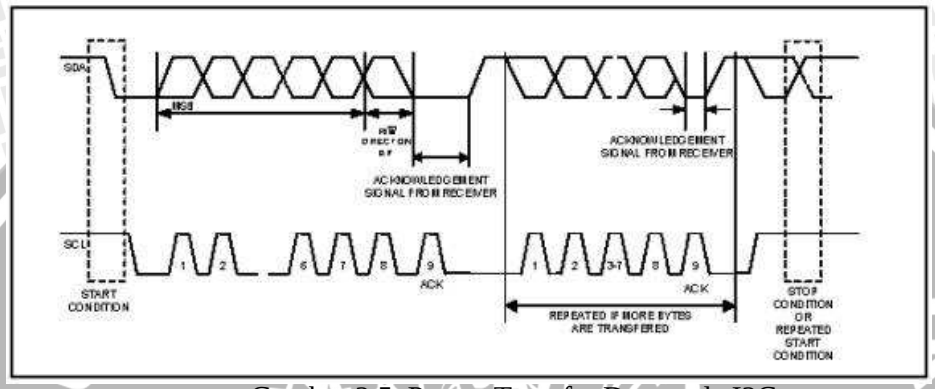
Pada aplikasi ini DS1307 bekerja dengan dua mode, yaitu:

1. **Mode *Slave* Penerima (Master Menulis Pada *Slave*):** Data serial dan clock diterima melalui SDA dan SCL. Setiap menerima *byte* data DS1307 akan

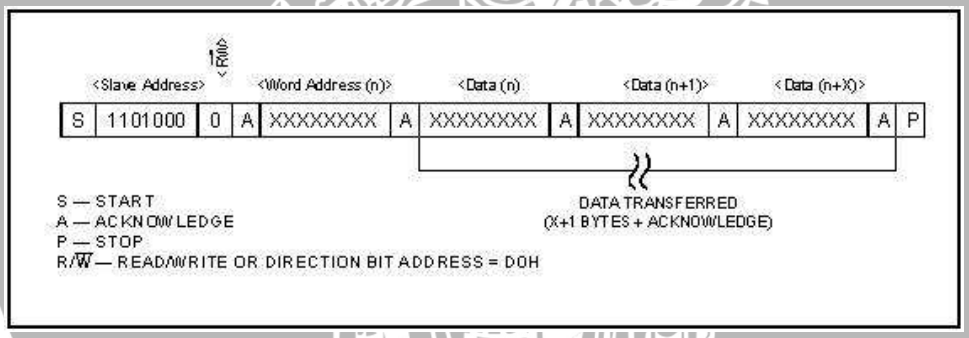
merespon dengan membangkitkan bit *acknowledge*. Untuk mengawali proses pengiriman data dari master menuju *slave* diawali dengan kondisi START dan diakhiri dengan kondisi STOP. Setiap *slave* akan membaca alamat yang dituju oleh master dan memeriksa apakah alamat tersebut sama dengan alamat *Slave* tersebut. *Byte* alamat *slave* adalah *Byte* pertama yang diterima *slave* setelah master membangkitkan kondisi START. *Byte* alamat terdiri dari 7 bit data, untuk DS1307 *Byte* alamat tersebut adalah 1101000b, dan diikuti oleh bit arah (R/W), yang mana untuk penulisan data ke *slave* adalah 0. Setelah menerima dan menganalisa *Byte* alamat, DS1307 membangkitkan tanda *acknowledge* pada jalur SDA. Kemudian master akan mengirimkan sebuah data word alamat pada DS1307 untuk mengeset regiter pointer pada DS1307. Setelah itu Master dapat mengakhiri proses pengiriman data ataupun melanjutkannya dengan mengirimkan *Byte* data pada DS1307. Register pointer akan bertambah nilainya secara otomatis setiap terjadi proses penulisan data. Untuk mengakhiri proses pengiriman data master membangkitkan kondisi STOP.

2. **Mode Slave Pengirim (Master Membaca Dari Slave):** *Byte* pertama diterima dan diolah oleh *slave* seperti pada mode penerima, tetapi bit arah bernilai 1. DS1307 mengirimkan data serial pada SDA ketika menerima sinyal clock pada SCL. Untuk memulai prose pengiriman data diawali dengan kondis START dan diakhiri dengan kondisi STOP. *Byte* yang berisi data alamat diterima setelah master membangkitkan kondisi START. *Byte* alamat DS1307 terdiri dari 7-bit alamat dan 1 bit arah. 7-bit alamat tersebut adalah 1101000 dan bit arah tersebut (R/W) adalah 0 untuk *read.* Setelah menerima dan mengolah data alamat, DS1307 akan membalas dengan membangkitkan bit *acknowledge* pada pin SDA. Kemudian DS1307 mulai mengirimkan data dimulai dari data yang terdapat pada alamat yang ditunjuk oleh register pointer. Nilai register pointer secara otomatis akan bertambah setiap terjadi proses pembacaan 1 *Byte* data. Untuk mengakhiri proses pengiriman data maka master harus mengirimkan tadan “not *acknowledge*” kepada *slave*.

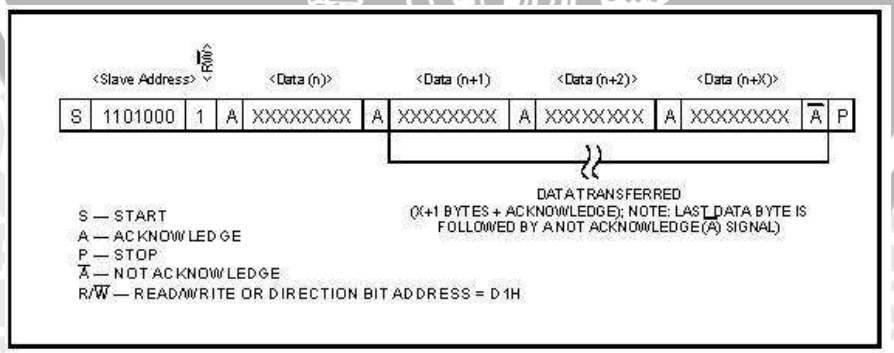
Untuk dapat mengambil nilai waktu dan tanggal maka master harus melakukan proses pembacaan data *(Read)* pada *slave* (DS1307), dengan alamat register sesuai dengan Tabel 2-3. Setiap nilai-nilai waktu atau tanggal disimpan pada register yang mempunyai alamat yang berbeda-beda, misalnya untuk register detik yang menyimpan nilai detik, menempati alamat register 00h.



Gambar 2.5. Proses Transfer Data pada I2C



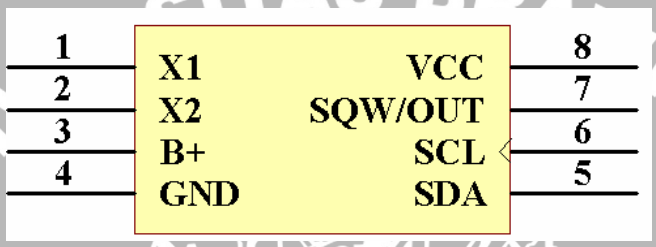
Gambar 2.6. Data Transfer dari Master Menuju *slave.*



Gambar 2.7. Data Transfer dari *Slave* menuju Master.

Tabel 2-3. Setting DS1307

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH



Gambar 2.8. RTC DS1307

Konfigurasi Pin DS1307:

- **Vcc,** + 5 Volt
- **X1 dan X2,** merupakan jalur untuk koneksi ke kristal 32,768 kHz.
- **B+(V_{BAT}),** Input untuk baterai *back-up* tegangan sebesar 3Volt.
- **SQW/OUT,** *Square Wave/Output Driver*
- **SCL,** *Serial Clock*
- **SDA,** *Serial Data*
- **GND,** *Ground*

2.3. Speak Call ISD 25120 (IC Data Perekam Suara)

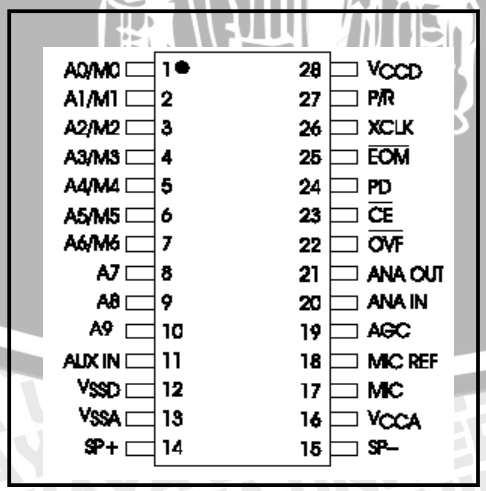
IC penyimpan suara yang digunakan merupakan jenis EEPROM *(Electrically Erasable Programable Read Only Memory)* yaitu ROM yang dapat diprogram, dihapus dan diprogram ulang secara elektrik dengan arus listrik, bukan sinar *ultraviolet*. IC ISD *(Information Strorage Device)*, yang dipakai yaitu ISD 25120. IC ini dapat merekam pesan maksimal 120 detik dan dapat dikaskade

sehingga pesan yang disimpan dapat diperpanjang sesuai dengan keinginan kita dengan 160 alamat yang berbeda.

Didalam ISD 25120 dilengkapi dengan internal *amplifier*, *internal automatic gain control (AGC), filter antialiasing* (perata) dan *speaker amplifier* (penguat speaker). Secara keseluruhan seri ISD 25120 dapat melakukan sebuah perekaman atau pemutaran ulang pesan dengan komponen sederhana seperti mikropon, *speaker*, beberapa komponen penunjang, duah buah saklar dan sumber tegangan.

Rekaman akan disimpan dalam sel memori yang tidak mudah hilang *(non volztile)*, memberikan tempat penyimpanan yang masih kosong. Cara unik ini yang membuat ISD disebut *Direct Analog Stroge Technology* (DAST) atau teknik penyimpanan analog langsung, dengan jalan sinyal suara (voice) dan bunyi disimpan secara langung dalam bentuk analog, kedalam memori EPROM. Penyimpanan analog langsung memungkinkan reproduksi suara secara alami dalam satu chip tunggal.

Susunan ISD 25120 DAST adalah dikelompokkan dalam 160 segmen dari alamat A0 sampai A7 yang menunjukkan akses tips segmen dalam kesatuan untuk alamat pesan. Kemampuan pemberian atau penyediaan alamat yang berupa pesan yang disimpan dalam bentuk kalimat dan suara.



Gambar 2.9. Pin-pin IC ISD 25120

- **Address Input (A0-A7) Pin 1-6 Dan 9-10**

  Input alamat ini mempunyai dua fungsi, tergantung dari level dari dua *Most Significant Bits* (MSB) dari alamat. Jika dua MSB ini keduanya low, maka semua input digunakan sebagai bit pengalamatan *(Adress Bits)* dan digunakan sebagai alamat untuk memulai *(Start Address)* dari perekaman atau pemutaran ulang *(Play Back).* Kaki-kaki dari pengalamat hanya merupakan masukan dan bukan merupakan informasi keluaran pengalamatan internal *(Output Internal Address Information).* Ketika proses operasional sedang berjalan dan pada saat kedua MSB ini high, maka sinyal input pengalamatan digunakan sebagai bits mode *(Mode Bit)* yang membuat mode operasi normal dan pengalamatan secara tidak langsung *(Simultanously).*

- **Vssd dan Vssa (Ground) Pin 12 dan 13**

  Sama seperti Vccd dan Vcca, analog input dan digital sirkuit di dalam ISD 25120 menggunakan *bus ground* yang terpisah untuk meminimalisasi *noise*. Pin ini harus dihubungkan sedekat mungkin dengan *ground*.

- **Speaker Output (SP +, SP -) Pin 14 dan 15**

  Pin SP + dan SP – digunakan untuk mengeluarkan suara yang telah direkam ke *speaker* atau ke *device* lainnya. Output ini mempunyai impedansi sebesar 16 Ohm.

- **Microphone Input (mic) Pin 17**

  Kaki mikropon ini terhubung dengan Vcc melalui beberapa kapasitor yang terhubung secara seri, bersamaan dengan resistor 10 KΩ yang berada didalam chip *(internal).* Harga dari kapasitor dari dalam perancangan ini menggunakan harga kapasitor sesuai dengan yang tertera dalam rangkaian data sheet ISD 25120.

- ***Microphone Reference* (MIC REF) Pin 18**

  Ketika MIC REF menghubungkan antara Vcc dengan mikropon *ground*, maka tingkat noise selama perekaman dapat dikurangi. Noise itu disebabkan oleh *pre-amplifier* yang terdapat didalam *chip*. Bila pin ini tidak digunakan,

maka tidak boleh dihubungkan dengan sinyal atau dengan tegangan apapun, harus dalam keadaan terbuka.

- ***Automatic Gain Control*** **(AGC) Pin 19**

  Kegunaan dari AGC adalah untuk menambah atau mengurangi secara otomatis penguatan *(Gain)* dari *pre-amplifier* yang juga meluaskan batas dari sinyal input yang dapat digunakan oleh mikropon tanpa terjadi distorsi. AGC ini dapat secara dinamis meluaskan batas dari suara yang terekam baik itu suara bisikan sampai suara yang keras. Untuk menggunakan fasilitas AGC ini, resistor dan kapasitor luar (eksternal) harus dihubungkan secara *parallel* antara pin AGC dengan *ground.* Harga yang direkomendasikan adalah R = 470 KΩ dan C = 4,7 µF (Dalam perancangan ini juga dipakai harga seperti diatas sama dengan data sheet ISD).

- ***Analog Input*** **(ANA IN) Pin 20**

  Kapasitor eksternal (luar) menghubungkan antara ANA IN ke ANA OUT pin harga-harga dari kapasitor luar bersama dengan 3 KΩ input impedansi di ANA IN dapat dipilih sendiri untuk memberikan keadaan *cut off* (terputus) pada frekuensi rendah sampai pada *pass band* suara. ANA IN juga dapat digunakan pada input sumber *alternative* dari sinyal analog pada sinyal mikropon terus ke kapasitor kopling.

- ***Analog Output*** **(ANA OUT) Pin 21**

  Sinyal dari mikropon dikuatkan dan dikeluarkan melalui ANA OUT pin. Penguatan tegangan dari *pre-amp* tergantung dari tingkat tegangan AGC *(Automatik Gain Control)* pin. *Pre-amplifier* ini mempunyai penguatan maksimum sekitar 24 dB untuk tingkat masukan kecil.

- ***Playback, Level – Activated*** **(PLAY L) Pin 23**

  Ketika sinyal ini berpindah dari *high* ke *low*, maka PLAY L akan berjalan. *Playback* akan berjalan sampai input ini tertekan *high,* tanda akhir dari pesan tercapai atau ruang memori sudah habis. ISD akan kembali ke mode stanby setelah playback ini berhenti.

- ***Playback Edge-Aktivated* (PLAY E) Pin 24**

  Ketika sinyal akan bepindah menuju *low (low-going transition)* terdeteksi di input ini, maka PLAY E akan berjalan. Playback berjalan sampai tanda akhir dari pesan tercapai (Akhir dari ruang memori tercapai). Setelah menyelesaikan *playback*, ISD secara otomatis akan kembali ke mode *stanby*, menekan PLAY E ke high pada waktu *playback* berjalan tidak akan menghentikan *playback*. Jadi *playback* akan berhenti bila mencapai akhir dari pesan atau ruang memori habis.

- ***Record LED Output* (RECLED) Pin 25**

  Selama proses perekaman output RECLED akan *low*. Maka output ini bisa digunakan untuk menjalankan sebuah led yang berfungsi untuk mengetahui bahwa terjadi proses perekaman. Ketika tanda akhir dari pesan tercapai pada saat *playback*, maka RECLED akan *low* sebentar.

- ***Optimal External Clock* (XCLK) Pin 26**

  Digunakan untuk penambahan kristal clock bila dibutuhkan pewaktuan yang lebih besar dan presisi. Bila input ini tidak digunakan, harus dihubungkan dengan ground.

- ***Record* (REC) Pin 27**

  Input sinyal REC akan aktif dalam kondisi *low*. ISD 25120 akan merekam bila REC dalam keadaan low, dan sinyal ini harus terus dalam keadaan low bila ingin terus merekam. Jika input REC ini tertekan low dalam keadaan masih memutar ulang pesan *(playback)*, maka *playback* akan berhenti dan ISD akan merekam.

- **VCCA Dan VCCD Pin 16 Dan 28**

  Analog dan digital sirkuit yang terdapat didalam chip ISD 25120 menggunakan bus power yang terpisah untuk meminimalisasi noise. Pin power ini harus dihubungkan sedekat mungkin dengan sumber tegangan.

2.4. LCD (Liquid Crystal Display) M1632

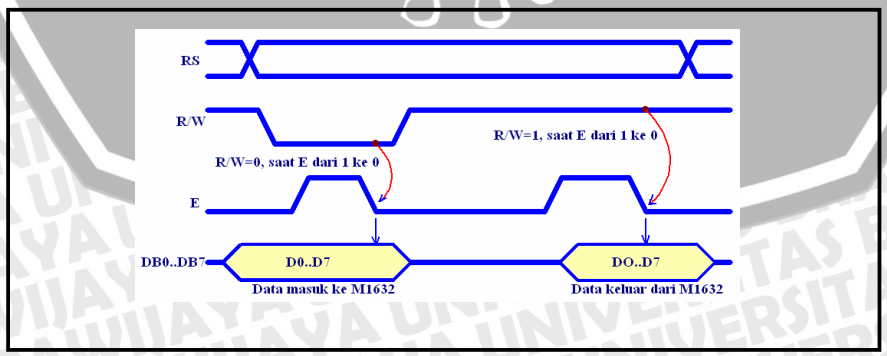
LCD Display Module M1632 buatan Seiko Instrument Inc. terdiri dari dua bagian, yang pertama merupakan panel LCD sebagai media penampil

informasi dalam bentuk huruf/angka dua baris, masing-masing baris bisa menampung 16 huruf/angka. Bagian kedua merupakan sebuah sistem yang dibentuk dengan mikrokontroler yang ditempelkan dibalik pada panel LCD, berfungsi mengatur tampilan informasi serta berfungsi mengatur komunikasi M1632 dengan mikrokontroler yang memakai tampilan LCD itu. Dengan demikian pemakaian M1632 menjadi sederhana, sistem lain yang M1632 cukup mengirimkan kode-kode ASCII dari informasi yang ditampilkan seperti layaknya memakai sebuah printer.

2.4.1. Sinyal interface M1632

Untuk berhubungan dengan mikrokontroler pemakai, M1632 dilengkapi dengan 8 jalur data (**DB0..DB7**) yang dipakai untuk menyalurkan kode ASCII maupun perintah pengatur kerjanya M1632. Selain itu dilengkapi pula dengan **E**, **R/W** dan **RS** seperti layaknya komponen yang kompatibel dengan mikroprosesor. Kombinasi lainya **E** dan **R/W** merupakan sinyal standar pada komponen buatan Motorola. Sebaliknya sinyal-sinyal dari MCS51 merupakan sinyal khas Intel dengan kombinasi sinyal **WR** dan **RD.**

RS, singkatan dari Register Select, dipakai untuk membedakan jenis data yang dikirim ke M1632, kalau **RS=0** data yang dikirim adalah perintah untuk mengatur kerja M1632, sebaliknya kalau **RS=1** data yang dikirim adalah kode ASCII yang ditampilkan. Demikian pula saat pengambilan data, saat **RS=0** data yang diambil dari M1632 merupakan data status yang mewakili aktivitas M1632, dan saat **RS=1** maka data yang diambil merupakan kode ASCII dari data yang ditampilkan. Proses mengirim/mengambil data ke/dari M1632 ditunjukkan dalam Gambar 2.10 bisa dijabarkan sebagai berikut :

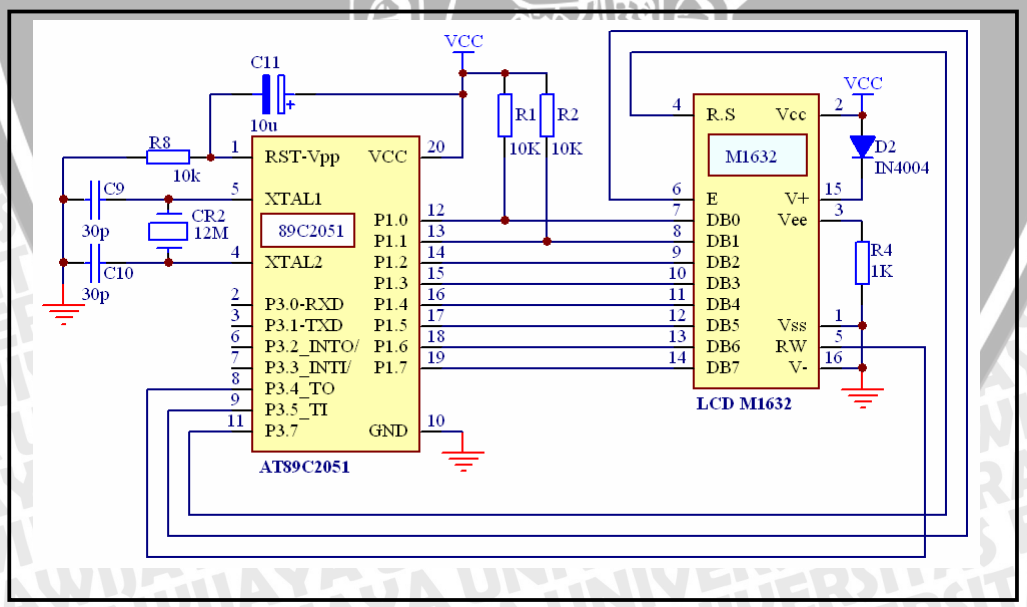


Gambar 2.10. Mengirim/Mengambil Data Ke/Dari M1632

1) **RS** harus dipersiapkan dulu, untuk menentukan jenis data seperti yang telah dibicarakan di atas.
2) **R/W** di-nol-kan untuk menandakan akan diadakan pengiriman data ke M1632. Data yang akan dikirim disiapkan di **DB0..DB7**, sesaat kemudian sinyal **E** di-satu-kan dan di-nol-kan kembali. Sinyal **E** merupakan sinyal sinkronisasi, saat **E** berubah dari 1 menjadi 0 data di **DB0 .. DB7** diterima oleh M1632.
3) Untuk mengambil data dari M1632 sinyal **R/W** di-satu-kan, menyusul sinyal **E** di-satu-kan. Pada saatu **E** menjadi 1, M1632 akan meletakkan datanya di **DB0 .. DB7**, data ini harus diambil sebelum sinyal **E** di-nol-kan kembali.

2.4.2. Interface Ke MCS-51

Sinyal-sinyal M1632 yang mengikuti standar teknik interface Motorola, tidak sesuai dengan sinyal dari MCS51, dengan demikian sinyal-sinyal itu disimulasikan melalui port MCS51. Sebagai contoh gambar 2.11 memperlihatkan hubungan AT89C2051 dengan M1632, dalam gambar tersebut **P3.7** dipakai untuk mensimulasikan sinyal **RS**, **P3.5** sebagai **E** dan **P3.4** sebagai **R/W**. Lewat program dibangkitkan sinyal-sinyal pada ketiga kaki Port 3 ini, sesuai dengan pesyaratan yang dikehendaki M1632.



Gambar 2.11. Hubungan M1632 ke MCS'51

Potongan Program 1 merupakan sub-rutin untuk mengendalikan M1632 yang dihubungkan ke AT89C2051 seperti terlihat di gambar 2.11, sebelum sub-rutin ini dipakai, tepatnya pada saat setelah reset harus dikirimkan perintah **CLR E**. Potongan program 1 terdiri dari dua bagian, yakni bagian mengirim data ke M1632 yang terdiri dari sub-rutin **Kirim Perintah** dan sub-rutin **Kirim ASCII**, sedangkan bagian mengambil data dari M1632 terdiri dari sub-rutin **Ambil Status** dan sub-rutin **Ambil ASCII**.

Sebelum mengiriman data, Akumulator A sudah terlebih dulu diisi dengan data yang akan dikirim. Data yang dikirim dengan sub-rutin **Kirim Perintah** akan diterima M1632 sebagai perintah untuk mengatur kerja M1632, dan data yang dikirim dengan sub-rutin **Kirim ASCII** akan ditampilkan di panel LCD.

Setelah M1632 menerima data, M1632 memerlukan waktu antara 40 sampai 1640 mikro-detik untuk mengolahnya, selama waktu itu M1632 untuk sementara tidak bisa menerima data, hal ini ditandai dengan bit 7 dari Register Status = '**1**'.

Proses pengiriman data ke M1632 dijelaskan sebagai berikut :

1) Perbedaan sub-rutin **Kirim Perintah** dan **Kirim ASCII** terletak pada nilai **RS** pada saat sub-rutin itu bekerja. Sub-rutin **Kirim Perintah** bekerja dengan **RS='0'** (baris 6), data yang dikirim AT89C2051 diterima M1632 sebagai perintah untuk mengatur kerja M1632. Sub-rutin **Kirim ASCII** bekerja dengan **RS='1'** (baris 10), data yang dikirim AT89C2051 diterima M1632 sebagai kode ASCII yang akan ditampilkan
2) Sinyal **RW** di-nol-kan agar M1632 siap menerima data (baris 12), setelah itu data di akumulator **A** diletakkan di **D0..D7** (Port 1 dari AT89C2051) di baris 13, baris 14 dan 15 membangkitkan sinyal sinkronisasi **E** dengan cara membuat **P3.5** menjadi '1' dan kemudian kembali menjadi '**0**'. Saat sinyal **E** kembali menjadi '**0**' data di Port 1 akan diterima oleh M1632.
3) Selesai mengirim data, program harus menunggu sampai M1632 siap menerima data lagi. Hal ini dilakukan dengan cara mengambil Status M1632 (baris 18), dan memeriksa bit 7-nya (baris 19), selama bit 7 bernilai '**1**' berarti M1632 masih sibuk mengurus diri, dan program menunggunya di Tunggu Dulu.

Proses pengambilan data dari M1632 dijelaskan sebagai berikut :

1) Seperti bahasan di atas, **RS** dipakai untuk memilih Register Perintah/Status, sub-rutin **Ambil Status** bekerja dengan **RS=0** dan sub-rutin **Ambil ASCII** bekerja dengan **RS='1'**.
2) Sinyal **RW** di-satu-kan agar M1632 siap memberi data (baris 29), setelah sinyal **E** menjadi '**1**' (baris 30) M1632 akan meletakkan data di **D0 .. D7**, setelah data ini diambil (baris 31) sinyal **E** dikembalikan menjadi '**0**'

Berikut adalah Potongan Program 1 AT89C2051 dengan M1632 :

```
01:     E       bit     P3.5            ; sinyal E   di P3.5
02:     RW      bit     P3.4            ; sinyal R/W di P3.4
03:     RS      bit     P3.7            ; sinyal RS  di P3.7
04:     ;Comment In Here
05:     KirimPerintah:
06:     CLR   RS                        ; RS=0 : register perintah
07:     SJMP  OutByte
08:     ;
09:     KirimASCII:
10:     SETB  RS                        ; RS=1 : Display Data RAM
11:     OutByte:
12:     CLR   RW                        ; RW = '0', kirim data
13:     MOV   P1,A                      ; siapkan data di D0..D7
14:     SETB  E                         ; buat pulsa positip
15:     CLR   E                         ; sesaat
16:     ;
17:     TungguDulu:
18:     ACALL AmbilStatus
19:     JB    A.7,TungguDulu
20:     RET
21:     AmbilStatus:
22:     CLR   RS                        ; RS=0 : register status
24:     SJMP  InByte
26:     AmbilASCII:
```

```
27:     SETB  RS            ; RS=1 : Display Data RAM
28:     InByte:
29:     SETB  RW            ; RW = '1', ambil data
30:     SETB  E             ; minta data pada M1632
31:     MOV   A,P1          ; ambil data
32:     CLR   E             ; kembalikan E ke '0'
33:     RET
```

2.4.3. Mengatur tampilan M1632

M1632 mempunyai seperangkat perintah untuk mengatur tata kerjanya, perangkat perintah tersebut meliputi perintah untuk menghapus tampilan, meletakkan kembali cursor pada baris huruf pertama baris pertama, menghidup/matikan tampilan dan lain sebagainya, semua itu dibahas secara terperinci dalam lembar data M1632.

Setelah diberi sumber daya, ada beberapa langkah persiapan yang harus dikerjakan dulu agar M1632 bisa dipakai, langkah-langkah tersebut antara lain adalah:

1) Tunggu dulu selama 15 mili-detik atau lebih.
2) Kirimkan perintah 30h, artinya trasfer data antar M1632 dan mikrokontroler dilakukan dengan mode 8 bit
3) Tunggu selama 4.1 mili-detik
4) Kirimkan sekali lagi perintah 30h
5) Tunggu lagi selama 100 mikro-detik

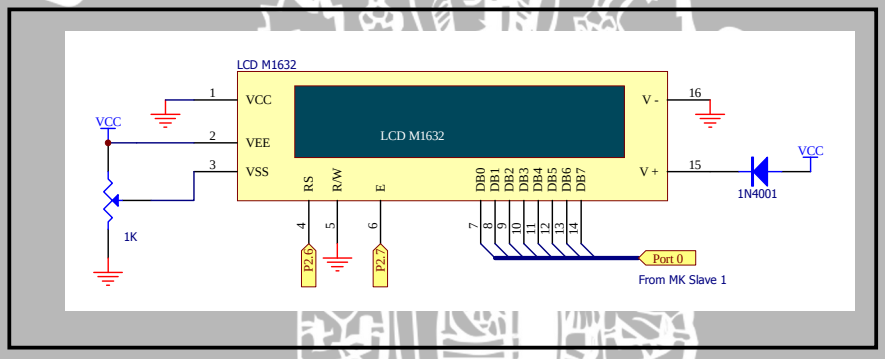
Setelah langkah-langkah tersebut di atas M1632 barulah bisa menerima data dan menampilkannya dengan baik. Pada awalnya tampilan akan nampak kacau, dengan demikian perlu segera dikirim perintah menghapus tampilan dan lain sebagainya, sesuai dengan petunjuk yang ada di Lembar Data.

Di atas dipakai AT89C2051 sebagai contoh, meskipun demikian semua yang dibahas di atas sepenuhnya bisa dipakai pada mikrokontroler MCS 51. Dalam pemakaiannya karena berbagai macam alasan, bisa saja sinyal **E**; **RW** dan **RS** tidak disimulasikan di **P3.5**; **P3.4** dan **P3.7**. Hal ini bisa diselesaikan dengan melakukan beberapa penyesuaian, yakni tentukan dulu perubahan rangkaian

sesuai dengan keadaan yang ada, dan perubahan rangkaian itu harus di sesuaikan di baris 1 sampai 3 pada potongan program di atas.

M1632 mempunyai 8 jalur data dan memerlukan 3 jalur kontrol, dalam suatu rangkaian yang memakai banyak port dari MC-S51, bisa terjadi kekurangan port untuk menghubungkan MCS51 ke M1632. Jika sampai terjadi hal semacam ini bisa ditempuh hal hal berikut :

1) M1632 dipakai dalam mode data 4 bit, yakni hanya memakai jalur data **D0..D3**
2) Dengan sedikit tambahan rangkaian sinyal **WR** dan **RD** diubah menjadi sinyal **E** dan **R/W** gaya Motorola, sehingga tidak perlu menyediakan port untuk men-simulasikan sinyal-sinyal tersebut. Berikut adalah gambar rangkaian LCD dengan komponen-komponen pendukung dengan pin-pin yang akan dihubungkan pada mikrokontroler MCS 51 :



Gambar 2.12. Rangkaian LCD M1632

LCD M1632 mempunyai spesifikasi sebagai berikut :

1) Memiliki 16 karakter dan dua baris tampilan yang terdiri dari 5 x 7 dot matrik ditambah dengan kursor.
2) Pembangkit karakter ROM untuk 192 jenis karakter.
3) Pembangkit karakter RAM untuk 8 jenis karakter.
4) 80 x 8 display data RAM (max 80 karakter).
5) Isolator didalam modul.
6) Memerlukan catu daya ± 5 volt.
7) Otomatis reset saat catu daya dinyalakan.

Tabel 2-4. Fungsi Pin – Pin LCD

No. PIN	Nama PIN	Fungsi
1	Vss	Terminal Ground
2	Vcc	Tegangan Catu + 5 volt
3	Vee	Mengendalikan kecerahan LCD
4	RS	Sinyal pemilihan register 0 = Tulis 1 = Baca
5	R/W	Sinyal seleksi tulis atau baca 0 = Tulis 1 = Baca
6	E	Sinyal operasi awal yang mengaktif kan data tulis atau baca
7 - 14	DB0 – DB7	Merupakan saluran data berisi perin tah data yang akan ditampilkan
15	V + BL	Back Light Supply 5 Volt (Volt)
16	V – BL	Back Ligth Supply 0 (Ground)

Pada LCD juga terdapat instruksi – instruksi sebagai berikut :

- Display clear : membersihkan tampilan yang ada pada LCD.
- Cursor home : hanya membersihkan tampilan dan kursor kembali ke semula.
- Empty mode Set : layar beraksi sebagai tampilan tulis.

  S : 1/0 = menggeser layar.

  1/0 : 1 = kursor bergerak ke kanan dan layar bergerak ke kiri.

- Display On/Off kontrol.

  D : 1 = layar on

D : 0 = layar off

C : 1 = kursor on

C : 0 = kursor off

B : 1 = kursor berkedip-kedip

B : 0 = kursor tidak berkedip – kedip

- Cursor Display Shift

  S/C : 1 = LCD diidentifikasikan sebagai layar

  S/C : 0 = LCD diidentifikasikan sebagai kursor

  R/L : 1 = menggeser satu spasi ke kanan

  R/L : 0 = menggeser satu spasi ke kiri

- Fuction Set

  DL : 1 = panjang data LCD pada 8 bit

  DL : 0 = panjang data LCD pada 4 bit

  Bit upper ditransfer terlebih dahulu kemudian diikuti dengan 4 bit lower.

  N : 1/0 = LCD menggunakan 2 atau 1 baris karakter

  P : 1/0 = LCD menggunakan 5 x 10 dot matrik

- CG RAM address set : menulis alamat RAM ke karakter
- DD RAM address set : menulis alamat RAM ke tampilan
- BF/address set : BF = 1/0, LCD dalam keadaan sibuk atau tidak sibuk.
- Data write to CG RAM or DD RAM : membaca byte dari alamat terakhir RAM yang dipilih.

BAB III
METODOLOGI

Pada bab ini akan diuraikan metodologi yang akan dilakukan dalam Perancangan Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51. Perancangan sistem ini mengacu pada rumusan masalah yang telah dibuat sebelumnya. Metodologi yang digunakan secara umum meliputi pengumpulan data berupa studi literatur, perancangan sistem, pembuatan perangkat keras *(hardware)* dan perangkat lunak *(software*), pengujian alat dan analisis hasil pengujian, serta pengambilan kesimpulan dan saran.

3.1 Studi Literatur

Mempelajari segala literatur yang berhubungan dengan pembuatan sistem pada skripsi ini, meliputi studi tentang karakteristik komponen yang akan digunakan serta semua teori berdasarkan blok diagram sistem yang telah dirancang, yaitu: rangkaian mikrokontroler, rangkaian ISD, rangkaian RTC, rangkaian display LCD, rangkaian push-button, dan rangkaian power supply.

3.2 Perencanaan dan Pembuatan Sistem

Perencanaan dan pembuatan sistem dilakukan dalam beberapa langkah, langkah-langkah perencanaan dan pembuatan sistem tersebut adalah:

a. Perancangan keseluruhan sistem

   Menghubungkan semua perangkat keras (*Hardware*) beserta program (*Software*) yang menunjang kinerja dari alat yang digunakan.

b. Penentuan spesifikasi sistem.

   Penentuan spesifikasi sistem dilakukan untuk memberikan keterangan mengenai hal-hal yang berkaitan dengan kerja sistem.

c. Pembuatan blok diagram sistem.

   Penyusunan blok diagram sistem dilakukan untuk mempermudah pemahaman mengenai alur kerja alat yang akan dibuat.

d. Perencanaan rangkaian untuk masing-masing blok :

1) Perencanaan perangkat keras (*Hardware*)

Perangkat keras pada sistem ini terdiri dari :

- Rangkaian mikrokontroler
- Rangkaian ISD
- Rangkaian LCD
- Rangkaian RTC
- Rangkaian push-button
- Rangkaian Power Supply

2) Perencanaan perangkat lunak (*Software*)

Perancangan perangkat lunak dimulai dengan proses membuat diagram alir. Diagram alir ini sebenarnya adalah langkah-langkah kerja yang disusun secara logis dan sistematis sebagai gambaran perintah- perintah yang akan dijalankan oleh mikrokontroler dalam megendalikan keseluruhan rangkaian. Langkah-langkah tersebut kemudian di implementasikan menjadi baris-baris perintah dalam bahasa assembly.

3.3 Pengujian Sistem

Pengujian bertujuan untuk menguji apakah alat yang dibuat dapat berjalan seperti yang diharapkan. Pengujian dilakukan pada perangkat keras maupun perangkat lunak. Metode pengujian alat adalah sebagai berikut:

1. Pengujian rangkaian power supply
2. Pengujian rangkaian mikrokontroler
3. Pengujian rangkaian ISD
4. Pengujian rangkaian LCD
5. Pengujian Rangkaian RTC
6. Pengujian rangkaian push-button
7. Pengujian rangkaian secara keseluruhan

3.4 Analisis

Analisis dilakukan setelah sistem diuji coba dan diambil datanya. Kemudian dianalisis apakah sesuai dengan teori yang ada, dibandingkan dengan spesifikasi alat serta penngevaluasian sistem kerja alat.

3.5 Kesimpulan dan Saran

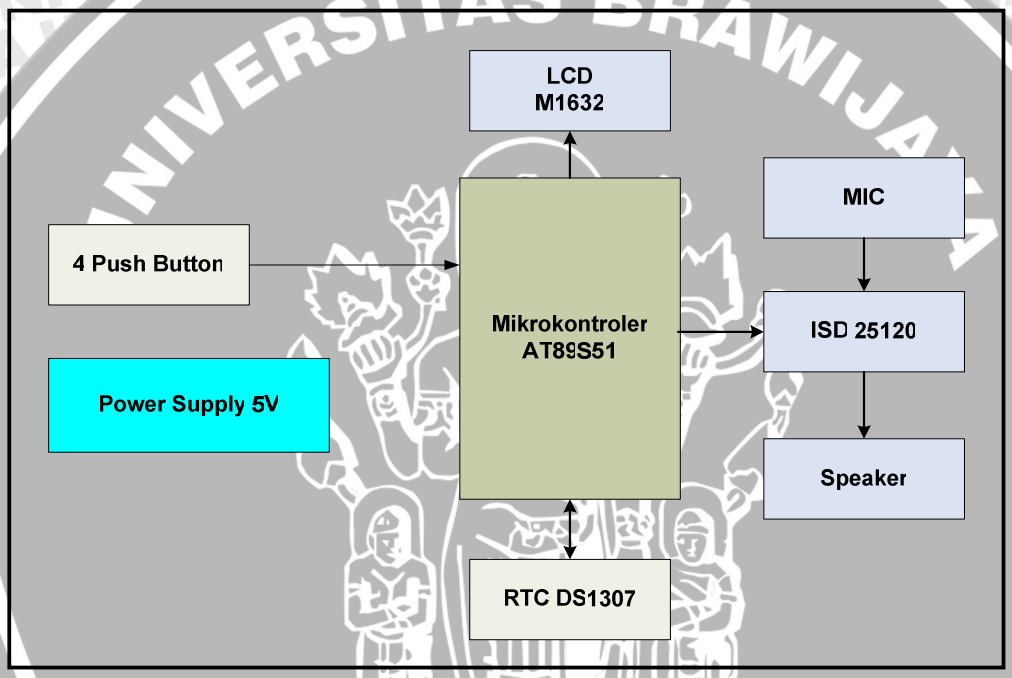
Kesimpulan didapat berdasarkan hasil perealisasian dan pengujian alat sesuai dengan tujuan dan rumusan masalah. Saran diberikan setelah melihat adanya kekurangan dalam sistem yang telah dibuat, dengan harapan agar alat ini dapat dikembangkan menjadi lebih baik.

BAB IV

PERANCANGAN DAN PEMBUATAN ALAT

4.1. Diagram Blok

Secara umum Perancangan Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51 yang akan dibuat, digambarkan secara diagram blok sebagai berikut :



Gambar 4.1. Blok Diagram

- Mikrokontroler AT89S51 sebagai pengendali utama keseluruhan rangkaian.
- RTC DS1307 sebagai pewaktu yang menghitung berdasarkan tahun, tanggal, jam, menit dan detik.
- ISD 25120 digunakan untuk merekam suara sehingga dihasilkan output suara.
- Loud Speaker digunakan untuk mengeluarkan output suara.
- Mic digunakan untuk proses perekaman suara pada ISD.
- 4 Push-button digunakan sebagai input untuk seting sistem bel sekolah, yaitu tombol menu, tombol up, tombol down, dan tombol reset.

- Buzzer digunakan sebagai alarm bel sekolah
- LCD digunakan sebagai display.
- Power Supply digunakan untuk memberikan tegangan kerja yang dibutuhkan untuk keseluruhan rangkaian.

4.2. Prinsip Kerja Alat

Pertama-tama kita dapat melakukan seting waktu bel/alarm melalui tombol push button menu, up, dan down. Pada saat waktu telah sesuai dengan input nilai waktu alarm, maka secara otomatis mikrokontroler akan mengeluarkan suara alarm sesuai dengan seting alarm yang telah di inputkan.

4.3. Mikrokontroler AT89S51

Mikrokontroller AT89S51 dirancang untuk dapat berdiri sendiri, karena sudah terdapat *EPROM*, *RAM* serta *Port I/O internal*. Untuk berhubungan dengan peralatan luar *chip* dibutuhkan 3 *Bus* yaitu :

1) *Data Bus*

   Yaitu jalur untuk *input - output* data yang lebarnya sesuai dengan data yang diolah oleh mikrokontroler, yaitu 8 *bit*.

2) *Address Bus*

   Yaitu jalur *input - output* atau dari *memori* yang dihubungi, sehingga pada suatu saat hanya ada satu *device* yang berhubungan dengan *CPU*. Lebar *address bus* mikrokontroler AT89S51 adalah 16 *bit* (A0 - A15).

3) *Control Bus*

   Berfungsi sebagai pengatur *sinkronisasi* hubungan antara CPU dengan *device* Luar.

4.3.1. Pemetaan *Memori*

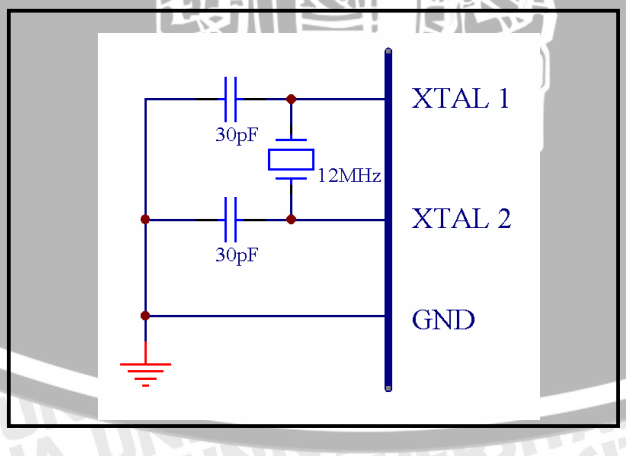
Mikrokontroller AT89S51 memiliki 16 *bit address* (A0 - A15) dengan demikian kapasitas maksimumnya adalah 2^{16} = 65536 *byte* = 64 *Kbyte* dengan

alamat 0000H-FFFFH. Jika alamat-alamat program lebih tinggi dari 0FFFH, yang melebihi kapasitas *RAM internal* menyebabkan mikrokontroler secara otomatis mengambil *code byte* dari program *memori* eksternal. *Code byte* juga hanya diambil dari *memori* eksternal dengan alamat 0000H-0FFFH dengan menghubungkan *Pin* EA ke *ground*.

Dalam perancangan ini hanya menggunakan 4 *Kbyte*, karena program sudah mencukupi, sehingga *pin* EA dihubungkan ke *VCC*. Mikrokontroler AT89S51 memiliki 4 *Kbyte memori internal* yang dapat diprogram dan dihapus sesuai dengan keinginan, dan bersifat *non volatile* (tidak hilang pada saat catu daya terputus).

4.3.2. Rangkaian *Clock*

Mikrokontroler AT89S51 ini memiliki *internal clock*, yang berfungsi sebagai sumber *clock*, tapi masih diperlukan rangkaian tambahan untuk membangkitkan *clock* yang diperlukan. Rangkaian ini terdiri dari 2 buah kapasitor dan sebuah kristal dengan ketentuan seperti gambar berikut :



Gambar 4.2. Rangkaian *Clock* AT89S51

4.3.3. Rangkaian *Reset*

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

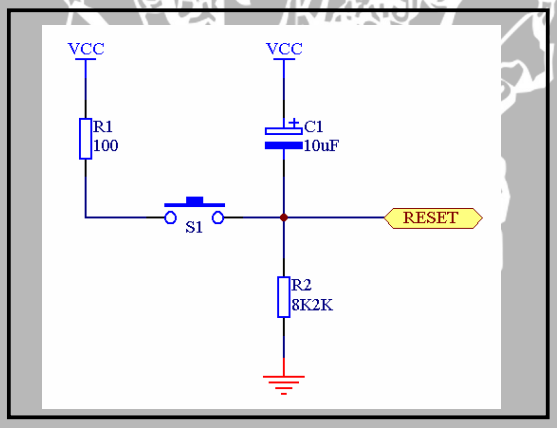
Jika saklar S1 ditekan , *reset* bekerja secara manual, aliran arus akan mengalir dari *VCC* melalui R1 menuju kaki RST. Tegangan di RST atau VR2 akan berubah menjadi :

$$VR2 = \frac{R2xVCC}{R1+R2}$$

$$= \frac{8200x5}{100+8200}$$

$$= 4{,}94\,V$$

Saat saklar dilepas, aliran arus dari *VCC* melalui R1 akan berhenti dan tegangan pada kaki RST akan turun menuju ke nol dan proses *reset* selesai.



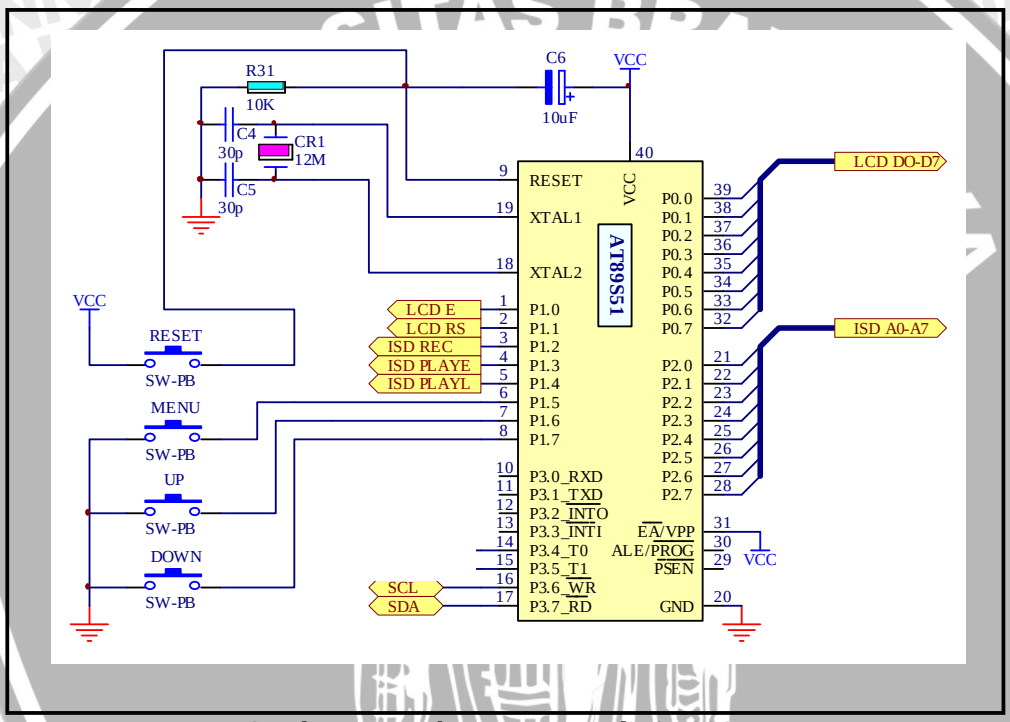
Gambar 4.3. Rangkaian *Reset*

4.3.4. Hubungan *Pin* Pada AT89S51

Diskripsi konfigurasi *Pin-Pin* AT89S51 pada rangkaian:

- *Pin* 1 dan 2 : dihubungkan ke *LCD* E dan RS
- *Pin* 3,4 dan 5 : dihubungkan ke rangkaian ISD PR, PD, CE, EOM

- *Pin* 6, 7 dan 8 : dihubungkan *Push-button Menu, Up,* dan *Down*
- *Pin* 9 : dihubungkan *Push-button reset*
- Pin 16 dan 17 : dihubungkan SCL dan SDA DS1307
- Pin 18 dan 19 : dihubungkan *X-tal* 12 MHz
- *Pin* 20 : dihubungkan ke GND
- *Pin* 32-39 : dihubungkan ke *LCD* D0-D7
- *Pin* 31 dan 40 : dihubungkan ke *VCC*



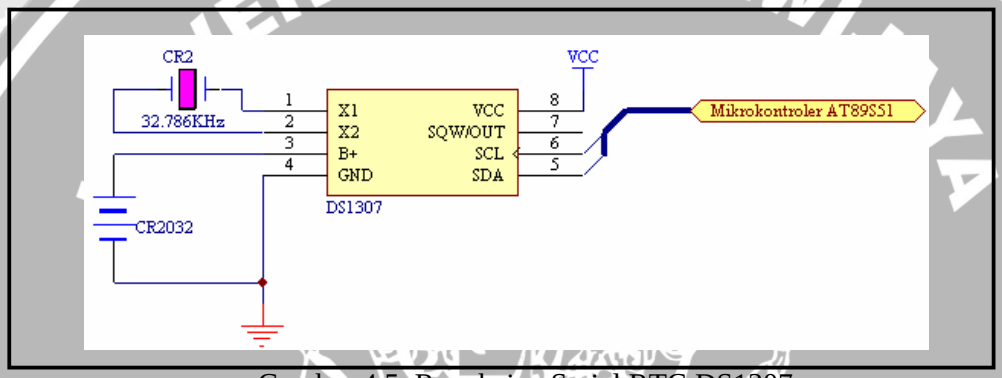
Gambar 4.4. Hubungan *Pin* Pada AT89S51

4.4. Rangkaian RTC DS1307

DS1307 adalah IC serial *Real Time Clock (RTC)* dimana alamat dan data ditransmisikan secara serial melalui sebuah jalur data dua arah. Karena menggunakan jalur data serial maka hanya memerlukan dua buah pin saja untuk komunikasi. Yaitu pin untuk data dan pin untuk sinyal clock. Dalam system yang kita rancang ini RTC DS1307 difungsikan sebagai pewaktu, yaitu peripheral yang menyediakan data detik, menit,jam, hari, tanggal, bulan dan tahun. Data waktu ini nantinya akan diolah oleh mikrokontroler dan ditampilkan pada LCD.

Pemilihan penggunaan serial RTC DS1307 pada perancangan ini adalah dikarenakan RTC DS1307 mempunyai beberapa kelebihan sebagai berikut::

- Harga yang relative lebih murah jika dibandingakan dengan RTC lainnya
- Karena akses data dilakukan secara serial, maka hanya butuh 2 pin saja, sehingga akan menghemat port mikrokontroler
- Tidak perlu lagi ada perbaikan penanggalan untuk penggunaan ditahun 2000 ke atas
- Sudah tersedia pin untuk baterai back-up, sehingga tidak perlu lagi dibuatkan rangkaian untuk baterai back-up, apabila kehilangan supply tegangan



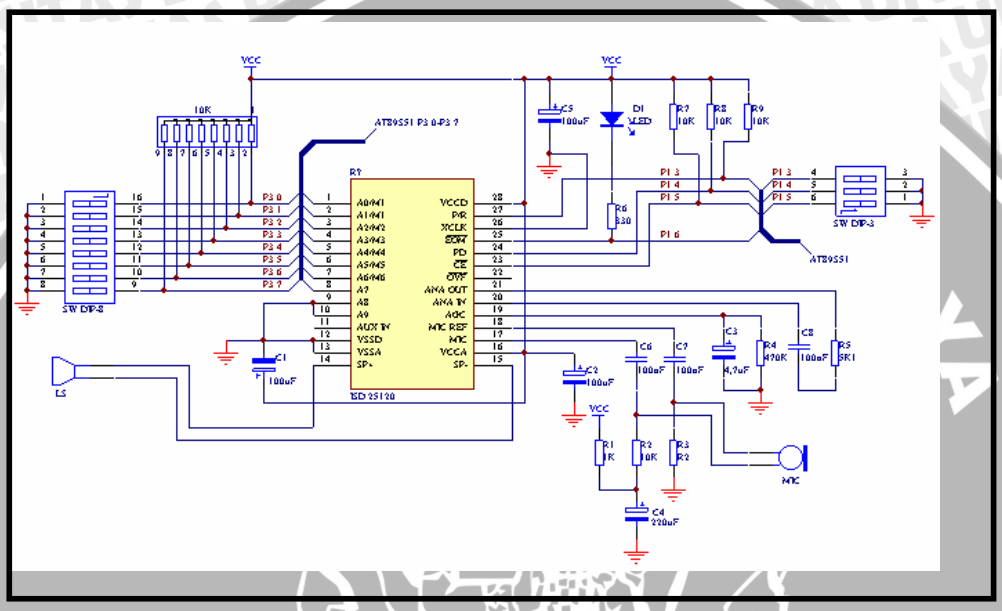
Gambar 4.5. Rangkaian Serial RTC DS1307

Baterai back-up yang digunakan adalah baterai back-up 3V CR2032, yang dapat bertahan untuk masa operasi 10 tahun, kondisi ini amat hemat biaya. Kristal yang digunakan adalah standart quartz kristal dengan dinilai 32,768KHz. Pin SDA dan SCL dari RTC serial DS1307 ini dihubungkan ke port mikrokontroler AT89S51 P3.6 dan P3.7.

4.5. Rangkaian Pemutar Suara ISD25120

Pada rangkaian ini, suara di rekam menggunakan *Information Storage Device* ISD 25120 yang disimpan dalam EEPROM. Berdasarkan data sheet, ISD 25120 ini mampu merekam suara dengan lama perekaman 120 detik dengan alamat yang berbeda. Rangkaian ISD 25120 ditunjukkan dalam Gambar 4.6. Pin *address* data menerima masukan 8 bit dari mikrokontroler *AT89S51* pada Port 2. Alamat-alamat ini akan memilih data suara yang mana yang akan dipanggil. Kemudian untuk pin *Chip Enable* (CE) berfungsi sebagai pengaktifan (*Enabled),*

secara aktif *low* yang terkoneksi dengan *Port* 1.5. Pada pin 25 *End Of Massage* (EOM) merupakan pulsa bahwa akhir dari data suara yang ditandai dengan aktif *low* sebentar, ini terhubung dengan *Port 1.6* dan indikator sebuah Led. Untuk *Play/Record* (P/R) pin 27 berfungsi untuk merekam dan memainkan lagi dengan memberikan aktif *low*, pin ini terhubung dengan mikrokontroler *Port 1.3*.



Gambar 4.6. Rangkaian ISD 25120

Sinyal suara merupakan bentuk sinyal analog kemudian diubah menjadi bentuk digital untuk disimpan ke dalam *memory*. Data-data digital yang sudah tersimpan yang berasal dari data analog (suara) dapat dipanggil kembali dengan memanggil alamat penyimpanan datanya. Proses perekaman pada ISD 25120 adalah sebagai berikut :

1) Pin *Chip Enable* (CE) pin 23 mendapat logika *low*.
2) Memberikan alamat dengan mengatur dip *switch.*
3) Kemudian Pin *Playback/Record* pin 27 mendapat logika *low*.
4) Perekaman dimulai dengan memasukkan data suara pada mikrofon.
5) Pin *Playback/Record* mendapat logika *high* kembali.
6) Mencari alamat terakhir yang ditandai pada *End Of Message* (EOM) pin 25 terjadi pulsa *low* sesaat kemudian *high* kembali.

7) Demikian seterusnya, untuk melanjutkan perekaman alamat terkhir dari sebelumnya diberikan spasi dan memulai perekaman kembali sampai batas waktu dari kemampuan ISD-nya.

Kemudian untuk prosedur pemanggilan data-data suara yang telah direkam adalah sebagai berikut :

1) Pin *Chip Enable* (CE) pin 23 mendapat logika *low*.
2) Memanggil alamat suara yang diinginkan, pada waktu perekaman tadi.
3) Kemudian memberikan logika *low* pada *Playback/Record* pin 27, yang berarti pemanggilan suara dimulai.
4) Menunggu pada *End Of Message* (EOM) pin 25 terdapat pulsa logika *low* sesaat.
5) Bila sudah terdapat pulsa logika *low* sesaat pada EOM, segera diberikan logika *high* pada pin *Chip Enable* CE, ini menandakan akhir dari data suara pada alamat itu.
6) Demikian seterusnya pada alamat-alamat selanjutnya, sesuai dengan data suara yang diinginkan.

4.6. Perancangan Rangkaian *LCD*

LCD Display Module M1632 buatan *Seiko Instrument Inc*. terdiri dari dua bagian, yang pertama merupakan panel *LCD* sebagai media penampil informasi dalam bentuk huruf/angka dua baris, masing-masing baris bisa menampung 16 huruf/angka. Bagian kedua merupakan sebuah sistem yang dibentuk dengan mikrokontroler yang ditempelkan dibalik pada panel *LCD*, berfungsi mengatur tampilan informasi serta berfungsi mengatur komunikasi M1632 dengan mikrokontroler yang memakai tampilan *LCD* itu..

Untuk berhubungan dengan mikrokontroler pemakai, M1632 dilengkapi dengan 8 jalur data (**DB0..DB7**) yang dipakai untuk menyalurkan kode *ASCII* maupun perintah pengatur kerjanya M1632. Selain itu dilengkapi pula dengan **E**, **R/W** dan **RS** seperti layaknya komponen yang kompatibel dengan mikroprosesor. Kombinasi lainya **E** dan **R/W** merupakan sinyal standar pada komponen buatan Motorola. Sebaliknya sinyal-sinyal dari MCS51 merupakan sinyal khas *Intel* dengan kombinasi sinyal **WR** dan **RD.**

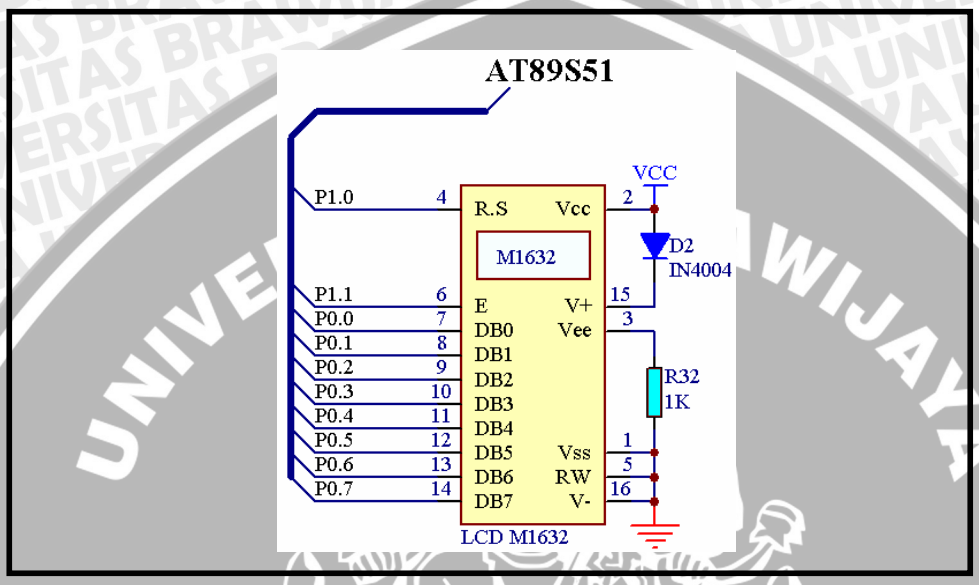
RS, singkatan dari *Register Select*, dipakai untuk membedakan jenis data yang dikirim ke M1632, kalau **RS**=**0** data yang dikirim adalah perintah untuk mengatur kerja M1632, sebaliknya kalau **RS**=**1** data yang dikirim adalah kode *ASCII* yang ditampilkan. Demikian pula saat pengambilan data, saat **RS**=**0** data yang diambil dari M1632 merupakan data status yang mewakili aktifitas M1632, dan saat **RS**=**1** maka data yang diambil merupakan kode *ASCII* dari data yang ditampilkan. Proses mengirim/mengambil data ke/dari M1632 bisa dijabarkan sebagai berikut :

1) **RS** harus dipersiapkan dulu, untuk menentukan jenis data seperti yang telah dibicarakan di atas.
2) **R/W** di-nol-kan untuk menandakan akan diadakan pengiriman data ke M1632. Data yang akan dikirim disiapkan di **DB0..DB7**, sesaat kemudian sinyal **E** di-satu-kan dan di-nol-kan kembali. Sinyal **E** merupakan sinyal sinkronisasi, saat **E** berubah dari 1 menjadi 0 data di **DB0 .. DB7** diterima oleh M1632.
3) Untuk mengambil data dari M1632 sinyal **R/W** di-satu-kan, menyusul sinyal **E** di-satu-kan. Pada saatu **E** menjadi 1, M1632 akan meletakkan datanya di **DB0 .. DB7**, data ini harus diambil sebelum sinyal **E** di-nol-kan kembali.

M1632 mempunyai seperangkat perintah untuk mengatur tata kerjanya, perangkat perintah tersebut meliputi perintah untuk menghapus tampilan, meletakkan kembali cursor pada baris huruf pertama baris pertama, menghidup/matikan tampilan dan lan sebagainya, semua itu dibahas secara terperinci dalam lembar data M1632. Setelah diberi sumber daya, ada beberapa langkah persiapan yang harus dikerjakan dulu agar M1632 bisa dipakai, langkah-langkah tersebut antara lain adalah :

1) Tunggu dulu selama 15 mili-detik atau lebih.
2) Kirimkan perintah 30h, artinya trasfer data antar M1632 dan mikrokontroler dilakukan dengan mode 8 *bit*
3) Tunggu selama 4.1 mili-detik
4) Kirimkan sekali lagi perintah 30h
5) Tunggu lagi selama 100 mikro-detik

Setelah langkah-langkah tersebut di atas M1632 barulah bisa menerima data dan menampilkannya dengan baik. Pada awalnya tampilan akan nampak kacau, dengan demikian perlu segera dikirim perintah menghapus tampilan dan lain sebagainya, sesuai dengan petunjuk yang ada di lembar data.



Gambar 4.7. Rangkaian *LCD* Pada Mikrokontroler

Diskripsi konfigurasi *Pin-Pin* pada *LCD*:

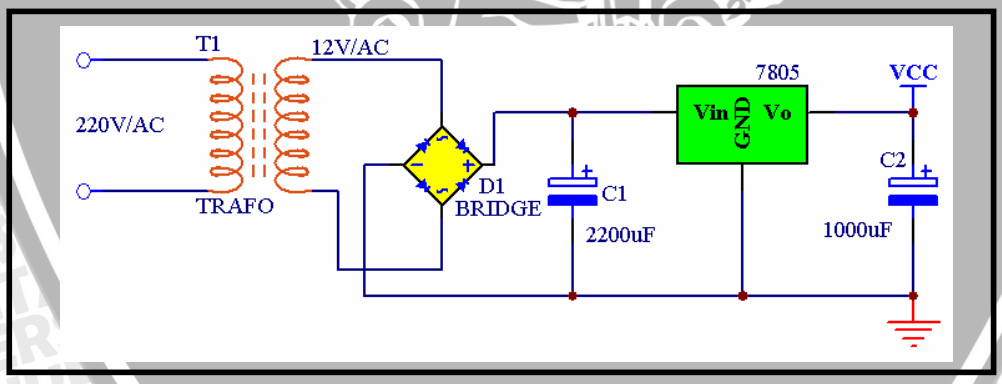
- *Pin* 1,3,5 dan 16 : dihubungkan dengan *ground*
- *Pin* 2 : dihubungkan dengan *VCC*
- *Pin* 4 : dihubungkan dengan P1.0 AT89S51
- *Pin* 6 : dihubungkan dengan P1.1 AT89S51
- *Pin* 7-14 : dihubungkan dengan AT89S51 (P0.0-P0.7)

Untuk tampilan dipergunakan *LCD* Dot Matrik 2 x 16 karakter. Sinyal-sinyal yang diperlukan oleh *LCD* adalah RS dan *Enable*, sinyal RS dan *Enable* dipergunakan sebagai *input* yang outputnya dipakai untuk mengaktifkan *LCD*. *LCD* akan aktif apabila mikrokontroler memberikan instruksi tulis pada LCD. Saat kondisi RS *don't care* dan *Enable* 0 maka *LCD* tetap pada kondisi semula, pengiriman data ke *LCD* dilakukan saat RS berlogika 0 dan *enable* berlogika 1.

Instruksi dikirim pada *LCD* bila keadaan RS 1 dan *Enable* 1. Pin *LCD* ini untuk data terkoneksi pada *Port 0* mikrokontoler AT89S51. Kemudian untuk RS dihubungkan pada *Port* 1.0, tulis/baca (*Read/Write)* diberikan logika *low* karena disini LCD bersifat menulis data, dan yang terakhir *Enable* (E) dikendalikan dengan *Port 1.1*.

4.7. Rangkaian *Power Supply*

Rangkaian *power supply* dibutuhkan sebagai sumber tegangan kerja untuk keseluruhan rangkaian. Rangkaian *power supply* mendapatkan sumber tegangan dari tegangan jala-jala PLN sebesar 220V/AC. Tegangan 220V/AC ini kemudian diturunkan menjadi 12V/AC menggunakan transformator penurun tegangan. Tegangan AC 12V kemudian disearahkan oleh dioda *bridge* menjadi tegangan DC. Keluaran dari dioda *bridge* ini kemudian diinputkan ke IC regulator LM7805 yang akan menghasilkan output tegangan DC sebesar +5V untuk memberikan *supply* tegangan pada tiap-tiap rangkaian. Elco 2200uF dan 1000uF digunakan untuk membuang *ripple* pada tegangan DC. Gambar rangkaian *power supply* ditunjukkan pada gambar berikut.

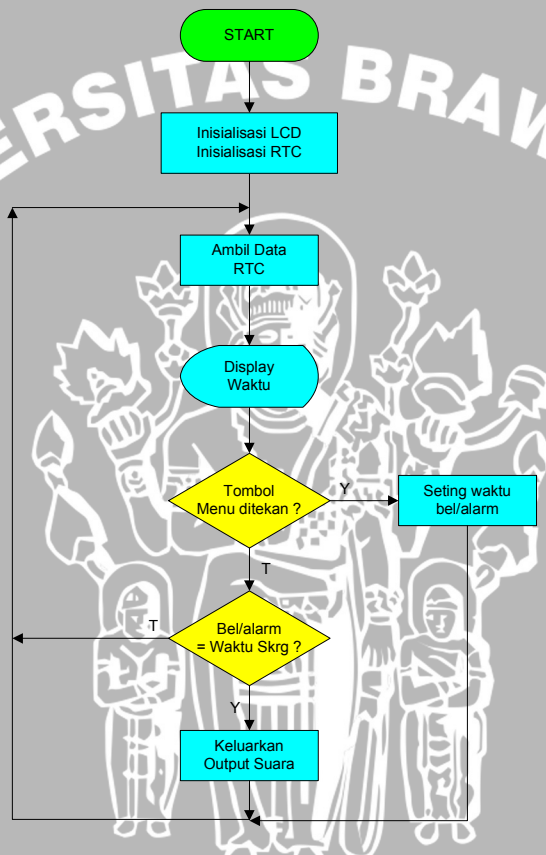


Gambar 4.8. Rangkaian *Power Supply*

4.8. Perangkat Lunak *(software)*

Perencanaan perangkat lunak sangat diperlukan untuk menjalankan *sistem* seperti yang diinginkan. Mikrokontroler AT89S51 tidak akan bisa dijalankan tanpa adanya *software*. Didalam perencanaan perangkat lunak ini di buat perintah-perintah yang berfungsi untuk mengatur urutan dan tata kerja dari keseluruhan sistem.

Flowchart



Gambar 4.9. *Flowchart* Keseluruhan Alat

BAB V
PENGUJIAN ALAT

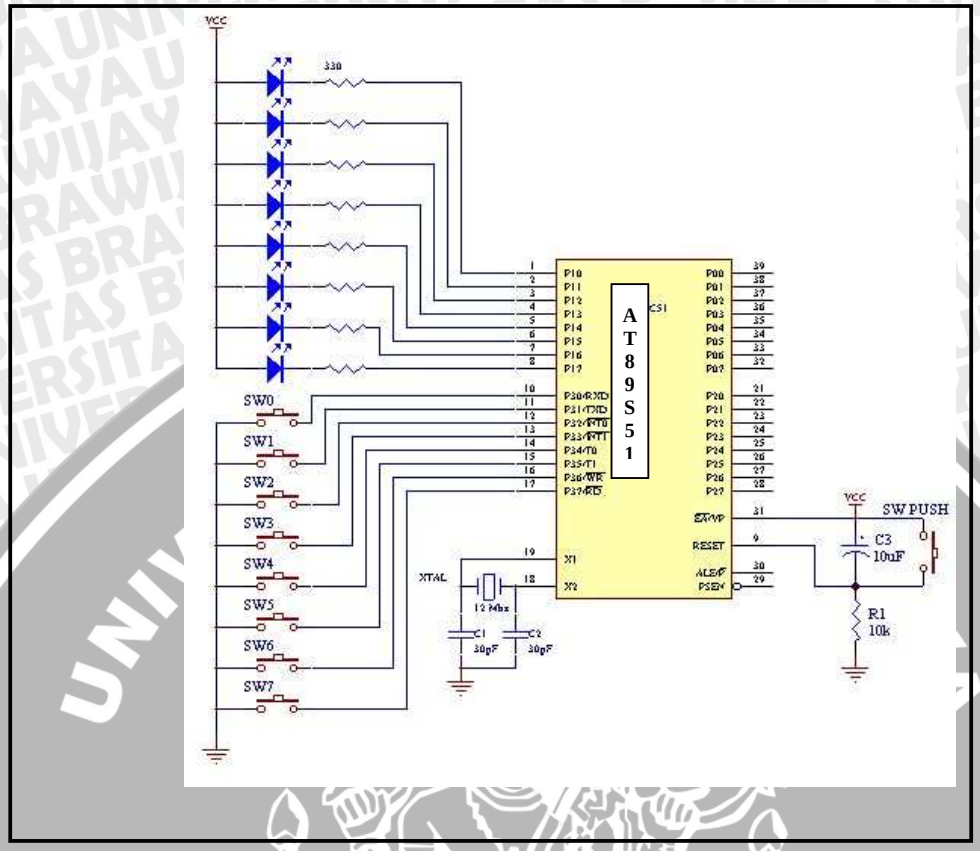
Setelah melakukan perancangan atau pembuatan sistem kontrol ini, maka kita perlu melakukan suatu pengujian sistem. Yang mana pengujian sistem ini bertujuan antara lain :

1) Mengetahui sejauh mana Sistem Reminder Berbasis Mikrokontroler AT89S51 ini berfungsi sebagaimana yang kita harapkan.
2) Mencari dan menemukan berbagai kendala yang mungkin timbul pada saat Sistem Bel/Alarm Sekolah Secara Otomatis Berbasis Mikrokontroler AT89S51 beroperasi untuk kemudian diperbaiki sampai pada tingkat kesalahan sistem yang sekecil mungkin sehingga didapatkan hasil yang sebaik-baiknya.

5.1. Pengujian Mikrokontroler AT89C51

Pengujian ini bertujuan untuk mengetahui apakah minimum sistem dan port-port pada mikrokontroler yang digunakan dapat berjalan dengan baik.

- **Peralatan yang Digunakan**
  1) Mikrokontroler AT89S51
  2) LED
  3) Mikrokontroler *writer*
- **Prosedur Pengujian**
  1) Merancang rangkaian seperti dalam Gambar 5.1
  2) Menguji dengan program
  3) Mencatat hasil keluaran pada tabel 5.1



Gambar 5.1. Rangkaian Pengujian Mikrokontroler dan Sistem Minimum

Listing Program :

```
        org 0h
mulai:  jnb p3.0,nol
        jnb p3.1,satu
        jnb p3.2,dua
        jnb p3.3,tiga
        jnb p3.4,empat
        jnb p3.5,lima
        jnb p3.6,enam
        jnb p3.7,tujuh
        jmp mulai
nol:    mov p1,#11111111b
        jmp mulai
satu:   mov p1,#11111110b
```

```
        jmp mulai
dua:    mov p1,#11111101b
        jmp mulai
tiga:   mov p1,#11111100b
        jmp mulai
empat:  mov p1,#11111011b
        jmp mulai
lima:   mov p1,#11111010b
        jmp mulai
enam:   mov p1,#11111001b
        jmp mulai
tujuh:  mov p1,#11111000b
        jmp mulai
end
```

Data Hasil Pengujian :

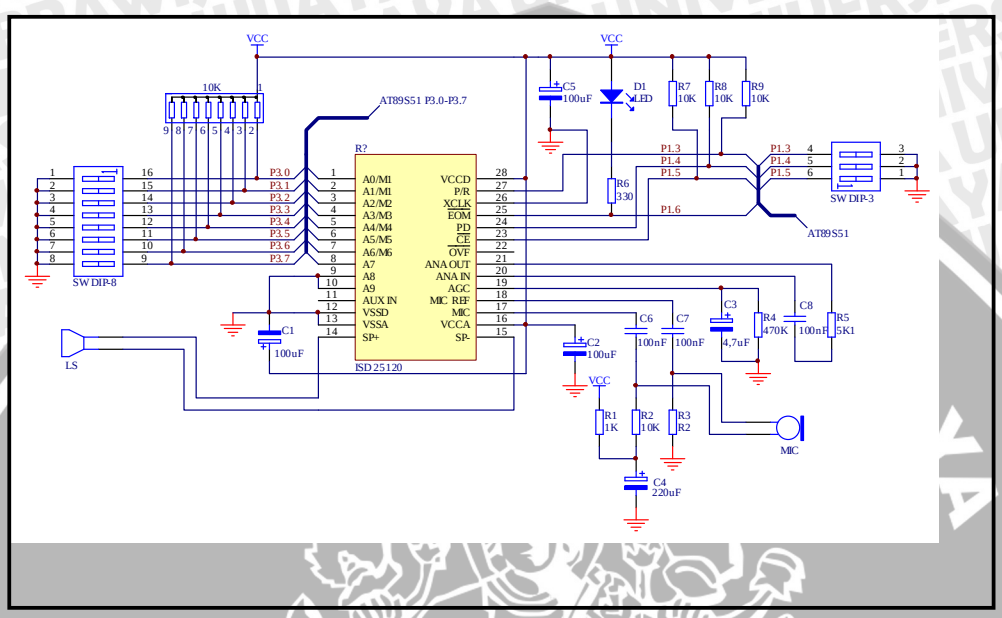
Tabel 5-1. Tabel Hasil Pengujian Mikrokontroler

Kondisi	Nyala LED
Tekan P3.0	1111 1111
Tekan P3.1	1111 1110
Tekan P3.2	1111 1101
Tekan P3.3	1111 1100
Tekan P3.4	1111 1011
Tekan P3.5	1111 1010
Tekan P3.6	1111 1001
Tekan P3.7	1111 1000

5.2. Pengujian Rangkaian Pemutar/Perekam Suara ISD 25120

Pengujian ini bertujuan untuk mengetahui apakah rangkaian ISD25120 yang telah dibuat apakah telah dapat bekerja seperti yang kita harapkan atau tidak. Selain itu juga untuk mengetahui alamat data suara yang disimpan pada memori ISD 25120. Adapun peralatan yang digunakan dalam pengujian ini adalah sebagai berikut:

1) *DIP Switch* 8 *bit.*
2) Rangkaian ISD 25120
3) *Power Supply* 5 Volt.
4) *Loudspeaker*.



Gambar 5.2. Rangkaian Pengujian ISD 25120

5.2.1. Langkah Pengujian

1) Memberikan alamat dengan memasukkan data pada *DIP-Switch.*
2) Menekan tombol *record*, untuk memulai merekam melalui *microphone.*
3) Mencatat alamat awal dan alamat akhirnya dan ditabelkan.
4) Memanggil alamat yang sudah direkam.
5) Menekan tombol *play* dan mendengarkan hasil suaranya.

5.2.2. Hasil Pengujian

Tabel 5-2. Hasil Pengujian Alamat Data Suara dan Hasil Suara

Hasil Suara	Alamat Data Suara 8 Bit (Biner)	Alamat Data Suara 8 Bit (Heksa)
“Nol”	0000 1000	08
“Satu”	0001 0000	10
“Dua”	0001 1000	18
“Tiga”	0010 0000	20
“Empat”	0010 1000	28
“Lima”	0011 0000	30
“Enam”	0011 1000	38
“Tujuh”	0100 0000	40
“Delapan”	0100 1000	48
“Sembilan”	0101 0000	50
“Sepuluh”	0101 1000	58
“Sebelas”	0110 0000	60

5.3. Pengujian Rangkaian Tampilan LCD

- **Tujuan**

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

- **Peralatan yang dibutuhkan**
  1) Power Supply 5 Volt
  2) Sistem Mikrokontroler dan LCD M1632
- **Prosedur Pengujian**
  1) Menyusun rangkaian seperti dalam Gambar 5.3
  2) Menjalankan program untuk menampilkan tulisan

'PROGRAM '

' TES LCD M1632 '

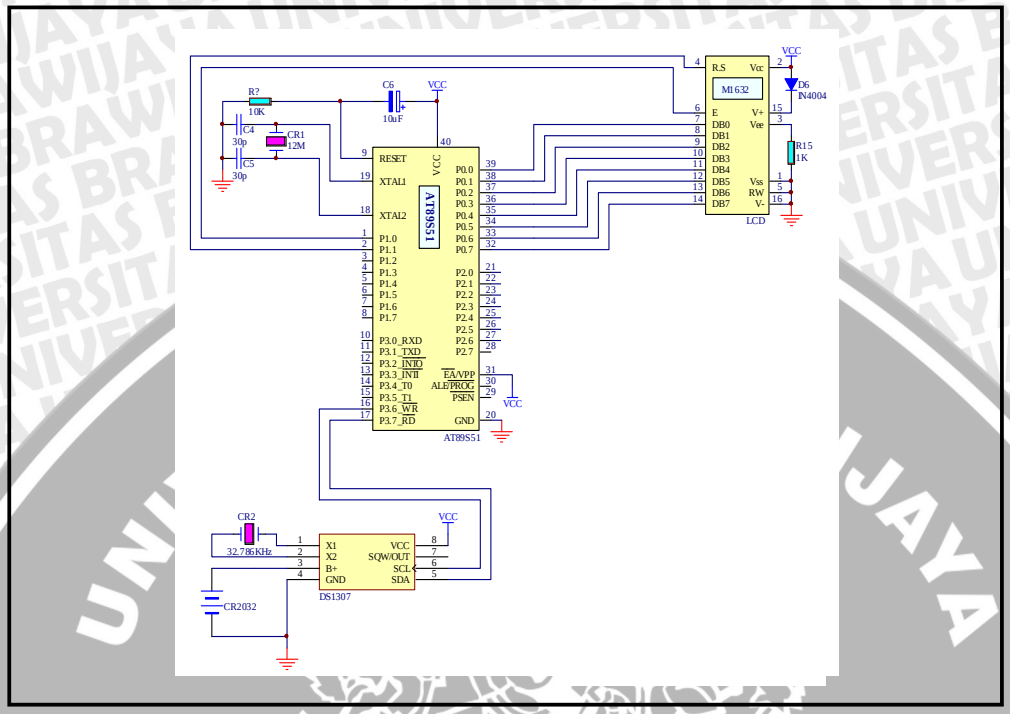
3) Mengamati keluaran pada LCD

Gambar 5.3. Foto Tampilan LCD Hasil Percobaan

5.4. Pengujian Terhadap Rangkaian RTC DS1307

RTC DS1307 merupakan modul *Real Time Clock*, yaitu sebuah modul yang mempunyai sistem jam digital yang bekerja secara independent. Artinya, sistem jam digital pada modul RTC DS1307 ini bekerja mengaktifkan sistem jam digital di mana besaran jam, menit dan detik tersimpan dalam register-register tertentu dalam modul RTC DS1307. Seperti pembahasan pada bab sebelumnya bahwa RTC DS1307 berfungsi sebagai pewaktu maka pengujian dapat dilakukan dengan cara sebagai berikut:

1) Membuat rangkaian seperti gambar 5.4
2) Memasukkan program untuk tes RTC pada mikrokontroler.
3) Menggunakan keypad untuk seting awal waktu.
4) Mengamati Waktu pada LCD.



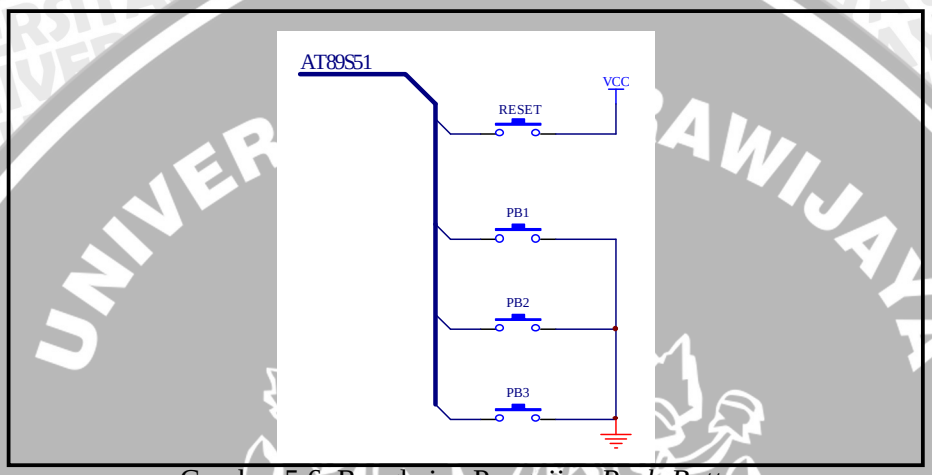
Gambar 5.4. Rangkaian Pengujian RTC DS1307

Gambar 5.5. Foto Hasil Pengujian Rangkaian RTC DS1307

5.5. Pengujian Terhadap Rangkaian *Push-button*

Rangkaian *push-button* berfungsi sebagai inputan mikrokontroler untuk seting sistem reminder. *Push-button* diberikan input tegangan sebesar 4,9 *Volt* dan dirangkai dengan menggunakan *common ground*. Sehingga apabila terjadi penekanan pada tombol *push-button*, maka akan terjadi arus hubung singkat yang menyebabkan kondisi tegangan pada pin *push-button* yang terhubung dengan mikrokontroler berubah dari kondisi *"high"* 4,9 *Volt* menjadi kondisi *"low"* 0 *Volt.*

Pengujian rangkaian *push-button* dilakukan dengan cara mengukur tegangan pada *pin* mikrokontroler yang terhubung dengan rangkaian *push-button* dengan menggunakan multimeter *DC* dengan batas 20V/*DC*. Kondisi tegangan pada pin mikrokontroler yang terhubung dengan rangkaian *push-button* sebelum adanya penekanan pada tombol *push-button* adalah 4,9V. Berikut cara pengukuran tegangan pada rangkaian *push-button*.



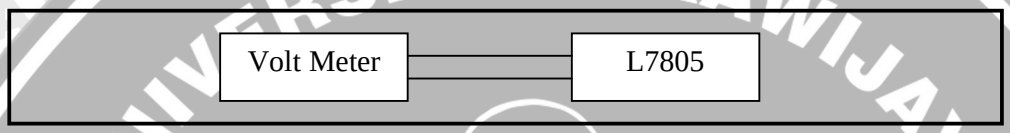
Gambar 5.6. Rangkaian Pengujian *Push-Button*

Tabel 5-3. Hasil Pengukuran Pada Rangkaian *Push-Button*

Push-Button (Tertekan)	P1.5 (V)	P1.6 (V)	P1.7 (V)	RESET (V)
PB 1 (*Menu)*	0	4,9	4,9	0
PB 2 *(Up)*	4,9	0	4,9	0
PB 3 *(Down)*	4,9	4,9	0	0
RESET	4,9	4,9	4,9	4,9

5.6. Pengujian Terhadap Rangkaian *Power Supply*

Pengujian ini bertujuan untuk mengetahui tegangan yang dikeluarkan oleh rangkaian *power supply* yang telah dibuat. Dengan begitu dapat diketahui apakah terjadi kesalahan terhadap rangkaian *power supply* atau tidak. Tegangan yang dibutuhkan untuk memberikan tegangan kerja untuk rangkaian keseluruhan adalah +5V. Untuk mengukur besarnya tegangan pada rangkaian *power supply* maka pada pengujian rangkaian *power supply* ini menggunakan Multimeter Digital. Berikut cara pengukuran untuk pengambilan data besarnya tegangan pada rangkaian *power supply*.



Gambar 5.7. Diagram Blok Pengujian Tegangan *Power Supply*

Untuk mengukur besarnya tegangan keluaran rangkaian *power supply* maka tegangan masukan pada rangkaian *power supply* juga harus tersambung dengan sumber tegangannya yaitu baterai 9 Volt. Multimeter digital di-seting pada *DC Volt* dengan batas maksimal pengukuran yaitu 20 *Volt DC*. Pengukuran tegangan dilakukan pada *output* keluaran dari *regulator* tegangan LM7805. Dari hasil pengukuran pada rangkaian *power supply* diperoleh data seperti pada tabel berikut ini.

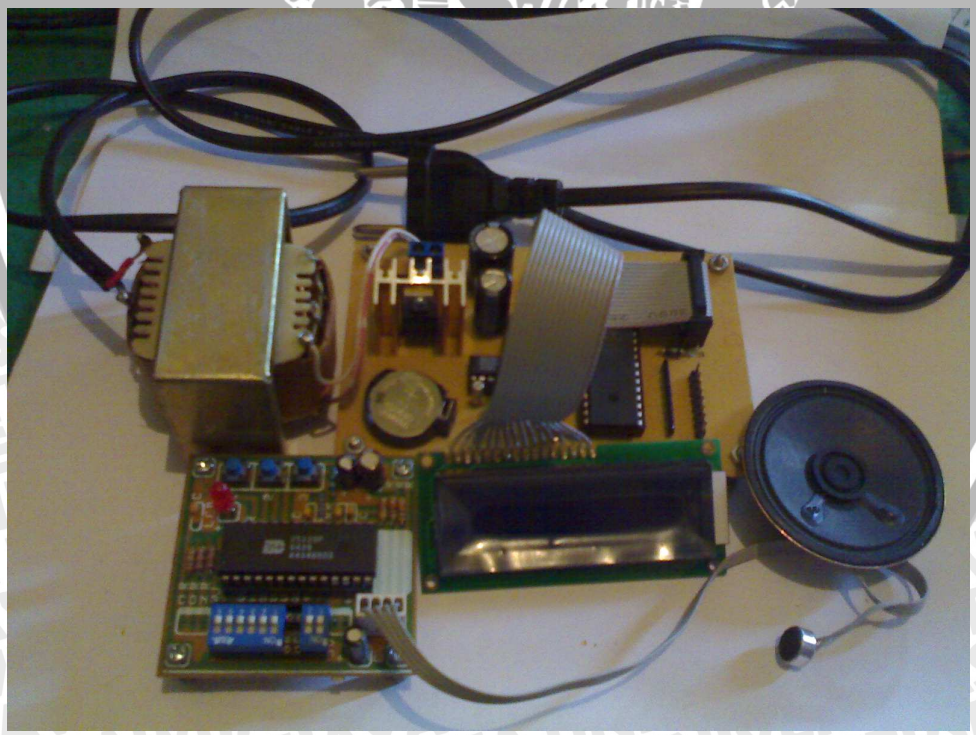
Tabel 5-4. Hasil Pengukuran Rangkaian *Power Supply*

Indek Pengujian	**Out 7805 (V/DC)**
1	4,9
2	5,0
2	5,0
3	4,9
4	4,9
5	4,9

Dari hasil pengukuran yang diperoleh dapat diketahui bahwa *power supply* ini dapat memberikan tegangan kerja untuk rangkaian keseluruhan sesuai dengan tegangan kerja yang dibutuhkan. Hasil tersebut, dikarenakan oleh adanya beberapa faktor yang mempengaruhi, diantaranya kualitas dari tiap-tiap komponen yang digunakan nilainya tidak ideal.

5.7 Pengujian Keseluruhan Sistem

Pada pengujian ini keseluruhan bagian dirangkai menjadi satu. Rangkaian dihubungkan dengan catu daya. Pertama-tama dilakukan seting waktu bel/alarm melalui tombol push button menu, up, dan down, dan kemudian memasukkan input suara agar nantinya dihasilkan output suara sebagai alarm. Pada saat waktu telah sesuai dengan input nilai waktu alarm,maka secara otomatis mikrokontroler akan mengeluarkan suara alarm dan output suara sesuai dengan seting alarm yang telah diinputkan.



Gambar 5.8. Foto Alat

BAB VI
PENUTUP

6.1. Kesimpulan

Pada bab ini akan dibahas mengenai kesimpulan dari hasil laporan Tugas Akhir ini. Kesimpulan yang dibuat tentu saja berdasarkan dari hasil perencanaan dan pembuatan alat seperti yang dibahas pada bab-bab sebelumnya. Kesimpulan yang didapat adalah sebagai berikut:

1. Rangkaian mikrokontroler sebagai pengendali keseluruhan rangkaian pada alat ini memiliki input data waktu digital RTC DS1307 dan 4 buah tombol push-button untuk seting waktu. Sedangkan output mikrokontroler adalah display LCD dan ISD 25120.
2. Rangkaian RTC DS1307 dilengkapi dengan baterai back-up tegangan sehingga data waktu yang tersimpan tidak hilang walapun power supply dimatikan.
3. Alarm bel sekolah diatur oleh mikrokontroler dengan cara membandingkan nilai waktu yang diterima dari RTC dengan nilai waktu alarm yang tersimpan dalam program mikrokontroler.
4. Output alarm berupa suara dan nada ditampilkan oleh mikrokontroler dengan cara memanggil alamat pada ISD.
5. Rangkaian power supply menggunakan sebuah IC regulator L7805 sehingga menghasilkan tegangan DC 5V untuk memberikan tegangan kerja keseluruhan rangkaian bel/alarm sekolah.
6. Dikarenakan Sistem yang telah dibuat dilengkapi dengan output suara menggunakan IC perekam suara ISD 25120, sehingga batas durasi maksimum output suara yang dapat direkam pada sistem ini adalah 120 detik.

6.2. Saran

1. Diharapkan alat ini dapat dikembangkan dengan menambahkan penguat suara.

DAFTAR PUSTAKA

Budiharto, Widodo. 2007. *12 Proyek Mikrokontroler untuk Pemula.* Jakarta: PT. Elex Media Komputindo.

Dwi Sunar Prasetyo. 2003. *Belajar Sistem Cepat Elektronika.*

Kf Ibrahim. 1996. *Teknik Digital.* Andi Off set. Yogyakarta.

Malvino. Terjemahan Hanafi Gunawan. 1992. *Prinsip-prinsip Elektronika.* Edisi Kedua. Erlangga. Jakarta.

Paulus Andi Nalwan. 2003. *Teknik Antar Muka Dan Pemrogaman Mikrokontroler AT89S51*. PT. Elex Media Komputindo. Jakarta.

Wasito S. 2001. *Vademekum Elektronika.* Edisi II. PT. Gramedia Pustaka Utama. Jakarta.

LAMPIRAN

```
        ORG     000H
        JMP     START

;===============
;LCD KONSTANTA
;===============
DISPCLR EQU 00000001B
FUNCSET EQU 00111000B
ENTRMOD EQU 00000110B
DISPON  EQU 00001100B
;==========
;PORT LCD
;==========
LCDE    BIT     P1.0
LCDRS   BIT     P1.1
PLCD    EQU     P0
BUZZ    BIT     P3.6
;==============
;RTC ADDRES
;==============
SCL     BIT     P3.5
SDA     BIT     P3.4
;==============
;==========
;PORT ISD
;==========
ISD_PD  BIT     P1.3
ISD_CE  BIT     P1.2
ISD_EOM BIT     P1.4
ISD_PR  BIT     P3.1
PISD    EQU     P2

;====================
;PUSH BUTTON ADDRES
;====================
TBDOWN  BIT     P1.7
TBUP    BIT     P1.6
TBMENU  BIT     P1.5

;==============
;=======================================
;ALAMAT VARIABLE
;=======================================
PUTR            EQU     030H
RTC_DAT         EQU     031H
RTC_ADR         EQU     032H
SEC             EQU     033H
```

```
MIN         EQU    034H
HOUR        EQU    035H
DATE        EQU    036H
MONTH       EQU    037H
YEAR        EQU    038H
JMLHR       EQU    039H
TEMP        EQU    03AH
MIN_P       EQU    03BH
HOUR_P      EQU    03CH
DATE_P      EQU    03DH
MONTH_P     EQU    03EH
YEAR_P      EQU    03FH
M           EQU    040H
R           EQU    041H
REC_NUM     EQU    042H
C_5Ms       EQU    022H

;=============
;MAIN PROGRAM
;=============
START:
     LCALL   INITLCD
     CALL    DELAY1D
     MOV     REC_NUM,#0
     SETB    ISD_PD
     SETB    ISD_CE
     SETB    ISD_PR
     SETB    ISD_EOM
       SETB  BUZZ
       LCALL       IKLAN
       ;JMP   RING_AGAIN

START_A:
       CALL  DIS_BACK

       CALL  CEK_ALARM
       JB    TBMENU,START_A
       JNB   TBMENU,$
       MOV   A,#DISPCLR
       CALL    LCDINS
       CALL  DIS_BACK
       CALL  SET_WAKTU          ;SET WAKTU
       MOV   A,#DISPCLR
       CALL    LCDINS
       JMP   START_A

TEST_ISD:
       CALL  DIS_BACK
```

```
        JB     TBMENU,TEST_ISD_OUT
        JNB    TBMENU,$
        CALL   INPUT_SUARA
        JMP    TEST_ISD

TEST_ISD_OUT:
        CALL   DIS_BACK
        JB     TBUP,ISI_RECNUM
        JNB    TBUP,$
        CALL   RING_AGAIN
        JMP    TEST_ISD

ISI_RECNUM:
        JB     TBDOWN,TEST_ISD
        JNB    TBDOWN,$
        MOV    A,#DISPCLR
        CALL    LCDINS
        CALL   DELAY1D
        CALL   DSP_ISI_REC
        JB     TBDOWN,$
        JNB    TBDOWN,$
        MOV    REC_NUM,#0
        MOV    A,#DISPCLR
        CALL    LCDINS
        CALL   DELAY1D
        JMP    TEST_ISD

;=======PROSEDUR MENU SUNTING INPUT SUARA=========
INPUT_SUARA:
IS1:
        MOV    A,#DISPCLR
      CALL    LCDINS
      CALL     DSP_RECORD
      MOV     A,REC_NUM
      MOV     PISD,A
      CLR     ISD_PR
      CLR     ISD_PD
      CLR     ISD_CE
IS2:
      JB      TBMENU,IS2
      JNB     TBMENU,$
      SETB    ISD_PR
      SETB    ISD_PD
      SETB    ISD_CE
      MOV      A,REC_NUM
      ADD      A,#28
      MOV      REC_NUM,A
      MOV      A,#DISPCLR
```

```
        CALL    LCDINS
        CALL     DELAY1D
        RET

RING_AGAIN:
        MOV      A,#DISPCLR
        CALL    LCDINS
        CALL     DSP_PLAY
        MOV     A,#28
        MOV     PISD,A
        SETB    ISD_PR
        CLR     ISD_PD
        CLR     ISD_CE
        JB     ISD_EOM,$
        SETB    ISD_EOM
        SETB    ISD_PD
        SETB    ISD_CE
        ;MOV     A,REC_NUM
        ;ADD     A,#28
       ;MOV      REC_NUM,A
        MOV      A,#DISPCLR
        CALL    LCDINS
        CALL     DELAY1D
        ;RET
        JMP      RING_AGAIN
BUZZERZ:
          CLR    BUZZ
          CALL  DELAY1D
          SETB  BUZZ
          CLR    BUZZ
          CALL  DELAY1D
          SETB  BUZZ
          CALL  DELAY1D
          CLR    BUZZ
          CALL  DELAY1D
          SETB  BUZZ
          RET

OUT_SUARA:
        MOV     PISD,A
        SETB    ISD_PR
        CLR     ISD_PD
        CLR     ISD_CE
        JB     ISD_EOM,$
        SETB    ISD_EOM
        SETB    ISD_PD
        SETB    ISD_CE
        MOV      A,#DISPCLR
```

```
        CALL    LCDINS
          RET

;=============================
CEK_ALARM:
;=============================
JAM_KE_1:                       ; 07.00
          MOV   A,HOUR
          CJNE  A,#7,JAM_KE_2
          MOV   A,MIN
          CJNE  A,#0,JAM_KE_2
          MOV   A,SEC
          CJNE  A,#0,JAM_KE_2
          MOV   A,#DISPCLR
        CALL    LCDINS
        CALL    DSP_1
        CALL    BUZZERZ
        MOV     A,#0
        CALL    OUT_SUARA
        RET

JAM_KE_2:                       ; 07.45
          MOV   A,HOUR
          CJNE  A,#7,JAM_KE_3
          MOV   A,MIN
          CJNE  A,#45,JAM_KE_3
          MOV   A,SEC
          CJNE  A,#0,JAM_KE_3
          MOV   A,#DISPCLR
        CALL    LCDINS
        CALL    DSP_2
        CALL    BUZZERZ
        MOV     A,#28
        CALL    OUT_SUARA
        RET
JAM_KE_3:                       ; 08.30
          MOV   A,HOUR
          CJNE  A,#8,JAM_KE_4
          MOV   A,MIN
          CJNE  A,#30,JAM_KE_4
          MOV   A,SEC
          CJNE  A,#0,JAM_KE_4
          MOV   A,#DISPCLR
        CALL    LCDINS
        CALL    DSP_3
        CALL    BUZZERZ
        MOV     A,#56
        CALL    OUT_SUARA
```

```
        RET

JAM_KE_4:                       ; 09.15
        MOV   A,HOUR
        CJNE  A,#9,JAM_KE_5
        MOV   A,MIN
        CJNE  A,#15,JAM_KE_5
        MOV   A,SEC
        CJNE  A,#0,JAM_KE_5
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_4
      CALL    BUZZERZ
      MOV     A,#84
      CALL    OUT_SUARA
      RET

JAM_KE_5:                       ; 10.00
        MOV   A,HOUR
        CJNE  A,#10,JAM_ISTIRAHAT
        MOV   A,MIN
        CJNE  A,#0,JAM_ISTIRAHAT
        MOV   A,SEC
        CJNE  A,#0,JAM_ISTIRAHAT
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_5
      CALL    BUZZERZ
      MOV     A,#112
      CALL    OUT_SUARA
      RET

JAM_ISTIRAHAT:                  ; 10.45
        MOV   A,HOUR
        CJNE  A,#10,JAM_KE_6
        MOV   A,MIN
        CJNE  A,#45,JAM_KE_6
        MOV   A,SEC
        CJNE  A,#0,JAM_KE_6
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_ISTIRAHAT
      CALL    BUZZERZ
      MOV     A,#140
      CALL    OUT_SUARA
      RET

JAM_KE_6:                       ; 11.00
```

```
        MOV   A,HOUR
        CJNE  A,#11,JAM_KE_7
        MOV   A,MIN
        CJNE  A,#0,JAM_KE_7
        MOV   A,SEC
        CJNE  A,#0,JAM_KE_7
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_6
      CALL    BUZZERZ
      MOV     A,#168
      CALL    OUT_SUARA
      RET
JAM_KE_7:                    ; 11.45
        MOV   A,HOUR
        CJNE  A,#11,JAM_PULANG
        MOV   A,MIN
        CJNE  A,#45,JAM_PULANG
        MOV   A,SEC
        CJNE  A,#0,JAM_PULANG
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_7
      CALL    BUZZERZ
      MOV     A,#196
      CALL    OUT_SUARA
      RET

JAM_PULANG:                  ; 12.30
        MOV   A,HOUR
        CJNE  A,#12,OUT_C
        MOV   A,MIN
        CJNE  A,#30,OUT_C
        MOV   A,SEC
        CJNE  A,#0,OUT_C
        MOV   A,#DISPCLR
      CALL    LCDINS
      CALL    DSP_PULANG
      CALL    BUZZERZ
      MOV     A,#224
      CALL    OUT_SUARA
      RET
OUT_C:
        RET
;==========================
 DIS_BACK:
;=========
      MOV    A,#081H
```

```
    CALL   LCDINS
    MOV      A,#'T'
    CALL     LCDDAT
    MOV      A,#'G'
    CALL     LCDDAT
    MOV      A,#'L'
    CALL     LCDDAT
    MOV     A,#0C1H
    CALL   LCDINS
    MOV      A,#'J'
    CALL     LCDDAT
    MOV      A,#'A'
    CALL     LCDDAT
    MOV      A,#'M'
    CALL     LCDDAT
DIS_R:
    MOV      A,#89H
    CALL     LCDINS
    MOV      A,#'-'
    CALL     LCDDAT
    MOV      A,#8DH
    CALL     LCDINS
    MOV      A,#'-'
    CALL     LCDDAT
    MOV      A,#0CAH
    CALL     LCDINS
    MOV      A,#':'
    CALL     LCDDAT
    MOV      A,#0CDH
    CALL     LCDINS
    MOV      A,#':'
    CALL     LCDDAT
    MOV      A,#085H
    CALL     LCDINS
    MOV      A,#':'
    CALL     LCDDAT
    MOV      A,#0C5H
    CALL     LCDINS
    MOV      A,#':'
    CALL     LCDDAT

    MOV     RTC_ADR,#04
    CALL   GET_RTC
    CALL   PINDAH_DATA_RTC
    MOV     A,#087H
    CALL   LCDINS
    MOV     A,DATE
    MOV     B,#010
```

```
DIV    AB
ADD    A,#030H
CALL   LCDDAT
MOV    A,B
ADD    A,#030H
CALL    LCDDAT
MOV    RTC_ADR,#05
CALL    GET_RTC
CALL   PINDAH_DATA_RTC
MOV    A,#08AH
CALL    LCDINS
MOV    A,MONTH
CALL   CARI_BULAN
CALL   LCDSTRING
MOV    RTC_ADR,#06
CALL   GET_RTC
CALL   PINDAH_DATA_RTC
MOV    A,#08EH
CALL   LCDINS
MOV    A,YEAR
MOV    B,#010
DIV    AB
ADD    A,#030H
CALL   LCDDAT
MOV    A,B
ADD    A,#030H
CALL   LCDDAT
MOV    RTC_ADR,#02
  CALL   GET_RTC
CALL   PINDAH_DATA_RTC
MOV    A,#0C8H
CALL   LCDINS
MOV    A,HOUR
MOV    B,#010
DIV    AB
ADD    A,#030H
CALL   LCDDAT
MOV    A,B
ADD    A,#030H
CALL   LCDDAT
MOV    RTC_ADR,#01
CALL    GET_RTC
CALL    PINDAH_DATA_RTC
MOV    A,#0CBH
CALL   LCDINS
MOV    A,MIN
MOV    B,#010
DIV    AB
```

```
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,B
        ADD     A,#030H
        CALL    LCDDAT
        MOV     RTC_ADR,#0
        CALL    GET_RTC
        CALL    PINDAH_DATA_RTC
        MOV     A,#0CEH
        CALL    LCDINS
        MOV     A,SEC
        MOV     B,#010
        DIV     AB
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,B
        ADD     A,#030H
        CALL    LCDDAT
        RET
        ;===============


;============================
;============================
GET_RTC:
;========
        CLR     EA
        ACALL   SIGNAL_START
        JC      ERROR_GET
          MOV   A,#11010000B        ;==> ADRESS RTC DAN PROSES WRITE
          ACALL         OUT8BIT
        JC      ERROR_GET
        MOV     A,RTC_ADR
          ACALL         OUT8BIT
        JC      ERROR_GET
        ACALL   SIGNAL_START
          MOV   A,#11010001B        ;==> ADRESS RTC DAN PROSES READ
          ACALL         OUT8BIT
        JC      ERROR_GET
          ACALL         IN8BIT
        MOV     RTC_DAT,A
        JC      ERROR_GET
        ACALL   SIGNAL_STOP
        SETB    EA
        RET

ERROR_GET:
        MOV     A,#080H
```

```
        CALL    LCDINS
        MOV     DPTR,#TXT_RTC_ERROR_GET
        CALL    LCDSTRING
        RET

;=============================================================
=====
SET_RTC:
;============================
        CLR     EA
        LCALL   SIGNAL_START
        JC      ERROR_SET
          MOV   A,#11010000B        ;==> ADRESS RTC DAN PROSES WRITE
          CALL  OUT8BIT
        JC      ERROR_SET
        MOV     A,RTC_ADR
          ACALL         OUT8BIT
        JC      ERROR_SET
        MOV     A,RTC_DAT
          ACALL         OUT8BIT
        JC      ERROR_SET
        ACALL   SIGNAL_STOP
        SETB    EA
          RET

ERROR_SET:
        MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#TXT_RTC_ERROR_SET
        CALL    LCDSTRING
        RET

;=============================================================
OUT8BIT:
;============================
          PUSH  B
          MOV   B,#8        ; akan digeser 8 kali (bit)
OUTLOOP:
          RLC   A           ; bit A.7 digeser ke C di PSW
        MOV     SDA,C       ; nilai C di SDA
          NOP               ; tunggu sebentar sebelum.. ..
        SETB    SCL         ; SCL dibuat = "1"
          NOP               ; tunggu lagi
          NOP
          NOP
          NOP
        CLR     SCL         ; SCL dibuat ="0", BYTE diambil "slave"
        DJNZ    B,OUTLOOP   ; ulangi terus sampai 8 kali
```

```
      SETB    SDA        ; SDA dibuat "1"
        NOP              ; agar 'slve' bisa mengirim ACK
        NOP
      SETB    SCL        ; clock ke 9 untuk menerima ACK
        NOP              ; tunggu dulu
        NOP
        NOP
        NOP
      MOV     C,SDA      ; ambil ACK yang dikirim "slave"
      CLR     SCL        ; SCK kembali ke "0"
        POP     B
        RET

;=====================================
IN8BIT:
;============================
        PUSH  B
      SETB    SDA        ; SDA="1" agar "slave" bisa kirim BYTE
        MOV   B,#8       ; akan digeser 8 kali (bit)
INLOOP:
      NOP              ; "slave" boleh mengubah BYTE selama SCK="0"
        NOP
        NOP
      SETB    SCL        ; SCK="1"
        NOP              ; tunggu sebentar
        NOP
      MOV     C,SDA      ; ambil kiriman bit dari "slave"
        RLC     A        ; ditampung di A
      CLR     SCL        ; "slave" boleh merubah BYTE selama SCK="0"
      DJNZ    B,INLOOP
        POP     B
        RET

;=======P=============================================================
======
SIGNAL_START:
;============================
      SETB  SDA
      SETB  SCL          ; Memastikan I2C Bus bisa dipakai

      JNB   SDA,BUSBUSY   ; kalau SDA=0 I2C Bus tidak siap pakai
      JNB   SCL,BUSBUSY   ; kalau SCL=0 I2C Bus tidak siap pakai

        ; Membuat sinyal START
        NOP              ; tunggu sebentar
      CLR   SDA
        NOP              ; tunggu agar BYTE benar-benar stabil
        NOP
```

```
        NOP
        NOP
        NOP
      CLR   SCL
        CLR   C          ; C=0 berarti berhasil membuat START
        RET
BUSBUSY:
        SETB  C          ; C=1 berarti gagal membuat START
        RET

;================================================
SIGNAL_STOP:
;============================
      CLR   SDA          ; SDA=0 low beberapa saat
        NOP
        NOP
      SETB  SCL          ; SCL=1
        NOP              ; tunggu sebentar
        NOP
        NOP
        NOP
        NOP
      SETB  SDA          ; SDA=1
        RET

;==============================
CARI_BULAN:
;==========================
      CJNE    A,#01,CB2
      MOV     DPTR,#BLN_1
      SJMP    CBX
CB2:
      CJNE    A,#02,CB3
      MOV     DPTR,#BLN_2
      SJMP    CBX
CB3:
      CJNE    A,#03,CB4
      MOV     DPTR,#BLN_3
      SJMP    CBX
CB4:
      CJNE    A,#04,CB5
      MOV     DPTR,#BLN_4
      SJMP    CBX
CB5:
      CJNE    A,#05,CB6
      MOV     DPTR,#BLN_5
      SJMP    CBX
CB6:
```

```
        CJNE    A,#06,CB7
        MOV     DPTR,#BLN_6
        SJMP    CBX
CB7:
        CJNE    A,#07,CB8
        MOV     DPTR,#BLN_7
        SJMP    CBX
CB8:
        CJNE    A,#08,CB9
        MOV     DPTR,#BLN_8
        SJMP    CBX
CB9:
        CJNE    A,#09,CB10
        MOV     DPTR,#BLN_9
        SJMP    CBX
CB10:
        CJNE    A,#010,CB11
        MOV     DPTR,#BLN_10
        SJMP    CBX
CB11:
        CJNE    A,#011,CB12
        MOV     DPTR,#BLN_11
        SJMP    CBX
CB12:
        MOV     DPTR,#BLN_12
CBX:
        RET
;=====================================
 SET_WAKTU:
;=========================
          MOV     A,#00001101B
        CALL    LCDINS
        JMP     RSJ4
                  ;<-- SET TANGGAL
RSJ0:
        MOV     A,MONTH
        CALL    C_JML_HR
        MOV     A,#087H
        CALL    LCDINS
        MOV     A,DATE
        MOV     B,#010
        DIV     AB
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,B
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,#088H
```

```
        CALL    LCDINS
RSJ1:
        JB      TBUP,RSJ2
        JNB     TBUP,$
        MOV     A,DATE
        CJNE    A,JMLHR,RSJ1_1
        MOV     DATE,#01
        JMP     RSJ0
RSJ1_1:
        INC     DATE
        JMP     RSJ0
RSJ2:
        JB      TBDOWN,RSJ3
        JNB     TBDOWN,$
        MOV     A,DATE
        CJNE    A,#01,RSJ2_1
        MOV     DATE,JMLHR
        JMP     RSJ0
RSJ2_1:
        DEC     DATE
        JMP     RSJ0
RSJ3:
        JB      TBMENU,RSJ1
        JNB     TBMENU,$
        MOV     A,DATE
        MOV     B,#10
        DIV     AB
        SWAP    A
        ADD     A,B
        MOV     DATE,A
        MOV     RTC_ADR,#04H
        MOV     RTC_DAT,DATE
        CALL    SET_RTC
        JMP     RSJ8

                ;<-- SET BULAN
RSJ4:
        MOV     A,#08AH
        CALL    LCDINS
        MOV     A,MONTH
        CALL    CARI_BULAN
        CALL    LCDSTRING
        MOV     A,#08CH
        CALL    LCDINS
        CALL    DELAY
        CALL    DELAY
RSJ5:
        JB      TBUP,RSJ6
```

```
        JNB     TBUP,$
        CALL      DELAY
        CALL      DELAY
        CALL      DELAY
        MOV     A,MONTH
        CJNE    A,#012,RSJ5_1
        MOV     MONTH,#01
        JMP     RSJ4
RSJ5_1:
        INC     MONTH
        JMP     RSJ4
RSJ6:
        JB      TBDOWN,RSJ7
        JNB     TBDOWN,$
        MOV     A,MONTH
        CJNE    A,#01,RSJ6_1
        MOV     MONTH,#012
        JMP     RSJ4
RSJ6_1:
        DEC     MONTH
        JMP     RSJ4
RSJ7:
        JB      TBMENU,RSJ5
        JNB     TBMENU,$
        MOV       A,MONTH
        MOV       B,#10
        DIV       AB
        SWAP      A
        ADD       A,B
        MOV       MONTH,A
        MOV     RTC_ADR,#05H
        MOV     RTC_DAT,MONTH
        CALL    SET_RTC
        JMP     RSJ0

                ;<-- SET TAHUN
RSJ8:
        MOV     A,#08EH
        CALL    LCDINS
        MOV     A,YEAR
        MOV     B,#010
        DIV     AB
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,B
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,#08FH
```

```
        CALL    LCDINS
RSJ9:
        JB      TBUP,RSJ10
        JNB     TBUP,$
        MOV     A,YEAR
        CJNE    A,#099,RSJ9_1
        MOV     YEAR,#0
        JMP     RSJ8
RSJ9_1:
        INC     YEAR
        JMP     RSJ8
RSJ10:
        JB      TBDOWN,RSJ11
        JNB     TBDOWN,$
        MOV     A,YEAR
        JNZ     RSJ10_1
        MOV     YEAR,#099
        JMP     RSJ8
RSJ10_1:
        DEC     YEAR
        JMP     RSJ8
RSJ11:
        JB      TBMENU,RSJ9
        JNB     TBMENU,$
        MOV     A,YEAR
        MOV     B,#10
        DIV     AB
        SWAP    A
        ADD     A,B
        MOV     YEAR,A
        MOV     RTC_ADR,#06H
        MOV     RTC_DAT,YEAR
        CALL    SET_RTC

                ;<-- SET JAM
RSJ12:
        MOV     A,#0C8H
        CALL    LCDINS
        MOV     A,HOUR
        MOV     B,#010
        DIV     AB
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,B
        ADD     A,#030H
        CALL    LCDDAT
        MOV     A,#0C9H
        CALL    LCDINS
```

```
RSJ13:
      JB     TBUP,RSJ14
      JNB    TBUP,$
      call     delay
      MOV    A,HOUR
      CJNE   A,#023,RSJ13_1
      MOV    HOUR,#0
      JMP    RSJ12
RSJ13_1:
      INC    HOUR
      JMP    RSJ12
RSJ14:
      JB     TBDOWN,RSJ15
      JNB    TBDOWN,$
      call     delay
      MOV    A,HOUR
      JNZ    RSJ14_1
      MOV    HOUR,#023
      JMP    RSJ12
RSJ14_1:
      DEC    HOUR
      JMP    RSJ12
RSJ15:
      JB     TBMENU,RSJ13
      JNB    TBMENU,$
      call     delay
      MOV    A,HOUR
      MOV    B,#10
      DIV    AB
      SWAP   A
      ADD    A,B
      MOV    HOUR,A
      MOV    RTC_ADR,#02H
      MOV    RTC_DAT,HOUR
      CALL   SET_RTC

             ;<-- SET MENIT
RSJ16:
      MOV    A,#0CBH
      CALL   LCDINS
      MOV    A,MIN
      MOV    B,#010
      DIV    AB
      ADD    A,#030H
      CALL   LCDDAT
      MOV    A,B
      ADD    A,#030H
      CALL   LCDDAT
```

```
        MOV     A,#0CCH
        CALL    LCDINS
RSJ17:
        JB      TBUP,RSJ18
        JNB     TBUP,$
        MOV     A,MIN
        CJNE    A,#059,RSJ17_1
        MOV     MIN,#0
        JMP     RSJ16
RSJ17_1:
        INC     MIN
        JMP     RSJ16
RSJ18:
        JB      TBDOWN,RSJ19
        JNB     TBDOWN,$
        MOV     A,MIN
        JNZ     RSJ18_1
        MOV     MIN,#059
        JMP     RSJ16
RSJ18_1:
        DEC     MIN
        JMP     RSJ16
RSJ19:
        JB      TBMENU,RSJ17
        JNB     TBMENU,$
        MOV     A,MIN
        MOV     B,#10
        DIV     AB
        SWAP    A
        ADD     A,B
        MOV     MIN,A
        MOV     RTC_ADR,#01H
        MOV     RTC_DAT,MIN
        CALL    SET_RTC
        CALL    DELAY
        MOV     A,#DISPON
        CALL    LCDINS
        CALL    DELAY
        RET

;==============================
PINDAH_DATA_RTC:
;==========================
        MOV     A,RTC_ADR
        JNZ     PDR1
        MOV     A,RTC_DAT
        ANL     A,#0FH
```

```
        MOV     TEMP,A
        MOV     A,RTC_DAT
        SWAP    A
        ANL     A,#0FH
        MOV     B,#010
        MUL     AB
        ADD     A,TEMP
        MOV     SEC,A
        SJMP    PDRX
PDR1:
        CJNE    A,#01,PDR2
        MOV     A,RTC_DAT
        ANL     A,#0FH
        MOV     TEMP,A
        MOV     A,RTC_DAT
        SWAP    A
        ANL     A,#0FH
        MOV     B,#010
        MUL     AB
        ADD     A,TEMP
        MOV     MIN,A
        SJMP    PDRX
PDR2:
        CJNE    A,#02,PDR3
        MOV     A,RTC_DAT
        ANL     A,#0FH
        MOV     TEMP,A
        MOV     A,RTC_DAT
        SWAP    A
        ANL     A,#0FH
        MOV     B,#010
        MUL     AB
        ADD     A,TEMP
        MOV     HOUR,A
        SJMP    PDRX
PDR3:
        CJNE    A,#04,PDR4
        MOV     A,RTC_DAT
        ANL     A,#0FH
        MOV     TEMP,A
        MOV     A,RTC_DAT
        SWAP    A
        ANL     A,#0FH
        MOV     B,#010
        MUL     AB
        ADD     A,TEMP
        MOV     DATE,A
        SJMP    PDRX
```

```
PDR4:
      CJNE    A,#05,PDR5
      MOV     A,RTC_DAT
      ANL     A,#0FH
      MOV     TEMP,A
      MOV     A,RTC_DAT
      SWAP    A
      ANL     A,#0FH
      MOV     B,#010
      MUL     AB
      ADD     A,TEMP
      MOV     MONTH,A
      SJMP    PDRX
PDR5:
      CJNE    A,#06,PDRX
      MOV     A,RTC_DAT
      ANL     A,#0FH
      MOV     TEMP,A
      MOV     A,RTC_DAT
      SWAP    A
      ANL     A,#0FH
      MOV     B,#010
      MUL     AB
      ADD     A,TEMP
      MOV     YEAR,A
PDRX:
      MOV     TEMP,#0
      RET

;=============================
C_JML_HR:
;=========================
      CJNE    A,#01,CJH2
      MOV     JMLHR,#031
      SJMP    CJHX
CJH2:
      CJNE    A,#02,CJH3
      MOV     JMLHR,#028
      SJMP    CJHX
CJH3:
      CJNE    A,#03,CJH4
      MOV     JMLHR,#031
      SJMP    CJHX
CJH4:
      CJNE    A,#04,CJH5
      MOV     JMLHR,#030
      SJMP    CJHX
CJH5:
```

```
        CJNE    A,#05,CJH6
        MOV     JMLHR,#031
        SJMP    CJHX
CJH6:
        CJNE    A,#06,CJH7
        MOV     JMLHR,#030
        SJMP    CJHX
CJH7:
        CJNE    A,#07,CJH8
        MOV     JMLHR,#031
        SJMP    CJHX
CJH8:
        CJNE    A,#08,CJH9
        MOV     JMLHR,#031
        SJMP    CJHX
CJH9:
        CJNE    A,#09,CJH10
        MOV     JMLHR,#030
        SJMP    CJHX
CJH10:
        CJNE    A,#010,CJH11
        MOV     JMLHR,#031
        SJMP    CJHX
CJH11:
        CJNE    A,#011,CJH12
        MOV     JMLHR,#030
        SJMP    CJHX
CJH12:
        CJNE    A,#012,CJHX
        MOV     JMLHR,#031
CJHX:
        RET
;=========================
BLN_1   : DB 'JAN',0
BLN_2   : DB 'FEB',0
BLN_3   : DB 'MAR',0
BLN_4   : DB 'APR',0
BLN_5   : DB 'MEI',0
BLN_6   : DB 'JUN',0
BLN_7   : DB 'JUL',0
BLN_8   : DB 'AGT',0
BLN_9   : DB 'SEP',0
BLN_10  : DB 'OKT',0
BLN_11  : DB 'NOV',0
BLN_12  : DB 'DES',0

TXT_RTC_ERROR_GET     :     DB     ' ERROR GET RTC  ',0
TXT_RTC_ERROR_SET     :     DB     ' ERROR SET RTC  ',0
```

```
;==========================
PRINTSTRINGLOOP:
;====================
      CALL   LCDDAT
      INC    DPTR
LCDSTRING:
      CLR    A
      MOVC   A,@A+DPTR
      JNZ    PRINTSTRINGLOOP
      RET

KURSOR:
        MOV    A,#00001101B
      CALL   LCDINS
      RET

DISPLON:
        MOV    A,#DISPON
      CALL   LCDINS
      MOV    A,#DISPCLR
      CALL   LCDINS
        RET

TAMPIL:
        ADD    A,#030H
      CALL   LCDDAT
      RET

;================================
LCDINS:
;======================
      MOV    PLCD,A
      CLR    LCDRS
      SJMP   LCDOUT
LCDDAT:
      MOV    PLCD,A
      SETB   LCDRS
LCDOUT:
      SETB   LCDE
      CALL    DELAY
      CLR    LCDE
        CALL  DELAY
      RET
DSP_ON:
        MOV    A,#DISPON
      CALL   LCDINS
      CALL    DELAY
```

```
    RET

;================================
DELAY4:
    MOV   R7,#0250
DELAY4_1:
    DJNZ  R7,DELAY4_1
    RET
DELAY3:
      MOV  PUTR,A
MUTERZ:
    CALL  DELAY2
    DJNZ  PUTR,MUTERZ
    RET
DELAY2:
    MOV   R5,#130
MUTERX:
    MOV   R6,#250
    CALL  DELAY
    DJNZ  R6,$
    DJNZ  R5,MUTERX
    RET
DELAY:
    MOV   R3,#08
MUTER:
    MOV   R4,#0255
      DJNZ  R4,$
      DJNZ  R3,MUTER
      RET

;============================

;============================
INITLCD:
;============================
    MOV   A,#DISPCLR
    CALL  LCDINS
    CALL  DELAY
    MOV   A,#FUNCSET
    CALL  LCDINS
    CALL  DELAY
    MOV   A,#DISPON
    CALL  LCDINS
    CALL  DELAY
    MOV   A,#ENTRMOD
    CALL  LCDINS
    CALL  DELAY
    MOV   A,#DISPCLR
```

```
        CALL    LCDINS
        CALL    DELAY2
        RET
;=================================
DSP_PLAY:
          MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#IDLE_1
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#IDLE_2
        CALL    LCDSTRING
          RET
DSP_RECORD:
          MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#IDLE_3
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#IDLE_4
        CALL    LCDSTRING
          RET

DSP_ISI_REC:
          MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#IDLE_5
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#IDLE_6
        CALL    LCDSTRING
          RET


;=============
;TEXT SECTION
;=============
IDLE_1  : DB 'PLAY SUARA      ',0
IDLE_2  : DB '  TESTING       ',0

IDLE_3  : DB '  Record Suara  ',0
IDLE_4  : DB '  Tekan PB Menu ',0

IDLE_5  : DB ' ISI RECNUM 0   ',0
IDLE_6  : DB ' TESTING        ',0
```

```
IDLE_9  : DB '  STOP RECORD   ',0
IDLE_A  : DB ' Tekan PB Menu  ',0

ABOUT_1 : DB '       BEL/ALARM',0
ABOUT_2 : DB '         SEKOLAH',0
ABOUT_3 : DB '        BERBASIS',0
ABOUT_4 : DB '     MCU AT89S51',0
ABOUT_7 : DB '       M RIZA YR',0
ABOUT_8 : DB '  NIM:0210633054',0
ABOUT_9 : DB '     PEMBIMBING1',0
ABOUT_A : DB ' PANCA M, ST.,MT',0
ABOUT_B : DB '     PEMBIMBING2',0
ABOUT_C : DB 'Ir. NURUSSA'ADAH',0

DS1    : DB '   JAM PELAJARAN',0
DS2    : DB '        PERTAMA',0
DS3    : DB '          KEDUA',0
DS4    : DB '         KETIGA',0
DS5    : DB '        KEEMPAT',0
DS6    : DB '         KELIMA',0
DS7    : DB '           JAM',0
DS8    : DB '      ISTIRAHAT',0
DS9    : DB '         KEENAM',0
DSA    : DB '        KETUJUH',0
DSB    : DB '         PULANG',0

DSP_1:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS2
      CALL    LCDSTRING
        RET
DSP_2:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS3
      CALL    LCDSTRING
        RET
```

```
DSP_3:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS4
      CALL    LCDSTRING
        RET
DSP_4:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS5
      CALL    LCDSTRING
        RET
DSP_5:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS6
      CALL    LCDSTRING
        RET
DSP_ISTIRAHAT:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS7
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
      MOV     DPTR,#DS8
      CALL    LCDSTRING
        RET
DSP_6:
        MOV     A,#080H
      CALL    LCDINS
      MOV     DPTR,#DS1
      CALL    LCDSTRING
      MOV     A,#0C0H
      CALL    LCDINS
```

```
        MOV     DPTR,#DS9
        CALL    LCDSTRING
          RET
DSP_7:
          MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#DS1
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#DSA
        CALL    LCDSTRING
          RET
DSP_PULANG:
          MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#DS7
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#DSB
        CALL    LCDSTRING
          RET

;============================
; DAFTAR IKLAN
;============================
IKLAN:
        MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#ABOUT_1
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#ABOUT_2
        CALL    LCDSTRING
        MOV     A,#03
        CALL    DELAY3
        MOV     A,#080H
        CALL    LCDINS
        MOV     DPTR,#ABOUT_3
        CALL    LCDSTRING
        MOV     A,#0C0H
        CALL    LCDINS
        MOV     DPTR,#ABOUT_4
        CALL    LCDSTRING
        MOV     A,#03
        CALL    DELAY3
```

```
        MOV    A,#080H
        CALL   LCDINS
        MOV    DPTR,#ABOUT_7
        CALL   LCDSTRING
        MOV    A,#0C0H
        CALL   LCDINS
        MOV    DPTR,#ABOUT_8
        CALL   LCDSTRING
        MOV    A,#03
        CALL   DELAY3
        MOV    A,#080H
        CALL   LCDINS
        MOV    DPTR,#ABOUT_9
        CALL   LCDSTRING
        MOV    A,#0C0H
        CALL   LCDINS
        MOV    DPTR,#ABOUT_A
        CALL   LCDSTRING
        MOV    A,#03
        CALL   DELAY3
        MOV     A,#DISPCLR
        CALL   LCDINS
        RET

DELAY1D:
        MOV  C_5mS,#0200
WAIT_1D:
        CALL DELAY_5MS
        DJNZ C_5mS,WAIT_1D
        RET

DELAY_5MS:
        PUSH TMOD
        MOV  TMOD,#21H
        MOV  TH0,#0EDH
        MOV  TL0,#0FFH
        SETB TR0
WAIT_5MS:
        JBC   TF0,DAH_5MS
        JMP   WAIT_5MS
DAH_5MS:
        CLR   TR0
        POP   TMOD
        RET
END
```