II. TINJAUAN PUSTAKA

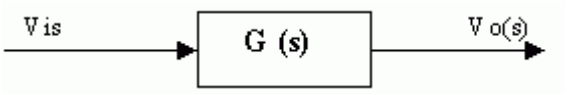
A. Sistem Kendali

Sistem Kendali adalah suatu sistem yang bertujuan untuk mengendalikan suatu proses agar output yang dihasilkan dapat dikontrol sehingga tidak terjadi kesalahan. Dalam hal ini *output* yang dikendalikan adalah kestabilan, ketelitian, dan kedinamisannya. Secara umum, sistem kendali dapat dibedakan menjadi dua jenis yaitu :

1. Sistem kendali untai terbuka
2. Sistem kendali untai tertutup

1. Sistem Kendali untai Terbuka

Sistem Kendali untai terbuka, outputnya tidak mempengaruhi input. Atau dengan kata lain sistem kendali untai terbuka *output* tidak dapat digunakan sebagai perbandingan umpan balik dengan *input*-nya. Akibatnya ketetapan dari sistem tergantung dari kalibrasi. Pada umumnya, sistem kendali untai terbuka tidak tahan terhadap gangguan luar. Di bawah ini adalah gambar diagram blok sistem kendali loop terbuka.



Gambar 1. Blok Diagram Sistem Kendali untai terbuka

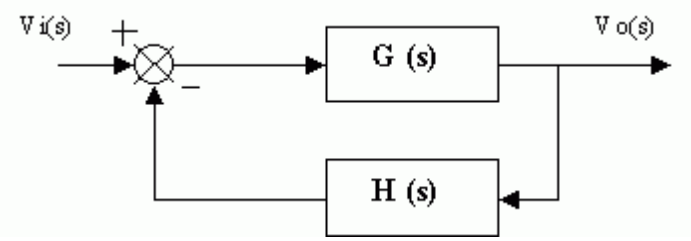
Fungsi alih sistem kendali untai terbuka adalah : Vo(s) = G(s) .Vi(s)

2. Sistem Kendali untai Tertutup

Sistem kendali untai tertutup seringkali disebut sistem kendali umpan balik. Pada sistem kendali untai tertutup, sinyal kesalahan yang bekerja, yaitu perbedaan antara sinyal input dan sinyal umpan balik di-*input*-kan ke pengendali sedemikian rupa untuk mengurangi kesalahan dan membawa keluaran sistem ke nilai yang dikehendaki. Pada umumnya sistem kendali untai tertutup tahan terhadap gangguan dari luar. Secara umum sistem kendali untai tertutup ini dibagi menjadi dua jenis, yaitu :

1. Sistem kendali kontinyu
2. Sistem kendali diskret

Secara umum gambar Sistem Kendali untai Tertutup adalah sebagai berikut :



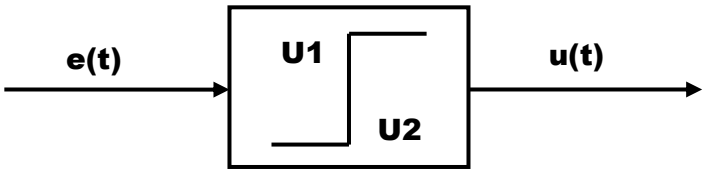
Gambar 2. Blok Diagram Sistem Kendali untai Tertutup

Fungsi alih sistem kendali untai tertutup adalah : Vo(s) / Vi(s) = G(s) / (1 + G(s).H(s))

Di mana G(s) : Fungsi alih sistem; H(s) : Fungsi alih.

B. Sistem Kendali On-off

Pada sistem kontrol ON-OFF, elemen pembangkit hanya memiliki dua posisi, yaitu ON dan OFF. Kontrol ON-OFF memiliki karakteristik sinyal keluaran dari kontroler u(t) tetap pada salah satu nilai maksimum atau minimum tergantung pada sinyal pembangkit kesalahan positif atau negatif. Diagram blok Sistem Kendali ON-OFF yang memiliki masukan e(t) dan keluaran u(t), ditunjukkan pada Gambar berikut.



Gambar 3. Blok Diagram Sistem Kendali ON-OFF

Aksi kontrol ON-OFF ditunjukkan pada persamaan berikut:

$$u(t) = \begin{cases} U1, & e(t) > 0 \\ U2, & e(t) < 0 \end{cases}$$

Persamaan di atas memiliki nilai U1 dan U2 yang konstan. Nilai minimum U2 dapat sebesar nol atau - U1. Pada sistem kontrol tertutup (*close loop*), sinyal e(t) merupakan sinyal kesalahan aktuasi (*error*) sebesar selisih antara sinyal *input* dengan sinyal umpan balik.

C. Slot Ekspansi ISA

ISA merupakan kependekan dari *Industry Standard Architecture*, yaitu suatu standardisasi bus pada komputer. Dalam ISA terdapat penyemat-penyemat yang berhubungan dengan bus-bus komputer. Di antaranya adalah penyemat

untuk data, control baca dan tulis, layanan *interrupt*. ISA mempunyai 8 bit bus data dan 20 bus alamat.

Walaupun belum ada spesifikasi baku dari ISA, namun ada dua hal yang menjadi acuan bus ISA hingga sekarang, yaitu:

1. Kecepatan bus ISA sebesar 8 MHz
2. Lebar data yang digunakan adalah 16 bit (2 byte).

Slot ISA sepenuhnya adalah 16 bit dan slot ISA 8 bit merupakan *subset* dari ISA 16 bit. Kartu ISA 8 bit bisa dipasang dan dioperasikan pada slot ISA 16 bit, tetapi sebaliknya slot ISA 16 bit tidak bisa dipasang pada slot ISA 8 bit. Hal ini karena slot ISA 8 bit terdiri atas 62 konektor (31 pin x 2 baris), sedangkan slot ISA 16 bit diperluas dari 62 (8 bit) menjadi 98 pin (31 pin x 2 baris + 18 pin x 2 baris).

ISA 8-bit

Bus ISA 8-bit merupakan varian dari bus ISA, dengan bus data selebar 8-bit, yang digunakan dalam IBM PC 5150 (model PC awal). Bus ini telah ditinggalkan pada sistem-sistem modern ke atas tetapi sistem-sistem Intel 286/386 masih memilikinya. Kecepatan bus ini adalah 4.77 MHz (sama seperti halnya prosesor Intel 8088 dalam IBM PC), sebelum ditingkatkan menjadi 8.33 MHz pada IBM PC/AT. Karena memiliki *bandwidth* 8-bit, maka *transfer rate* maksimum yang dimilikinya hanyalah 4.77 Mbyte/detik atau 8.33 Mbyte/detik. Meskipun memiliki *transfer rate* yang lamban, bus ini termasuk mencukupi kebutuhan saat itu, karena bus-bus I/O semacam *serial*

port, parallel port, controller floppy disk, controller keyboard dan lainnya sangat lambat. Slot ini memiliki 62 konektor.

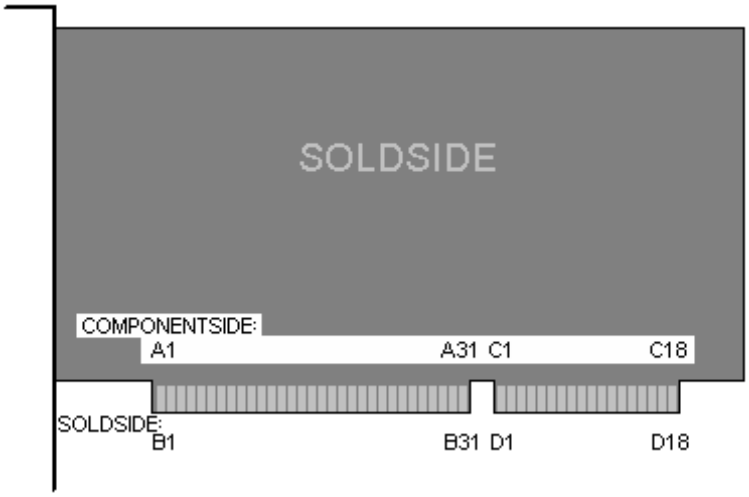
Meski desainnya sederhana, IBM tidak langsung mempublikasikan spesifikasinya saat diluncurkan tahun 1981, tetapi harus menunggu hingga tahun 1987, sehingga para manufaktur perangkat pendukung agak kerepotan membuat perangkat berbasis ISA 8-bit.

ISA 16-bit

Bus ISA 16-bit adalah sebuah bus ISA yang memiliki *bandwidth* 16-bit, sehingga mengizinkan *transfer rate* dua kali lebih cepat dibandingkan dengan ISA 8-bit pada kecepatan yang sama. Bus ini diperkenalkan pada tahun 1984, ketika IBM merilis IBM PC/AT dengan mikroprosesor Intel 80286 di dalamnya. Mengapa IBM meningkatkan ISA menjadi 16 bit adalah karena Intel 80286 memiliki bus data yang memiliki lebar 16-bit, sehingga komunikasi antara prosesor, memori, dan *motherboard* harus dilakukan dalam ordinal 16-bit. Meski prosesor ini dapat diinstalasikan di atas *motherboard* yang memiliki bus I/O dengan *bandwidth* 8-bit, hal ini dapat menyebabkan terjadinya *bottleneck* pada bus sistem yang bersangkutan.

Daripada membuat bus I/O yang baru, IBM ternyata hanya merombak sedikit saja dari desain ISA 8-bit yang lama, yakni dengan menambahkan konektor ekstensi 16-bit (yang menambahkan 36 konektor, sehingga menjadi 98 konektor), yang pertama kali diluncurkan pada Agustus tahun 1984, tahun yang sama saat IBM PC/AT diluncurkan. Ini juga menjadi sebab mengapa

ISA 16-bit disebut sebagai AT-bus. Hal ini memang membuat interferensi dengan beberapa kartu ISA 8-bit, sehingga IBM pun meninggalkan desain ini, ke sebuah desain di mana dua slot tersebut digabung menjadi satu slot.

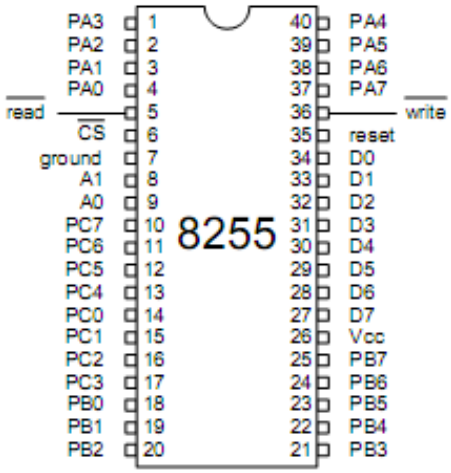


Gambar 4. Konfigurasi pin *male* kartu komputer

Gambar 5. Konfigurasi pin female slot ISA 16-bit

D. PPI 8255 (*PROGRAMMABLE PERIPHERAL INTERFACE 8255*)

PPI 8255 adalah IC yang dirancang untuk membuat port masukan dan keluaran paralel, IC ini mempunyai 24 bit I/O yang terorganisasi menjadi tiga port 8 bit (24 jalur) dengan nama Port A, Port B, dan Port C.



Gambar 6. Pin *IC* PPI 8255

Konfigurasi dari 24 jalur I/O ini bisa digunakan untuk masukan, keluaran, ataupun *biderectional* (dua arah). Pada I/O, yang dikontrol secara *software* akan lebih mudah bila dibandingkan dengan pengontrolan secara *hardware*.

1. Deskripsi fungsi 8255

a. Data bus buffer

Buffer bidirectional theree state ini digunakan untuk antar muka 8255 ke sistem bus data,data dikirim dan diterima oleh *buffer* berdasarkan eksekusi *input* atau *output* dari CPU. *Control Word* dan *information status* juga dikirimkan melalui *buffer* data bus.

b. *Read*/*Write* dan kontrol logika.

Fungsi dari blok ini adalah untuk mengatur semua pengiriman baik internal maupun eksternal dari data dan *Control Word*. Blok ini menerima *input* dari alamat CPU dan bus kontrol dan selanjutnya blok ini mengirimkan perintah ke kedua *group* kontrol.

c. *Chip Select*

Chip Select, jika *logika low* pada pin input ini maka komunikasi antara 8255 dan CPU akan *enable*.

d. *Read*

Read, jika *logika low* pada pin input ini maka 8255 akan mengirimkan data atau status informasi ke CPU pada bus data.

e. *Write*

Jika *Logika low* pada pin input ini maka CPU dapat menulis data atau kata kontrol ke 8255.

f. A0 dan A1

Port select 0 dan *port select 1*,sinyal input ini berhubungan dengan input RD dan WR, mengontrol pemilihan satu dari tiga port atau register kontrol pin tersebut umumnya dihubungkan ke *least significant bit* dari *bus addres* (A0 dan A1).

g. *Reset*

Logika *high* pada input pin ini akan menyebabkan *reset* pada *register control* dan semua port (A,B,C) akan berfungsi dalam mode *input*.

h. Port A, B dan C

8255 terdiri atas tiga buah port 8 bit (A, B dan C). semuanya dapat dikonfigurasikan dalam berbagai variasi fungsi bergantung pada sistem software yang diberikan.

Port A (PA0-PA7). 8 bit data *Output latch buffer* dan 8 bit data *input latch*.

Port B (PB0-PB7). 8 bit data *Output latch buffer* dan 8 bit data *input latch*.

Port C (PC0-PC7). 8 bit data *Output latch buffer* dan 8 bit data *input latch*.

Tiap 4 bit port terdiri atas 4 bit *latch* dan dapat digunakan untuk sinyal output kontrol dan sinyal input status.

2. Control Group

Control group dibagi menjadi dua *group,* yaitu *group* A dan *group* B. *Group* tersebut menerima Read/Write Control.

Control group A digunakan untuk :

a. Mengatur port A yang bisa diseting sebagai *input/output latch buffer*
b. Mengatur 4 upper bit (C4..C7), port C sebagai *input buffer* atau *output latch/buffer* jika bekerja pada mode 0.
c. Mengatur 4 upper bit (C4..C7), port C sebagai *control group* A jika bekerja pada mode 1 atau 2.

Control group B digunakan untuk :

a. Mengatur port B yang bisa di-*setting* sebagai *latch buffer input/output*
b. Mengattur 4 lower bit (C0..C3), port C sebagai *input buffer* atau *output latch/ buffer* jika bekerja pada mode 0
c. Mengatur 4 lower bit (C0..C3), port C sebagai *control group* B jika bekerja pada mode 1 atau 2

Tabel 1. Operasi dasar PPI 8255

A1	**A0**	**RD**	**WR**	**CS**	**Operasi Read/Write**
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	1	1	0	0	Data Bus → Port C
x	x	x	x	1	Data Bus → Three State
1	1	0	1	0	Illegal Condition
x	x	1	1	0	Data Bus → Three State

3. Mode/Protokol komunikasi

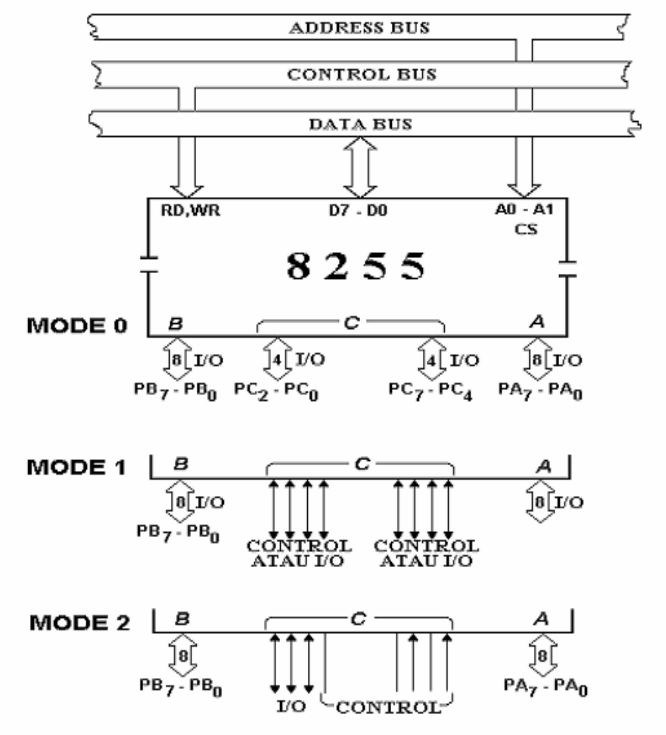
Transfer data pada PPI 8255 dibagi menjadi tiga protokol komunikasi:

1. Mode 0 (*Simple protocol*)/*Basic input-output*
2. Mode 1 (*Single handshaking protocol*)/*Strobed input-output*
3. Mode 2 (Double handshaking protocol)/Bi-directional bus

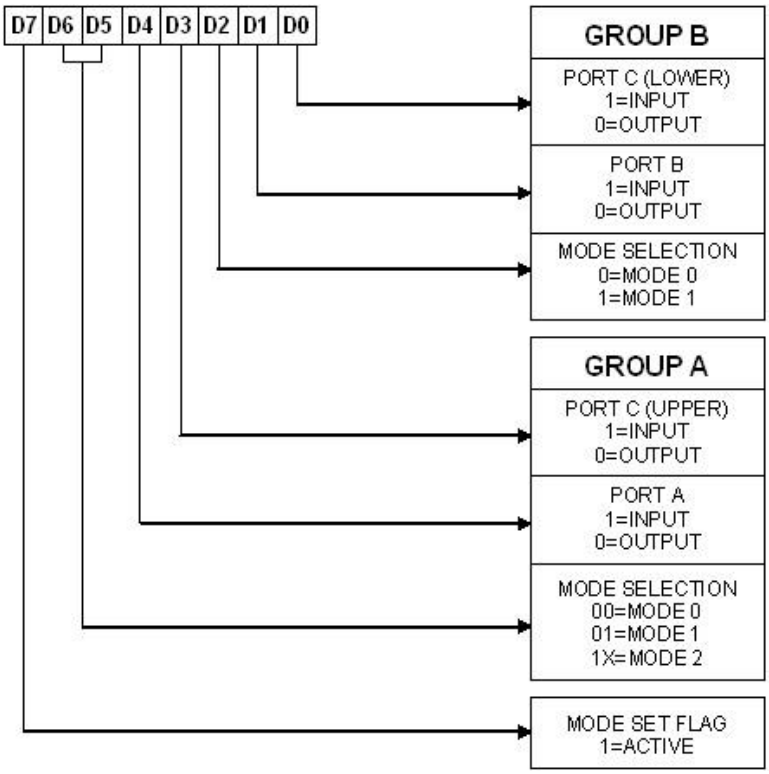
Transfer data pada mode 0 tidak memerlukan sinyal khusus yang menandakan apakah telah terjadi transfer data atau belum. Semua Port I/O dipakai sebagai Input dan Output. Tidak diperlukan sinyal "*Handshake*". Data langsung ditulis atau dibaca dari port yang bersangkutan.

Fungsi dasar dari 8255 mode 0 adalah :

1. Dua Port-8 bit (Port A & B) serta 2 Port - 4 bit (Port C).
2. Setiap Port dapat dipakai sebagai *input* atau *output*.
3. *Output* di-*latch* (ditahan); *input* tidak di-*latch* (ditahan).
4. Menyediakan 16 kombinasi konfigurasi *input/output* pada mode ini.



Gambar 7. Group *Control* PPI 8255



Gambar 8. Format *Control Word* PPI 8255

Control Word Port digunakan untuk inisialisasi awal yang menentukan PPI 8255 bekerja pada mode 0, 1 dan 2. *Control Word* juga menentukan port-port mana saja yang digunakan sebagai *input* dan *output* serta sebagai sinyal pengendali.

4. Set/Reset Bit

Pada PPI 8255 terdapat port untuk set dan reset sebuah bit, di mana jika terjadi Set atau Reset hanya salah satu port yang dipakai pada Port C.

Contoh:

1. Jika Port C saat ini datanya adalah FFH (1111 1111), jika kita akan me-*reset* Port C5 (PC5) maka port C hasilnya adalah EFH (1101 1111).
2. Jika Port C saat ini datanya adalah 1FH (0001 1111), jika kita akan me-*set* Port C7 (PC7) maka Port C hasilnya adalah 9FH (1001 1111).

E. Visual Basic 6.0

1. Pengertian Visual Basic 6.0

Visual Basic 6.0 merupakan salah satu bahasa pemrograman yang dapat digunakan untuk menyusun dan membuat program aplikasi pada lingkungan sistem operasi Windows. Dengan menggunakan Visual Basic 6.0, kemampuan Windows dapat dimanfaatkan secara optimal. Kecanggihan yang dimiliki oleh Visual Basic 6.0 akan menjadikan betapa mudahnya menyusun program aplikasi dengan tampilan grafis yang menawan dalam waktu yang relatif singkat. Program

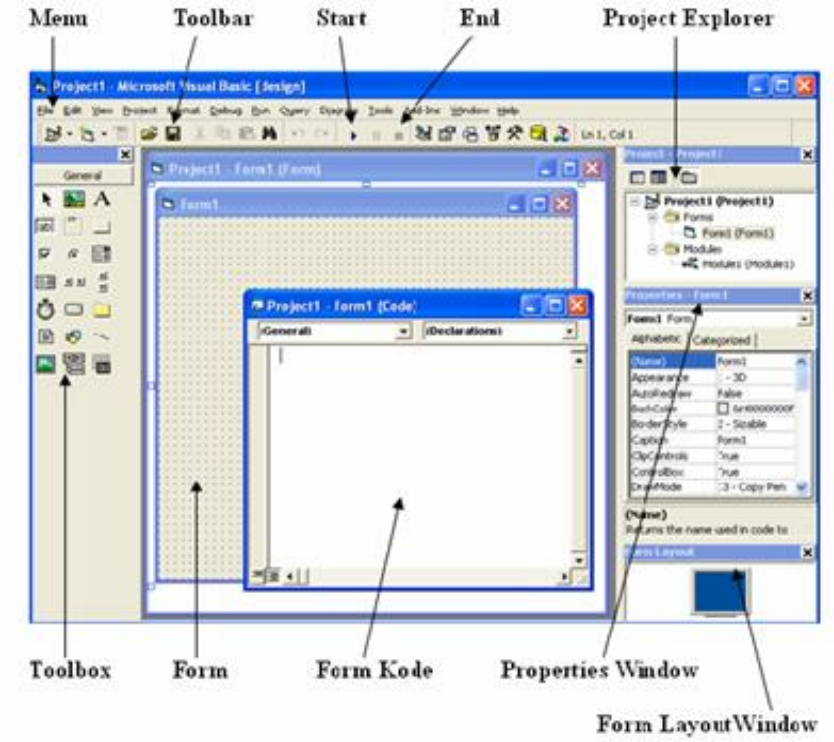
aplikasi dapat berupa program *database*, program grafis, program kendali, dan lain sebagainya. Di dalam Visual Basic 6.0 sudah terdapat komponen-komponen yang sangat membantu pembuatan program aplikasi.

Beberapa keuntungan menggunakan Visual Basic 6.0 daripada bahasa pemrograman yag lain di antaranya :

1) Tampilan grafis (*under Windows*) sehingga lebih "bersahabat".
2) Cara pemrograman relatif lebih mudah sehingga cocok untuk segala tingkat programer.
3) Hubungan dengan perangkat luar (*hardware*) tidak begitu rumit sehingga cukup mudah untuk meng-implementasikan sebagai pengendali peralatan elektronik.

2. IDE Visual Basic 6.0

Langkah pertama dalam membuat program aplikasi dengan Visual Basic 6.0 adalah membuat sebuah *project*. Pembuatan sebuah project dapat dilakukan dengan beberapa cara, di antaranya dengan meng-klik **Start | Program | Microsoft Visual Studio 6.0 | Microsoft Visual Basic 6.0**. Setelah itu akan terlihat tampilan pilihan jenis *New Project*, pilih *Standart EXE* maka akan terlihat tampilan *IDE (Integrated Development Environment)* Visual Basic 6.0.



Gambar 9. Tampilan IDE Visual Basic 6.0

1) Menu

Visual Basic mempunyai tiga belas menu dan masing-masing menu mempunyai fungsi yang berbeda.

2) Toolbar

Toolbar mempunyai fungsi yang sama dengan menu, hanya saja berupa icon-icon gambar dan digunakan sebagai jalan pintas.

3) Toolbox

Toolbox merupakan tempat kontrol-kontrol yang akan digunakan untuk membantu pembuatan program aplikasi.

4) Project Explorer

Project Explorer merupakan tempat yang digunakan untuk melihat daftar *forms*, *modules*, *class modules*, dan *designers*.

5) Properties Window

Properties Window berfungsi untuk mengatur properti dari setiap objek kontrol atau form. Pada Properties Window semua objek kontrol dapat diatur karakteristiknya.

6) Form Layout Window

Form layout window berfungsi untuk melihat atau mengetahui posisi tampilan form saat program dijalankan.

7) Form Objek

Form objek digunakan untuk menempatkan atau meletakkan objek dari kontrol-kontrol yang akan digunakan untuk merancang dan membuat program aplikasi.

8) Form Kode

Form kode digunakan sebagai tempat untuk menulis kode-kode program aplikasi.

F. Bahasa Pemrograman pada Visual Basic 6.0

1. Variabel

Setiap melakukan pemrograman, akan selalu memerlukan tempat penyimpanan data, misalnya untuk menampung data hasil perhitungan, menampung data hasil pembacaan register, atau lainnya. Tempat penyimpanan data itu dinamakan Variabel yang merupakan pointer yang menunjuk pada alamat memori fisik tertentu di komputer.

Dalam penggunaannya variabel harus mempunyai nama dan tipe data

tertentu. Nama variabel menunjuk pada suatu tempat pada memori komputer, sedangkan tipe data mengontrol besarnya memori yang disediakan untuk variabel tersebut. Berikut ini adalah tipe data pada Visual Basic beserta ukuran byte dan range tipe data tersebut.

Tabel 2. Tipe data pada *Visual Basic*

Tipe Data	**Ukuran byte**	**Range**
Integer	2 byte	-32768 s.d. 32767
Long	4 byte	-2.147.483.648 s.d. 2.147.483.647
Single	4 byte	Negatif: -3.402823E38 s.d. -1.401298E-45 Positif: 1.401298E-45 s.d. 3.402823E38
Double	8 byte	Negatif: -1.79769313486232E308 s.d. -4.94065645841247E-324 Positif: 4.94065645841247E-324 s.d. 1.79769313486232E308
Currency	8 byte	-922337203685477.5808 s.d. 922337203685477.5807
String	1 byte per karakter	0 s.d. 2 milyar karakter
Byte	1 byte	0 s.d. 255
Boolean	2 byte	True (Benar) atau False (salah)
Date	8 byte	1 Januari 100 s.d. 31 Desember 9999
Object	4 byte	Referensi objek
Variant	16 byte + 1 byte per karakter	Null, error, dan seluruh tipe data yang lain (Boolean, numeric, string, objek, array)

Pada Visual basic terdapat dua cara untuk mendeklarasikan sebuah variabel, yaitu dengan cara deklarasi eksplisit dan cara deklarasi implisit. Deklarasi eksplisit menggunakan pernyataan “Dim” diikuti nama dan tipe datanya, sedangkan deklarasi implisit menggunakan simbol di belakang nama variabel yang mempresentasikan tipe data yang digunakan.

Berikut ini adalah contoh deklarasi eksplisit :

Dim Text As String

Contoh deklarasi implisit :

Tabel 3. Deklarasi Implisit

Tipe data	Simbol karakter	Contoh pemakaian
Integer	%	Angka% = 100
Long Integer	&	Angka& = 2147483647
Single	!	Angka! = 2147483647000
Double	#	Konstanta_Pi# = 3.1415926535
Currency	@	saldo@ = 1000.50
String	$	Nama$ = “Awan”

Pada Visual Basic juga terdapat Konstanta yang merupakan variabel tetapi nilainya tetap. Dengan konstanta, kode program yang dibuat akan lebih mudah dibaca dan mencegah penulisan yang salah pada kode program yang dibuat. Visual Basic telah menyediakan konstanta-konstanta siap pakai yang dalam penamaannya diawali dengan karakter “vb”, contoh vbRed yang merupakan konstanta untuk warna merah.

2. Kontrol Program

Dengan kontrol program, alur eksekusi program dapat dikendalikan serta dapat menentukan keputusan apa yang harus dikerjakan oleh program pada kondisi tertentu. Kontrol program pada Visual Basic meliputi kontrol pertimbangan kondisi dan keputusan, kontrol pengulangan serta kontrol penyaluran alternatif. Beberapa kontrol program pada Visual Basic yang digunakan pada pemrograman ini :

1) If ... Then

Pernyataan ini mengetes suatu kondisi berdasarkan syarat kondisi kemudian menentukan suatu tindakan jika kondisi tersebut dipenuhi yang berupa pernyatan.

If <syarat kondisi> Then <pernyataan>

End If

2) If ... Then ... Else

Pernyataan ini hampir sama dengan If ... Then ..., yaitu digunakan untuk mengetes suatu kondisi tertentu. Hanya saja, jika suatu kondisi tidak terpenuhi, maka alur program akan mengeksekusi pernyataan yang lain kemudian menentukan suatu tindakan jika salah satu kondisi tersebut terpenuhi.

If <syarat kondisi 1> Then <pernyataan pertama>

ElseIf <syarat kondisi 2> Then <pernyataan kedua>

-

ElseIf <syarat kondisi n> Then <pernyataan ke-n>
Else <pernyataan>

End If

3) Select ... Case

Pada dasarnya perintah ini sama dengan perintah *If ... Then ... Else*, yaitu akan mengeksekusi satu blok pernyataan dari beberapa pilihan blok pernyataan. Hanya saja penulisannya lebih ringkas dan lebih mudah dimengerti.

Select Case <kondisi yang diuji>

Case <syarat kondisi 1>

<blok pernyataan pertama>

Case <syarat kondisi 2>

<blok pernyataan kedua>

-

Case Else

<blok pernyataan ke-n>

End Select

4) Do ... Loop

Perintah *Do ... Loop* digunakan untuk perulangan suatu blok pernyataan sampai dipenuhinya syarat kondisi yang ditetapkannya.

Do

<blok pernyataan>

Loop Until <syarat kondisi>

5) For ... Next

Perintah ini sama dengan melakukan perulangan seperti perintah *Do ... Loop*, tetapi dengan *For ... Next* bisa ditentukan nilai awal dan nilai akhir perulangan serta nilai kenaikannya.

For <nama_variabel> = <nilai awal> To <nilai akhir>

<blok pernyataan>

Next <nama_variabel>

3. Prosedur

Pembuatan program akan lebih mudah dengan memecah program menjadi blok-blok komponen yang lebih kecil yang disebut Prosedur. Pengaturan ini sangat berguna ketika ada bagian program yang sering melakukan tugas yang sama berulang-ulang atau bermaksud membagikannya pada program yang lain.

1) Sub Procedure

Salah satu jenis prosedur yang ada di dalam Visual Basic adalah *Sub Procedure*. *Sub Procedure* adalah blok kode yang dijalankan

sebagai tanggapan atas terbentuknya even, baik even itu merupakan even pemanggilan dari prosedur lain maupun even yang terjadi dari pemakaian program, misal even penekanan tombol kiri mouse.

[Private/Public] [Static] Sub

<nama_prosedur> (argumen)

<blok pernyataan>

End Sub

Setiap kali prosedur dipanggil, blok pernyataan yang ada di antara Sub dan End Sub akan dijalankan.

2) Operator

Operator digunakan pada Visual Basic untuk memanipulasi data maupun untuk melakukan perhitungan. Operator pada Visual Basic dapat dikelompokkan menjadi tiga kelompok yaitu operator matematik, operator perbandingan dan operator logika.

a) Operator Matematik

Operator matematik digunakan untuk melakukan perhitungan matematik.

Tabel 4. Operator matematik

Operator	Operasi	Contoh pemakaian
^	Pemangkatan	Nilai% = 2^2 ‘menghasilkan 4
-	Tanda negatif	Nilai% = -5 ‘menghasilkan negatif 5
*, /	Perkalian dan pembagian	Nilai% = (2*3)/6 ‘menghasilkan 1
\	Pembagian integer	
Mod	Modulus (sisa Pembagian)	Nilai% = 10 Mod 5 ‘menghasilkan 2
+, -	Penambahan dan Pengurangan	Nilai% = 6 + 2 - 4 ‘menghasilkan 4
&	Penggabungan string	Teks$ = “aw” & “an” ‘menghasilkan “awan”

b) Operator Perbandingan

Operator perbandingan digunakan untuk membandingkan suatu ekspresi dengan ekspresi yang lain dan akan menghasilkan nilai Boolean (*False* atau *True*).

Tabel 5. Operator perbandingan

Operator	Operasi	Contoh pemakaian
=	Sama dengan	Nilai = (1 + 2) = 3 ‘menghasilkan True
<>	Tidak sama dengan	Nilai = (1 + 2) <> 3 ‘menghasilkan False
<	Lebih kecil	Nilai = 2 < 3 ‘menghasilkan True
>	Lebih besar	Nilai = 2 > 3 ‘menghasilkan False
<=	Lebih Kecil atau sama dengan	Nilai = 2 <= 3 ‘menghasilkan True
>=	Lebih besar atau sama dengan	Nilai = 2 >= 3 ‘menghasilkan False
Like	Mempunyai cirri yang sama	Nilai = “abba” Like “a*” ‘True Nilai = “abba” Like “a??a” ‘True Nilai = “abba” Like “a?a” ‘False Nilai = “a” Like “[a-z]” ‘True
Is	Mempunyai referensi obyek yang sama	Nilai = Command1 Is Label1

c) Operator Logika

Operator logika biasanya digunakan untuk mengekspresikan satu atau lebih ekspresi logika yang akan menghasilkan nilai Boolean.

Tabel 6. Operator logika

Operator	Keterangan	Tabel Kebenaran Operasi	
		Operan	Hasil
Not	Akan menghasilkan nilai kebalikan dari nilai operan	Not True	False
		Not False	True
And	Akan menghasilkan true jika kedua operan-nya berlogika true	True And True	True
		True And False	False
		False And True	False
		False And False	False

Tabel 6. (lanjutan)

Operator	Keterangan	Tabel Kebenaran Operasi	
		Operan	Hasil
Or	Akan menghasilkan True jika salah satu operan-nya berlogika True	True Or True	True
		True Or False	True
		False Or True	True
		False Or False	False
Xor	Akan menghasilkan True jika operan-nya berlogika berbeda	True Xor True	False
		True Xor False	True
		False Xor True	True
		False Xor False	False
Eqv	Akan menghasilkan True jika operan-nya berlogika sama	True Eqv True	True
		True Eqv False	False
		False Eqv True	False
		False Eqv False	True

G. Pengaksesan Port Paralel Komputer Pada Visual Basic 6.0

Pada Visual Basic semua permintaan pengaksesan *hardware* harus melalui Windows dengan menggunakan program eksternal. Program eksternal itu adalah sebuah file DLL (*Dynamic Link Library*), dalam program ini penulis menggunakan sebuah file DLL dengan nama **io.dll**. Untuk menggunakannya, file DLL ini harus diletakkan di direktori //windows/system32 atau diikutkan dalam satu folder program.

Cara penggunaannya adalah sebagai berikut:

File DLL dideklarasikan ke dalam Visual Basic. Pendeklarasiannya adalah sebagai berikut :

Public Declare Sub PortOut Lib "io.dll" _

(ByVal Port As Integer, ByVal Data As Byte)

Public Declare Sub PortWordOut Lib "io.dll" _

(ByVal Port As Integer, ByVal Data As Integer)

```
Public Declare Sub PortDWordOut Lib "io.dll" _
(ByVal Port As Integer, ByVal Data As Long)

Public Declare Function PortIn Lib "io.dll" _
(ByVal Port As Integer) As Byte

Public Declare Function PortWordIn Lib "io.dll" _
(ByVal Port As Integer) As Integer

Public Declare Function PortDWordIn Lib "io.dll" _
(ByVal Port As Integer) As Long

Public Declare Sub SetPortBit Lib "io.dll" _
(ByVal Port As Integer, ByVal Bit As Byte)

Public Declare Sub ClrPortBit Lib "io.dll" _
(ByVal Port As Integer, ByVal Bit As Byte)

Public Declare Sub NotPortBit Lib "io.dll" _
(ByVal Port As Integer, ByVal Bit As Byte)

Public Declare Function GetPortBit Lib "io.dll" _
(ByVal Port As Integer, ByVal Bit As Byte) As Boolean

Public Declare Function RightPortShift Lib "io.dll" _
(ByVal Port As Integer, ByVal Val As Boolean) As Boolean

Public Declare Function LeftPortShift Lib "io.dll" _
```

(ByVal Port As Integer, ByVal Val As Boolean) As Boolean

Public Declare Function IsDriverInstalled Lib "io.dll" () As Boolean

Selanjutnya penggunaan fungsi dan prosedur sebagai berikut:

fungsi *PortIn* membutuhkan dua parameter yaitu alamat perangkat keras dan variabel hasil pembacaan data dari perangkat keras dengan tipe data byte. Sedangkan prosedur *PortOut* membutuhkan dua parameter juga, yaitu alamat perangkat keras dan nilai atau variabel yang menyimpan nilai yang akan dikirimkan ke perangkat keras yang bersangkutan.

PortOut	Mengirim data dalam format byte (8-bit) ke port tertentu.
PortWordOut	Mengirim data dalam format word (16-bit) ke port tertentu.
PortDWordOut	Mengirim data dalam format double word (32-bit) ke port tertentu.
PortIn	Membaca data dalam format byte (8-bit) dari port tertentu.
PortWordIn	Membaca data dalam format word (16-bit) dari port tertentu.
PortDWordIn	Membaca data dalam format double word (32-bit) dari port tertentu.
GetPortBit	Membaca status dari bit tertentu.
SetPortBit	Set bit (=1) pada port tertentu.
ClrPortBit	Reset bit (=0) pada port tertentu.
NotPortBit	Lakukan inversi (NOT) bit pada port tertentu
RightPortShift	Geser bit dari port tertentu ke kanan, LSB MSB.
LeftPortShift	Geser bit dari port tertentu ke kiri, MSB LSB.

IsDriverInstalled Akan memberikan nilai bukan-NOL jika **io.dll** sudah terinstal dan berfungsi. Tujuan utama dari fungsi ini adalah untuk memastikan bahwa penggerak mode kernel pada NT/2000/XP telah diinstal dan dapat diakses.

Untuk mengirimkan data ke port paralel, digunakan fungsi PortOut.

Sintak penggunaannya adalah sebagai berikut:

PortOut [Alamat_Port], [Nilai]

Perintah di atas membutuhkan dua parameter, yaitu Alamat_Port dan Nilai yang merupakan alamat port dan nilai data yang ingin dikirimkan ke port tersebut.

Untuk menerima data dari port parallel, digunakan fungsi PortIn.

Sintak penggunaannya adalah sebagai berikut:

Variabel Hasil Pembacaan= PortIn([Alamat_Port])

Perintah di atas membutuhkan yaitu alamat perangkat keras dan variabel hasil pembacaan data dari perangkat keras dengan tipe data byte.