BAB II
LANDASAN TEORI

2.1 RF Smart Card

RFID (*Radio Frequency Identification)* adalah suatu teknologi yang secara konseptual mirip dengan *Barcode* yang banyak dijumpai di banyak barang dagangan, atau *magnetic strips* yang banyak digunakan dilingkungan kartu kredit. Bedanya, *RFID* terdiri atas sebuah *chip eletronik* yang bisa dijejali dengan aneka macam data/informasi sesuai kebutuhan. Data itu misalnya saja berupa tipe barang, tanggal pembuatan, tujuan pengiriman barang, dan data-data lainnya. Perbedaan lainnya bilamana *barcode* memerlukan proses pemindaian sebelum datanya terbaca, *RFID* menggunakan teknologi radio pada frekuensi tertentu, dan frekuensi ini kemudian dibaca sebuah alat yang dinamakan *RFID Reader*. *RFID* terdiri dari 2 bagian yaitu *Transponder* dan *Reader*. *Transponder* bisa berupa *Card* (kartu), *Tag* (kancing), atau *Wristband* (jam tangan). Pada kartu ini hanya mempunyai tingkat frekuensi sebesar 125 KHz. Sedangkan *Reader* berfungsi untuk membaca nomor data ID yang dikirimkan oleh *Transponder*. Efisiensi tinggi yang diberikan oleh sistem *RFID* mendatangkan masalah keamanan dan privasi. Sistem *RF Smart Card* rentan terhadap serangan secara fisik, mudah dipalsukan, disadap, saluran komunikasinya mudah dibanjiri dengan *noise* sehingga dapat mematikan peralatan komunikasi. Serangan ini disebut serangan *Denial of Service* dengan tujuan menutup saluran komunikasi. Berdasarkan lubang keamanan, keamanan dapat diklasifikasikan menjadi empat, yaitu : keamanan bersifat fisik, keamanan yang berhubungan dengan orang, keamanan dari data dan media, teknik komunikasi serta keamanan dalam operasi.

1. Keamanan yang bersifat fisik

Keamanan yang bersifat fisik dilakukan penyerangan terhadap *RFID* yaitu antara lain berupa pemindahan material kartu maupun menukar kartu sehingga data yang ada di dalam kartu tidak sesuai dengan keadaan yang sebenarnya. Serangan *Denial of Service* juga dikategorikan dalam kelas ini yaitu dengan mengacaukan siaran *(broadscast)*, menutup pesan yang akan masuk pada *reader* dengan cara membanjiri saluran komunikasi dengan *noise* yang banyak.

2. Keamanan yang berhubungan dengan personal

Keamanan yang berhubungan dengan manusia pada sistem *RFID* adalah dengan adanya keteledoran perilaku manusia yang mempunyai akses ke *RFID*, misalkan memberikan *password* yang tidak berhak atau membuang kartu *RFID* yang masih berisi data tentang *database*, sehingga penyadap yang menemukan dapat membaca dan melakukan penyerangan yang sama pada kartu yang masih aktif sehingga dapat mengganggu privasi dari orang yang berhak.

3. Keamanan dari data media serta teknik komunikasi

Keamanan *RFID* yang termasuk kelas ini adalah kelemahan dalam *software* yang digunakan untuk mengelola data pada kartu sehingga seseorang yang tidak mempunyai hak akses dapat membuka isi dari kartu karena *password* yang ada di kartu dapat dibuka.

4. Keamanan dalam operasi

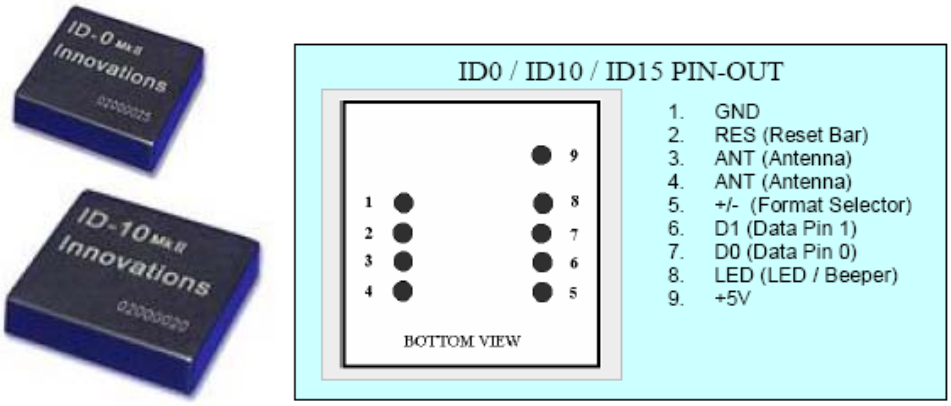
Keamanan *RFID* yang termasuk dalam kelas ini adalah prosedur yang digunakan untuk mengatur dan mengelola tentang sistem keamanan dan juga termasuk prosedur setelah terjadi serangan.

ID Smart card sering disebut juga sebagai *chip card* atau *integrated circuit* (IC) *card* . Definisi *chip card* sendiri yaitu kategori umum yang mencakup *smart card* dan *memory card*. *ID Smart card* adalah *plastic card* yang mengandung *memory chip* dan *microprocessor*. Kartu ini bisa menambah, menghapus, mengubah informasi yang terkandung. Keunggulannya adalah *ID smart card* tidak perlu mengakses *database* di *server* karena sudah ada sebagian terkandung di kartu. Sedangkan *memory card* dipasangi *memory silicon* tanpa *microprocessor*. Fungsi dasar suatu *ID smart card* adalah untuk mengidentifikasi *card holder* ke sistem komputer. *Card holder* disini adalah pemilik kartu tersebut, identifikasi ini menyangkut otentifikasi organisasi yang membuat kartu tersebut dan *card holder* dan hak aksesnya. Beberapa hal yang harus dipertimbangkan dalam metode identifikasi kartu adalah :

1. Mengkonfirmasi identitas *card holder* (pemegaang kartu) sebelum mengakses data.
2. Memberikan data untuk konfirmasi ke alat eksternal, sistem atau perorangan.
3. Menyediakan data ke sistem tanpa pengecekan orang yang menggunakan.

2.1.1. RFID 10

RFID 10 merupakan alat sebuah sistem pengamanan yang sama halnya dengan *Barcode*. RFID 10 yang umum digunakan, mempunyai *internal antenna* yang bisa membaca kartu yang ditanami dengan *chip* sejauh kurang lebih 12 cm hingga 16 cm tergantung frekuensi yang digunakan, namun bila menggunakan *external antenna* jarak maksimum yang dihasilkan bisa mencapai 25 cm. Cara kerja *RFID* 10 ini adalah ketika kartu didekatkan pada *RFID Reader* maka kumparan/lilitan yang ada pada kartu akan membangkitkan suatu tegangan yang berubah menjadi suatu data serial. Bentuk *RFID* 10 terlihat pada Gambar 2.1 dan fungsi pin pada *RFID* 10 terlihat pada Tabel 2.1.



Gambar 2.1 RFID 10

Tabel 2.1 Fungsi Pin Pada RFID 10

Pin No.	Description
Pin 1	Zero Volts and Tuning Capacitor Ground
Pin 2	Strap to +5V
Pin 3	To External Antenna and Tuning Capacitor
Pin 4	To External Antenna
Pin 5	Format Selector (+/-)
Pin 6	Data 1
Pin 7	Data 0
Pin 8	3.1 kHz Logic
Pin 9	DC Voltage Supply

2.2 Regulator IC LM7805

IC dengan tiga kaki yang digunakan sebagai komponen pendukung dari vcc untuk menghasilkan tegangan 5 volt. Tegangan pada suatu transformator tidak selalu tetap atau berubah-ubah, agar transformator stabil maka diperlukan tambahan regulator IC LM7805 yang digunakan untuk menstabilkan tegangan 5 volt tersebut. Komponen dari regulator IC LM7805 bisa dilihat pada Gambar 2.2.

Karakteristik IC regulator :

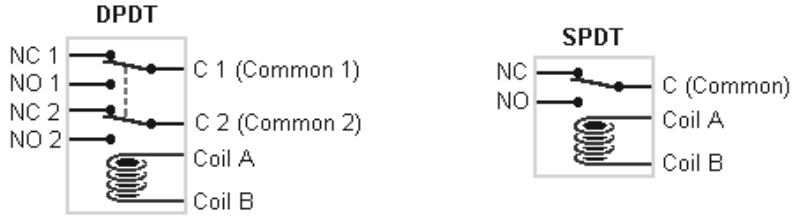
1. Ada tipe positif untuk meregulasi tegangan positif.
2. Ada tipe negatif untuk meregulasi tegangan negatif.
3. Penerapan IC mengharuskan Vi > Vo
4. Dalam praktek, minimum Vi ~Vo + 3 Volt
5. Agar dapat menerapkan Vi > Vo perlu menggunakan komponen-komponen lepasan.
6. Transistor dalam konfigurasi tunggal-emitor, dimana V_i > Vo hanya kalau transistornya jenuh.



Gambar 2.2 Regulator IC LM7805

2.3 Relay DPDT *(double pole/ double throw)*

Switch yang paling umum digunakan dan paling banyak variasi konfigurasi kontaknya adalah *toggle switch*. Setiap *switch* terdiri dari satu atau lebih kutub yang masing-masing kutub adalah *switch*. Salah satu konfigurasi kontak yang paling sederhana adalah SPST *(single pole/single throw)* yang hanya memiliki satu kontak untuk terbuka atau tertutup. Jenis lain adalah SPDT *(single pole/double throw)* memiliki dua kontak A dan B. Lengan penggerak disebut *common* atau *wiper*, DPDT *(double pole/double throw)* yang terdiri dari dua SPDT di dalam suatu rumah (lihat Gambar 2.3.). Seluruh jenis ini memiliki dua posisi yaitu on atau off. Arus listrik yang mengalir melalui suatu penghantar akan menghasilkan medan magnet, jika penghantar tersebut diselubungi kumparan, medan magnet yang dihasilkan akan diarahkan sepanjang kumparan. Jika arus diperbesar maka medan magnet yang dihasilkan akan lebih besar. Jika sebuah benda magnetik diselubungi pada kumparan untuk tujuan menggerakan benda maka kumparan tersebut diberi energi dengan cara mengalirkan arus listrik (istilahnya: *energized*) maka kumparan ini disebut *solenoid*. Benda magnetik yang dapat bergerak ini disebut *armatur*, dan kebanyakan *armatur* dapat digerakan baik oleh DC maupun AC. Sebuah kumparan/*armatur* relay mungkin dirancang untuk menghubungkan lebih dari satu kontak. Kontak-kontak tersebut dapat berupa *normally-open* (NO), *normally-closed* (NC) atau kombinasi keduanya. Yang dimaksud keadaan normal adalah ketika kumparan relay tidak diberi energi *(de-energized)*. Kontaktor relay dapat berupa logam, tabung-merkuri atau tabung magnetik.

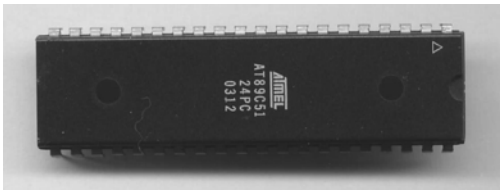


(a) (b)

Gambar 2.3 (a) Relay DPDT; (b) Relay SPDT

2.4 Mikrokontroler AT89C51

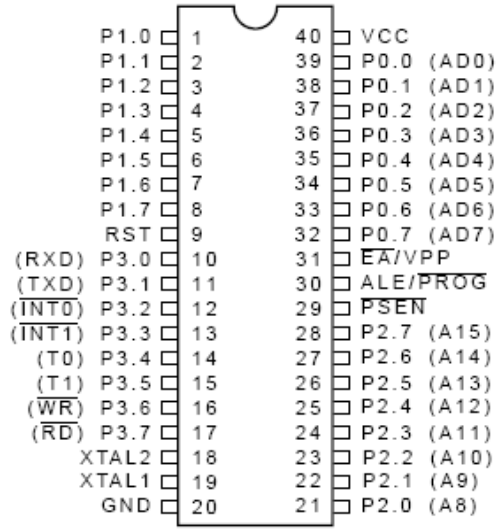
AT89C51 merupakan mikrokontroler 8 bit dengan 4 kbyte Flash PEROM *(programmable and eraseble read only)* yang konfigurasi dan instruksinya kompatibel dengan standar 80C51 dan 89C52. AT89C51 mampu ditulis dan dihapus sebanyak 1000 kali. AT89C51 memiliki 128 x 8 bit RAM internal, 32 jalur I/O *programmable*, dua buah *Timer/Counter* 16 bit, tujuh sumber interupsi, kanal *programmable* serial. Selain itu AT89C51 memiliki mode *low-power Idle* dan *power-down* dan tiga tingkat pengunci program memori. AT89C51 dapat beroperasi statis dari 0-24 MHz. Gambar AT89C51 dapat dilihat pada Gambar 2.4 :



Gambar 2.4 Mikrokontroler AT89C51

2.4.1 Konfigurasi dan Fungsi Kaki Pin Mikrokontroler AT89C51

Mikrokontroler AT89C51 mempunyai 40 kaki, 32 kaki diantaranya adalah kaki untuk keperluan *port parallel*. Satu *port parallel* terdiri dari 8 kaki, dengan demikian 32 kaki tersebut membentuk 4 buah *port parallel*. Konfigurasi pin mikrokontroler AT89C51 bisa terlihat pada Gambar 2.5 :



Gambar 2.5 Konfigurasi AT89C51

1. Port 0

   Port 0 adalah 8 bit *bidirectional* port I/O. Port 0 akan memultipleks data dan alamat ketika mengakses program eksternal dan memori data eksternal, dalam mode ini port 0 mempunyai *pull-ups* internal.

2. Port 1

   Port 1 adalah 8 bit *bidirectiona*l port I/O dengan *pull-ups* internal

3. Port 2

   Port 2 adalah 8 bit *bidirectional* port I/O. Dengan *pull-ups* internal. Port 2 dapat digunakan untuk mengeluarkan alamat 8 bit teratas ketika mengakses memori eksternal.

4. Port 3

   Port 3 adalah 8 bit *bidirectiona*l port I/O dengan *pull-ups* internal. Selain itu port 3 juga memiliki fungsi lain yang dapat dilihat pada Tabel 2.2, yaitu :

Tabel 2.2 Fungsi lain pin port 3

Pin Port	Fungsi Lain
P3.0	RXD (port input serial)
P3.1	TXD (port output serial)
P3.2	INT0 (interupsi eksternal 0)
P3.3	INT1 (interupsi eksternal 1)
P3.4	T0 (input eksternal timer 0)
P3.5	T1 (input eksternal timer 1)
P3.6	WR (sinyal write pada data memori eksternal)
P3.7	RD (sinyal read pada data memori eksternal)

5. RST

   RST adalah input *reset* (aktif logika high) pulsa transisi dari *low* ke *high* akan *mereset* AT89C51. Pin ini dihubungkan dengan rangkaian *power reset.*

6. ALE/PROG *(Address Latch Enable /Programmed)*

   ALE/PROG adalah pin yang digunakan untuk menahan alamat. Memori eksternal selama pelaksanaan instruksi.

7. PSEN *(Programed Store Enable)*

   PSEN adalah pin yang berfungsi pada saat mengeksekusi program yang terletak pada memori eksternal. PSEN akan aktif dua kali setiap *cycle.*

8. EA/VPP *(Eksternal Access Enable)*

   EA/VPP adalah pin yang berfungsi untuk menentukan mikrokontroller mengeksekusi program eksternal atau internal. Apabila EA diberi logika *low* maka mikrokontroller akan mengeksekusi program eksternal dan apabila diberi logika *high* maka mikrokontroller akan mengeksekusi program internal.

9. XTAL 1

   XTAL 1 adalah pin input pada rangkaian osilator internal.

10. XTAL 2

    XTAL 2 adalah pin output pada rangkaian osilator internal.

11. VCC

    VCC adalah input catu daya sebesar 5 V.

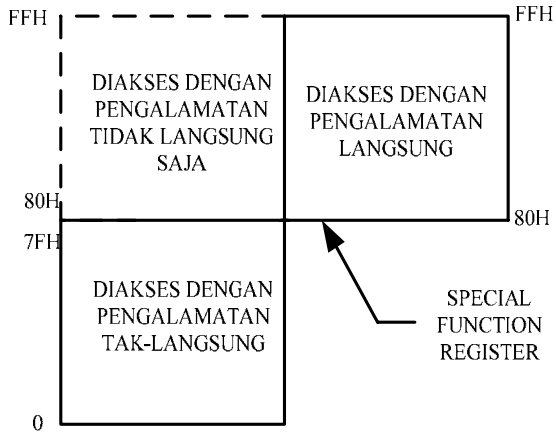
12. GND

    GDD adalah input *ground.*

2.4.2 Struktur Memori

Semua produk mikrokontroler Flash AT89C51 dari *atmel* memiliki ruang alamat memori data dan program yang terpisah. 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.

Memori program hanya bisa dibaca saja. Memori data menempati suatu ruang alamat yang terpisah dari memori program. Memori eksternal dapat diakses secara langsung hingga 64 Kbyte dalam ruang memori data eksternal. CPU akan memberikan sinyal baca dan tulis, RD dan WR selama pengaksesan memori data eksternal. Suatu konfigurasi perangkat keras untuk mengakses RAM hingga 2 Kbyte, dalam hal ini CPU mengeksekusi instruksi pada Flash internal. Port 0 bertindak sebagai bus alamat/data yang termultipleks ke RAM dan 3 jalur pada port 2 digunakan untuk menghalami *(page)* RAM. Alamat satu-byte sering digunakan untuk membantu satu atau lebih jalur I/O dalam peng-halaman-an RAM. Alamat dua-byte dapat dipakai dalam kasus jika *byte* tinggi dikirim melalui port 2. ruang memori dibagi menjadi tiga blok, yang dikenal sebagai 128 bawah *(lower 128)*, 128 atas *(upper 128)* dan Register Fungsi Khusus *(Special Function Register*=SFR*).* Memori data internal dapat dilihat pada Gambar 2.6 :



Gambar 2.6 Memori Data Internal

Mikrokontroler AT89C51 mempunyai struktur memori yang terdiri dari *RAM internal*, *special function register*, dan *Flash PEROM*. Spesifikasi dari masing-masing struktur memori tersebut adalah sebagai berikut :

1. *RAM Internal* : merupakan memori sebesar 128 byte yang biasanya digunakan untuk menyimpan variabel atau data yang sifatnya sementara. RAM Internal dialamati oleh RAM *Address Register*.
2. *Special Function Register* : memori yang berisi *register-register* yang mempunyai fungsi-fungsi khusus yang disediakan oleh mikrokontroler tersebut, seperti *timer, serial,* dan lain-lain. AT89C51 mempunyai 21 *special function register* yang terletak antara alamat 80H hingga FFH.
3. *Flash PEROM* : memori yang digunakan untuk menyimpan instruksi-instruksi MCS-51. *Flash PEROM* dialamati oleh program *Address Register*. Mikrokontroler AT89C51 mempunyai 4Kb *Flash PEROM*, yaitu ROM yang dapat ditulis ulang atau dihapus menggunakan sebuah perangkat *programmer*, yang mempunyai kemampuan ditulis ulang hingga 1000 kali. Program yang ada pada *Flash PEROM* akan dijalankan pada saat sistem di-reset, pin EA/VP berlogika 1 sehingga mikrokontroler aktif berdasarkan program yang ada pada *Flash PEROM*-nya. Namun jika pin EA/VP berlogika 0, maka mikrokontroler aktif berdasarkan program yang ada pada memori eksternal.

2.4.3 Struktur Port Mikrokontroler AT89C51

Pada *port serial* mikrokontroler AT89C51 penerimaan dan pengiriman data *port serial* melalui *register SBUF (Serial Buffer)*. Penulisan ke *SBUF* berarti membaca *register* penerimaan *SBUF*. *Port serial* pada mikrokontroler AT89C51 bisa digunakan dalam 4 mode kerja yang berbeda, terdiri dari 1 mode bekerja secara *sinkron* dan 3 lainnya bekerja secara *asinkron*.

Adapun mode kerja dari *port serial*, antara lain yaitu :

1. *Mode O* : mode ini bekerja secara *sinkron*, data serial dikirim dan diterima melalui kaki P3.0 (Rxd), sedangkan kaki P3.1 (Txd) digunakan untuk meyalurkan detak pendorong data *serial* yang dibangkitkan oleh mikrokontroler AT89C51. Data dikirim dan diterima 8 bit sekaligus dimulai dari bit LSB dan diakhiri dengan bit MSB. Kecepatan *baudrate* 1/12 frekuensi kristal yang digunakan.
2. *Mode 1* : Pada mode ini data dikirim melalui kaki P3.1 (Txd) dan diterima melalui kaki P3.0 (Rxd) secara *asinkron* (juga mode 2 dan mode 3). Pada mode 1 data dikirim dan diterima 10 bit sekaligus, diawali dengan 1 bit start, disusul dengan 8 bit data yang dimulai dari bit LSB dan diakhiri dengan 1 bit stop. Pada mikrokontroler AT89C51 yang berfungsi sebagai penerima bit stop adalah RB8 dalam register SCON *(Serial Control)*. Kecepatan *baudrate* bisa diatur sesuai dengan keperluan dengan menggunakan timer, Mode 2 dan 3 yang umum dikenal dengan UART.
3. *Mode 2* : Data dikirim dan diterima 11 bit sekaligus, diawali dengan 1 bit start, disusul 8 bit data, kemudian bit ke 9 yang bisa diatur lebih lanjut, diakhiri dengan 1 bit stop. Pada mikrokontroler AT89C51 yang berfungsi sebagai pengirim, bit 9 tersebut berasal dari bit TB8 dalam *register* SCON dan yang berfungsi sebagai penerima, bit 9 ditampung pada bit RB8 dalam *register* SCON, sedangkan bit stop diabaikan tidak ditampung. *Baudrate* bisa dipilih antara 1/32 atau 1/64 frekuensi kristal yang digunakan.
4. *Mode 3* : mode ini sama dengan mode 2 hanya saja *baudrate*-nya bisa diatur sesuai dengan keperluan seperti mode 1.

2.4.4 Mode Pengalamatan Mikrokontroler AT89C51

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

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

   Cara ini menggunakan konstanta, misalnya **MOV A,#20.** data konstanta merupakan data yang menyatu dengan instruksi, contoh instruksi diatas mempunyai arti bahwa data konstantanya yaitu 20h.

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

   Cara ini dipakai untuk menunjukan data yang berada di suatu lokasi memori dengan cara menyebut lokasi (alamat) memori tempat data tersebut berada. Misalnya : **MOV A,30h.**

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

   Cara ini digunakan untuk mengakses data yang berada di dalam memori, tetapi lokasi memori tidak disebut secara langsung tetapi dititipkan ke *register* lain. Misalnya : **MOV A,@R0.**

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

   Instruksi ini mempunyai arti bahwa data dalam *register* serbaguna R5 disalin ke akumulator A. Instruksi ini menjadikan *register* serbaguna R0 sampai R7 dapat digunakan sebagai tempat penyimpan data yang praktis dan kerjanya sangat cepat. Misalnya : **MOV A,R5**

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

   MCS-51 mempunyai cara penyebutan data dalam memori program yang dilakukan secara tak langsung, misalnya **: MOVC A,A+@DPTR.** Instruksi ini mengganti **MOV** dengan **MOVC**, tambahan huruf C dimaksudkan untuk membedakan bahwa instruksi ini digunakan untuk memori program. Tanda ‘@’ digunakan untuk menandai A+DPTR yang berfungsi untuk menyatakan lokasi memori yang isinya disalin ke akumulator A, dalam hal ini nilai yang tersimpan dalam DPTR *(Data Pointer Rregister-2 byte)* ditambah dengan nilai yang tersimpan dalam akumulator A *(1 byte)* sama dengan lokasi memori program yang diakses.

2.5 Komunikasi serial RS-232

Komunikasi data pada umumnya dapat dilakukan dengan dua cara, yaitu secara serial dan secara paralel. Komunikasi data serial dilakukan dengan mengirimkan dan menerima data 8 bit secara satu per satu, sedangkan komunikasi data paralel dilakukan dengan mengirimkan dan menerima data 8 bit secara bersamaan atau sekaligus. Protokol RS-232 merupakan standar komunikasi serial untuk sistem komputer. Dimaksudkan awalnya untuk komunikasi antara komputer. Protokol RS-232 umumnya digunakan untuk komunikasi dengan periferal yang lambat.

Terdapat 25 sinyal untuk protokol RS-232 namun hanya tiga yang digunakan untuk banyak kasus, yaitu :

1. RxD *(received data)* digunakan untuk menerima data.
2. TxD *(transmited data)* digunakan untuk mengirim data.
3. GND *(ground)*

RS-232 memerlukan tegangan non TTL, yaitu :

1. - 3 s/d – 25 Volt logika 1
2. + 3 s/d + 25 Volt logika 0

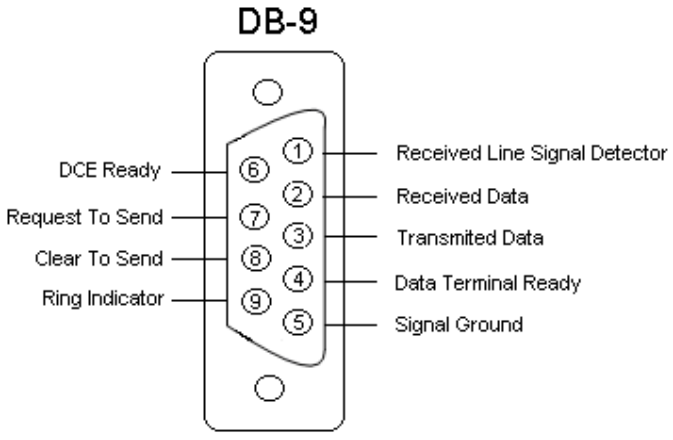
Mikrokontroler AT89C51 merupakan level TTL, oleh karena itu diperlukan antarmuka untuk menghubungkan AT89C51 dengan jalur RS-232. Antarmuka tersebut adalah IC MAX 232. Fungsi IC MAX-232 ini adalah untuk menerima standar level logika TTL dan menghasilkan level RS-232 dengan *supply* normal + 5 Volt. RS-232 (*recommended standard number 232*) merupakan seperangkat alat yang diciptakan oleh *Electrical Industries Assosiation* yang berfungsi sebagai antarmuka dalam mentransfer data dengan komputer yang mana pengiriman data dilakukan dengan mengirimkan kode biner. Pada seperangkat komputer biasanya tersedia *Communication Port* atau sering disebut dengan COM. Biasanya terdapat dua buah *Communication Port*, yaitu COM1 dan COM2.

Pada dasarnya ada dua jenis komunikasi data *serial*, yaitu komunikasi data *serial sinkron* dimana pengiriman *clock* dilakukan secara bersamaan dengan data *serial* dan komunikasi data *serial asinkron* dimana pengiriman *clock* dilakukan secara dua tahap, yaitu saat data dikirimkan dan saat data diterima. RS-232 pada komputer mempunyai dua jenis konektor, yaitu konektor dengan 25 pin atau sering disebut DB-

25 *connector* dan konektor dengan 9 pin atau sering disebut DB-9 *connector* (lihat Gambar 2.7). Perlu diperhatikan bahwa dalam pengiriman data *serial* semakin jauh jarak kirim maka kemungkinan *noise* atau gangguan semakin besar.

Dalam setiap proses transfer data *serial*, RS-232 memerlukan sebuah *Data Terminal Equipment* atau sering disebut DTE dan *Data Communication Equipment* atau sering disebut DCE pada masing-masing terminal. Pengiriman data dilakukan secara bit per bit, misalnya jika ingin mengirim suatu karakter "A" yang dalam format ASCII adalah 41H atau 01000001 biner maka data akan dikirim mulai dari bit pertama, kedua, sampai bit terakhir. Kecepatan transfer data harus sama antara penerima dan pengirim, jika kecepatannya tidak sama maka akan terjadi *overflow*. Kecepatan transmisi tersebut biasanya sering disebut dengan *boud rate*. *Boud rate* yang sering dipakai diantaranya adalah 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 230400, 460800, dan 921600. panjang data bit yang sering digunakan diantaranya adalah 4, 5, 6, 7, dan 8 bit.

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



Gambar 2.7 Konektor Serial DB-9

Fungsi dari pin DB 9 dapat dilihat pada Tabel 2.3. Berikut ini adalah tabel penggunaan pin, dan jenis sinyal konektor DB 9 :

Tabel 2.3 Fungsi Dari Pin DB 9

Pin (DB-9)	Nama Sinyal	Fungsi	Jenis
1	Data Carrier Detect	Untuk mendeteksi boleh atau tidaknya DTE menerima data	Input
2	Received Data (RxD)	Sebagai jalur pengiriman data dari DCE ke DtE	Input
3	Transmited Data (TxD)	Sebagai jalur pengiriman data dari DTE ke DCE	Output
4	Data Terminal Ready (DTR)	Untuk memberitahu kesiapan terminal DTE	Output
5	Ground	Sebagai ground	-
6	Data Set Ready (DSR)	Untuk menyatakan bahwa status data tersambung pada DCE	Input
7	Request To Send (RTS)	Untuk mengirim sinyal informasi dari DTE ke DCE bahwa akan ada data yang akan dikirim	Output
8	Clear To Send (CTS)	Untuk memberitahu pada DTE bahwa DCE siap untuk menerima data.	Input
9	Ring Indicator	Untuk memberitahu DTE bahwa ada terminal yang menginginkan komunikasi dengan DCE	Input

2.6 Format Pengiriman Data

Port serial AT89C51 dapat digunakan untuk komunikasi data secara *sinkron* maupun *asinkron*. Komunikasi data serial secara *sinkron* merupakan bentuk komunikasi data serial yang memerlukan sinyal *clock* untuk sinkronisasi. Sinyal *clock* tersebut akan tersulut pada setiap bit pengiriman data, sedangkan komunikasi *asinkron* tidak memerlukan sinyal *clock* sebagai sinkronisasi. Pengiriman data pada komunikasi serial AT89C51 dilakukan mulai dari bit yang paling rendah (LSB) hingga bit yang paling tinggi (MSB).

1. Komunikasi *Sinkron*

   Sinyal *clock* pada komunikasi sinkron diperlukan oleh peralatan penerima data untuk mengetahui adanya pengiriman setiap bit data. Peralatan atau komponen penerima akan mengetahui adanya pengiriman bit yang pertama ataupun perubahan bit data dengan mendeteksi sinyal *clock*. Pada aplikasinya, komunikasi *sinkron* dari serial port AT89C51 selalu digunakan untuk mengakses *shift register*. PISO *(Parallel In Serial Out)* untuk proses penerimaan data dari PISO ke port serial AT89C51 atau SIPO *(Serial In Parallel Out)* untuk proses pengiriman data dari AT89C51 ke SIPO.

2. Komunikasi *Asinkron*

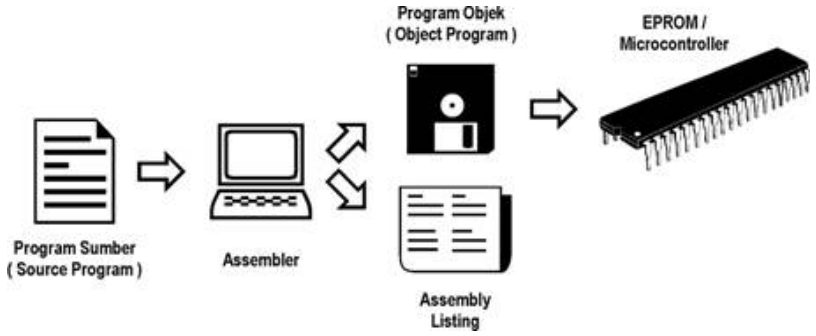
   Komunikasi *asinkron* tidak memerlukan sinyal *clock* sebagai sinkronisasi, namun pengiriman data ini harus diawali dengan start bit dan diakhiri dengan stop bit. Sinyal *clock* yang merupakan *baudrate* dari komunikasi data ini dibangkitkan oleh masing-masing baik penerima maupun pengirim data dengan frekuensi yang sama. Pada aplikasinya proses komunikasi *asinkron* ini selalu digunakan untuk mengakses komponen-komponen yang mempunyai fasilitas *UART (Universal Asynchronous Reciver/Transmitter)* seperti port serial PC atau port serial mikrokontroler yang lain.

2.7 Teori Pemrograman

Tulisan-tulisan atau *syntax-syntax* yang ada di komputer yang digunakan sebagai perintah atau instruksi pada komputer agar komputer bisa menjalankan tugasnya sesuai dengan yang kita inginkan. Bahasa pemrograman terdiri dari bahasa pemrograman tingkat rendah dan bahasa pemrograman tingkat tinggi *(high level).*

2.7.1 Bahasa Pemrograman Assembly

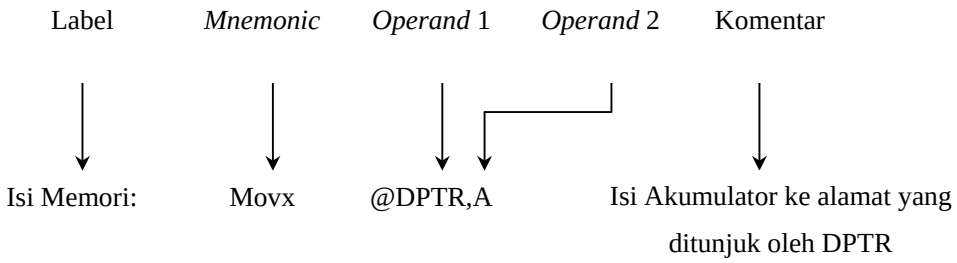
Assembler adalah program komputer yang mentranslasi program dari bahasa *assembly* ke bahasa mesin. Sedangkan bahasa *Assembly* adalah ekuivalensi bahasa mesin dalam bentuk *alphanumerik*. *Mnemonics alphanumerik* digunakan sebagai alat bantu bagi *programmer* untuk memprogram mesin komputer daripada menggunakan bahasa mesin yang panjang dan rumit. Proses cara pembuatan program *Assembly* terlihat pada Gambar 2.8.



Gambar 2.8 Proses Assembly

Program sumber *Assembly* terdiri dari kumpulan baris-baris perintah dan biasanya disimpan dengan *extension* .ASM dengan 1 baris untuk satu perintah, setiap baris perintah tersebut bisa terdiri atas beberapa bagian (*field*), yakni bagian label, bagian *mnemonic*, bagian *operand* yang bisa lebih dari satu dan terakhir bagian komentar. Program sumber *(Source Code)* dibuat dengan program *editor* biasa yaitu *Note Pad* pada *Windows*, selanjutnya program sumber diterjemahkan ke bahasa mesin dengan menggunakan program *Assembler*. Hasil kerja program *Assembler* adalah program objek dan juga *assembly listing*, dengan ketentuan sebagian berikut :

1. Masing-masing bagian dipisahkan dengan spasi atau TAB, khusus untuk *operand* yang lebih dari satu masing-masing *operand* dipisahkan dengan koma.
2. Bagian-bagian tersebut tidak harus semuanya ada dalam sebuah baris, jika ada satu bagian yang tidak ada maka *spasi* atau *TAB* sebagai pemisah bagian tetap harus ditulis.
3. Bagian Label ditulis mulai huruf pertama dari baris, jika baris bersangkutan tidak mengandung label, maka label tersebut digantikan dengan *spasi* atau *TAB*, yakni sebagai tanda pemisah antara bagian Label dan bagian *mnemonic* seperti pada Gambar 2.9.



Gambar 2.9 Bentuk Program Sumber Assembly

Assembler Directive digunakan untuk mengatur kerja dari program *Assembler*. *Mnemonic* untuk *Assembler Directive* tergantung pada program *Assembler* yang dipakai.

Ada beberapa *Assembler Directive* yang umum, yang sama untuk banyak macam program *Assembler* diantaranya adalah :

1. ORG (*Origin)*
   Digunakan untuk menyatakan lokasi memori tempat instruksi atau perintah yang ada di bawahnya disimpan.
2. EQU (*Equate)*
   EQU digunakan untuk mendefinisikan sebuah simbol atau lambang *assembl*er secara bebas.
3. DB (*Define Byte)*
   Digunakan untuk memberikan nilai tertentu pada *memory-program*.
4. DW (*Define Word)*
   Dipakai untuk memberi nilai 2 *byte* ke *memory-program* pada baris bersangkutan. *Assembler Directive* ini biasa dipakai untuk membentuk suatu tabel yang isinya adalah nomor-nomor *memory-program*.
5. DS (*Define Storage)*
   Assembler Directive ini dipakai untuk membentuk variabel. Sebagai variabel tentu saja memori yang dipakai adalah *memory-data* (RAM) bukan *memory-program* (ROM).

2.7.2 Bahasa Pemrograman Delphi 7

Borland Delphi adalah paket bahasa pemrograman yang mempunyai cakupan kemampuan yang luas dan sangat canggih. Secara umum, kemampuan *Delphi* adalah menyediakan komponen-komponen dan bahasa pemrograman yang andal, sehingga memungkinkan kita untuk membuat program aplikasi sesuai dengan keinginan, dengan tampilan dan kemampuan yang canggih.

Borland Delphi merupakan bahasa Pemrograman yang mudah, karena *Delphi* adalah bahasa perograman tingkat tinggi (*high level*) sehingga sangat memudahkan *user* untuk bermain-main di tingkat ini. Pada Pemrograman *Delphi* kita tinggal *click* dan *drag* ke *form*, dan jadilah program aplikasi yang kita inginkan.

Delphi memiliki kelebihan dibandingkan dengan program aplikasi lainnya, karena *Delphi* bersifat produktifitas dari pengembangan perangkat lunak yang dapat dibagi menjadi 5 atribut penting, yaitu :

1. Kualitas lingkungan pengembangan *visual.*
2. Kecepatan *compiler* dibandingkan dengan kompleksitasnya.
3. Kekuatan bahasa pemrograman dibandingkan dengan kompleksitasnya.
4. *Fleksibilitas* arsitektur basis data.
5. Pola desain dan pemakaian yang diwujudkan oleh *framework*-nya.

Pada saat pertama kali dipanggil, akan muncul beberapa *tools* utama dalam *Delphi* IDE, yaitu *Menu, Toolbar, Component Palette, Object Inspector, Object TreeView, Code Editor, Project Manager*, dan beberapa *tools* lain. Pada dasarnya IDE (I*ntegrated Development Environment*) milik *Delphi* dibagi menjadi 6 bagian utama :

1. *Menu*

   Dari menu ini kita dapat memanggil atau menyimpan program, menjalankan dan melacak *bug* program.

2. *Speed Bar*

   Disebut juga *toolbar* berisi kumpulan tombol, setiap tombol pada *Speed Bar* menggantikan salah satu item menu, contoh seperti pada Gambar 2.10 :



Gambar 2.10 *Speed Bar*

3. *Component Pallete*

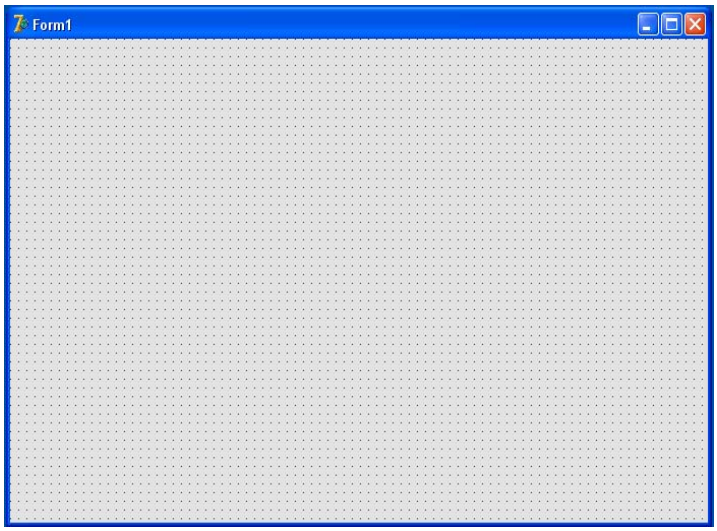
   Component Pallete berisi komponen-komponen *Delphi* (visual/non visual VCL dan CLX) yang dapat dipergunakan untuk menyusun tampilan program dalam *form*. Komponen yang ada dalam *component palllete* disusun dan dikelompokkan dalam *page* sesuai fungsinya. Bila kita letakkan *mouse pointer* diatas sebuah komponen dalam *component pallete* dan kita diamkan beberapa saat, maka akan muncul nama komponen tersebut dalam bentuk *tooltip*, seperti yang tampak pada Gambar 2.11 berikut :



Gambar 2.11 *Component Pallete*

4. *Form Designer*

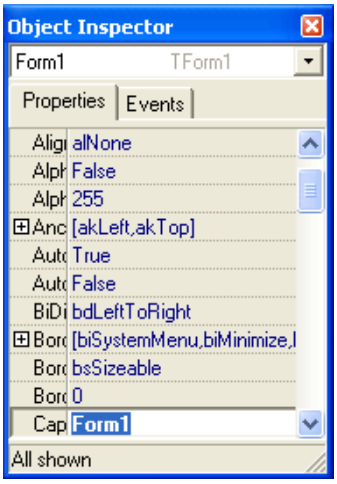
   Form Designer adalah *form* tempat kita mengatur komponen sesuai kebutuhan tampilan program yang kita buat. Objek yang sudah ada dalam *form* dapat diatur posisinya sesuai kebutuhan cukup dengan *click-drag* pada objek tersebut. Objek yang terseleksi akan ditandai dengan delapan kotak kecil di bagian tepi dan sudutnya. *Form Designer* tampak pada Gambar 2.12 :



Gambar 2.12 ***Form Designer***

5. *Object Inspector*

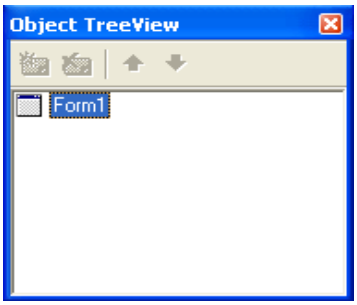
   Object Inspector digunakan untuk mengubah nilai *property* dari objek terseleksi yang ada dalam *Form Designer*. *Object Inspector* terdiri atas dua *page*, yaitu *property* dan *event.* Setiap jenis komponen akan memiliki *property* dan *event* yang berbeda dengan jenis komponen lainnya (lihat Gambar 2.13).



Gambar 2.13 ***Object Inspector***

6. *Object TreeView*

Object TreeView menampilkan hubungan *parent-child* antar komponen dalam bentuk hirarki. *Object TreeView, Form Designer, dan Object Inspector* saling tersinkronisasi, sehingga bila fokus kita ubah dalam *Form Designer* maka fokus dalam *Object TreeView* dan *Object Inspector* juga akan berpindah, gambar dari *Object TreeView* dapat dilihat pada Gambar 2.14 :



Gambar 2.14 *Object TreeView*