BAB II

DASAR TEORI

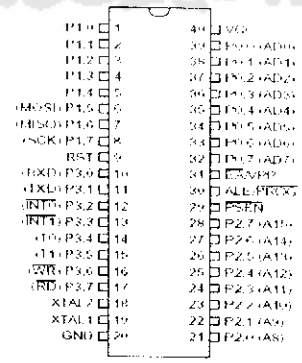
2.1. Mikrokontroler AT89S51

Mikrokontroler AT89S51 merupakan mikrokontroler 8 bit berbasis arsitektur MCS-51 yang memiliki :

1. Memori program (*flash memory)* 4 Kbytes,
2. Memori data 256 byte,
3. 32-bit *Programmable* I/O,
4. 2 buah *Timer/Counter* 16-bit,
5. *Programmable UART serial channel,*
6. Modul SPI (Serial Peripheral Interface).

2.1.1. Deskripsi kaki-kaki mikrokontroler AT89S51

Mikrokontroler AT89S51 memiliki 40 pin seperti yang ditunjukkan pada Gambar 2.1 berikut:



Gambar 2. 1. Pin-pin pada mikrokontroler AT89S51

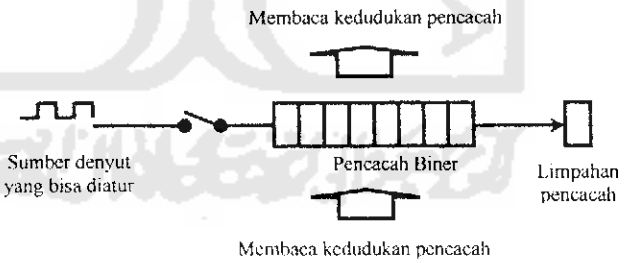
Fungsi dari tiap-tiap pin adalah sebagai berikut.

1. VCC merupakan suplay tegangan sebesar 4 – 6 volt
2. GND merupakan pin *ground*
3. *Port* 0 merupakan *port* masukan-keluaran dua arah dan dikonfigurasikan sebagai multiplek dua *bus* alamat rendah (A0-A7) dan data selama pengaksesan program memori dan data memori eksternal.
4. *Port* 1 merupakan *port* masukan-keluaran dengan internal *pull-up*.
5. *Port* 2 merupakan *port* masukan-keluaran dengan internal *pull-up.* Port 2 mengeluarkan alamat byte tinggi selama pengaksesan ke memori data eksternal. Selain itu *port* 2 juga tetap menghasilkan isi SFR (*Special Function Register*) dari P2.
6. *Port* 3 merupakan *port* masukan-masukan dengan internal *pull-up*. *Port* 3 juga memiliki fungsi khusus, yaitu: RXD (P3.0) *port* masukan serial, TXD (P3.1) *port* keluaran serial, INT0 (P3.2) interupsi eksternal 0, INT1 (P3.3) interupsi eksternal 1, T0 (P3.4) *input external timer* 0, T1 (P3.5) *input external timer* 1, WR (P3.6) *strobe* tulis data memori eksternal, RD (P3.7) *strobe* baca data memori eksternal.
7. RST merupakan input *reset*
8. ALE/PROG, pulsa keluaran ALE digunakan untuk proses '*latching*' byte *address* rendah (A0-A7) selama pengaksesan ke eksternal memori.
9. PSEN merupakan *strobe* baca ke program memori ekternal.

10. EA/VPP adalah *External Address enable* (EA) di-*ground*-kan jika mengakses memori eksternal. Untuk mengakses memori internal maka dihubungkan ke V_{CC}.
11. X-TALL1 dan X-TALL2 kaki ini dihubungkan dengan kristal bila digunakan osilator internal. X-TALL1 merupakan masukan *inverting* osilator *amplifier* sedangkan X-TALL2 merupakan keluaran *inverting* osilator *amplifier*.

2.1.2. Timer/Counter

Pada dasarnya sarana masukan ini merupakan seperangkat pencacah biner yang terhubung langsung ke saluran data mikrokontroler sehingga mikrokontroler bisa membaca kondisi pencacah dan bila diperlukan dapat pula mengubah kondisi pencacah tersebut. Seperti layaknya pencacah biner, saat sinyal *clock* yang diberikan sudah melebihi kapasitas pencacah maka pencacah akan memberikan sinyal *overflow*, seperti yang ditunjukkan pada Gambar 2.2 berikut:



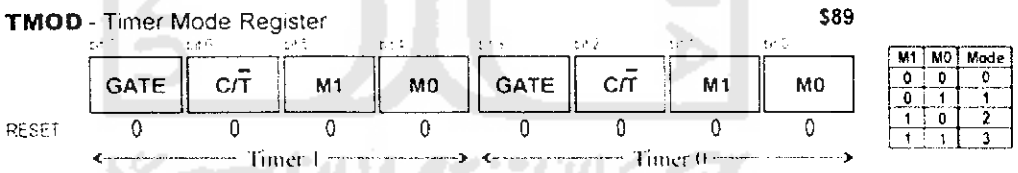
Gambar 2. 2. Konsep dasar *timer/counter* pada AT89S51

Sinyal *clock* yang diberikan ke pencacah dibedakan menjadi 2 macam yaitu sinyal *clock* dengan frekuensi tetap dan bervariasi. Jika sebuah pencacah bekerja dengan frekuensi tetap pencacah tersebut bekerja sebagai *timer* atau pewaktu, karena pencacah tersebut identik dengan waktu yang bisa ditentukan

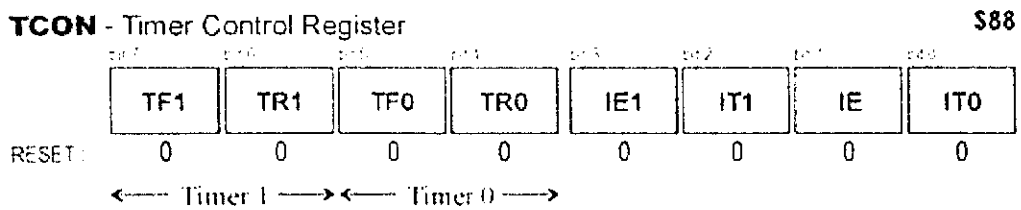
secara pasti. Sedangkan jika sebuah pencacah bekerja dengan frekuensi bervariasi, pencacah tersebut bekerja sebagai *counter* atau pencacah.

Mikrokontroler AT89S51 memiliki 3 perangkat *timer/counter* yaitu *timer* 0, *timer* 1, dan *timer* 2. Untuk mengakses *timer/counter* digunakan register khusus yang tersimpan dalam SFR. Pencacah biner *timer* 0 diakses melalui register TL0 (*Timer 0 low byte*, pada alamat 6Ah) dan register TH0 (*Timer 0 high byte*, pada alamat 6Ch). Pencacah biner *timer* 1 diakses melalui register TL1 (*Timer 1 low byte,* pada alamat 6Bh) dan register TH1 (*Timer 1 high byte,* pada alamat 6Dh). Untuk mengatur kerja *timer/counter* tersebut digunakan 2 register tambahan yaitu register TCON (*Timer control register* pada alamat 88h) dan register TMOD (*Timer mode register* pada alamat 89h).

Timer 2 merupakan sebuah *timer/counter* 16 bit. Register-register yang digunakan untuk mengakses *timer* 2 adalah TL2 dan TH2, register kontrol *timer* T2CON dan *register-register capture*: RCAP2L dan RCAP2H.



Gambar 2. 3. *Timer Mode Register* (TMOD)



Gambar 2. 4. *Timer Control Register* (TCON)

Register TMOD dan register TCON merupakan register yang digunakan untuk mengatur kerja timer 0 dan timer 1 susunan bit TMOD dan TCON seperti pada gambar diatas. Register TMOD dibagi menjadi 2 bagian secara simetris yaitu bit 0 sampai bit 3 untuk timer 0 dan bit 4 sampai bit 7 untuk mengatur timer 1. Bit M0/M1 dipakai untuk menentukan mode timer seperti Tabel 2.1 berikut :

Tabel 2. 1. Mode kerja register TMOD

M1	M0	MODE	Keterangan
0	0	0	*Timer/counter* 13 bit
0	1	1	*Timer/counter* 16 bit
1	0	2	*Timer/counter* 8 bit isi ulang
1	1	3	Gabungan antara 16 bit dan 8 bit

Bit 6 (bit C/T) pada register TMOD dipakai untuk mengatur sumber sinyal detak yang diberikan ke pencacah biner. Bit TFx (TF0 atau TF1) pada register TCON merupakan bit penampung limpahan, TFx akan menjadi '1' setiap kali pencacah biner yang terhubung padanya melimpah sedangkan bit TRx (TR0 atau TR1) merupakan bit pengatur saluran sinyal detak, bila bit ini = 0 sinyal detak tidak disalurkan ke pencacah biner sehingga pencacah berhenti melakukan pencacahan.

2.1.3. Penyimpan (memori)

Pada mikrokontroler AT89S51 terdapat dua bagian memori yaitu : memori data dan memori program. Memori data menempati suatu ruang alamat yang terpisah dari memori program. Memori data dan memori program eksternal

dapat dikombinasikan dengan cara menggabungkan sinyal $\overline{RD}$ dan $\overline{PSEN}$ melalui gerbang NAND dan keluarannya sebagai tanda baca ke memori data/program eksternal.

2.1.3.1. Memori data

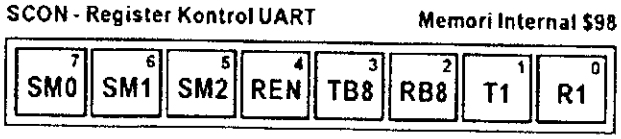
Memori data internal dibagi menjadi dua blok yang dikenal sebagai 128 byte bawah (*lower 128 byte*) yang menempati alamat 00h – 007h dan 128 byte atas (*upper 128 byte*) yang merupakan register fungsi khusus SFR (*Spesial Function Register*) pada alamat 80h – FFh yang hanya dapat diakses dengan pengalamatan tidak langsung.

2.1.3.2. Memori program

Memori program sering juga disebut dengan *flash memory* dengan kapasitas sebesar 4K-byte yang hanya digunakan untuk membaca saja. Memori ini dapat diakses pada alamat 0000H – 1FFFH.

2.1.4. UART (*Universal Asynchronous Receiver/Transmitter*)

Komunikasi UART ini digunakan dalam komunikasi infra merah antara rangkaian pemancar dan penerima. Port serial pada AT89S51 bisa digunakan dalam 4 *mode* kerja yang berbeda. Dari 4 *mode* tersebut, 1 *mode* diantaranya bekerja secara sinkron dan 3 lainnya bekerja secara asinkron. *Register control* dan status untuk *Port serial* berada dalam SCON seperti pada Gambar 2.5. *Register* ini mengandung *bit-bit* pemilih *mode* kerja *port serial*, *bit* data ke-9 pengiriman dan penerimaan (TB8 dan RB8) serta *bit-bit* interupasi *port serial* (TI dan RI).



Gambar 2. 5. Susunan bit register SCON

Bit SM0 dan SM1 (bit 7 dan 6 pada register SCON) dipakai untuk menentukan *mode* kerja port serial. Setelah *reset* kedua *bit* ini bernilai'0' dan penentuan kerja *port serial* seperti pada Tabel 2.2 berikut :

Tabel 2. 2. Mode kerja *port* serial

SM0	SM1	Mode	Keterangan	Baud Rate
0	0	0	Register Geser	Tetap (f osc/2)
0	1	1	UART 8-bit	Bisa diubah-ubah (dgn timer)
1	0	2	UART 9-bit	Tetap (f osc/64 atau f osc/32)
1	1	3	UART 9-bit	Bisa diubah-ubah (dgn timer)

Kecepatan transmisi (*baud rate*) merupakan suatu hal yang amat penting dalam komunikasi data seri asinkron, mengingat dalam komunikasi data seri asinkron *clock* tidak ikut dikirimkan, sehingga harus diusahakan bahwa kecepatan transmisi mengikuti standar yang sudah ada.

Dalam AT89S51, *clock* untuk transmisi data dibangkitkan dengan sarana *timer* 1. Untuk keperluan ini, *timer* 1 dioperasikan sebagai 8 bit *auto reload timer* (*mode* 2), artinya TL1 bekerja sebagai *timer* 8 bit menerima *clock* dari osilator kristal yang frekuensinya sudah dibagi 12 terlebih dulu, setiap kali pencacah (*counter*) nilainya menjadi 0 maka nilai yang sebelumnya sudah disimpan di TH1 secara otomatis diisikan lagi ke TL1, sehingga TL1 akan menghasilkan *clock* yang frekuensinya diatur oleh TH1, *clock* ini berikutnya dibagi lagi dengan 32 sebelum

dipakai sebagai *clock* untuk UART. Sehingga dapat dinyatakan dengan persamaan 2.1 berikut :

$$Baudrate = \frac{f_{osilator}}{12 \times [256 - TH1] \times 32} \quad (2.1)$$

Kalau kecepatan transmisi (*baudrate*) sudah ditentukan dan frekuensi kristal sudah dipastikan, maka nilai yang disimpan di TH1 bisa dihitung berdasarkan persamaan 2.2 berikut :

$$TH1 = 256 - \frac{K \times f_{osilator}}{12 \times 32 \times Baudrate} \quad (2.2)$$

Dalam persaman di atas, K adalah konstanta yang nilainya 1 atau 2, tergantung pada nilai yang tersimpan di bit SMOD dalam register PCON. Jika SMOD ='0' K bernilai 1 dan kalau SMOD='1' K akan bernilai 2. Perlu dicacat, setelah AT89S51 di-reset, SMOD akan bernilai '0', artinya jika tidak diatur lebih lanjut K bernilai 1. $f_{osilator}$ adalah frekuensi osilator kristal yang digunakan.

2.1.5. Interupsi

Ketika mikrokontroler sedang melaksanakan suatu program, interupsi dapat menghentikan pelaksanaan program tersebut sementara dan kemudian akan mengembalikannya ke program sebelum terjadi interupsi. Apabila CPU mendapat permintaan interupsi, *Program Counter* (PC) akan diisi alamat dari vektor interupsi. CPU kemudian melaksanakan rutin pelayanan interupsi mulai dari alamat tersebut. Bila rutin pelayanan interupsi selesai dilaksanakan, CPU kembali melaksanakan program utama yang ditinggalkannya.

Mikrokontroler AT89S51 mempunyai beberapa saluran interupsi. Pada mikrokontroler jenis ini interupsi dibedakan menjadi dua jenis, yaitu:

1. Interupsi yang tak dapat dihalangi oleh perangkat lunak *(non maskable interrupt)*, yaitu *reset.*
2. Interupsi yang dapat dihalangi perangkat lunak (*maskable interrupt*), meliputi INT0 dan INT1 (eksternal), *Timer/Counter* 1 dan interupsi dari port serial (internal).

Instruksi RET1 (*Return from Interrupt Routine*) harus digunakan untuk kembali dari layanan rutin interupsi. Instruksi ini dapat dipakai agar saluran interupsi kembali dapat dipakai. Alamat awal layanan rutin interupsi dari setiap sumber interupsi ditunjukkan pada Tabel 2.3.

Interupsi eksternal INT0 dan INT1 masing-masing dapat diaktifkan berdasarkan level atau transisi tergantung pada bit IT0 dan IT1 dalam TCON. *Flag* yang menghasilkan interupsi ini adalan bit dalam IE0 dan IE1 dari TCON. Interupsi *timer* 0 dan *timer* 1 dihasilkan oleh TF0 dan TF1. Terdapat dua buah register untuk mengontrol interupsi, yaitu IE (*Interrupt Enable*) dan IP (*Interrupt Priority*). Prosesor tidak akan menanggapi interupsi jika suatu interupsi belum dilaksanakan secara lengkap.

Tabel 2. 3. Alamat layanan rutin interupsi

Nama	Lokasi	Alat Interupsi
Reset	00H	*Power on reset*
INT0	03H	INT0
*Timer*0	0BH	*Timer*0
INT1	13H	INT1
*Timer*1	1BH	*Timer*1
SINT	23H	Port I/O serial

2.1.6. IE (*Interrupt Enable*)

Setiap sumber interupsi dapat diaktifkan maupun dilumpuhkan secara individual dengan mengatur satu bit di SFR (*Special Function Register*) yang bernama IE. Bit-bit IE didefinisikan dalam Tabel 2.4.

Jika akan megaktifkan interupsi 0 (INT0), nilai yang harus diberikan ke IE adalah 81H (yaitu memberikan logika 1 ke EEA dan EX0). Pengaturan nilai IE ini dapat juga dilakukan dengan pengalamatan secara it (*bit addressable*).

Tabel 2. 4. *Interrupt Enable* (IE)

	MSB							LSB
	EA	-	-	ES	ET1	EX1	ET0	EX0
Bobot	80H	40H	20H	10H	08H	04H	02H	01H

Dimana :

EA: *Disable* semua interupsi apabila bit ini *clear*. Bila bit ini *clear*, maka apapun kondisi bit lain dalam register ini, semua interupsi tidak akan dilayani, olch karena itu untuk mengaktifkan salah satu interupsi, bit ini harus *set*

ES: *Enable/disable Serial Port Interrupt, set = enable, clear = disable.* Apabila *serial port interrupt* aktif maka interupsi akan terjadi setiap ada data yang masuk ataupun keluar melalui serial port yang membuat *flag* RI (*Receive Interrupt Flag*) ataupun TI (*Transmit Interrupt Flag*).

ET1: *Enable/disable Timer* 1 *Interrupt, set = enable, clear = disable.* Apabila interupsi ini *enable* maka interupsi akan terjadi pada saat Timer 1 *overflow*.

EX1: *Enable/disable External Interrupt* 1, *set = enable, clear = disable.* Apabila interupsi ini *enable* maka interupsi akan terjadi pada saat terjadi pulsa *low* pada INT1

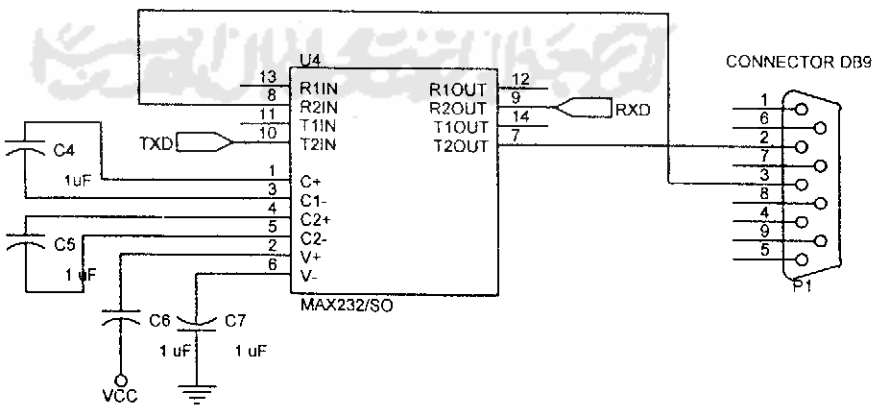
ET0: *Enable/disable Timer 0 Interrupt, set = enable, clear = disable.* Apabila interupsi ini *enable* maka interupsi akan terjadi pada saat Timer 0 *overflow*.

EX0: *Enable/disable External Interrupt 0, set = enable, clear = disable.* Apabila interupsi ini *enable* maka interupsi akan terjadi pada saat terjadi pulsa *low* pada INT0

2.2. Komunikasi RS232

Komunikasi serial RS232 digunakan sebagai antarmuka antara komputer dengan mikrokontroler. Agar level tegangan data serial dari mikrokontroler setara dengan level tegangan komunikasi port serial PC, diperlukan MAX232 untuk mengubah ke tegangan TTL/CMOS *logic level* RS232. .

Ada tiga hal pokok yang diatur dalam MAX232, antara lain bentuk sinyal dan level tegangan yang dipakai, penentuan jenis sinyal dan konektor yang dipakai, susunan sinyal pada kaki-kaki di konektor serta penentuan tata cara pertukaran informasi antara komputer dan alat-alat pelengkapnya. Gambar 2.6 berikut menunjukkan rangkaian koneksi RS232 dengan mikrokontroler.



Gambar 2. 6. Rangkaian koneksi MAX232 dengan mikrokontroler

2.3. Perangkat lunak *Borland Delphi*

Perkembangan bahasa pemrograman berlangsung dengan pesat, mulai dari bahasa tingkat rendah (yang lebih dekat dengan *hardware*) seperti bahasa *assembly*, ataupun bahasa tingkat tinggi, seperti bahasa C dan Pascal. Pada bahasa ini agak susah mengembangkannya dan membuat tampilannya menjadi menarik, apalagi bila menggunakan bahasa pemrograman ini dalam bentuk grafik. Perkembangan bahasa pemrograman terus berlanjut dengan dikeluarkannya Windows oleh pihak Microscoft. Pada Windows diperkenalkan model OOP (*Object Oriented Programming*) yaitu yang lebih menampilkan kemudahan dan tampilan yang menarik (seperti *window-window* pada aplikasi yang sering gunakan pada *MS Word*). Delphi merupakan salah satu program yang berbasis pada OOP, jadi dengan bahasa Delphi dapat dibuat program-program yang menarik untuk dilihat dan fleksibel serta *user friendly*.

2.3.1. Form

Berbeda dengan Pascal, pada Delphi dikenal OOP (*Object oriented programming*), jadi bila pada Pascal tampilannya menjemukan, pada Delphi tampilannya dapat diatur semenarik mungkin pada *form* yang digunakan. Untuk mengatur tampilan *form* caranya cukup mudah, hanya dengan menaruh komponen-komponen yang diinginkan pada *form* tersebut, dan memfungsikan masing-masing komponen sesuai dengan yang diinginkan.

2.3.2. Unit

Setiap perubahan pada *form* akan berakibat perubahan pada unit. Untuk pindah dari form ke unit, dapat dilakukan dengan menekan tombol **F12.** Berikut

ini adalah bentuk unit yang diberikan Delphi saat pertama kali membuka sebuah *form* :

```
unit Unit1;
interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
TForm1 = class(TForm)

private
{ Private declarations }

public
{ Public declarations }

end;
var
Form1: TForm1;

implementation
{$R *.DFM}
end.
```

2.3.3. Komponen

Dalam membuat program, Delphi telah menyediakan banyak kemudahan, yaitu dengan disediakannya komponen-komponen. Komponen ini merupakan sebuah prosedur/program yang sudah di kompile dan langsung dapat digunakan, sesuai dengan fungsinya masing-masing. Untuk menggunakan komponen ini dengan meng-klik komponen yang diinginkan, kemudian klik di *form*, maka komponen tersebut akan muncul di *form*. Tampilan *component palette standard* dari delphi seperti pada Gambar 2.7 berikut

Gambar 2. 7. *Component palette standard* dari delphi

Kegunaan dari beberapa komponen adalah sebagai berikut:

a. *Button/ Bitbtn*

Komponen ini biasa digunakan sebagai tombol kendali. Perbedaan antara bitbtn dengan btn adalah : bitbtn dapat menyisipkan warna pada tombol dan *icon* tertentu, lain halnya dengan menggunakan btn.

b. *Panel*

Panel berfungsi untuk mengelompokan komponen-komponen didalamnya.

c. *Label*

Kita dapat menamakan atau memberi keterangan pada program.

d. *Edit*

Edit berfungsi sebagai masukan data (*input*) dalam bentuk *string*, dari bentuk *string* dapat diubah menjadi bentuk *integer* atau bentuk lainnya. Yang kemudian dapat digunakan untuk operasi selanjutnya.

e. *Chart*

Dengan *Chart* data-data yang telah dianalisa, dapat ditampilkan ke dalam grafik, sehinga memudahkan untuk menganalisanya.

f. *Stringgrid*

Stringgrid berguna untuk menaruh data *string* kedalam bentuk kolom tabel, seperti pada Excel.

g. PopupMenu

PopupMenu berfungsi sebagai perintah yang aktif bila meng-klik kanan *mouse*, Untuk mengaktifkannya harus mengaktifkan popup menu pada komponen yang dinginkan, caranya : ubah pada *object inspector*.

h. MainMenu

Contoh *main menu* adalah *Option* pada tiap aplikasi porgram, dengan komponen ini, fungsi-fungsi program dapat ditaruh seperti pada aplikasi umumnya.

i. ComboBox

Combo Box berfungsi sebagai petunjuk untuk pemilihan berbagai masukan.

j. CheckBox

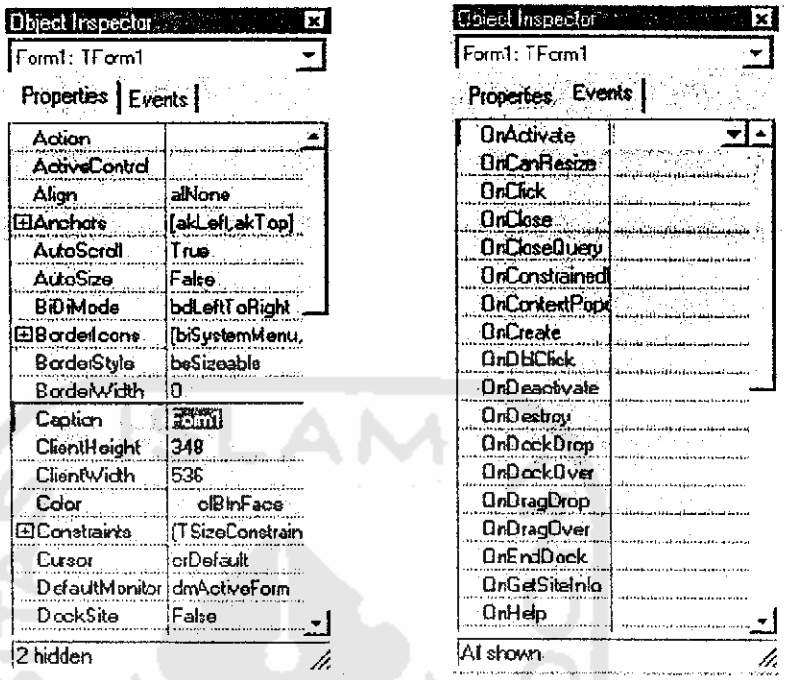
Bila komponen ini di *check* maka ada aplikasi yang bisa disetting untuk bekerja dibawahnya.

k. RadioButton

Prinsip kerjanya hampir sama dengan *check box*, cuma tampilannya saja yang berbeda.

2.3.4. *Object Inspector*

Object inspector berguna sebagai menu pilihan dari masing-masing komponen. Dengan *object inspector* komponen-komponen yang digunakan dapat dimanipulasi. Pada Gambar 2.8 berikut ditunjukan tampilan *object inspector*.



(a) Properties (b) Events

Gambar 2. 8. *Object inspector*

2.3.5. Komponen serial Delphi

Ada beberapa metode yang digunakan untuk mengakses data melalui port serial menguunakan Deplhi salah satunya adalah dengan menggunakan komponen ComPort. Pada menu komponen standar yang digunakan pada Delphi 7 tidak terdapat komponen ComPort sehingga untuk dapat menggunakan komponen tersebut terlebih dulu komponen ComPort diinstallkan ke Delphi 7. Cara menginstall Komponen ComPort ke Delphi 7 adalah sebagai berikut:

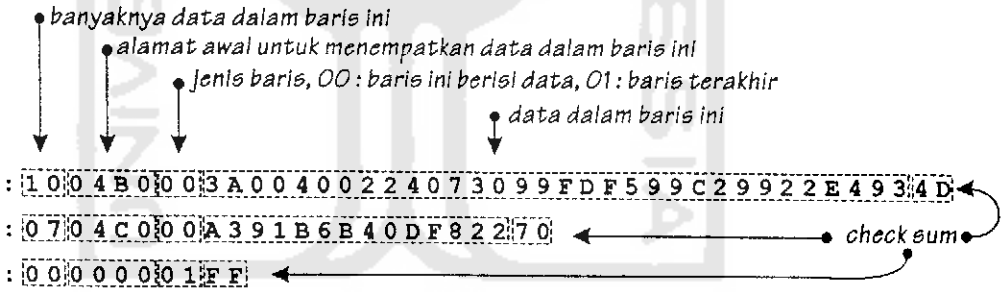
1. Buat nama folder baru didalam folder Delphi 7, kemudian ekstrak file Comport.Zip ke folder baru. File Comport.Zip dapat didownload pada www.sourceforge.net/projects/comport

2. Klik menu component kemudian klik install packages kemudian buka file CPortLib7.dpk, kemudian klik tombol "compile".
3. Kemudian buka file DsgnCPort7.dpk, kemudian klik tombol "compile" setelah dicompile kemudian klik tombol install.

2.4. Format HEX dari intel

Program *assembler* menyimpan kode mesin hasil terjemahannya ke dalam file teks dengan format khusus, format yang paling banyak dipakai assembler MCS51 adalah 'format HEX dari intel'.

File kode mesin dengan format HEX, merupakan baris-baris tulisan seperti terlihat dalam Gambar 2.9, *file* semacam itu bisa dibaca dengan *text editor* biasa, misalnya EDIT.COM dalam DOS, atau NOTEPAD dalam Windows.



Gambar 2. 9. Anatomi baris-baris dalam file format HEX Intel

Setiap baris mengandung informasi tentang berapa banyak data dalam baris tersebut, alamat awal tempat penyimpanan data dalam baris tersebut, jenis baris dan sarana untuk memastikan kebenaran data yang dinamakan sebagai *check sum*. Dalam baris tersebut, setiap huruf (kecuali huruf pertama) mewakili satu

bilangan heksadesimal, dengan demikian setiap 2 huruf membentuk data satu *byte* yang terdiri dari 2 bilangan heksadesimal.

Rincian dari format tersebut sebagai berikut :

1. Huruf pertama dalam baris, selalu berisi tanda ":", merupakan kode identitas yang menyatakan baris tersebut berisikan kode-kode mesin yang disimpan dalam format HEX dari Intel.
2. Huruf ke-2 dan ke-3 dipakai untuk menyatakan banyaknya data dalam baris yang dinyatakan dengan bilangan heksa-desimal, sehingga banyaknya data dalam 1 baris maksimal adalah 255 (atau heksadesimal FF).
3. Huruf ke 4 sampai 7, merupakan 4 buah bilangan heksa-desimal yang dipakai untuk menyatakan alamat awal tempat penyimpanan kode-kode dalam baris teks tersebut.
4. Huruf 8 dan 9 dipakai untuk menyatakan jenis teks data. Nilai 00 dipakai untuk menyatakan baris tersebut berisikan data biasa, 01 menyatakan baris tersebut merupakan baris terakhir.
5. Huruf ke 10 dan seterusnya adalah data. Setiap 2 huruf mewakili data 1 *byte*, sehingga jumlah huruf pada bagian ini adalah dua kali banyaknya data yang disebut pada butir 2 di atas.
6. Dua huruf terakhir dalam baris merupakan *check sum*. *Byte-byte* yang disebut dalam butir 2 sampai 5 di atas dijumlahkan, hasil penjumlahan dibalik (*inverted*) sebagai bilangan *check sum*. Hasil penjumlahan bisa menghasilkan nilai yang lebih besar dari 2 bilangan heksadesimal, namun hanya 2 bilangan heksadesimal yang bobotnya terkecil yang dipakai.