JILID 3

Agus Putranto, dkk.

Teknik Otomasi Industri

untuk

Sekolah Menengah Kejuruan

Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

Agus Putranto dkk

TEKNIK OTOMASI INDUSTRI

SMK

JILID 3

Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

TEKNIK OTOMASI INDUSTRI

Untuk SMK

JILID 3

Penulis : Agus Putranto
Abdul Mukti
Djoko Sugiono
Syaiful Karim
Arie Eric Rawung
Sodikin Susa'at
Sugiono

Perancang Kulit : TIM

Ukuran Buku : 18,2 x 25,7 cm

PUT t PUTRANTO, Agus
Teknik Otomasi Industri untuk SMK Jilid 3 /oleh Agus Putranto, Abdul Mukti, Djoko Sugiono, Syaiful Karim, Arie Eric Rawung, Sodikin Susa'at, Sugiono ---- Jakarta : Direktorat Pembinaan Sekolah Menengah Kejuruan, Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah, Departemen Pendidikan Nasional, 2008.
xii, 232 hlm
Daftar Pustaka : LAMPIRAN A.
Glosarium : LAMPIRAN B.
ISBN : 978-979-060-089-8
ISBN : 978-979-060-092-8

Diterbitkan oleh
Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional
Tahun 2008

KATA SAMBUTAN

Puji syukur kami panjatkan kehadirat Allah SWT, berkat rahmat dan karunia Nya, Pemerintah, dalam hal ini, Direktorat Pembinaan Sekolah Menengah Kejuruan Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Departemen Pendidikan Nasional, pada tahun 2008, telah melaksanakan penulisan pembelian hak cipta buku teks pelajaran ini dari penulis untuk disebarluaskan kepada masyarakat melalui *website* bagi siswa SMK.

Buku teks pelajaran ini telah melalui proses penilaian oleh Badan Standar Nasional Pendidikan sebagai buku teks pelajaran untuk SMK yang memenuhi syarat kelayakan untuk digunakan dalam proses pembelajaran melalui Peraturan Menteri Pendidikan Nasional Nomor 12 tahun 2008.

Kami menyampaikan penghargaan yang setinggi-tingginya kepada seluruh penulis yang telah berkenan mengalihkan hak cipta karyanya kepada Departemen Pendidikan Nasional untuk digunakan secara luas oleh para pendidik dan peserta didik SMK di seluruh Indonesia.

Buku teks pelajaran yang telah dialihkan hak ciptanya kepada Departemen Pendidikan Nasional tersebut, dapat diunduh (*download*), digandakan, dicetak, dialihmediakan, atau difotokopi oleh masyarakat. Namun untuk penggandaan yang bersifat komersial harga penjualannya harus memenuhi ketentuan yang ditetapkan oleh Pemerintah. Dengan ditayangkannya *soft copy* ini akan lebih memudahkan bagi masyarakat untuk mengaksesnya sehingga peserta didik dan pendidik di seluruh Indonesia maupun sekolah Indonesia yang berada di luar negeri dapat memanfaatkan sumber belajar ini.

Kami berharap, semua pihak dapat mendukung kebijakan ini. Selanjutnya, kepada para peserta didik kami ucapkan selamat belajar dan semoga dapat memanfaatkan buku ini sebaik-baiknya. Kami menyadari bahwa buku ini masih perlu ditingkatkan mutunya. Oleh karena itu, saran dan kritik sangat kami harapkan.

Jakarta, 17 Agusutus 2008
Direktur Pembinaan SMK

KATA PENGANTAR

Puji syukur kehadirat Alloh, dengan tersusunnya buku Teknik Otomasi Industri ini semoga dapat menambah khasanah referensi khususnya di bidang tekologi industri yang akhir-akhir ini mulai berkembang di Indonesia.

Isi buku ini sengaja disajikan secara praktis dan lengkap sehingga dapat membantu para siswa Sekolah Menengah Kejuruan (SMK), mahasiswa, guru serta para praktisi industri. Teknik Otomasi Industri yang selama ini dideskripsikan secara variatif dan adaptif terhadap perkembangan serta kebutuhan berbagai kalangan praktisi industri. Penekanan dan cakupan bidang yang dibahas dalam buku ini sangat membantu dan berperan sebagai sumbangsih pemikiran dalam mendukung pemecahan permasalahan yang selalu muncul didalam disain, pengendalian / pemgontrolan suatu sistem.

Oleh karena itu, buku ini disusun secara integratif antar disiplin ilmu yaitu teknik elektronika analog, elektronika daya,teknik digital, pemrograman dan elektronika daya yang saling mendukung sehingga skill yang diperlukan terkait satu dengan lainnya. Secara tuntas, kualitas maupun manajemen proses control standar yang berlaku di tingkat internasional termasuk didalam wilayah pembahasan.

Tim penulis mengucapkan terima kasih kepada berbagai pihak yang telah membantu materi naskah serta dorongan semangat dalam penyelesaian buku ini. Kami sangat berharap dan terbuka untuk masukan serta kritik konstruktif dari para pembaca sehingga dimasa datang buku ini lebih sempurna dan implementatif.

Tim Penulis

GET IT ON Google Play

Download di App Store

iOS segera hadir

Unduh buku lainnya melalui aplikasi. Gratis.

Buku BSE dilengkapi dengan daftar isi untuk memudahkan navigasi. Tersedia juga majalah, tabloid, buku dan koran yang lebih hemat hingga 80% dibanding edisi cetak.

Unduh aplikasi myedisi reader gratis

myedisi.com/reader

myedisi

Buku BSE terbaru belum tersedia di myedisi? Sampaikan melalui email **bse@myedisi.com**

DAFTAR ISI

BAB VIII MIKROKONTROLER

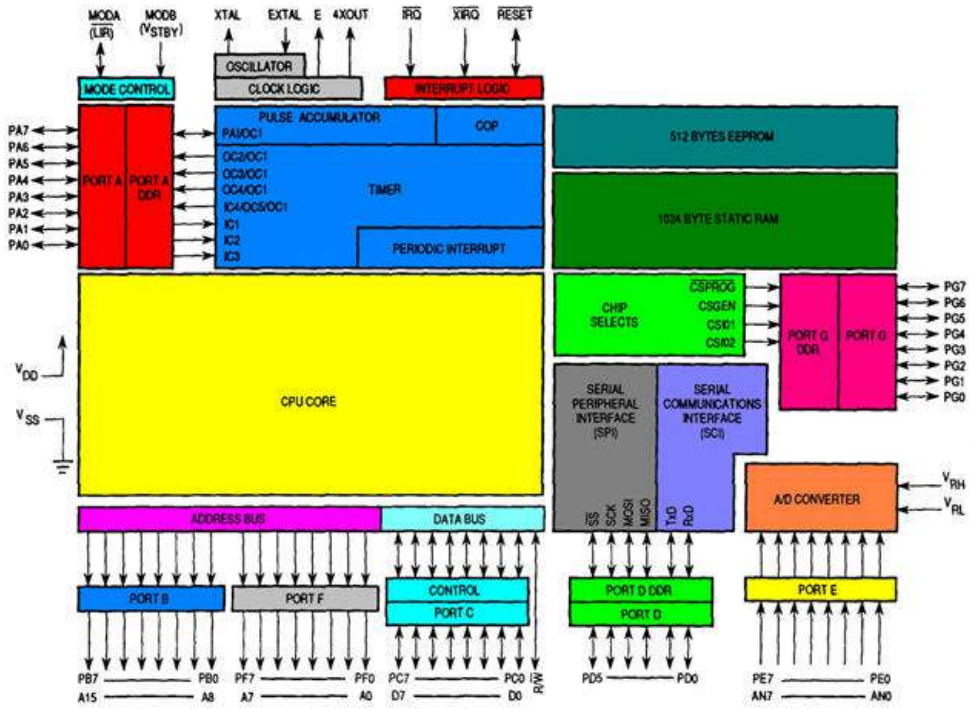
Micro Controller Unit (MCU) adalah sebuah chip mikrokomputer yang didalamnya terdapat mikroprosessor (CPU), memory RAM, ROM, EEPROM, I/O Port dan bahkan ADC-DAC serta beberapa fasilitas penunjang lainnya seperti misalnya Timer dan PWM yang sudah terintegrasi dalam satu IC.

Gambar 8.01 IC Mikrokontroller 68HC11F1

8.1. Mikrokontroller MC68HC11F1

Salah satu jenis mikrokontroller yang dibahas pada buku ini adalah MC68HC11F1 buatan Motorolla yang memiliki feature sebagai berikut :

- Sistim timer expanded 16 bit dengan empat tingkat prescaler yang dapat diprogram
- Serial Coomunication Interface (misalnya untuk RS232)
- Serilal Peripheral Interface (misalnya untuk LCD)
- Delapan masukan analog 8 bit ADC
- Enam Port Digital I/O 8 bit
- Block Protect Mechanism untuk EEPROM dan CONFIG
- Nonmultiplexed Expanded Bus
- 68 pin PLCC
- Power saving STOP dan STOP
- 64 K Memory Addressability
- 512 bytes EEPROM
- 1024 bytes RAM
- 8 bit Pulse Accumulator Circuit
- Bit Test dan instruksi percabangan
- Real-Time Interrupt
- Empat Programmable Chip Select
- Computer Operating Properly (COP) Watchdog system



Gambar 8.02 Blok Diagram MC68HC11F1

8.2. Mode Operasi dan Deskripsi Sinyal

8.2.1. Mode Operasi

MC68HC11F1 menyediakan fasilitas untuk memilih salah satu dari empat mode operasi. Ada dua mode operasi normal dan dua mode operasi khusus. Mode operasi normal yaitu mode sibgle-chip dan mode expanded yang tidak dimultiplex. Sedangkan mode operasi khusus yaitu bootstrap dan mode test.
Pemilihan mode dilakukan dengan mengatur logika masukan pada pin MODA dan MODB seperti pada table beikut ini.

Tabel 8.01 Mode Operasi MC68HC11

MODA	MODB	Mode Operasi
0	1	Single Chip
1	1	Expanded Nonmultiplexed
0	0	Special Bootstrap
1	0	Special Test

8.2.1.1. Mode Single-Chip

Gambar 8.03. Mikrokontroler Mode Single Chip

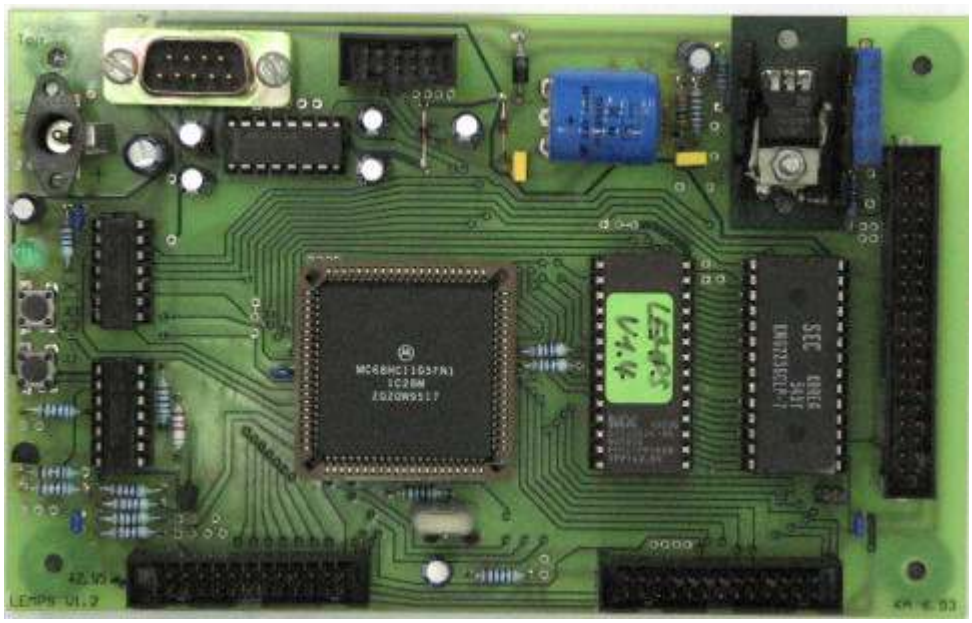
Pada mode ini, mikrokontroller bekerja terbatas sesuai dengan kemampuan yang tersedia pada satu chip mikrokontroller itu sendiri tanpa memiliki saluran alamat dan data keluar.
Semua kode program disimpan pada EEPROM sebesar 512 byte yang beralamatkan $FE00 - $FFFF.
Pada mode ini semua pin dapat dipergunakan sebagai input/output port dan semua aktivitas alamat serta data harus berada pada internal memory yang di dalam mikrokontroler.
Untuk fungsi-fungsi penggunaan yang mudah dan pembuatan program yang relatif kecil serta dapat ditampung sesuai kapasitas memori intern, mikrokontroler dapat dioperasikan atau dapat dibangun pada “Single Chip Mode”. Program dibuat dan ditempatkan pada EEPROM yang tersedia di dalam chip mikrokontroler itu sendiri sedangkan data temporernya disimpan pada internal RAM.
Pada fungsi ini, mikrontroller bekerja hanya dengan dirinya sendiri tidak dengan bantuan perangkat memori dan peripheral input output dari luar.

Sinyal masukan dan keluaran langsung disambungkan ke pin-pin pada PORT yang tersedia dalam IC mikrokontroler itu sendiri. Dengan demikian secara phisik suatu kontrol dengan mikrokontroler single chip tidak memerlukan tempat yang besar. Karena kapasitas memori yang tersedia pada single chip biasanya kecil, penggunaan mikrokontroler pada mode single chip menjadi sangat terbatas untuk program-program pendek saja.
Dengan adanya fungsi-fungsi seperti timer, watchdog-system, analog to digital conversion untuk keperluan masukan analog, interface untuk komunikasi data serial, port paralel dan serial serta port-port digital lainnya sebuah chip mikrokontroler dapat difungsikan sebagai ”Single

Chip", sedangkan untuk mikroprosessor tambahan fungsi-fungsi di atas tidak ditemukan.

8.2.1.2. Mode Expanded-Nonmultiplexed

Pada mode ini, mikrokontroller dapat mengakses alamat sampai 64 K byte.
Port B dan port F berubah fungsi sebagai saluran alamat satu arah keluar dan port C berfungsi sebagai saluran data dua arah.
Pin-pin pada port B menjadi saluran alamat orde tinggi dan pin-pin pada port F menjadi saluran alamat orde rendah.



Gambar 8.04 Mikrokontroler Mode Expanded Nonmultiplexed

Pin baca/tulis ($R/\overline{W}$) digunakan untuk mengontrol arah aliran pada saluran data port C.
Programable chip selects dapat menggunakan port G pada pin PG7 – PG4
Untuk fungsi-fungsi penggunaan yang besar, program dan data ditempatkan pada external memori dan untuk keperluan tersebut mikrokontroler beroperasi atau dibangun pada "Expanded Mode".
Pada mode ini akan terbentuk saluran data dan saluran alamat untuk keperluan perangkat tambahan luar seperti penambahan EPROM dan RAM dengan kapasitas yang jauh lebih besar.

Karena pada mode expand beberapa input output berubah berubah fungsi menjadi saluran data dan saluran alamat, sebagai konsekuensinya jumlah port input output akan berkurang dibandingkan apabila mikrokontroler ini dibangun sebagai single chip.
Untuk keperluan yang besar yang membutuhkan banyak port input output, kita dapat menggunakan saluran data dan alamat tersebut untuk disambungkan ke beberapa peripheral input utput sebanyak yang kita inginkan.
Dengan penambahan fungsi-fungsi dan fasilitas lainnya yang disimpan dalam EPROM yang besar di luar chip mikrokontroler.
Penggunaan mikrokontroler dengan mode expand menjadi sangat luas dan mempermudah dalam pembuatan program-program panjang baik untuk keperluan pelatihan maupun program-program aplikasi mikrokontroler untuk pengontroller mesin-mesin industri yang kompleks.

8.2.1.3. Mode Bootstrap

Mode khusus bootstrap ini sama dengan mode single-chip. Program resident bootloader mengijinkan pengisian program panjang kedalam RAM mikrokontroller memalui port SCI.
Kontrol program dilalukan ke RAM ketika medapatkan sedikitnya empat karakter yang sesuai. Pada mode ini semua vector interrupt dipetakan ke dalam RAM seperti dalam tabel sehingga jika diperlukan kita dapat mengatur lokasi loncat yang sesuai dengan table berikut ini.

Tabel 8.02. Bootstrap Mode Jump Vectors

8.2.1.3.1.1. Alamat	**Vector**
00C4	SCI
00C7	SPI
00CA	Pulse Accumulator Input Edge
00CD	Pulse Accumulator Overflow
00D0	Timer Overflow
00D3	Timer Output Compare 5 / Input Capture 4
00D6	Timer Output Compare 4
00D9	Timer Output Compare 3
00DC	Timer Output Compare 2
00DF	Timer Output Compare 1
00E2	Input Capture 3
00E5	Input Capture 2
00E8	Input Capture 1

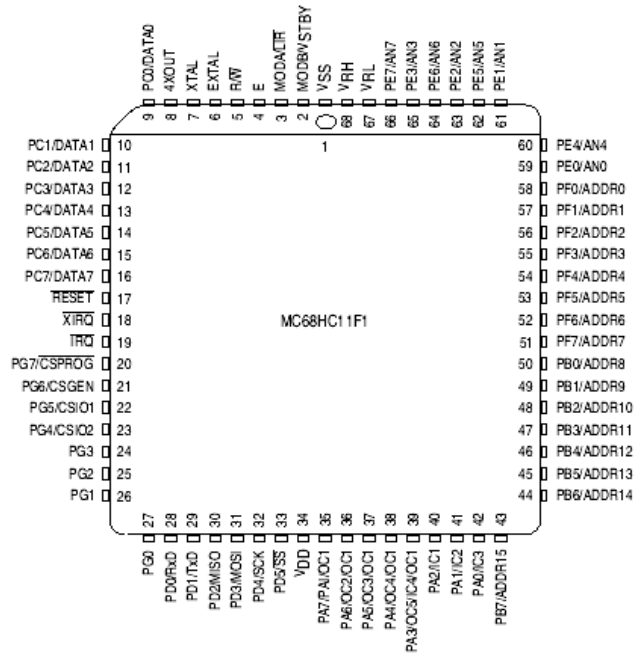
00EB	Real-Time Interrupt
00EE	$\overline{\text{IRQ}}$
00F1	$\overline{\text{XIRQ}}$
00F4	SWI
00F7	Illegal Opcode
00FA	COP Fail
00FD	Clock Monitor
BF00 (boot)	Reset

8.2.1.4. Mode Test

Mode expanded khusus ini pada umumnya dipergunakan untuk pengujian. Kadang digunakan pula untuk program kalibarsi dara personal kedalam internal EEPROM. 512 Byte EEPROM ini inisialnya disettidak aktif.
Kita dapat mengakses beberapa pengujian khusus. Reset dan interrupt vector diakses dari luar dari lokasi $BFC0 - $BFFF

8.2.2. Deskripsi Sinyal

Mikrokontroler MC68HC11F1 dalam kemasan Plastic leaded chip carrier (PLCC) terdiri 68 pin dengan sesusunan sebagai berikut :



Gambar 8.05 Pin IC MC68HC11G1

8.2.2.1. VDD dan VSS

Power supply disambungkan ke mikrokontroler m dua buah pin. V_{DD} disambung ke positiv 5 V (± 10 %) dan V_{SS} disambung ke ground (0 V). Mikrokontroler bekerja dengan tegangan power supply tunggal 5 volt nominal. Perubahan sinyal yang cepat pada pin power supply ini dapat menyebabkan kerusakan. Untuk mencegahnya gunakanlah power supply yang baik atau dapat pula dipasangkan kapasitor bypaas untuk meredam frekueni tinggi atau noise yang mungkin terjadi.

8.2.2.2. Reset ($\overline{\text{RESET}}$)

Adalah signal control dua arah aktif low yang dipergunakan untuk menginisialisasi mikrokontroler sebagai tanda awal kerja stsrt-up.

8.2.2.3. XTAL dan EXTAL

Pin-pin ini menyediakan interface suatu kristal atau suatu CMOS-compatible clock untuk menkontrol rangkaian generator clock internal. Frekuensi kristal yang diterapkan harus empat kali lebih tinggi dari frekuensi clok yang diinginkan.

8.2.2.4. E Clock

Pin ini menyediakan keluaran pulsa E clock yang dihasilkan oleh internal clock generator yang dapat dipergunakan sebagai timing referensi. Frekuensi keluaran E clock adalah seperempat dari freuensi kristal yang dipasang.

8.2.2.5. 4XOUT

Pin ini menyediakan keluaran pulsa yang telah diperkuat yang besarnya frekuensi adalah empat kali E clock. Keluaran pin ini dapat dipergunakan sebagai clock masukan bagi prosessor lain.

8.2.2.6. $\overline{\text{IRQ}}$

Negative edge-sensitive atau level- sensitive. Pemilihan ini dapat dilakukan dengan mngeset bit IRQE pada register OPTION dimana pin ini defaultnya setelah reset diset pada mode level- sensitive.

Pin ini adalah aktif low sehingga untuk rancang baunnya diperlukan sebuah resistor pull up yang dipasang antara pin ini dengan V_{DD}.
Pin ini menyediakan fasilitas interrupts asinkron yang dapat ditriger berdasarkan

8.2.2.7. $\overline{\text{XIRQ}}$

Pin ini menyediakan fasilitas nonmaskable interrupt. Setelah reset, bit X pada condition code register diset menutup beberapa interup sampai dibuka kembali melui software. Input pin interrupt ini adalah level-sensitive yang aktif low sehingga memerlukan resistor pull up yang siambungkan ke V_{DD}.

8.2.2.8. MODA/ $\overline{\text{LIR}}$ dan MODB/V_{STBY}

Pada saat reset pin ini berfungsi untuk memilih mode operasi, yitu dua mode operasi normal dan dua mode operasi khusus.

Keluaran $\overline{\text{LIR}}$ dapat dipergunakan sebagai salah satu cara debugging setelah proses reset selesai. Pin $\overline{\text{LIR}}$ ini adalah open-drain yang akan

berlogika low selama siklus pertama E-Clock pada setiap instruksi dan akan tetap bertahan selama siklus tersebut.
Masukan V_{STBY} dipergunakan untuk mempertahankan isi RAM selama tegangan power supply tidak ada.

8.2.2.9. V_{RL} dan V_{RH}

Pin ini dipergunakan untuk tegangan referensi
Analog To Digital Converter.

8.2.2.10. R/$\overline{W}$

Keluaran R/$\overline{W}$ dipergunakan untuk mengontrol arah aliran data pada saluran data external dalam mode operasi expanded-nonmultiplexed. Sinyal berlogika rendah pada pin ini menunjukkan bahwa data akan dituliskan pada saluran data external. Sedangkan sinyal berlogika tinggi pada pin ini menunjukkan bahwa proses pembacaan data sedang berlangsung.

Pada mode single-chip dan bootstrap pin R/$\overline{W}$ akan berlogika tinggi.

8.2.2.11. Port Input/Output

Pada IC Mikrokontroler MC68HC11F1 terdapat 54 pin input/output (I/O) yang terbagi dalam enam port yang masing-masing port terdiri dari 8 bit , yaitu Port A, Port B, Port C, Port E, Port F dan Port G. Sedangkan satu port lagi yaitu Port D terdiri dari enam bit.

Kebanyakan dari port tersebut memiliki banyak fungsi tergantung pada mode operasi yang dipilih. Tabel di bawah ini memperlihatkan fungsi setiap port sesuai dengan mode operasinya.

Tabel 8.03 Fungsi sinyal port

Port	Bit	Single-ChipBootstrap	ExpandedSpecal-test
A	0	PA0/IC3	PA0/IC3
A	1	PA1/IC2	PA1/IC2
A	2	PA2/IC1	PA2/IC1
A	3	PA3/IC4/OC5	PA3/IC4/OC5

A	4	PA4/OC4/OC1	PA4/OC4/OC1
A	5	PA5/OC3/OC1	PA5/OC3/OC1
A	6	PA6/OC2/OC1	PA6/OC2/OC1
A	7	PA7/PAI/OC1	PA7/PAI/OC1
B	0	PB0	A8
B	1	PB1	A9
B	2	PB2	A10
B	3	PB3	A11
B	4	PB4	A12
B	5	PB5	A13
B	6	PB6	A14
B	7	PB7	A15
C	0	PC0	D0
C	1	PC1	D1
C	2	PC2	D2
C	3	PC3	D3
C	4	PC4	D4
C	5	PC5	D5
C	6	PC6	D6
C	7	PC7	D7
D	0	PD0/RxD	PD0/RxD
D	1	PD1/TxD	PD1/TxD
D	2	PD2/MISO	PD2/MISO
D	3	PD3/MOSI	PD3/MOSI
D	4	PD4/SCK	PD4/SCK
D	5	PD5/$\overline{ss}$	PD5/$\overline{ss}$
E	0	PE0/AN0	PE0/AN0
E	1	PE1/AN1	PE1/AN1
E	2	PE2/AN2	PE2/AN2
E	3	PE3/AN3	PE3/AN3
E	4	PE4/AN4	PE4/AN4
E	5	PE5/AN5	PE5/AN5
E	6	PE6/AN6	PE6/AN6
E	7	PE7/AN7	PE7/AN7
F	0	PF0	PF0
F	1	PF1	PF1
F	2	PF2	PF2
F	3	PF3	PF3

F	4	PF4	PF4
F	5	PF5	PF5
F	6	PF6	PF6
F	7	PF7	PF7
G	0	PG0	PG0
G	1	PG1	PG1
G	2	PG2	PG2
G	3	PG3	PG3
G	4	PG4	PG4/CSIO2
G	5	PG5	PG5/CSIO1
G	6	PG6	PG6/CSGEN
G	7	PG7	PG7/$\overline{\text{CSPROG}}$

8.3. Memory dan Kontrol dan Register Status

8.3.1. Memory

MC68HC11F1 pada dasarnya mampu mengakses 64 K byte alamat memori external. Satu chip IC memiliki 1 K byte static RAM, 512 byte EEPROM dan 96 byte status dan code register. Gambar 1.6 berikut mengilustrasikan peta memory untuk semua mode operasi.

8.3.2. Pemetaan Memory Subsystems

Menggunkan register INIT, dapat dilakukan pemetaan untuk 96 byte blok register kontrol dan register status dserta 1K RAM static ke dalam 4K boundary di memory.

Setelah reset lokasi Ram adalah pada alamat $0000 sampai dengan $03FFF dan lokasi register berada pada alamat $1000 sampai dengan $105F.

EEPROM dapat di-enable-kan dengan mengeset bit EEON pada register CONFIG.

Pada mode expanded-nonmultiplexed dan special-test, alamat EEPROM ada pada lokasi memory $xE00 sampai dengan $xFFF, dimana x mewakili nilai dari empat bit tertinggi pada register CONFIG.

Pada mode single-chip, alamat EEPROM adalah pada $FE00 sampai dengan $FFFF sedangkan pada mode bootstrap ROM berada pada alamat $BF00 sampai dengan $BFFF pada saat perubahan ke mode bootstrap.

Pengalamatan memory seharusnya tidak boleh terjadi konflik, prioritas utama adalah blok regioster dan berikutnya adalah RAM. Sedangkan pada mode bootstrap ROM mendapatkan prioritas utama setelah itu baru EEPROM.

8.3.3. Control dan Status Register

Ada 96 byte ststus register yang digunakan untuk mengontrol operasi mikrokontroller. Alamat register ini dapat direlokasikan sebesar 4 K boundary di dalam internal RAM yang defaultnya setelah reset adalah $1000 - $105F. Tabel 1.3 berikut ini adalah daftar register serta alamat yang dipakai.

8.3.4. RAM dan I/O Mapping Register (INIT)

Register INIT adalah register 8 bit yang khusus dipergunakan untuk inisialisasi merubah default lokasi alamat RAM dan alamat register kontrol yang terdapat pada peta memory internal MCU. Perubahan ini hanya dapat dilakukan selama 64 siklus pertama E-clock setelah sinyal reset pada mode normal. Setelah itu register INIT menjadi register yang hanya dapat dibaca saja.

RAM and I/O Mapping

	Bit 7	6	5	4	3	2	1	Bit 0
$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG4	REG1	REG0
RESET:	0	0	0	0	0	0	0	1

RAM[3:0] — Internal RAM Map Position
REG[3:0] — 128-Byte Register Block Map Position

Gambar 8.06 RAM dan I/O Mapping Register

Sejak register INIY diset &1 oleh reset, default alamat awal RAM adalah $0000 dan blok register kontrol dan register status berawal pada alamat $1000. RAM[3:0] khusus untuk mengatur alamat awal 1KByte RAM dan REG[3:0] khusus untuk mengatur alamat blok register kontrol dan register status. Dalam hal ini kombinasi dari empat bit RAM dan REG menjadi empat bit terbesar (Most Significant Bit MSB) dari 16-bit alamat RAM atau register yang ditulisi. Pada contoh di atas, register INIT diset dengan nilai $01 menunjukkan bahwa alamat awal blok register kontrol dan register status diatur pada posisi $1000. Sedangkan alamat awal RAM pada posisi $0000. Berikut ini adalah daftar register kontrol dan register status dengan alamat awal $1000

Tabel 8.04 Daftar register kontrol dan register status

	Bit 7	6	5	4	3	2	1	Bit 0	
$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA
$1001	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
$1002	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PORTG
$1003	DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDRG
$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PORTB
$1005	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	PORTF
$1006	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PORTC
$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	PORTD
$1009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE
$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC
$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M
$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D
$100E	Bit 15	14	13	12	11	10	9	Bit 8	TCNT (Hi)
$100F	Bit 7	6	5	4	3	2	1	Bit 0	TCNT (Lo)
$1010	Bit 15	14	13	12	11	10	9	Bit 8	TIC1 (Hi)
$1011	Bit 7	6	5	4	3	2	1	Bit 0	TIC1 (Lo)
$1012	Bit 15	14	13	12	11	10	9	Bit 8	TIC2 (Hi)
$1013	Bit 7	6	5	4	3	2	1	Bit 0	TIC2 (Lo)
$1014	Bit 15	14	13	12	11	10	9	Bit 8	TIC3 (Hi)
$1015	Bit 7	6	5	4	3	2	1	Bit 0	TIC3 (Lo)
$1016	Bit 15	14	13	12	11	10	9	Bit 8	TOC1 (Hi)
$1017	Bit 7	6	5	4	3	2	1	Bit 0	TOC1 (Lo)
$1018	Bit 15	14	13	12	11	10	9	Bit 8	TOC2 (Hi)
$1019	Bit 7	6	5	4	3	2	1	Bit 0	TOC2 (Lo)
$101A	Bit 15	14	13	12	11	10	9	Bit 8	TOC3 (Hi)
$101B	Bit 7	6	5	4	3	2	1	Bit 0	TOC3 (Lo)
$101C	Bit 15	14	13	12	11	10	9	Bit 8	TOC4 (Hi)
$101D	Bit 7	6	5	4	3	2	1	Bit 0	TOC4 (Lo)
$101E	Bit 15	14	13	12	11	10	9	Bit 8	TI4/O5 (Hi)
$101F	Bit 7	6	5	4	3	2	1	Bit 0	TI4/O5 (Lo)
$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1
$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2
$1022	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I	TMSK1

$1023	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F	TFLG1
$1024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
$1025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2
$1026	0	PAEN	PAMOD	PEDGE	0	I4/O5	RTR1	RTR0	PACTL
$1027	Bit 7	6	5	4	3	2	1	Bit 0	PACNT
$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR
$102A	Bit 7	6	5	4	3	2	1	Bit 0	SPDR
$102B	TCLR	0	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD
$102C	R8	T8	0	M	WAKE	0	0	0	SCCR1
$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR
$102F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SCDR
$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL
$1031	Bit 7	6	5	4	3	2	1	Bit 0	ADR1
$1032	Bit 7	6	5	4	3	2	1	Bit 0	ADR2
$1033	Bit 7	6	5	4	3	2	1	Bit 0	ADR3
$1034	Bit 7	6	5	4	3	2	1	Bit 0	ADR4
$1035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT
$1036									Reserved
$1037									Reserved
$1038	GWOM	CWOM	CLK4X	0	0	0	0	0	OPT2
$1039	ADPU	CSEL	IRQE	DLY	CME	FCME	CR1	CR0	OPTION
$103A	Bit 7	6	5	4	3	2	1	Bit 0	COPRST
$103B	ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM	PPROG
$103C	RBOOT	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO
$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT
$103E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	0	TEST1
$103F	EE3	EE2	EE1	EE0	1	NOCOP	1	EEON	CONFIG
$1040									Reserved
to									
$105B									Reserved
$105C	IO1SA	IO1SB	IO2SA	IO2SB	GSTHA	GSTHB	PSTHA	PSTHB	CSSTRH
$105D	IO1EN	IO1PL	IO2EN	IO2PL	GCSPR	PCSEN	PSIZA	PSIZB	CSCTL
$105E	GA15	GA14	GA13	GA12	GA11	GA10	0	0	CSGADR
$105F	IO1AV	IO2AV	0	GNPOL	GAVLD	GSIZA	GSIZB	GSIZC	CSGSIZ

8.4. Port Input/Output

MC68HC11F1 menyediakan 6 buah port input/outout 8 bit (port A,B,C,E,F dan G) dan satu buah port input/output 6 bit (port D).
Fungsi input/output port B,C,F dan G diatur menurut mode operasi yang dipilih. Pada mode single-chip dan bootstrap port tersebut berfungsi sebagai port parallel input/output. Sedangkan pada mode expanded-nonmultiplexed dan mode test, port B, C, F, G dan pin R/$\overline{W}$ dikonfigurasi sebagai saluran expansi memory, yang mana port B dan F sebagai saluran alamat, port C sebagai saluran data dan pin R/$\overline{W}$ sebagai kontro arah data serta empat bit atas (MSB) port G berfungsi sebagai external chip selects.
Sementara fungsi umum port input/output A, C, D dan G diatur oleh register pengarah data (Data Direction Register DDR) dari register yang bersangkutan.

8.4.1. Port A

Port A adalah 8 bit digital I/O port untuk penggunaan umum dengan data register (PORTA) dan data direction register (DDRA). Selain itu port A dapat dikonfigurasi untuk fungsi timer input capture (IC), timer output compare(OC) atau pulsa accumulator.

8.4.1.1. Data Register Port A (PORTA)

	Bit 7	6	5	4	3	2	1	Bit 0
$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
RESET:	HiZ	0	0	0	HiZ	HiZ	HiZ	HiZ
Alt. Pin Func.:	PAI	OC2	OC3	OC4	OC5/IC4	IC1	IC2	IC3
And/or:	OC1	OC1	OC1	OC1	OC1	—	—	—

Gambar 8.07 Register PORTA

Port A dapat dibaca kapan saja dan ketika sebagai keluaran, data yang telah dikeluarkan ke port A akan disimpan dalam internal latch.

8.4.1.2. Data Direction for Register Port A (DDRA)

	Bit 7	6	5	4	3	2	1	Bit 0
$1001	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.07 Data Penunjuk I/O Register PORTA

1 = Pin yang dimaksud akan dikonfigurasi sebagai keluaran
0 = Pin yang dimaksud akan dikonfigurasi sebagai masukan

8.4.2. Port B (PORTB)

Pada mode operasi single-chip, semua pin pada port B hanya dapat digunakan sebagai keluaran. Pada mode operasi expanded-nonmultiplexed, semua pin port B berfungsi sebagai saluran alamat orde tinggi (A15-A8).

	Bit 7	6	5	4	3	2	1	Bit 0
$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
S. Chip or Boot:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:	0	0	0	0	0	0	0	0
Expan. or Test:	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	ADDR 10	ADDR 9	ADDR 8

Gambar 8.08 Register PORTB

8.4.3. Port C

Pada mode operasi single-chip, port C adalah 8 bit digital I/O port untuk penggunaan umum dengan data register (PORTC) dan data direction register (DDRC). Pada mode operasi expanded-nonmultiplexed, port C berfungsi sebagai saluran data (D7-D0) dua arah yang dikontrol oleh signal R/$\overline{W}$.

8.4.3.1. Data Register Port C (PORTC)

	Bit 7	6	5	4	3	2	1	Bit 0
$1006	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
S. Chip or Boot:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:	0	0	0	0	0	0	0	0
Expan. or Test:	DATA 7	DATA 6	DATA 5	DATA 4	DATA 3	DATA 2	DATA 1	DATA 0

Gambar 8.09 Register PORTC

Port C dapat dibaca kapan saja dan ketika sebagai keluaran, data yang telah dikeluarkan ke port C akan disimpan dalam internal latch.

8.4.3.2. Data Direction for Register Port C (DDRC)

	Bit 7	6	5	4	3	2	1	Bit 0
$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.10 Data Penunjuk I/O Register PORTC

1 = Pin yang dimaksud akan dikonfigurasi sebagai keluaran
0 = Pin yang dimaksud akan dikonfigurasi sebagai masukan

8.4.4. Port D

Port D adalah 6 bit digital I/O port untuk penggunaan umum dengan data register (PORTD) dan data direction register (DDRD). Pada semua mode operasi, enam bit port D (D5-D0) selain dapat dipergunakan sebagai I/O dapat pula menjadi subsystem SCI dan SPI.

8.4.4.1. Data Register Port D (PORTD)

	Bit 7	6	5	4	3	2	1	Bit 0
$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0
RESET:	0	0	0	0	0	0	0	0
Alt. Pin Func.:	—	—	$\overline{SS}$	SCK	MOSI	MISO	TxD	RxD

Gambar 8.11 Register PORTD

Port D dapat dibaca kapan saja dan ketika sebagai keluaran, data yang telah dikeluarkan ke port D akan disimpan dalam internal latch.

8.4.4.2. Data Direction for Register Port D (DDRD)

	Bit 7	6	5	4	3	2	1	Bit 0
$1009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.11 Data Penunjuk I/O Register PORTD

Ketika port D dipergunakan sebagai I/O, maka kontrol untuk DDRD adalah sebagai berikut :

1 = Pin yang dimaksud akan dikonfigurasi sebagai keluaran
0 = Pin yang dimaksud akan dikonfigurasi sebagai masukan

Ketika port D difungsikan sebagai subsystem SPI, bit 5 berfungsi sebagai masukan slave select ($\overline{ss}$).
Pada mode slave SPI, DDD5 tidak memiliki arti (tidak berpengaruh). Sedangkan pada mode master SPI, DDD5 dapat diatur sebagai beikut :

1 = Port D bit 5 dikonfigurasi sebagai general-purpose output line
0 = Port D bit 5 dikonfigurasi sebagai masukan untuk medeteksi kesalahan pada SPI

8.4.5. Port E

Pada semua mode operasi, Port E dapat dipergunakan sebagai masukan 8 bit digital general-purpose (E7-E0) atau sebagai masukan 8 kanal analog (AN0-AN7).

	Bit 7	6	5	4	3	2	1	Bit 0
$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:	U	U	U	U	U	U	U	U
Alt. Pin Func.:	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Gambar 8.12 Register PORTE

8.4.6. Port F (PORTF)

Pada mode operasi single-chip, semua pin pada port F hanya dapat digunakan sebagai keluaran. Pada mode operasi expanded-nonmultiplexed, semua pin port F berfungsi sebagai saluran alamat orde rendah (A7-A0).

	Bit 7	6	5	4	3	2	1	Bit 0
$1005	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:	0	0	0	0	0	0	0	0
Alt. Pin Func.:	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0

Gambar 8.13 Register PORTF

8.4.7. Port G (PORT G)

Port G adalah 8 bit digital I/O port untuk penggunaan umum dengan data register (PORTG) dan data direction register (DDRG).

8.4.7.1. Data Register Port G (PORTG)

	Bit 7	6	5	4	3	2	1	Bit 0
$1002	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
RESET:	0	0	0	0	0	0	0	0
Alt. Pin Func.:	R/$\overline{W}$	—	XA18	XA17	XA16	XA15	XA14	XA13

Gambar 8.14 Register PORTG

Port G dapat dibaca kapan saja dan ketika sebagai keluaran, data yang telah dikeluarkan ke port G akan disimpan dalam internal latch.

8.4.7.2. Data Direction for Register Port G (DDRG)

	Bit 7	6	5	4	3	2	1	Bit 0
$1003	DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.15 Register PORTG

1 = Pin yang dimaksud akan dikonfigurasi sebagai keluaran
0 = Pin yang dimaksud akan dikonfigurasi sebagai masukan

8.5. Chip Selects

Fungsi dari chip select ini adalah untuk mengeliminasi kebutuhan akan tambahan komponen external dan mengantarmukai dengan perangkat perangkat pada mode operasi expanded-nonmultliplexed, seperti misalnya factor polaritas, ukuran blok alamat dan clock stretching dikontrol menngunakan register chip select.

Ada empat programmable chip select pada MC68HC11F1 yang dapat kita di-enablel-kan melalui chip-select control register (SCCTL) dan didesain supaya tidak terjadi konflik antar memori internal. Keempat chip select tersebut yaitu :

- Dua external I/O (CSIO1 dan CSIO2)
- Satu external program space ($\overline{\text{CSPROG}}$)
- Satu general-purpose chip select

8.5.1. Programmable Chip Select ($\overline{\text{CSPROG}}$)

External program space chip select ini mulai pada akhir dari alamat memory dan berlanjut maju sampai pada awal memeori dalam hitungan pangkat dua, dari 8K sampai 64K. Chip select ini aktif low dan aktif hanya selama waktu alamat yang valid. $\overline{\text{CSPROG}}$ dapat dienablekan melalui bit PCSEN pada chip-select control register (CSCTL) dan besarnya blok alamat diatur melalui bit PSIZA dan PSIZB dari register CSCTL. Sedangkan prioritas dikontrol oleh bit GCSPR.

8.5.2. I/O Chip Selects (CSIO1 dan CSIO2)

Chip select ini untuk memilih external device. Alamat-alamat blok diatur pada peta memory sebesar kelipatan 4 K. CSI1 memetakan memory mulai alamat $1060 sampai $17FF dan CSIO2 memetakan memory mulai alamat $1800 sampai dengan $1FFF dimana angka "1" adalah karakter yang mewakili nilai orde tinggi nible dari alamat blok register. Enable dan polaritas CSIO1 dan CSIO2 dikontrol oleh register CSCTL pada bit IO1EN, IO1PL, IO2EN dan IO2PL.
Bit IO1AV dan IO2AV pada register CSGSIZ menentukan chip select mana yang valid selama waktu alamat atau E-clock yang valid.

8.5.3. Chip-Select Control Register (CSCTL)

	Bit 7	6	5	4	3	2	1	Bit 0
$105D	IO1EN	IO1PL	IO2EN	IO2PL	GCSPR	PCSEN	PSIZA	PSIZB
RESET:	0	0	0	0	0	—	0	0

Gambar 8.16 CSCTL Register

IO1EN Enable for I/O Chip-Select 1
1 = Chip select is enabled
0 = Chip select is disabled

IO1PL Polarity select for I/O Chip-Select 1
1 = Chip select is active high
0 = Chip select is active low

IO2EN Enable for I/O Chip-Select 2
1 = Chip select is enabled
0 = Chip select is disabled

IO2PL Polarity select for I/O Chip-Select 2
1 = Chip select is active high
0 = Chip select is active low

GCSPR General-Purpose Chip-Select Priority
1 = General-purpose chip select has priority
0 = Program chip select has priority

PCSEN Enable for Program Chip-Select
1 = Program chip select is enabled. Reset sets PCSEN in expanded-nonmultilexed mode
0 = Program chip select is disabled. Reset clears PCSEN in single-chip mode

Tabel 8.05 PSIZA and PSIZB Program Chip-Select Address Sizes

PSIZA	PSIZB	Size (Bytes)	Address Range
0	0	64 K	$0000–$FFFF
0	1	32 K	$8000–$FFFF
1	0	16 K	$C000–$FFFF
1	1	8 K	$E000–$FFFF

8.5.4. General-Purpose Chip Select (CSGEN)

Chip select ini paling fleksibel diantara empat chip select dan memiliki kontrol bit paling banyak.
Polaritas, Alamat terhadap E-clock dan besarnya blok alamat ditentukan oleh bit-bit GNPOL,GAVLD, GSIZA, GSIZB dan GSIZC pada register CSGSIZ. Permulaan alamat dipilih oleh bit GCSPR pada register CSCTL.

8.6. Reset, Interrupts dan Low Power Modes

8.6.1. Resets

MCU memiliki empat macam reset, yaitu :
- Pin masukan external reset aktif low
- Fungsi power on reset
- Clock monitoring failur
- Computer operating properly (COP) watchdog-timer timeout

8.6.1.1. Pin Reset

Untuk memenuhi keperluan reset external, disediakan pin untuk reset aktif low. Yang mana lamanya waktu reset logika low adalah paling cepat 8 kali siklus E-clock.

8.6.1.2. Poweron Reset (POR)

Poweron reset adalah pendeteksian sinyal reset ketika terjadi perubahan tegangan VDD dari positip ke logika rendah. Pada prakteknya poweron reset ini adalah pembuatan delay selama beberapa watu pada pin reset agar ketika power supply dipasangkan akan menghasilkan logika rendah pada pin RESET dengan cara membuat rangkaian seri resistor capasitor.

8.6.1.3. Computer Operating Properly (COP)

Dalam MC68HC11F1 terdapat sebuah timer watchdog yang secara otomatis menghitung waktu time out program dalam tetapan waktu yang spesific. Jika timer COP watchdog mengijinkan untuk time out, maka suatu reset akan dilakukan, dimana pin $\overline{\text{RESET}}$ akan di-drive ke logika low untuk mereset mikrokontroller dan sistin external.
COP Watchdog dapat menguji atau melihat apakah program berjalan dengan baik atau terjadi kesalahan, untuk keperluan ini kita harus membuat software untuk menetapkan waktu watchdog. Jika watchdog tidak diset ulang maka dia akan me-reset system yang berarti akan kembali ke program awal.
Fungsi reset COP dapat dilakukan dengan memprogram bit kontrol NOCOP dari register sistim konfigurasi (CONFIG).
Pertama kali diprogram, kontrol bit ini akan dibersihkan kalau tidak ada power supply, dan fungsi COP ini aktif atau tidak tergantung dari software.
Bit kontrol proteksi (CR1 dan CR0) dalam register pilihan konfigurasi (OPTION) memberikan kemungkinan untuk memilih satu dari empat rate timeout. Tabel di bawah ini memperlihatkan hibungan antara CR1 dan CR0 terhadap periode COP timeout untuk beberapa variasi frekenuensi clock.

Tabel 8.06 COP Timeout Periods

CR [1:0]	Divide E/2^{15} By	XTAL = 8.0 MHz Timeout –0/+16.4 ms	XTAL = 12.0 MHz Timeout –0/+10.9 ms	XTAL = 16.0 MHz Timeout –0/+8.2 ms
0 0	1	16.384 ms	10.923 ms	8.192 ms
0 1	4	65.536 ms	43.691 ms	32.768 ms
1 0	16	262.14 ms	174.76 ms	131.07 ms
1 1	64	1.049 sec	699.05 ms	524.29 ms
	E =	2.0 MHz	3.0 MHz	4.0 MHz

Urutan langkah untuk mere-setting timer watchdog adalah sebagai berikut :
1. Tulis $55 ke register reset COP (COPRST)
2. Tulis $AA ke register COPRST

8.6.2. Interrupt

Selain interrupt type reset, masih terdapat 17 interrupt hardware dan satu interrupt software yang dapat dilakukan dari banyak kemungkinan sumber.
Interrupt ini dapat dibedakan menjadi dua macam, yaitu maskable dan nonmaskable interrupt.
Limabelas interrupt dapat dimasker melalui bit I pada register kode kondisi (Condition code register CCR).
Semua interrupt hardware pada chip MCU dikontrol oleh bit local secara individual.
Interrupt software adalah nonmaskable.
Pin masukan interrupt external $\overline{\text{XIRQ}}$ adalah interrupt yang nonmaskable karena $\overline{\text{XIRQ}}$ tidak dapat dimasker oleh software sejak dienablekan. Meskipun demikian $\overline{\text{XIRQ}}$ dapat dimasker selama reset.

Opcode illegal juga termasuk interrupt yang nonmaskable.

Real-time interrupt menyediakan sebuah programmable periodic interrupt yang ter-maskable oleh bit I dalam register CCR atau bit RTI enable pada register timer interrupt mask 2 (TMSK2). Rate berabasis pada E-cock dan software untuk memilih factor pembagi E-clock sebesar $E \div 2^{13}$, $E \div 2^{14}$, $E \div 2^{15}$ atau $E \div 2^{16}$.

Tabel 8.07 Daftar vector interrupt

Vector Address	Interrupt Source	CCR Mask Bit	Local Mask
FFC0, C1 – FFD4, D5	Reserved	—	—
FFD6, D7	SCI Serial System*	I	
	• SCI Receive Data Register Full		RIE
	• SCI Receiver Overrun		RIE
	• SCI Transmit Data Register Empty		TIE
	• SCI Transmit Complete		TCIE
	• SCI Idle Line Detect		ILIE
FFD8, D9	SPI Serial Transfer Complete	I	SPIE
FFDA, DB	Pulse Accumulator Input Edge	I	PAII
FFDC, DD	Pulse Accumulator Overflow	I	PAOVI
FFDE, DF	Timer Overflow	I	TOI
FFE0, E1	Timer Input Capture 4/ Output Compare 5	I	I4/O5I
FFE2, E3	Timer Output Compare 4	I	OC4I
FFE4, E5	Timer Output Compare 3	I	OC3I
FFE6, E7	Timer Output Compare 2	I	OC2I
FFE8, E9	Timer Output Compare 1	I	OC1I
FFEA, EB	Timer Input Capture 3	I	IC3I
FFEC, ED	Timer Input Capture 2	I	IC2I
FFEE, EF	Timer Input Capture 1	I	IC1I
FFF0, F1	Real-Time Interrupt	I	RTII
FFF2, F3	$\overline{\text{IRQ}}$ (External Pin)	I	None
FFF4, F5	X$\overline{\text{IRQ}}$ Pin	X	None
FFF6, F7	Software Interrupt	None	None
FFF8, F9	Illegal Opcode Trap	None	None
FFFA, FB	COP Failure	None	NOCOP
FFFC, FD	Clock Monitor Fail	None	CME
FFFE, FF	$\overline{\text{RESET}}$	None	None

*Interrupts generated by SCI; read SCSR to determine source

8.7. Programmable Timer

Sistim pewaktuan MC68HC11 terdiri atas lima buah pembagi clock. Pembagi clock utama merupakan free-running counter 16 bit yang dikendalikan oleh prescaler.
Programmable prescaler timer utama menyediakan empat pilihan pembagi clock yang dapat dipilih dengan mengatur dua bit kontrol yaitu PR1 dan PR0
Keluaran prescaler adalah pulsa clock yang telah dibagi dengan nilai pembagi 1, 4, 8 atau 16. Keluaran pulsa clock yang lebih lambat ini dipergunakan sebagai pulsa accumulataor, Real-Time Interrerupt (RTI) dan computer operating properly (COP) watchdog subsystem.

Semua aktivitas sistim timer utama disesuaikan free-running counter ini. Counter mulai dengan hitungan naik dari $0000 seperti (ketika MCU direset), dan berlanjut sampai hitungan maksimum, $FFFF. Dan kemudian kembali ke $0000, dan mengeset register flag overflow dan melakukan hitungan lagi seperti semula.
Pada mode operasi normal, sama sekali tidak mungkin untuk mereset, mengubah, atau interupsi counter ini.
Programmable timer bermula dari sebuah free-running counter 16 bit yang mendapatkan clock dari E-clock yang dibagi dengan bit kontrol yang dapat diset melalui bit PR1 dan PR0 dari register TMSK2 sebagi berikut

8.7.1. Timer Interrupt Mask Register 2 (TMSK2)

	Bit 7	6	5	4	3	2	1	Bit 0
$1024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.17 Timer Interrupt Mask Register 2 (TMSK2)

Tabel 8.08 PR1 dan PR0 Timer Prescaler Select

PR1	PR0	Divide By
0	0	1
0	1	4
1	0	8
1	1	16

Bit control prescaler hanya dapat diset selama 64 pertama E-clock setelah reset. Free-running counter (register TCNT) dapat dibaca kapan saja tanpa merubah isi register dan hanya dapat di-clear dengan reset saja. Free-running counter ini akan menghitung mulai $0000 sampai dengan $FFFF dan setiap terjadi overflow bit maka timer verflow flag

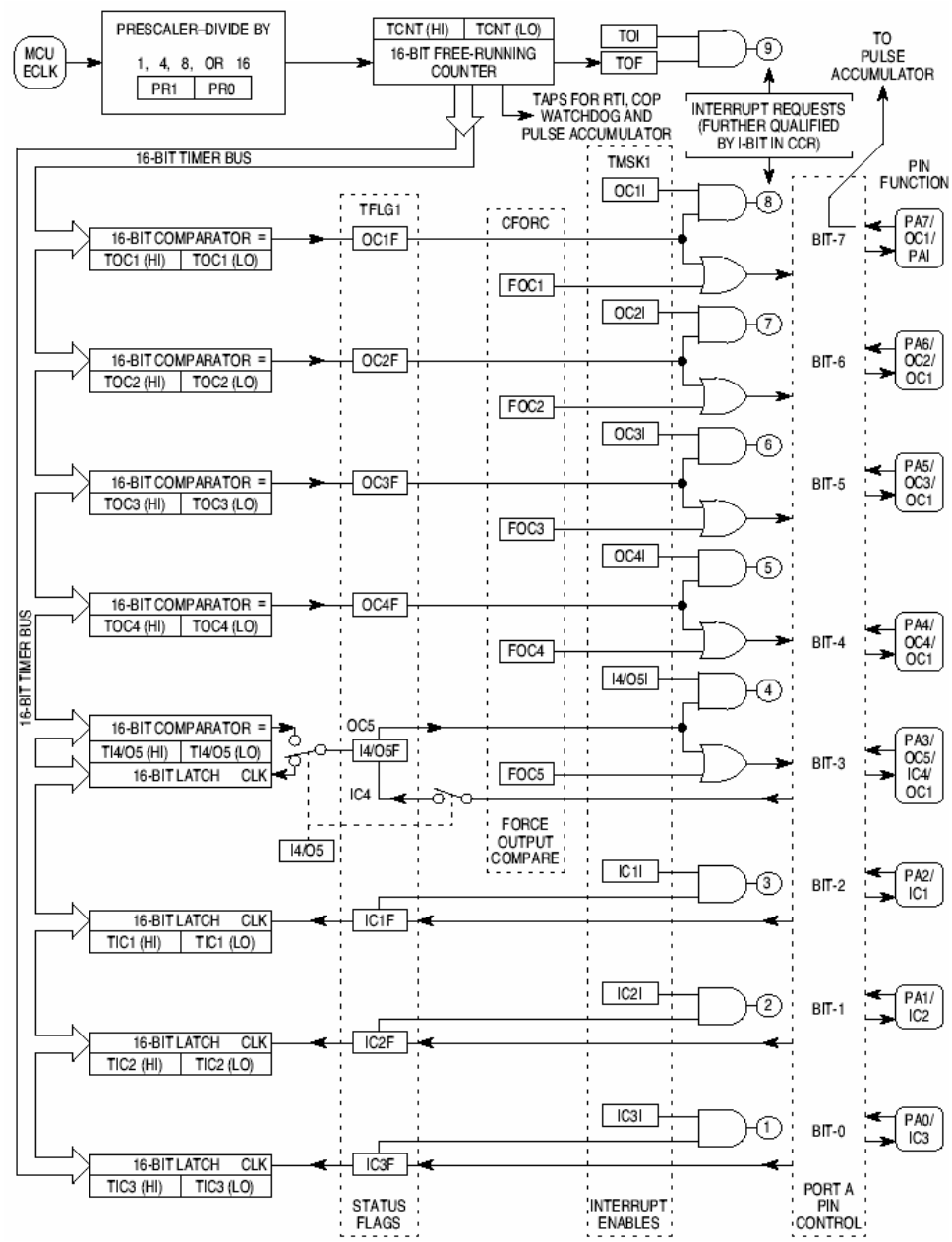
(TOF) di register TFLG2 akan diset dan bit timer overflow interrupt enable (TOI) di register TMSK2.
Programmable timer ini memiliki tiga register input capture dan empat register output compare yang dapat difungsikan dengan kontrol software.

8.7.2. Pulse Accumulator

Akkumulator pulsa adalah sebuah counter 8 bit yang dapat beroperasi satu dari dua mode tergantung dari kondisi kontrol bit pada register PACTL.
Mode tersebut adalah :

1. Event counting mode
   8 bit counter ini mendapatkan signal clock dari pin external yang besarnya frekuensi maksimal adalah setengah E-clock.

2. Gated time accumulation mode
   8 bit counter ini mendapatkan signal clock dari internal free running E-clock yang besarnya adalah 1/64 dari E-clock selama pin masukan external PAI diaktifkan.



Gambar 8.18 Digram Blok Timer

Akkumulator pulsa menggunakan port A bit 7 sebagai masukan,meskipun demikian pin ini juga masih dapat dipergunakan sebagai general-purpose I/O ataupun sebagai output compare. Dan ketika port A 7 ini telah dikonfigurasi sebagai output, pin ini tetap berfungsi segai masukan bagi akkumulator pulsa.

8.7.2.1. Pulse Accumulator Control Register (PACTL)

	Bit 7	6	5	4	3	2	1	Bit 0
$1026	0	PAEN	PAMOD	PEDGE	0	I4/O5	RTR1	RTR0
RESET:	0	0	0	0	0	0	0	0

Gambar 8.19 Pulse Accumulator Control Register (PACTL)

Tiga bit dari register ini mengontrol sistim 8 bit akkumulator pulsa. Satu bit yang lain berfungsi untuk mengenablekan output compare 5 atau input capture 4, sedangkan bit lainnya untuk memilih rate untuk sistim real-time interrupt.

PAEN Pulse Accumulator System Enable
1 = Pulse Accumulator on
0 = Pulse Accumulator off

PAMODPulse Accumulator Mode
1 = Gated time accumulator mode
0 = Event counter mode

Tabel 8.09 Pulse Accumulator Edge Control (PEDGE)

PAMOD	PEDGE	Action on clock
0	0	PAI Falling Edge Increments the counter
0	1	PAI Rising Edge Increments the counter
1	0	A Zero on PAI Inhibits Counting
1	1	A One on PAI Inhibits Counting

I4/O5 Configure TI4O5 Register for IC or OC
1 = IC4 function enabled
0 = OC5 function enabled

Tabel 8.10 RTR1 and RTR0 Real-Time Interrupt (RTI) Rate

RTR [1:0]	Divide E By	XTAL = 8.0 MHz	XTAL = 12.0 MHz	XTAL = 16.0 MHz
0 0	2^{13}	4.096 ms	2.731 ms	2.048 ms
0 1	2^{14}	8.192 ms	5.461 ms	4.096 ms
1 0	2^{15}	16.384 ms	10.923 ms	8.192 ms
1 1	2^{16}	32.768 ms	21.845 ms	16.383 ms
	E =	2.0 MHz	3.0 MHz	4.0 MHz

8.7.2.2. Pulse Accumulator Count Register (PACNT)

	Bit 7	6	5	4	3	2	1	Bit 0
$1027	Bit 7	6	5	4	3	2	1	Bit 0

Gambar 8.20 Pulse Accumulator Count Register (PACNT)

Register ini berisi hasil counter dari external input PAI pada mode external input events atau selama PAI ini aktiv pada mode gated time accumulation mode

8.8. 8 Electrically Erasable Programm-able Read-Only Memory (EEPROM)

Di dalam mikrokontroler MC68HC11F1 terdapat 512 byte EEPROM yang dapat dipetakan ke 4 K boundary di dalam memory. Alamat $xE00 - $xFFF, dimana x mewakili nilai orde tinggi di dalam register CONFIG dan nilai ini merupakan nilai awal dari 4 K boundary. Dalam mode single-chio dan mode bootstrap, EEPROM diset pada alamat $FE00 - $FFFF. Dalam mode special test, EEPROM awalnya diset tidak aktif dan untuk mengaktifkannya harus men-set bit EEON di register CONFIG.
Pemrograman EEPROM dikontrol oleh register PPROG dan register BPROT. EEPROM diset enable jika bit EEON pada register CONFIG diset dan EEPROM akan disable apabila bit EEON ini di-clear. Untuk menulis dan menghapus isi EEPROM menggunakan tegangan tinggi yang dibangkitkan secara internal di dalam chip. Dengan E-clock 2 MHz diperlukan waktu sekitar a0 mili detik untuk memprogram atau menghapus EEPROM, dan dengan E-clock antara 1 – 2 mili detik dieprlukan waktu tang lebih lama sekitar 20 mili detik.
EEPROM dapat dihapus berdasarkan per byte ataupun bulk.
Untuk mengeset byte alamat orde tinggi x kita harus mengeset bit EE3-EE0 dan untuk mengaktifkan EEPROM dengan mngeset bit EEON di register CONFIG sebagi berikut

8.8.1. EEPROM Block Protect Register (BPROT)

	Bit 7	6	5	4	3	2	1	Bit 0
$1035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0
RESET:	1	1	1	1	1	1	1	1

Gambar 8.21 EEPROM Block Protect Register

PTCON Protect CONFIG Register
1 = Programming/erasure of CONFIG register disabled

0 = Programming/erasure of CONFIG register alowed

BPRT3-BPRT0 Block Protect

1 = A set bit protects a block of EEPROM against programming or erasing

0 = A cleared bit permits programming or erasure of the associated lock.

Tabel 8.11 BPROT Address Sizes

Bit	Block Protected	Block Size
BPRT0	$xE00 - $xE1F	32 Bytes
BPRT1	$xE20 - $xE5F	64 Bytes
BPRT2	$xE60 - $xEDF	128 Bytes
BPRT3	$xEE0 - $xEFF	288 Bytes

8.8.2. Configuration Control Register (CONFIG)

	Bit 7	6	5	4	3	2	1	Bit 0
$103F	EE3	EE2	EE1	EE0	1	NOCOP	1	EEON
RESET:	—	—	—	—	1	—	1	—

Gambar 8.22 Configuration Control Register (CONFIG)

EE3-EE0 EEPROM Map Position

EEPROM berlokasi pada alamat $xE00 - $xFFF, dimana ‘x’ adalah bilangan hexadecimal yang diwakili oleh keempat bit ini.

EEON EEPROM Enable

1 = EEPROM aktif dalam peta memory dengan lokasi sesuai dengan bit EE3-EE0

0 = EEPROM tidak aktif dalam peta momory.

8.8.3. Menghapus EEPROM

EEPROM yang telah dihapus akan berisi data $FF. Untuk menghapusnya diperlukan langkah-langkah sebagai berikut :

1. Mengeset bit ERASE, EELAT dan appropriate BYTE serta ROW di register PPROG
2. Menulis suatu data ke dalam appropriate alamat EEPROM
3. Mengeset bit ERASE, EELAT dan EEPGM appropriate BYTE serta ROW di register PPROG
4. Tunda selama 10 ms atau lebih
5. Meng-clear bit EEPGM di register PPROG untuk mematikan tegangan tinggai

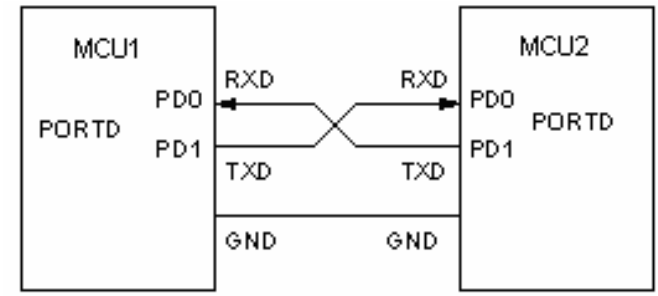
8.8.4. Memprogram EEPROM

Apabila lokasi EEPROM yang akan diprogram sudah berisi dengan bit data nol, maka sebelum memprogram lokasi memory yang berisi bit nol tersebut harus dihapus terlebih dahulu.
Untuk memprogram EEPROM, yakinkan bahwa register BPROT sudah jelas dan selanjutnya langkah-langkah pemrograman adalah dengan mengatur bit bit di register PPROG sebagai berikut :

1. Mengeset bit EELAT
2. Menulis data ke alamat yang diinginkan
3. Mengeset bit EELAT dan EEPGM
4. Tunda selama 10 mili detik atau lebih
5. Meng-clear bit EEPGM untuk mematikan tegangan tinggi

8.9. Serial Communication Interface (SCI)

SCI memungkinkan suatu mikrokontroler dapat berhubungan dengan peralatan lain dengan efisien dalam format data serial asynchronous. SCI mempergunakan format standar non-return-zero (NRZ) dengan berbagai kecepatan baud rate sesuai dengan kristal yang dipasang pada rangkaian mikrokontroler. Sambungan SCI ini disediakan dengan menggunakan pin-pin pada port D. PD0 dipewrgunakan untuk menerima data (RxD) dan PD1 dipergunakan untuk mengirim data (TxD). Baud rate diatur dengan mengeset prescaler untuk membagi E-clock.



Gambar 8.23 Interfacing dua MCU melalui SCI

Langkah-langkah mengakses SCI :

Mengirim data :
1. Memasukkan data ke Akkumulator
2. Menunggu sampai bit TDRE di register SCSR telah diset
3. Mengisikan data ke register SCDAT

Menerima data :

1. Menunggu sampai bit RDRF di register SCSR telah diset
2. Membaca data dari register SCDAT

8.9.1.1. PORTD

PD 7	PD 6	PD 5	PD 4	PD 3	PD 2	PD 1	PD 0
-	-	SS'	SCK	MOSI	MISO	TXD	RXD

Inisialisasi port D :
```
ldaa  #%00000011
staa  DDRD
```

Serial Communications Control Register 1
SCCR1, $102C

bit 7	6	5	4	3	2	1	0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Inisialisasi SCCR1 :
```
ldaa  #%00000000
staa  SCCR1
```

Inisialisasi ini seharusnya diletakkan pada program monitor

Serial Communications Control Register 2
SCCR2, $102D

bit 7	6	5	4	3	2	1	0
R8	T8	0	M	WAKE	0	0	0

Inisialisasi SCCR2 :

```
ldaa  #%00001100
staa  SCCR2
```

Inisialisasi ini seharusnya diletakkan pada program monitor

Baud Rate Register (BAUD), $102B

bit 7	6	5	4	3	2	1	0
TCLR	0	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0

Inisialisasi baud rate :

```
ldaa  #%00110000
staa  BAUD
```

Inisialisasi ini seharusnya diletakkan pada program monitor

Serial Communications Status Register
(SCSR), $102E

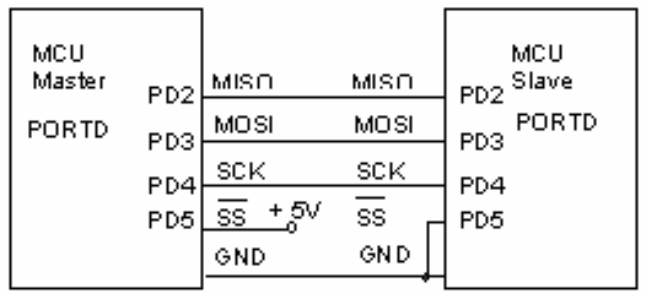
bit 7	6	5	4	3	2	1	0
TDRE	TC	RDRF	IDLE	OR	NF	FE	0

Serial Communications Data Register
(SCDAT), $102F

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

8.10. Serial Peripheral Interface (SPI)

SPI adalah sistim I/O serial synchronous berkecepatan tinggi. SPI dapat dipergunakan untuk perluasan tambahan port I/O secara serial ataupun sebagai sarana interkoneksi antar mikrokontroler dalam konfigurasi multimaster. Kecepatan clock dan polaritas dapat diprogram melalui software serta dapat disambungkan dengan banyak perangkat.. SPI dapat dikonfigurasi sebagai master atau slave.
SPI terdiri dari empat sinyal dasar, yaitu :
MOSI = Master-Out Salve-In
MISO = Master-In slave-Out
SCK = Serial Clock
$\overline{SS}$ = Slave select
keempat sinyal tersebut tersambung padaport D dan harus disetting sesuai arah data dengan menginisialisasi register DDRD



Gambar 8.25 Interfacing dua MCU melalui SPI

Langkah-langkah mengakses SPI :
1. Inisialisasi
2. Mengeset SS' dengan 0
3. Mengeluarkan data ke register SPDAT
4. Menunggu sampai SPSR diset
5. Membaca data dari register SPDAT
6. Mengeset SS' dengan 1

PORTD

PD 7	PD 6	PD 5	PD 4	PD 3	PD 2	PD 1	PD 0
-	-	SS'	SCK	MOSI	MISO	TXD	RXD

Inisialsisasi port :

```
ldaa  #%00111100
staa  DDRD
```

Control Register (SPCR), $1028

bit 7	6	5	4	3	2	1	0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Inisialisasi sebagai Master :

```
ldaa  #%01010011
staa  SPCR
```

Inisialisasi sebagai Slave :

```
ldaa  #%01000011
staa  SPCR
```

Status Register (SPSR), $1029

bit 7	6	5	4	3	2	1	0
TDRE	TC	RDRF	IDLE	OR	NF	FE	0

Data I/O Register (SPDAT), $102A

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

8.11. Analog-To-Digital Converter

MC68HC11F1 memiliki delapan kanal masukan analog yang pembacaannya dilakukan secara multiplex dan menggunakan metode successive-approximation sample and hold. Tegangan referensi masukan diberikan dari luar melalui pin V_{RL} dan V_{RH}. Hasil konversi berupa data 8 bit yang diperoleh setelah 32 E-clock cycle.

Jika tegangan masukan sama dengan V_{RL} maka data yang diperoleh sebagai hasil dari konversi adalah $00, dan jika tegangan masukan sama dengan V_{RH} maka data yang diperoleh sebagai hasil dari konversi adalah $FF (skala penuh) tanpa adanya indicator overflow.

Masukan analog dihubung pada mikrokontroller melalui masukan AN0-AN7 dan hasil konversi dapat dilihat pada register ADR1, ADR2, ADR3 dan ADR4.

Untuk mengakses ADC dapt dilakukan dengan memberikan kontrol pada register ADCTL dengan beberapajenis pengoperasian. Untuk

mengaktivkan ADC, sebuah jenis pengoperasian harus dipilih dengan mengeset bit bit pada register ADCTL.

A/D Control/Status Register (AD CTL)

	Bit 7	6	5	4	3	2	1	Bit 0
$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA
RESET:	U	0	U	U	U	U	U	U

Gambar 8.27 A/D Control Status Register

CCF Conversion Complete Flag
Bit ini akan di-set setelah proses konversi telah selesai dilakukan dan tetep tidak akan berubah sampai sampai pada penulisan ADCTL kembali

SCAN Contunous Scan Control
1 = melakukan 4 konversi secara terus menerus
0 = melakukan 4 konversi dan stop sampai penulisan ADCTL kembali

MULT Multiple-Channel/Single-Channel Control
1 = melakukan 4 konversi pada kanal yang dipilih untuk pembacaan 8 kanal
0 = melakukan 4 konversi pada kanal yang dipilih untuk pembacaan 4 kanal

CD-CA Channel Selects (D-A)
Digunakan untuk memilih satu dari delapan kanal pada mode multi kanal (MULT=1)

Tabel 8.12 Chanel Selects A-D

CD	CC	CB	CA	Channel Signal	Result
0	0	0	0	AN0	ADR1
0	0	0	1	AN1	ADR2
0	0	1	0	AN2	ADR3
0	0	1	1	AN3	ADR4
0	1	0	0	AN4	ADR1
0	1	0	1	AN5	ADR2
0	1	1	0	AN6	ADR3
0	1	1	1	AN7	ADR4
1	0	0	0	Reserved	ADR1
1	0	0	1	Reserved	ADR2
1	0	1	0	Reserved	ADR3
1	0	1	1	Reserved	ADR4
1	1	0	0	V_{RH} Pin *	ADR1
1	1	0	1	V_{RL} Pin *	ADR2
1	1	1	0	$(V_{RH})/2$ *	ADR3
1	1	1	1	Reserved*	ADR4

* Group ini hanya digunakan pada saat pengujian di pabrik

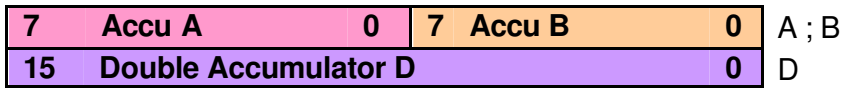
8.12. Informasi Pemrograman

8.12.1. Model Pemrograman

Di dalam mikrokontroller MC68HC11F1 terdapat delapan register central processing unit (CPU)

8.12.1.1. Accumulator (A,B dan D)

Accumulator A dan B adalah register 8 Bit, sebagai penampung lintas data ke dan dari ALU (Arithmetic Logic Unit), oleh karena itu selalu disebut dengan singkat Accu A atau Accu B. Operasi Arithmatik atau juga manipulasi data sebagian besar dilaksanakan dengan isi Accu ini dan pada register/Accu ini pula hasil operasi disimpan. Accumulator A dan B (masing-masing satu Byte) dapat digabungkan menjadi dua byte accumulator yang disebut Double Accumulator D (Accu D).



Gambar 8.28 Accumulator (A,B dan D)

8.12.1.2. Index Register X dan Y (IX dan IY)

Register ini adalah register 16 Bit yang digunakan untuk indexed addressing mode. Pada pengalamatan yang menggunakan indeks, isi dari indeks register 16 bit ditambah dengan 8 bit offset. Kedua register ini dapat juga digunakan sebagai register counter dan juga sebagai penyimpan sementara.

8.12.1.3. Stack Pointer (SP)

SP adalah register 16 Bit yang selalu berisi next free location pada stack. Stack adalah penyimpan yang mempunyai konfigurasi seperti LIFO (Last-In-First-Out → yang masuk terakhir akan keluar pertama kali)
Stack digunakan untuk pemanggilan program bagian (menyimpan alamat instruksi berikutnya setelah program bagian selesai),selama intterupt(menyimpan isi semua register CPU)dan instruksi Push-Pull

(menyimpan data sementara). Setiap kali satu Byte didalam Stack diambil atau pulled maka SP secara otomatis bertambah satu(increament). Pada aplikasi inisialisasi SP dilakukan pertama kali

15	Stack Pointer	0	SP

8.12.1.4. Program Counter (PC)

PC adalah register 16 bit yang berisi alamat instruksi berikutnya yang akan dikerjakan.

15	Program Counter	0	PC

8.12.1.5. Condition Code Register (CCR)

CCR berisi 5 bit sebagai indikator status, 2 bit interrupt masking dan 1 bit STOP disable. Ke lima bit indikator status tersebut adalah H,N,Z,V,dan C yang merefleksikan hasil operasi arithmatik dan operasi lainnya yang dilakukan CPU Flag H digunakan untuk operasi aritmathik BCD, sedangkan status bit pada flag N,Z,V dan C digunakan sebagai syarat untuk instruksi percabangan (loncat). Masing-masing Bit dapat diterangkan seperti di bawah ini :

S	V	H	I	N	Z	V	C	CCR

Gambar 8.29 Condition Code Register (CCR)

Keterangan :

S : Stop Disable
X : X Interrupt Mask
H : Half Carry (dari bit 3)
I : Interrupt Mask
N : Negative
Z : Zerro
V : Overflow
C : Carry

Carry/Borrow (C)

Bit Carry/Borrow di set, jika dalam operasi arithmatik yang telah dijalankan, hasil telah melebihi atau dibawah daerah bilangan register ALU yang dipakai. Bit ini juga akan terpengaruh pada instruksi Geser(shift) dan Putar(rotate).

Overflow (V)

Bit V di set, jika pada operasi arithmetik (Bit tertinggi sebagai bit tanda) menghasilkan hasil yang melampaui daerah bilangan.

Zero (Z)

Bit Z di set, jika hasil operasi arithmatik,logik dan juga manipulasi data yang telah dilaksanakan hasilnya NOL(zero).

Negative (N)

Bit N di set ,jika hasil operasi arithmatik,logik dan juga manipulasi data yang telah dilaksanakan hasilnya pada daerah Negatip. Hasil berada pada daerah negatip, jika MSB (bit tertinggi) adalah 1.

Interrupt Mask (I)

Bit I dapat di set melalui Hardware atau Software. untuk menutup/mencegah (disable --> Mask) semua maskable interrupt Bit I harus '1', sedangkan untuk mengijinkan atau melalukan semua maskable interrupt Bit I harus '0'. Bit I dapat di set atau di reset dengan instruksi(Software) SEI atau CLI

Half Carry (H)

Bit H di set, jika terjadi carry antara bit ke 3 dan 4 dalam operasi penjumlahan. Bit ini biasanya digunakan pada kalkulasi dalam BCD.

X Interrupt Mask(X)

Bit X hanya dapat di set melalui Hardware ($\overline{\text{RESET}}$ atau $\overline{\text{XIRQ}}$), dan dapat dihapus ('0') melalui instruksi transfer A ke CC Register (TAP) atau Return from Interrupt (RTI).

Stop Disable (S)

Bit ini dapat dipengaruhi melalui software, pada kejadian S = '1', instruksi STOP adalah disable (dicegah).

8.12.2. Instruction Set

Dalam buku **HC11 MC68HC11F1 PROGRAMMING REFERENCE GUIDE** terdapat informasi tentang kode operasi dari setiap instruksi yang digunakan oleh mikrokontroller MC68HC11F1.
Ada dua macam tabel kode operasi yang disediakan, yaitu berdasarkan urutan angka kode operasi (lihat halaman 10 s.d. 17) dan tabel kode operasi berdasarkan instruksi urut sesuai abjad (lihat halaman 18 s.d. 35).

Mnemonic
Tata tulis singkat untuk instruksi Assembler

Operation
Penjelasan pelaksanaan operasi instruksi assembler.

Description
Menggambarkan instruksi assembler dengan Symbol.

Adr. Mode
Pilihan/kemungkinan macam-macam pengalamatan dari instruksi assembler yang sesuai

Tabel 8.13 Cuplikan contoh tabel

Source Form **Mnemonics**	Operations **Operasi**	Boolean Expression **Deskripsi**	Addresing Mode for Operand **Adr. Mode**	Maschine Coding (Hexadecimal) Opcode **Opcode**	Operand(s) **Operands**	Bytes	Cycles	Condition Codes S X H I N Z V C
LDAA (opr)	Load Accumulator A	M → A	A IMM A DIR A EXT A IND,X A IND,Y	86 96 B6 A6 18 A3	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	– – – – Δ Δ o –

INH (inherent) Instruksi hanya terdiri dari satu Byte OpCode, tanpa operand.

IMM (immediate)Data yang akan diolah pada immediate addresing mode langsung berada pada byte setelah OpCode. Jumlah byte tergantung dari register mana yang akan digunakan, sehingga instruksinya dapat berupa instruksi dua, tiga atau empat byte.

Contoh :
LDAA #$3A
bilangan heksa $3A diambil ke Accu A.

LDAA #22
bilangan desimal 22 diambil ke Accu A.

LDAA #@22
bilangan octal 22 diambil ke Accu A.

LDAA #'A
karakter ASCII A diambil ke Accu A.

EORB #$34
ex-or bilangan heksa $34 dengan Accu B.

CMPA #%1001
membandingkan isi Accu A dengan bil Biner.

LDD #$1234
bilangan heksa $1234 diambil ke Accu D

Penulisan operand harus dimulai dengan menulis karakter '#', yang digunakan oleh assembler untuk mendeteksi bahwa mode yang digunakan adalah IMM.

catatan :	Awalan	Definisi
none	bilangan desimal	
$	bilangan heksa Desimal	
@	bilangan octal	
%	bilangan biner	
'	Satu Karakter ASCII	

EXT
(extended) Instruksi ini berhubungan langsung dengan lokasi atau alamat memori yang isinya akan diolah. Instruksi ini terdiri dari tiga atau empat Byte yaitu satu atau dua byte berupa OpCode sedangkan dua Byte berikutnya berupa alamat.

Contoh :

LDAA $2000
Isi dari alamat memori $2000 diambil ke Accu A (Isi dari memori tetap)

STAB $1002
Isi dari Accu B diletakan pada lokasi memori $1000(isi dari Accu B tetap)

IND

(indexed adressing) Instruksi ini berfungsi untuk mengambil atau meletakan data dari/ke memori, sedangkan alamat memori terlebih dahuli harus berada di indeks register (X atau Y). Alamat effektif sangat variatif tergantung dari isi IX atau IY 16 bit dan offset 8 bit.

Contoh :

LDX #$1000
Harga(yang dalam hal ini sbg. alamat) $1000 secara langsung diambil ke register X.
(sekarang isi register X =$1000)

STAB X
Isi dari Accu B diletakan pada lokasi memori yang alamatnya telah tersimpan di register X ($1000)

STAB ,X
Isi dari Accu B diletakan pada lokasi memori yang alamatnya telah tersimpan diregister X ($1000) → sama dengan di atas

STAB 0,X
Isi dari Accu B diletakan pada lokasi memori yang alamatnya telah tersimpan di register X ($1000) → sama dengan di atas

STAB 4,X
Isi dari Accu B diletakan pada lokasi memori yang alamatnya telah tersimpan di register X+4 ($1000+$4) → $1004

STAB 8/2+6,X
Isi dari Accu B diletakan pada lokasi memori yang alamatnya telah tersimpan di register X+(8/2+6) → $100A

DIR

(direct) Pengalamatan langsung hanya memungkinkan didaerah $0000..$00FF

Contoh :

LDAA $3B
Isi dari alamat memori $003B diambil ke Accu A.

REL

(relativ) Hanya digunakan untuk percabangan (Branch) dari Program.Daerah Offset adalah -128 sampai +127

Contoh :

BRA 03
Selalu loncat 3 Byte (lokasi memori) ke atas (kearah alamat yang lebih tinggi)

Operand
Informasi tambahan yang diperlukan oleh OpCode yang dapat berupa (Alamat, Data, atau Bitmask).

Perhatikan :
ii 1Byte harga bilangan
ll LSB dari alamat
hh MSB dari alamat
dd Alamat di dalam DIR Mode
ff 8 Bit Offset Positip
MSB Most Significant Byte / Bit (Byte/Bit tertinggi)
LSB Last Significant Byte / Bit (Byte/Bit terendah)

Bytes
Jumlah memori yang digunakan untuk satu instruksi.

Cycles
Jumlah/hitungan E-Clock Cycles, yang digunakan untuk pelaksanaan instruksi.

8.12.3. Instruksi Transfer Data

Kebanyakan operasi transfer data didapat dengan menggunakan instruksi LD (load). Data dapat ditransfer dalam unit-unit 8 bit atau 16 bit. Instruksi-instruksi seperti TBA, TAB, LDAA, LDAB, STAA ataupun STAB adalah menstransfer data dalam 8 bit sedangakan untuk transfer data 16 bit biasanya digunakan XGDX, XGDY, LDD, LDX, LDY, STD, STX dan STY.

Kemungkinan arah transfer data adalah:

Dari akkumulator ke akkumulator, misalnya TBA, TAB

Bertukar data antara akkumulator dan register, misalnya XGDX, XGDY

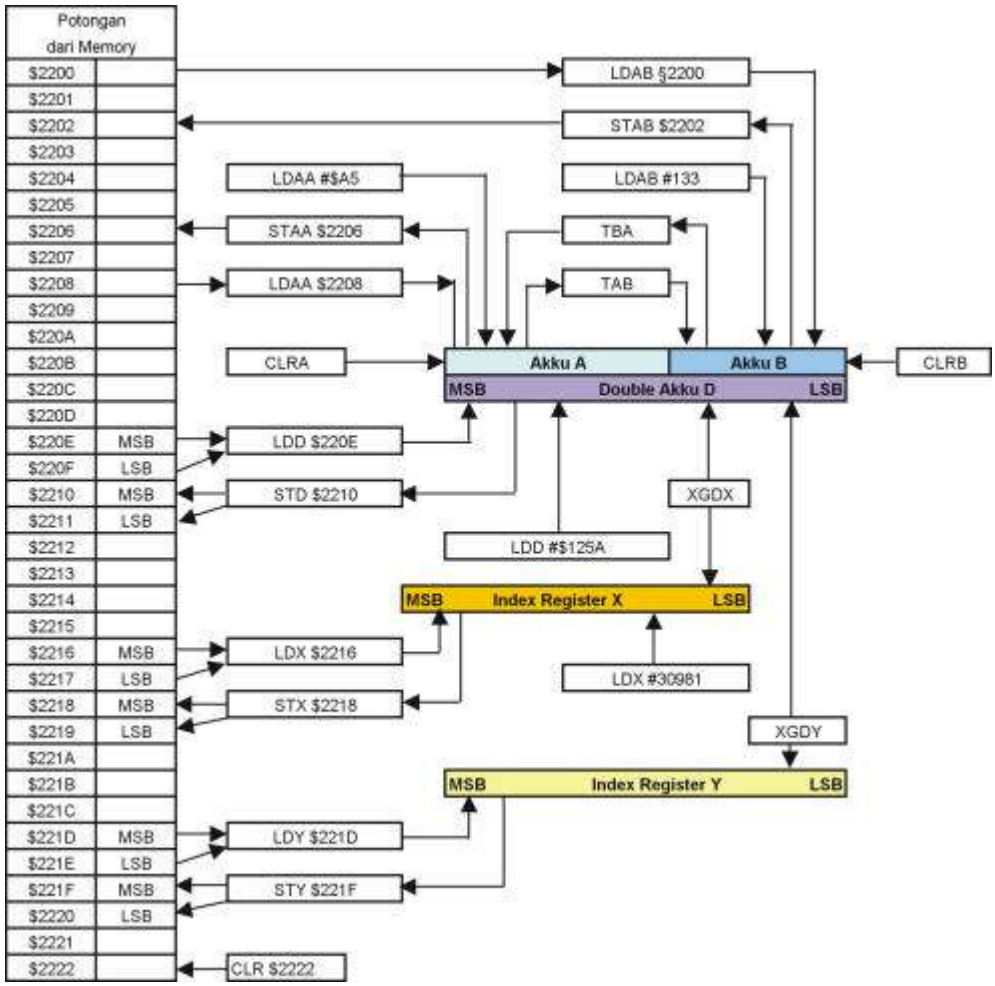
Dari akkumulator ke memory, misalnya STAA $1000, STAB $3000, STX $2200, STY $3400, STD $3454

Dari memory ke register, misalnya LDX $2000, LDY $2000

Dari memory ke akkumulator, misalnya LDAA $2000, LDAB $3457, LDD $3000

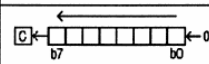
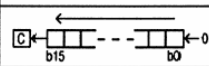
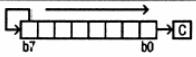
Dari data langsung ke akkumulator, misalnya LDAA #$01, LDAB #$34, LDD #$123A

Dari data langsung ke register, misalnya LDX #$ABCD, LDY #$8976



Gambar 8.30 Blok Diagram Instruksi Transfer

Tabel 8.14 Daftar Instruksi MC68HC11F1

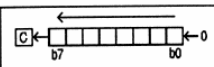
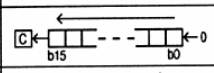
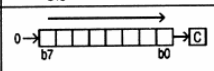
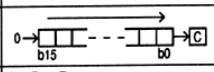
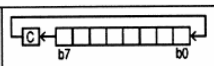
Source Forms	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal) Opcode	Operand(s)	Bytes	Cycles	Condition Codes S X H I N Z V C
ABA	Add Accumulators	A + B → A	INH	1B		1	2	— — Δ — Δ Δ Δ Δ
ABX	Add B to X	IX + 00:B → IX	INH	3A		1	3	— — — — — — — —
ABY	Add B to Y	IY + 00:B → IY	INH	18 3A		2	4	— — — — — — — —
ADCA (opr)	Add with Carry to A	A + M + C → A	A IMM	89	ii	2	2	— — Δ — Δ Δ Δ Δ
			A DIR	99	dd	2	3	
			A EXT	B9	hh ll	3	4	
			A IND,X	A9	ff	2	4	
			A IND,Y	18 A9	ff	3	5	
ADCB (opr)	Add with Carry to B	B + M + C → B	B IMM	C9	ii	2	2	— — Δ — Δ Δ Δ Δ
			B DIR	D9	dd	2	3	
			B EXT	F9	hh ll	3	4	
			B IND,X	E9	ff	2	4	
			B IND,Y	18 E9	ff	3	5	
ADDA (opr)	Add Memory to A	A + M → A	A IMM	8B	ii	2	2	— — Δ — Δ Δ Δ Δ
			A DIR	9B	dd	2	3	
			A EXT	BB	hh ll	3	4	
			A IND,X	AB	ff	2	4	
			A IND,Y	18 AB	ff	3	5	
ADDB (opr)	Add Memory to B	B + M → B	B IMM	CB	ii	2	2	— — Δ — Δ Δ Δ Δ
			B DIR	DB	dd	2	3	
			B EXT	FB	hh ll	3	4	
			B IND,X	EB	ff	2	4	
			B IND,Y	18 EB	ff	3	5	
ADDD (opr)	Add 16-Bit to D	D + M:M + 1 → D	IMM	C3	jj kk	3	4	— — — — Δ Δ Δ Δ
			DIR	D3	dd	2	5	
			EXT	F3	hh ll	3	6	
			IND,X	E3	ff	2	6	
			IND,Y	18 E3	ff	3	7	
ANDA (opr)	AND A with Memory	A • M → A	A IMM	84	ii	2	2	— — — — Δ Δ 0 —
			A DIR	94	dd	2	3	
			A EXT	B4	hh ll	3	4	
			A IND,X	A4	ff	2	4	
			A IND,Y	18 A4	ff	3	5	
ANDB (opr)	AND B with Memory	B • M → B	B IMM	C4	ii	2	2	— — — — Δ Δ 0 —
			B DIR	D4	dd	2	3	
			B EXT	F4	hh ll	3	4	
			B IND,X	E4	ff	2	4	
			B IND,Y	18 E4	ff	3	5	
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	3	6	— — — — Δ Δ Δ Δ
			IND,X	68	ff	2	6	
			IND,Y	18 68	ff	3	7	
			A INH	48		1	2	
			B INH	58		1	2	
ASLD	Arithmetic Shift Left Double		INH	05		1	3	— — — — Δ Δ Δ Δ
ASR (opr)	Arithmetic Shift Right		EXT	77	hh ll	3	6	— — — — Δ Δ Δ Δ
			IND,X	67	ff	2	6	
			IND,Y	18 67	ff	3	7	
			A INH	47		1	2	
			B INH	57		1	2	
BCC (rel)	Branch if Carry Clear	?C = 0	REL	24	rr	2	3	— — — — — — — —
BCLR (opr) (msk)	Clear Bit(s)	M • (mm) → M	DIR	15	dd mm	3	6	— — — — Δ Δ 0 —
			IND,X	1D	ff mm	3	7	
			IND,Y	18 1D	ff mm	4	8	
BCS (rel)	Branch if Carry Set	?C = 1	REL	25	rr	2	3	— — — — — — — —
BEQ (rel)	Branch if = Zero	?Z = 1	REL	27	rr	2	3	— — — — — — — —
BGE (rel)	Branch if ≥ Zero	?N ⊕ V = 0	REL	2C	rr	2	3	— — — — — — — —
BGT (rel)	Branch if > Zero	?Z + (N ⊕ V) = 0	REL	2E	rr	2	3	— — — — — — — —
BHI (rel)	Branch if Higher	?C + Z = 0	REL	22	rr	2	3	— — — — — — — —
BHS (rel)	Branch if Higher or Same	?C = 0	REL	24	rr	2	3	— — — — — — — —
BITA (opr)	Bit(s) Test A with Memory	A • M	A IMM	85	ii	2	2	— — — — Δ Δ 0 —
			A DIR	95	dd	2	3	
			A EXT	B5	hh ll	3	4	
			A IND,X	A5	ff	2	4	
			A IND,Y	18 A5	ff	3	5	
BITB (opr)	Bit(s) Test B with Memory	B • M	B IMM	C5	ii	2	2	— — — — Δ Δ 0 —
			B DIR	D5	dd	2	3	
			B EXT	F5	hh ll	3	4	
			B IND,X	E5	ff	2	4	
			B IND,Y	18 E5	ff	3	5	

BLE	(rel)	Branch if ≤ Zero	?Z + (N ⊕ V) = 1		REL	2F	rr			2	3	—	—	—	—	—	—	—	—
BLO	(rel)	Branch if Lower	?C = 1		REL	25	rr			2	3	—	—	—	—	—	—	—	—
BLS	(rel)	Branch if Lower or Same	?C + Z = 1		REL	23	rr			2	3	—	—	—	—	—	—	—	—
BLT	(rel)	Branch if < Zero	?N ⊕ V = 1		REL	2D	rr			2	3	—	—	—	—	—	—	—	—
BMI	(rel)	Branch if Minus	?N = 1		REL	2B	rr			2	3	—	—	—	—	—	—	—	—
BNE	(rel)	Branch if Not = Zero	?Z = 0		REL	26	rr			2	3	—	—	—	—	—	—	—	—
BPL	(rel)	Branch if Plus	?N = 0		REL	2A	rr			2	3	—	—	—	—	—	—	—	—
BRA	(rel)	Branch Always	?1 = 1		REL	20	rr			2	3	—	—	—	—	—	—	—	—
BRCLR	(opr) (msk) (rel)	Branch if Bit(s) Clear	?M • mm = 0		DIR IND,X IND,Y	13 1F 18 1F	dd ff ff	mm mm mm	rr rr rr	4 4 5	6 7 8	—	—	—	—	—	—	—	—
BRN	(rel)	Branch Never	?1 = 0		REL	21	rr			2	3	—	—	—	—	—	—	—	—
BRSET	(opr) (msk) (rel)	Branch if Bit(s) Set	?(M) • mm = 0		DIR IND,X IND,Y	12 1E 18 1E	dd ff ff	mm mm mm	rr rr rr	4 4 5	6 7 8	—	—	—	—	—	—	—	—

BSET	(opr) (msk)	Set Bit(s)	M + mm → M		DIR IND,X IND,Y	14 1C 18 1C	dd ff ff	mm mm mm		3 3 4	6 7 8	—	—	—	Δ	Δ	0	—
BSR	(rel)	Branch to Subroutine	See Special Ops		REL	8D	rr			2	6	—	—	—	—	—	—	—	—
BVC	(rel)	Branch if Overflow Clear	?V = 0		REL	28	rr			2	3	—	—	—	—	—	—	—	—
BVS	(rel)	Branch if Overflow Set	?V = 1		REL	29	rr			2	3	—	—	—	—	—	—	—	—
CBA		Compare A to B	A – B		INH	11				1	2	—	—	—	Δ	Δ	Δ	Δ
CLC		Clear Carry Bit	0 → C		INH	0C				1	2	—	—	—	—	—	—	—	0
CLI		Clear Interrupt Mask	0 → I		INH	0E				1	2	—	—	—	0	—	—	—	—
CLR	(opr)	Clear Memory Byte	0 → M		EXT IND,X IND,Y	7F 6F 18 6F	hh ff ff	ll		3 2 3	6 6 7	—	—	—	0	1	0	0
CLRA		Clear Accumulator A	0 → A	A	INH	4F				1	2	—	—	—	0	1	0	0
CLRB		Clear Accumulator B	0 → B	B	INH	5F				1	2	—	—	—	0	1	0	0
CLV		Clear Overflow Flag	0 → V		INH	0A				1	2	—	—	—	—	—	—	0	—
CMPA	(opr)	Compare A to Memory	A – M	A A A A A	IMM DIR EXT IND,X IND,Y	81 91 B1 A1 18 A1	ii dd hh ff ff	ll		2 2 3 2 3	2 3 4 4 5	—	—	—	Δ	Δ	Δ	Δ
CMPB	(opr)	Compare B to Memory	B – M	B B B B B	IMM DIR EXT IND,X IND,Y	C1 D1 F1 E1 18 E1	ii dd hh ff ff	ll		2 2 3 2 3	2 3 4 4 5	—	—	—	Δ	Δ	Δ	Δ
COM	(opr)	1's Complement Memory Byte	$FF – M → M		EXT IND,X IND,Y	73 63 18 63	hh ff ff	ll		3 2 3	6 6 7	—	—	—	Δ	Δ	0	1
COMA		1's Complement A	$FF – A → A	A	INH	43				1	2	—	—	—	Δ	Δ	0	1
COMB		1's Complement B	$FF – B → B	B	INH	53				1	2	—	—	—	Δ	Δ	0	1
CPD	(opr)	Compare D to Memory 16-Bit	D – M:M + 1		IMM DIR EXT IND,X IND,Y	1A 83 1A 93 1A B3 1A A3 CD A3	jj dd hh ff ff	kk ll		4 3 4 3 3	5 6 7 7 7	—	—	—	Δ	Δ	Δ	Δ

CPX	(opr)	Compare X to Memory 16-Bit	IX – M:M + 1		IMM DIR EXT IND,X IND,Y	8C 9C BC AC CD AC	jj dd hh ff ff	kk ll		3 2 3 2 3	4 5 6 6 7	—	—	—	Δ	Δ	Δ	Δ
CPY	(opr)	Compare Y to Memory 16-Bit	IY – M:M + 1		IMM DIR EXT IND,X IND,Y	18 8C 18 9C 18 BC 1A AC 18 AC	jj dd hh ff ff	kk ll		4 3 4 3 3	5 6 7 7 7	—	—	—	Δ	Δ	Δ	Δ
DAA		Decimal Adjust A	Adjust Sum to BCD		INH	19				1	2	—	—	—	Δ	Δ	Δ	Δ
DEC	(opr)	Decrement Memory Byte	M – 1 → M		EXT IND,X IND,Y	7A 6A 18 6A	hh ff ff	ll		3 2 3	6 6 7	—	—	—	Δ	Δ	Δ	—
DECA		Decrement Accumulator A	A – 1 → A	A	INH	4A				1	2	—	—	—	Δ	Δ	Δ	—
DECB		Decrement Accumulator B	B – 1 → B	B	INH	5A				1	2	—	—	—	Δ	Δ	Δ	—
DES		Decrement Stack Pointer	SP – 1 → SP		INH	34				1	3	—	—	—	—	—	—	—
DEX		Decrement Index Register X	IX – 1 → IX		INH	09				1	3	—	—	—	—	—	Δ	—	—

DEY	Decrement Index Register Y	IY − 1 → IY	INH	18 09		2	4	— — — — — Δ — —
EORA (opr)	Exclusive OR A with Memory	A ⊕ M → A	A IMM	88	ii	2	2	— — — — Δ Δ 0 —
			A DIR	98	dd	2	3	
			A EXT	B8	hh ll	3	4	
			A IND,X	A8	ff	2	4	
			A IND,Y	18 A8	ff	3	5	
EORB (opr)	Exclusive OR B with Memory	B ⊕ M → B	B IMM	C8	ii	2	2	— — — — Δ Δ 0 —
			B DIR	D8	dd	2	3	
			B EXT	F8	hh ll	3	4	
			B IND,X	E8	ff	2	4	
			B IND,Y	18 E8	ff	3	5	
FDIV	Fractional Divide 16 by 16	D/IX → IX; r → D	INH	03		1	41	— — — — — Δ Δ Δ
IDIV	Integer Divide 16 by 16	D/IX → IX; r → D	INH	02		1	41	— — — — — Δ 0 Δ
INC (opr)	Increment Memory Byte	M + 1 → M	EXT	7C	hh ll	3	6	— — — — Δ Δ Δ —
			IND,X	6C	ff	2	6	
			IND,Y	18 6C	ff	3	7	
INCA	Increment Accumulator A	A + 1 → A	A INH	4C		1	2	— — — — Δ Δ Δ —
INCB	Increment Accumulator B	B + 1 → B	B INH	5C		1	2	— — — — Δ Δ Δ —
INS	Increment Stack Pointer	SP + 1 → SP	INH	31		1	3	— — — — — — — —
INX	Increment Index Register X	IX + 1 → IX	INH	08		1	3	— — — — — Δ — —
INY	Increment Index Register Y	IY + 1 → IY	INH	18 08		2	4	— — — — — Δ — —
JMP (opr)	Jump	See Special Ops	EXT	7E	hh ll	3	3	— — — — — — — —
			IND,X	6E	ff	2	3	
			IND,Y	18 6E	ff	3	4	
JSR (opr)	Jump to Subroutine	See Special Ops	DIR	9D	dd	2	5	— — — — — — — —
			EXT	BD	hh ll	3	6	
			IND,X	AD	ff	2	6	
			IND,Y	18 AD	ff	3	7	
LDAA (opr)	Load Accumulator A	M → A	A IMM	86	ii	2	2	— — — — Δ Δ 0 —
			A DIR	96	dd	2	3	
			A EXT	B6	hh ll	3	4	
			A IND,X	A6	ff	2	4	
			A IND,Y	18 A6	ff	3	5	
LDAB (opr)	Load Accumulator B	M → B	B IMM	C6	ii	2	2	— — — — Δ Δ 0 —
			B DIR	D6	dd	2	3	
			B EXT	F6	hh ll	3	4	
			B IND,X	E6	ff	2	4	
			B IND,Y	18 E6	ff	3	5	
LDD (opr)	Load Double Accumulator D	M → A, M + 1 → B	IMM	CC	jj kk	3	3	— — — — Δ Δ 0 —
			DIR	DC	dd	2	4	
			EXT	FC	hh ll	3	5	
			IND,X	EC	ff	2	5	
			IND,Y	18 EC	ff	3	6	
LDS (opr)	Load Stack Pointer	M:M + 1 → SP	IMM	8E	jj kk	3	3	— — — — Δ Δ 0 —
			DIR	9E	dd	2	4	
			EXT	BE	hh ll	3	5	
			IND,X	AE	ff	2	5	
			IND,Y	18 AE	ff	3	6	
LDX (opr)	Load Index Register X	M:M + 1 → IX	IMM	CE	jj kk	3	3	— — — — Δ Δ 0 —
			DIR	DE	dd	2	4	
			EXT	FE	hh ll	3	5	
			IND,X	EE	ff	2	5	
			IND,Y	CD EE	ff	3	6	
LDY (opr)	Load Index Register Y	M:M + 1 → IY	IMM	18 CE	jj kk	4	4	— — — — Δ Δ 0 —
			DIR	18 DE	dd	3	5	
			EXT	18 FE	hh ll	4	6	
			IND,X	1A EE	ff	3	6	
			IND,Y	18 EE	ff	3	6	

LSL (opr)	Logical Shift Left			EXT	78	hh	ll	3	6	—	—	—	—	Δ	Δ	Δ	Δ
				IND,X	68	ff		2	6								
				IND,Y	18 68	ff		3	7								
			A	INH	48			1	2								
			B	INH	58			1	2								
LSLD	Logical Shift Left Double			INH	05			1	3	—	—	—	—	Δ	Δ	Δ	Δ
LSR (opr)	Logical Shift Right			EXT	74	hh	ll	3	6	—	—	—	—	0	Δ	Δ	Δ
				IND,X	64	ff		2	6								
				IND,Y	18 64	ff		3	7								
			A	INH	44			1	2								
			B	INH	54			1	2								
LSRD	Logical Shift Right Double			INH	04			1	3	—	—	—	—	0	Δ	Δ	Δ
MUL	Multiply 8 by 8	A × B → D		INH	3D			1	10	—	—	—	—	—	—	—	Δ
NEG (opr)	2's Complement Memory Byte	0 − M → M		EXT	70	hh	ll	3	6	—	—	—	—	Δ	Δ	Δ	Δ
				IND,X	60	ff		2	6								
				IND,Y	18 60	ff		3	7								
NEGA	2's Complement A	0 − A → A	A	INH	40			1	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGB	2's Complement B	0 − B → B	B	INH	50			1	2	—	—	—	—	Δ	Δ	Δ	Δ
NOP	No Operation	No Operation		INH	01			1	2	—	—	—	—	—	—	—	—
ORAA (opr)	OR Accumulator A (Inclusive)	A + M → A	A	IMM	8A	ii		2	2	—	—	—	—	Δ	Δ	0	—
			A	DIR	9A	dd		2	3								
			A	EXT	BA	hh	ll	3	4								
			A	IND,X	AA	ff		2	4								
			A	IND,Y	18 AA	ff		3	5								
ORAB (opr)	OR Accumulator B (Inclusive)	B + M → B	B	IMM	CA	ii		2	2	—	—	—	—	Δ	Δ	0	—
			B	DIR	DA	dd		2	3								
			B	EXT	FA	hh	ll	3	4								
			B	IND,X	EA	ff		2	4								
			B	IND,Y	18 EA	ff		3	5								
PSHA	Push A onto Stack	A → Stk, SP = SP − 1	A	INH	36			1	3	—	—	—	—	—	—	—	—
PSHB	Push B onto Stack	B → Stk, SP = SP − 1	B	INH	37			1	3	—	—	—	—	—	—	—	—
PSHX	Push X onto Stack (Lo First)	IX → Stk, SP = SP − 2		INH	3C			1	4	—	—	—	—	—	—	—	—
PSHY	Push Y onto Stack (Lo First)	IY → Stk, SP = SP − 2		INH	18 3C			2	5	—	—	—	—	—	—	—	—
PULA	Pull A from Stack	SP = SP + 1, A ← Stk	A	INH	32			1	4	—	—	—	—	—	—	—	—
PULB	Pull B from Stack	SP = SP + 1, B ← Stk	B	INH	33			1	4	—	—	—	—	—	—	—	—
PULX	Pull X from Stack (Hi First)	SP = SP + 2, IX ← Stk		INH	38			1	5	—	—	—	—	—	—	—	—
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY ← Stk		INH	18 38			2	6	—	—	—	—	—	—	—	—
ROL (opr)	Rotate Left			EXT	79	hh	ll	3	6	—	—	—	—	Δ	Δ	Δ	Δ
				IND,X	69	ff		2	6								
				IND,Y	18 69	ff		3	7								
			A	INH	49			1	2								
			B	INH	59			1	2								
ROR (opr)	Rotate Right			EXT	76	hh	ll	3	6	—	—	—	—	Δ	Δ	Δ	Δ
				IND,X	66	ff		2	6								
				IND,Y	18 66	ff		3	7								
			A	INH	46			1	2								
			B	INH	56			1	2								
RTI	Return from Interrupt	See Special Ops		INH	3B			1	12	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	Return from Subroutine	See Special Ops		INH	39			1	5	—	—	—	—	—	—	—	—
SBA	Subtract B from A	A − B → A		INH	10			1	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA (opr)	Subtract with Carry from A	A − M − C → A	A	IMM	82	ii		2	2	—	—	—	—	Δ	Δ	Δ	Δ
			A	DIR	92	dd		2	3								
			A	EXT	B2	hh	ll	3	4								
			A	IND,X	A2	ff		2	4								
			A	IND,Y	18 A2	ff		3	5								
SBCB (opr)	Subtract with Carry from B	B − M − C → B	B	IMM	C2	ii		2	2	—	—	—	—	Δ	Δ	Δ	Δ
			B	DIR	D2	dd		2	3								
			B	EXT	F2	hh	ll	3	4								
			B	IND,X	E2	ff		2	4								
			B	IND,Y	18 E2	ff		3	5								

SEC	Set Carry	1 → C		INH	0D			1	2	—	—	—	—	—	—	—	1
SEI	Set Interrupt Mask	1 → I		INH	0F			1	2	—	—	—	1	—	—	—	—
SEV	Set Overflow Flag	1 → V		INH	0B			1	2	—	—	—	—	—	—	1	—
STAA (opr)	Store Accumulator A	A → M	A	DIR	97	dd		2	3	—	—	—	—	Δ	Δ	0	—
			A	EXT	B7	hh	ll	3	4								
			A	IND,X	A7	ff		2	4								
			A	IND,Y	18 A7	ff		3	5								
STAB (opr)	Store Accumulator B	B → M	B	DIR	D7	dd		2	3	—	—	—	—	Δ	Δ	0	—
			B	EXT	F7	hh	ll	3	4								
			B	IND,X	E7	ff		2	4								
			B	IND,Y	18 E7	ff		3	5								
STD (opr)	Store Accumulator D	A → M, B → M + 1		DIR	DD	dd		2	4	—	—	—	—	Δ	Δ	0	—
				EXT	FD	hh	ll	3	5								
				IND,X	ED	ff		2	5								
				IND,Y	18 ED	ff		3	6								
STOP	Stop Internal Clocks			INH	CF			1	2	—	—	—	—	—	—	—	—
STS (opr)	Store Stack Pointer	SP → M:M + 1		DIR	9F	dd		2	4	—	—	—	—	Δ	Δ	0	—
				EXT	BF	hh	ll	3	5								
				IND,X	AF	ff		2	5								
				IND,Y	18 AF	ff		3	6								
STX (opr)	Store Index Register X	IX → M:M + 1		DIR	DF	dd		2	4	—	—	—	—	Δ	Δ	0	—
				EXT	FF	hh	ll	3	5								
				IND,X	EF	ff		2	5								
				IND,Y	CD EF	ff		3	6								
STY (opr)	Store Index Register Y	IY → M:M + 1		DIR	18 DF	dd		3	5	—	—	—	—	Δ	Δ	0	—
				EXT	18 FF	hh	ll	4	6								
				IND,X	1A EF	ff		3	6								
				IND,Y	18 EF	ff		3	6								

SUBA (opr)	Subtract Memory from A	A – M → A	A	IMM	80	ii		2	2	—	—	—	—	Δ	Δ	Δ	Δ
			A	DIR	90	dd		2	3								
			A	EXT	B0	hh	ll	3	4								
			A	IND,X	A0	ff		2	4								
			A	IND,Y	18 A0	ff		3	5								
SUBB (opr)	Subtract Memory from B	B – M → B	B	IMM	C0	ii		2	2	—	—	—	—	Δ	Δ	Δ	Δ
			B	DIR	D0	dd		2	3								
			B	EXT	F0	hh	ll	3	4								
			B	IND,X	E0	ff		2	4								
			B	IND,Y	18 E0	ff		3	5								
SUBD (opr)	Subtract Memory from D	D – M:M + 1 → D		IMM	83	jj	kk	3	4	—	—	—	—	Δ	Δ	Δ	Δ
				DIR	93	dd		2	5								
				EXT	B3	hh	ll	3	6								
				IND,X	A3	ff		2	6								
				IND,Y	18 A3	ff		3	7								
SWI	Software Interrupt	See Special Ops		INH	3F			1	14	—	—	—	1	—	—	—	—
TAB	Transfer A to B	A → B		INH	16			1	2	—	—	—	—	Δ	Δ	0	—
TAP	Transfer A to CCR	A → CCR		INH	06			1	2	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
TBA	Transfer B to A	B → A		INH	17			1	2	—	—	—	—	Δ	Δ	0	—
TEST	TEST (Only in Test Modes)	Address Bus Counts		INH	00			1	*	—	—	—	—	—	—	—	—
TPA	Transfer CCR to A	CCR → A		INH	07			1	2	—	—	—	—	—	—	—	—
TST (opr)	Test for Zero or Minus	M – 0		EXT	7D	hh	ll	3	6	—	—	—	—	Δ	Δ	0	0
				IND,X	6D	ff		2	6								
				IND,Y	18 6D	ff		3	7								
TSTA		A – 0	A	INH	4D			1	2	—	—	—	—	Δ	Δ	0	0
TSTB		B – 0	B	INH	5D			1	2	—	—	—	—	Δ	Δ	0	0
TSX	Transfer Stack Pointer to X	SP + 1 → IX		INH	30			1	3	—	—	—	—	—	—	—	—
TSY	Transfer Stack Pointer to Y	SP + 1 → IY		INH	18 30			2	4	—	—	—	—	—	—	—	—
TXS	Transfer X to Stack Pointer	IX – 1 → SP		INH	35			1	3	—	—	—	—	—	—	—	—
TYS	Transfer Y to Stack Pointer	IY – 1 → SP		INH	18 35			2	4	—	—	—	—	—	—	—	—
WAI	Wait for Interrupt	Stack Regs and WAIT		INH	3E			1	* *	—	—	—	—	—	—	—	—
XGDX	Exchange D with X	IX → D, D → IX		INH	8F			1	3	—	—	—	—	—	—	—	—
XGDY	Exchange D with Y	IY → D, D → IY		INH	18 8F			2	4	—	—	—	—	—	—	—	—

NOTES:

Cycle:
- * = Infinity or until reset occurs
- ** = 12 cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycle (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (total = 14 + n).

Operands:
- dd = 8-bit direct address $0000–$00FF. (High byte assumed to be $00.)
- ff = 8-bit positive offset $00 (0) to $FF (255) added to index.
- hh = High order byte of 16-bit extended address.
- ii = One byte of immediate data.
- jj = High order byte of 16-bit immediate data.
- kk = Low order byte of 16-bit immediate data.
- ll = Low order byte of 16-bit extended address.
- mm = 8-bit mask (set bits to be affected).
- rr = Signed relative offset $80 (–128) to $7F (+127). Offset relative to the address following the machine code offset byte.

Condition Codes:
- — = Bit not changed
- 0 = Always cleared (logic 0).
- 1 = Always set (logic 1).
- Δ = Bit cleared or set depending on operation.
- ↓ = Bit may be cleared, cannot become set.

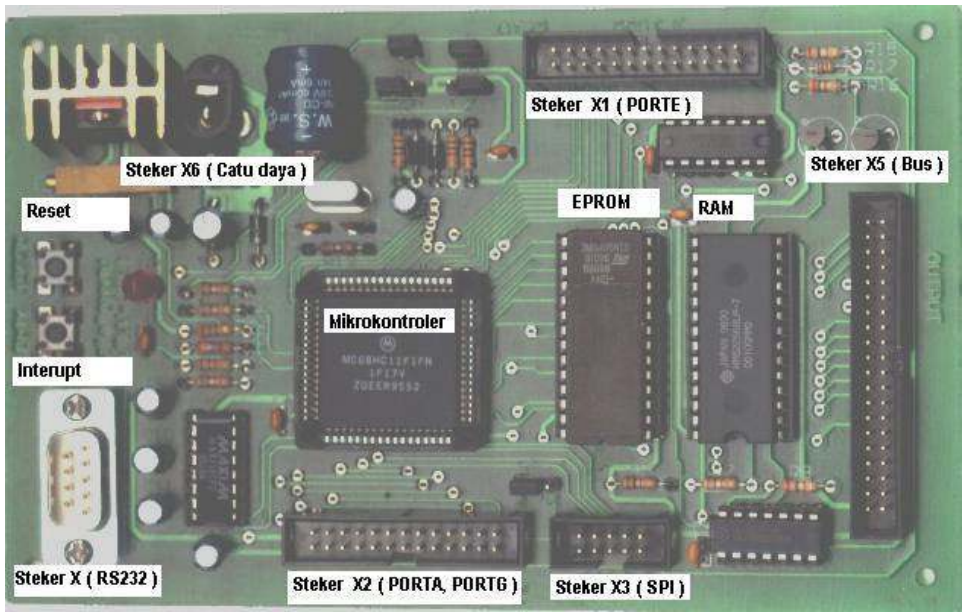
8.13. Modul Mikrokontroler VEDCLEMPS

Agar chip IC mikrokontroler dapat dipergunakan untuk berbagai keperluan, IC mikrokontroler harus dirangkai pada suatu board dan harus dilengkapi dengan rangkaian pendukung agar MCU tersebut dapat berinterkasi dengan banyak peralatan. Pada bagian ini kita akan menggunakan MCU yang dirangkai pada suatu board dengan mode expanded yang disebut dengan VEDCLEMPS

VEDCLEMPS adalah modul mikrokontroler yang dibangun dari chip IC MC68HC11F1 (Motorola) dalam mode ”EXPANDED” yang dilengkapi dengan extended RAM 32 KByte dan EPROM 32 KByte, dikembangkan bersama dengan Herr Bruno Warnister dari GIB Bern Switzerland.

Modul ini dilengkapi dengan software VEDCLEMPSWIN ditulis dengan software DELPHI di bawah operasi windows yang dalam penampakannya pada layar monitor (MENU dan keterangan lainnya) berbahasa Indonesia.

VEDCLEMPSWIN memungkinkan pembuatan program aplikasi menjadi lebih mudah dan menrik untuk segala kebutuhan baik di dunia industri maupun untuk keperluan pendidikan dan pelatihan di sekolah dan perguruan tinggi.



Gambar 8.31 Modul Mikrokontroler VEDCLEMPS

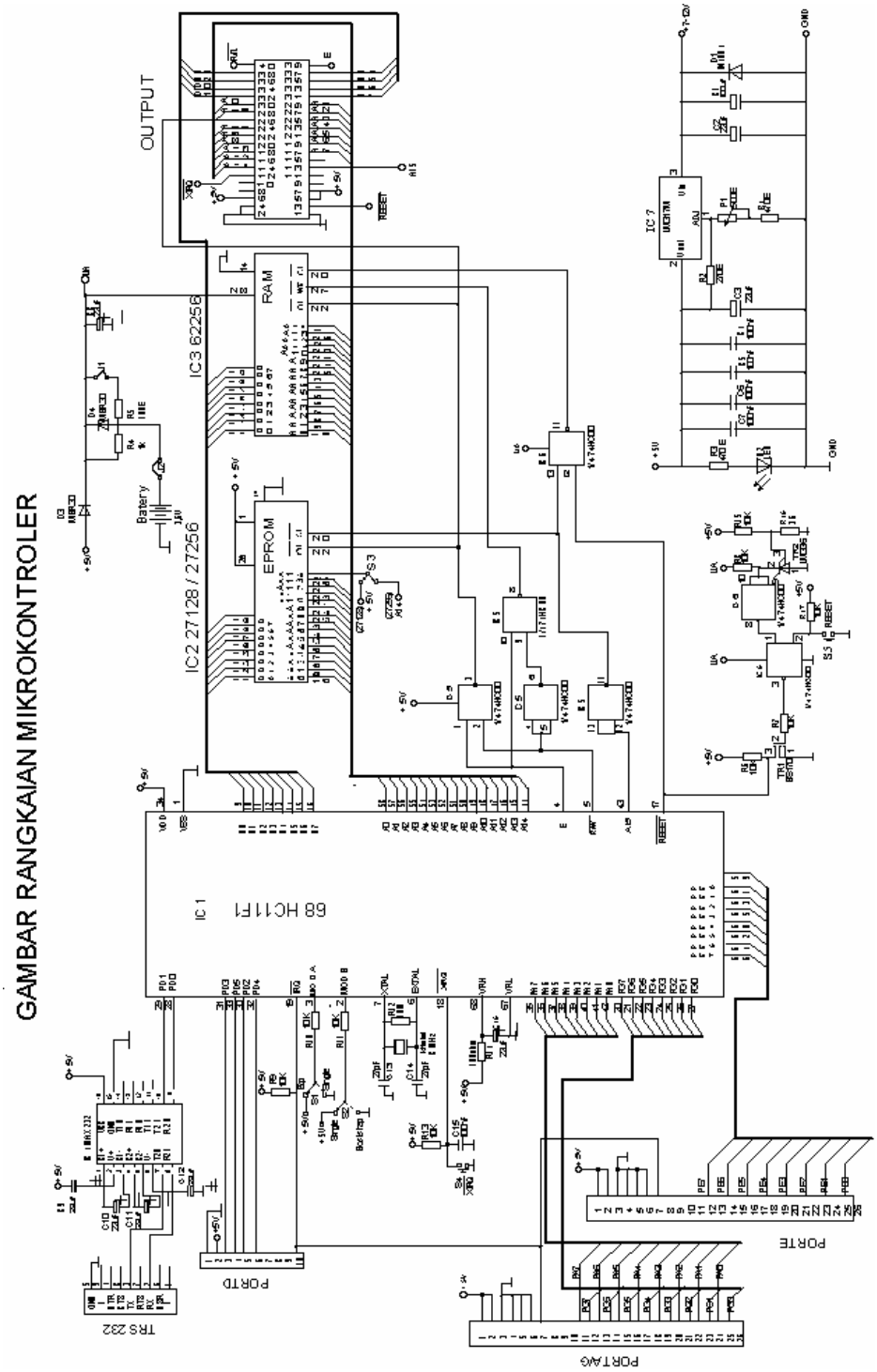
Dengan software ini kita dapat menulis, mengedit, menyimpan, meng-compile serta Download Program Assembler dari Personal Computer ke modul Microcontroller melalui sambungan serial PORT RS232.
System mikrokontroler pada dasarnya diprogram dengan bahasa Assembler, tetapi dapat pula dengan bahasa C atau Pascal yang kemudian diubah ke dalam kode-kode mikrokontroler yang sesuai.
Hampir pada semua 8 bit mikrokontroler mempunyai bangun yang hampir sama. Bagian yang paling utama adalah CPU (Central Prosessing Unit). CPU menginterprestasikan kode-kode pemrograman, mengatur jalannya program serta melaksanakan operasi aritmetik dan operasi logika di dalam ALU (Aritmetik Logic Unit).
Tidak semua CPU dapat dioperasikan dnegan bahasa Assembler yang sama, tetapi tergantung dari pabrik pembuatnya. Untuk famili Motorola MC68HC11 dapat digunakan bahasa Assembler yang dikeluarkan (Freeware) secara khusus. Dengan freeware ini VEDCLEMPSWIN for windows dikemas menjadi software pemrograman yang menarik untuk pembuatan program-program mikrokontroler.

8.13.1. Software VEDCLEMPS

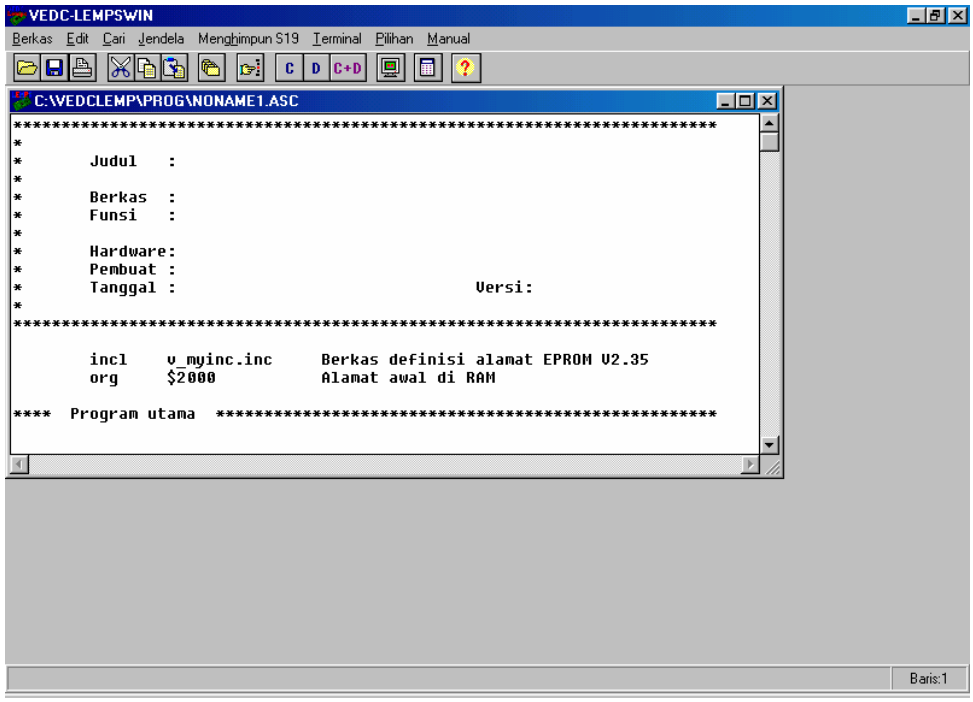
Mikrokontroler VEDCLEMPS dilengkapi dengan software VEDCLEMPSWIN For Windows yang tersedia dalam satu disket HD 1,44

MB. Untuk menginstall software ini, kita hanya memasukan disket VEDCLEMPS ke drive A dan jalankan file SETUP.EXE
yang terdapat pada direktori A:\VEDCLEMP.
Dari hasil setup, kita akan mendapatkan program VEDCLEMPS berbahasa Indonesia yang dipergunakan untuk membuat program mikrokontroler.

Selain program utama VEDCLEMPS WIN, disertakan pula beberapa contoh program aplikasi mikrokontroler yang disediakan dalam direktori c:/VEDCLEMP/PROG. Pada direktori ini diperlihatkan contoh program digital untuk menyalakan deteran led, aplikasi pwm untuk program suara notasi lagu , mengakses LCD 4 baris 20 kolom, contoh progran interupt, lampu lalu lintas dan led matrik.
Selain program contoh diatas ,disertakan pula sebuah program aplikasi under windows Analag dan Digital Input/ Output Test yang tampil pada layar monitor sekaligus sambung melalui RS 232 ke modul mikrokontroler VEDCLEMPS.



Gambar 8.31 Rangkaian modul Mikrokontroller



Gambar 8.33 Jendela utama software VEDCLEMPS

Dari program aplikasi ini kita dapat mengkomunikasikan komputer dan mikrokontroler melalui RS232 untuk membaca dan mengirim data . Melalui tombol mouse kita dapat menghidupkan deretan led yang tersambung pada PORTA , membaca deretan saklar pada PORTG serta data analog yang masuk ke PORTE dari mikrokontroler VEDCLEMPS.

Gambar 8.34 Trainer mikrokontroler VEDCLEMPS

Modul Mikrokontroler VEDCLEMPS memungkinkan untuk dipergunakan sebagai alat pelatihan mikrokontroler yang ideal karena selain didukung oleh software yang baik juga dilengkapi dengan beberapa modul lain yang mendukung proses pembelajaran antara lain :

- Modul Sevent Segment, untuk display counter, jam, stop watch, scoring board.
- Modul Input Output Test, untuk membuat simulasi program besar, penampil biner 8 bit, masukan 8 bit serta aneka program deretan led.
- Modul Input Analog Test, untuk pembuatan program masukan analog, voltmeter, kecepatan motor, PWM, simulasi tegangan ke perubahan teperatur dan lainnya.
- Modul suara, untuk pembuatan program suara berupa alam, lagu dan PWM.
- Modul Led Matrik, untuk display teks panjang yang atau bergerak horizontal maupun vertikal serta segala animasi teks.
- Modul Motor Steper, untuk program penaturan putaran potor steper.
- Modul Model Lampu Lalu Lintas.

Selain dukungan hardware dengan tersedianya macam-macam modul percobaan serta program VEDCLEMPSWIN, pada extended EPROM yang terpasang pada modul mikrokontroler juga dilengkapi fasilitas tambahan program monitor yang lengkap dengan fungsi-fungsi yang sangat diperlukan dalam pembuatan program.

8.13.2. Program Bagian EPROM Versi 2.35

Tabel 8.14 Program Bagian EPROM Versi 2.35

Nama	Fungsi
BLINKER	Led pada PORTA kiri-kanan dengan tunda waktu selama 200 ms. Stop program -> tekan tombol reset.
PORT_GA	Test Program Input-Output Membaca data PORTG dan dikeluarkan ke PORTA in : PORTG out : PORTA
PORT_EA	Test Program Input-Output Membaca data PORTE dan dikeluarkan ke PORTA in : PORTE out : PORTA
TEST_ADC	Test Program Analog to Digital Converter Membaca data ADC kanal 1 dan dikeluarkan ke PORTA in : ADC kanal 1 out : PORTA
TEST_SPI	Test Program Input-Output melalui SPI dengan pin No. 3 MOSI dan 5

	MISO dihubung singkat. Membaca data PORTG dan dikeluarkan ke PORTA in : PORTG out : PORTA
XYACOPY	Mengkopi data sebanyak A Byte dari alamat X ke alamat Y. in : Akku A -> Banyaknya Byte in : Reg. X -> Alamat sumber in : Reg. Y -> Alamat tujuan
XkeBCD	Mengubah bilangan Heksa ke Desimal in : Reg. X -> data dalam Heks a out : Reg. X -> data dalam Desimal
XkeHeksa	Mengubah bilangan Desimal ke Heksa in : Reg. X -> data dalam Desimal out : Reg. X -> data dalam Heksa
Tunda500ms	Tunda waktu selama 500 mili detik in : - out : -
Tunda1s	Tunda waktu selama 1 detik in : - out : -
TundaXms	Tunda waktu selama X mili detik in : Reg. X -> data lamanya tunda waktu out : -
REGI	Menampilkan isi Akku dan Register ke layar Monitor. in : - out : -
gan_bar1	Kursor pada Mode Terminal turun satu baris in : - out : -
Tulis_M	Menulis pada layar monitor suatu teks in : Reg. X contoh ldx #kata kata fcc '"teks"' fcb 0 out : Layar monitor -> teks
Baca_Byte	Membaca dari PC karakter 1 Byte in : Karakter dari RS232 out : Akku B -> Karakter dalam ASCII
Baca_2Byte	Membaca 2 Byte ASCII dari PC ke 1 Byte Heksa in : Karakter dari RS232 2 Byte out : Akku A -> Heksa 1 Byte
Baca_4Byte	Membaca 4 Byte ASCII dari PC ke 2 Byte Heksa in : Karakter dari RS232 4 Byte out : Reg. X -> MSB dan LSB 2 Byte Heksa
Tulis_Byte	Memberi ke monitor karakter 1 byte ASCII in : Akku B -> Data dalam ASCII out : ke RS232
Tul_HekAscii	Merubah dari bilangan Heksa ke Ascii dan mengirimkannya ke Monitor

	in : Akku A -> Data Heksa yang akan diubah out : Akku A -> ASCII MSB Akku B -> ASCII LSB
Baca_Tulis	Membaca dan menulis ke monitor karakter 1 Byte ASCII in : Akku B -> karakter dari RS232 ASCII out : Akku B ke RS232 ASCII
HEKSA_Ascii	Merubah dari bilangan Heksa ke kode Ascii in : Akku A -> data dalam heksa out : A = ASCII MSB dan B = ASCII LSB
ASCII_Heksa	Merubah dari kode Ascii ke bilangan heksa in : A = ASCII MSB dan B = ASCII LSB out : Akku A -> data dalam heksa
PWM	Modulator Lebar Pulsa in : Reg. X -> Periode positip '1' (T1) in : Reg. Y -> Periode (T2) in : Akku A-> Bit pada PORTA keluaran PWM out : PORTA , bit yang dipilih Periode, T = 8 us * T2 Frekuensi, f = 1/T Hz. Dutycyle, D = T1/T2 * 100 %

8.13.3. Not Lagu VEDCLEMPS

VEDCLEMPS menyediakan not lagu 3 oktaf dilengkapi pula dengan not setengah dan beberapa tempo lambat (Largo) sampai ke tempo cepat (Marsmo) serta sela.

Not-not ini adalah sub program yang disimpan di dalam EPROM mulai alamat 8001 yang dapat dipanggil dengan perintah JSR.

Contoh :

```
        jsr     do1
        jsr     Moderato
        jsr     re1
        jsr     Moderato
        jsr     mi2
        jsr     Moderato
        rts
```

Daftar nama not penuh :

```
si0   do1   re1   mi1   fa1   sol1  la1   si1
      do2   re2   mi2   fa2   sol2  la2   si2
      do3   re3   mi3   fa3   sol3  la3   si3
```

Daftar nama not setengah :

di1	ri1	fi1	sel1	li1
di2	ri2	fi2	sel2	li2
di3	ri3	fi3	sel3	li3

Daftar tempo :

largo	(lambat)
modagio	
adagio	
moderato	
marsla	
marsgio	
marsada	
marsmo	(cepat)
garis	
sela	
sela1	
sela2	
sela3	

8.13.4. Program Bagian Liquid Crystal Display (LCD)

jsr InitDisp	Inisialisasi SPI Inisialisasi Tampilan
jsr WriteLCD	Menulis text / data pada LCD In: X = penunjuk lokasi string dengan kata dan karakter pengontrol
jsr AHexDes	Menampilkan isi Akku pada LCD dalam format Desimal In: A = Data dalam format Hexadesimal
jsr Curs_On	Menampilkan kursor pada posisi kursor
jsr Curs_Off	Mematikan kursor pada posisi kursor
jsr ClearLCD	Menghapus tampilan, pada kursor B1,C1
jsr LED_On	Menyalakan LED
jsr LED_Off	Memadamkan LED
jsr Back_On	Menyalakan Back ground
jsr Back_Off	Memadamkan Back ground
jsr SetCursor	Meletakkan Cursor pada posisi A In: A = Posisi kursor (tergantung dari jenis LCD yang digunakan seperti tabel dibawah ini)

LM093LN	Baris 1 : $00..$0F Baris 2 : $40..$27
LM032L	Baris 1 : $00..$14; Baris 2 : $40..$54

LM044L Baris 1 : $80..$93
Baris 2 : $C0..$D3
Baris 3 : $94..$A7
Baris 4 : $D4..$E7

Dengan tersedianya segala fasilitas yang diinstall pada komputer serta yang terdapat pada EPROM memungkinkan pengunaan mikrokontroler menjadi lebih luas untuk segala keperluan di industri dan pelatihan.

8.13.5. Port VEDCLEMPS

Steker X1 (PORTE)

+5V	1	2	+5V
GND	3	4	GND
GND	5	6	GND
	7	8	
	9	10	
	11	12	PE7/AN7
	13	14	PE6/AN6
	15	16	PE5/AN5
	17	18	PE4/AN4
	19	20	PE3/AN3
	21	22	PE2/AN2
	23	24	PE1/AN1
	25	26	PE0/AN0

Gambar 8.35 Konfigurasi Steker X1 PORTE VEDCLEMPS

Steker X2 (PORTA, PORTG)

+5V	1	2	+5V
GND	3	4	GND
GND	5	6	GND
	7	8	
	9	10	
PA7/PAI/OC1	11	12	PG7
PA6/OC2/OC1	13	14	PG6
PA5/OC3/OC1	15	16	PG5
PA4/OC4/OC1	17	18	PG4
PA3/IC4/OC5	19	20	PG3
PA2/IC1	21	22	PG2
PA1/IC2	23	24	PG1
PA0/IC3	25	26	PG0

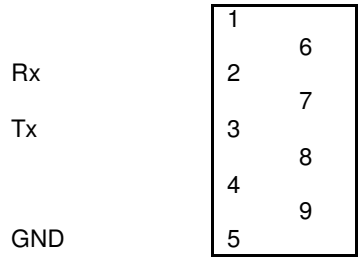
Gambar 8.36 Konfigurasi Steker X2 PORTG VEDCLEMPS

Steker X3 (SPI)

GND	1	2	+5V
MOSI/PD3	3	4	SS'/PD5
MISO/PD2	5	6	SCK/PD4
	7	8	

Gambar 8.37 Konfigurasi Steker X3 SPI VEDCLEMPS

Steker X4 (RS232)



Gambar 8.38 Konfigurasi Steker X4 RS232 VEDCLEMPS

Steker X5 (BUS)

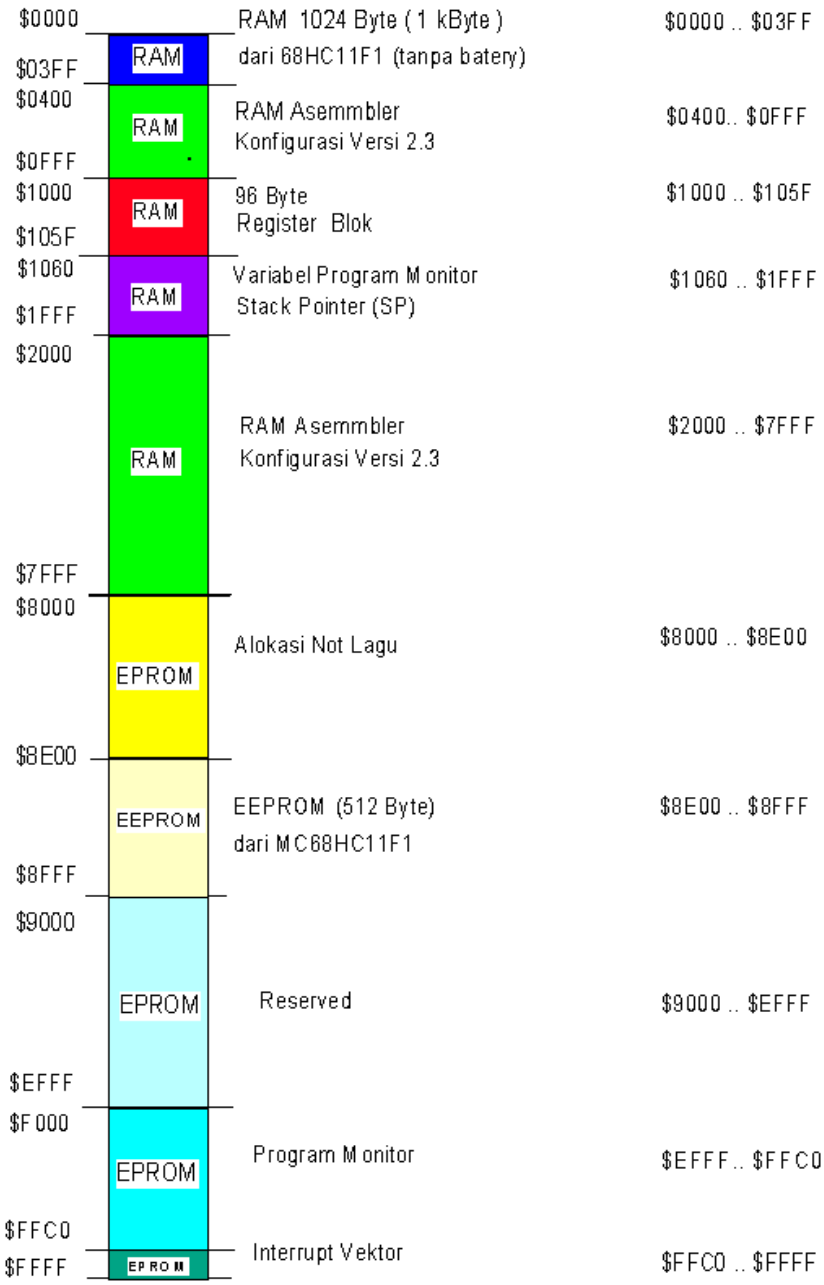
+5V	1	2	+5V
GND	3	4	GND
GND	5	6	GND
	7	8	
	9	10	
	11	12	IRQ
A15	13	14	A14
UA	15	16	A12
A7	17	18	A13
A6	19	20	A8
A5	21	22	A9
A4	23	24	A11
A3	25	26	OE'
A2	27	28	A10
A1	29	30	A0
D7	31	32	D1
D6	33	34	D0
D5	35	36	D2
D4	37	38	D3
E	39	40	R/W'

Gambar 8.39 Konfigurasi Steker X5 BUS VEDCLEMPS

Steker X6 (Steker Catu Daya)

```
          ---
GND ----| O-|-----  +8 s.d +12V
          ---
```

8.13.6. Peta Memory VEDCLEMPS



Gambar 8.40 Peta memory

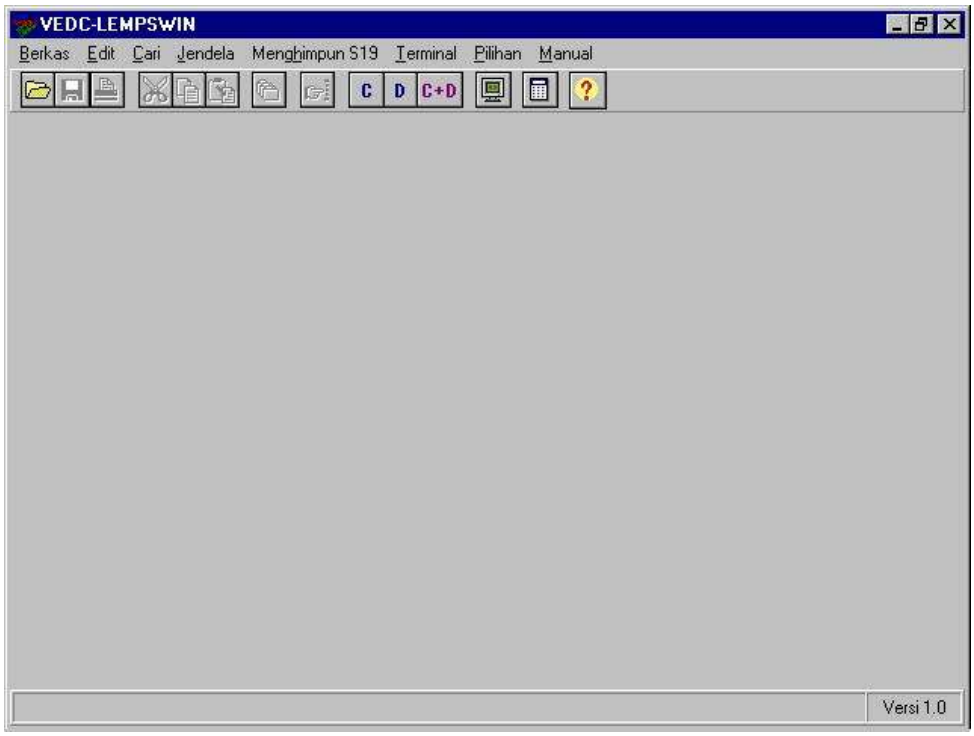
8.14. Software VEDCLEMPSWIN

VEDCLEMPSWIN dijalankan dengan cara double-klick pada icon yang tersedia pada Group VEDCLEMPS atau pada windows 95 ke atas , jalankan dengan melalui START - PROGRAMS - VEDCLEMPS - VEDC LEMPSWIN.

Gambar 8.41 Icon VEDCLEMPSWIN

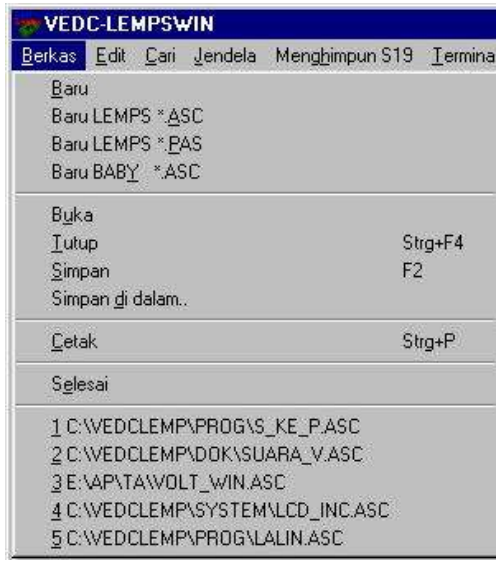
Gambar 8.42 Membuka program utama VEDCLEMPSWIN

Berikutnya akan muncul window VEDCLEMPS seperti berikut :



Gambar 8.43 Jendela utama VEDCLEMPSWIN

8.14.1. Menu Berkas



Gambar 8.44 Menu Berkas

BARU
Membuat file baru dengan format kosong (diisi sendiri)

Baru LEMPS *.ASC
Membuat file baru dengan format yangtelah disediakan untuk penulisan program LEMPS dengan assembler.

Baru LEMPS *.PAS
Membuat file baru dengan format yangtelah disediakan untuk penulisan program LEMPS dengan bahasa PASCAL.

Baru BABY *.ASC
Membuat file baru dengan format yangtelah disediakan untuk penulisan program BABY LEMPS dengan assembler.

Buka
Membuka file yang pernah dibuat.

Tutup
Menutup file yang aktif.

Simpan
Menyimpan file yang aktif ke disk.

Simpan di dalam
Menyimpan file dengan nama lain.

Cetak
Mencetak berkas yang aktif ke printer.

Selesai
Menutup program VEDC LEMPSWIN

8.14.2. Menu Edit

Gambar 8.45 Menu Edit

Memotong
Memotong/menghilangkan teks yang diblok

Mengkopi
Mengkopi teks yang telah diblok ke dalam clipboard.

Menambah
Menambahkan isi clipboard (teks yang telah dikopi) ke tempat dimana kursor ditempatkan.

Hapus baris
Menghapus satu baris dimana kursor ditempatkan.

8.14.3. Menu Cari

Gambar 8.46 Menu Cari

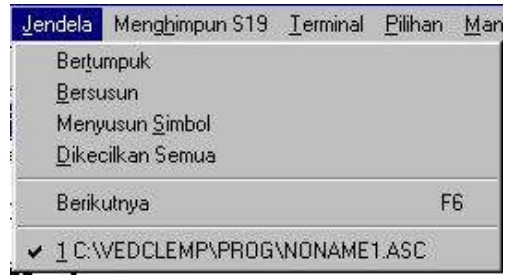
Cari
Mencari teks

Mengganti
Mencari dan sekaligus mengganti sutau teks dengan teks lain.

Cari lagi
Mengulang mencari teks yang telah dicari sebelumnya.

Cari Kesalahan [^]
Setelah meng-compile (Menghimpun) apabila terjadi kesalahan, maka apa yang salah tersebut dapat dilihat dengan menu ini dimana apa yang salah akan ditandai dengan tanda **^**

8.14.4. Menu Jendela



Gambar 8.47 Menu Cari

Bertumpuk
Menyusun jendela-jendela editor yang telah dibuka dalam susunan kaskada

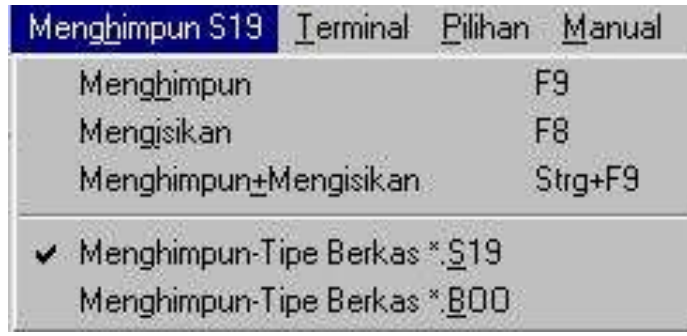
Bersusun
Menyusun jendela-jendela editor yang telah dibuka menjadi tampak semua.

Menyusun simbol
Menyusun jendela-jendela editor yang telah dibuka secara bebas.

Dikecilkan semua
Jendela-jendela editor yang telah dibuka dikecilkan semua.

Berikutnya
Mengaktifkan jendela editor berikutnya satu persatu.

8.14.5. Menu Menghimpun



Gambar 8.48 Menu Menghimpun

Menghimpun
Meng-compile file *.ASC menjadi file *.S19 atau *.BOO

Mengisikan
(Download) Mengirim file *.S19 melalui kabel RS232 ke modul mikrokontroller. Atau mengirim file *.BOO ke modul mikrokontroller Baby LEMPS.

Menghimpun+Mengisikan
Meng-compile sekaligus mengirim file melalui kabel RS232 ke modul mikrokontroller.

Menghimpun-Type berkas *.S19
Menetapkan bahwa file hasil compile adalah dalam format *.S19

Menghimpun-Type berkas *.BOO
Menetapkan bahwa file hasil compile adalah dalam format *.BOO yaitu format untuk pengisian EPROM.

8.14.6. Menu Terminal

Gambar 8.49 Menu Terminal

Terminal
Menampilkan mode terminal yaitu editoruntuk komunikasi antara komputer dan modul mikrokontroller.

Gambar 8.50 Jendela Terminal

8.14.7. Menu Pilihan

Gambar 8.51 Menu Pilihan

Sistim[RS232 & Path]
Mengatur sambungan PORT COM, BAUD Rate dan Direktori yang dipergunakan.

Terminal Tombol Fungsi
Mengatur kegunaan tombol yang disediakan pada mode terminal. Pemakai dapat mengubah fungsi tombol sesuai dengan keingginannya.

Jenis Huruf
Mengatur jenis huruf yang dipergunakan pada jendela editor,

Kalkulator
Membuka jendela kalkulator VEDC LEMPSWIN (Kalkulator jenis HP bukan CASIO !)

8.14.8. Menu Manual

Gambar 8.52 Menu Manual

Pengantar LEMPS
Berisi informasi bagaimana meng"hidupkan" mikrokontroller VEDCLEMPS dan mencobanya pada mode terminal dengan beberapa perintah "Token" BACA, TULIS dan GOTO

Pengantar BABY
Berisi informasi bagaimana meng"hidupkan" mikrokontroller BABYLEMPS dan mencobanya pada mode terminal dengan beberapa perintah "Token" R, W dan G

Hardware LEMPS
Berisi informasi tentang tata letak steker beserta urutan pin-pin pada PORT VEDCLEMPS dan Pembagian Memori (Memori map).

Hardware BABY
Berisi informasi tentang tata letak steker beserta urutan pin-pin pada PORT BABYLEMPS dan Pembagian Memori (Memori map)

Monitor LEMPS
Berisi informasi tentang :
Langkah-langkah menjalankan program
Sub Program monitor EPROM V2.35/VEDC
NOT lagu VEDCLEMPS
Sub program Liquid Crystal Dysplay (LCD)
Tokens monitor VEDCLEMPS
Alamat interupt vector

Monitor BABY
Berisi informasi tentang program monitor BABYLEMPS

Assembler M68HC11
Berisi informasi tentang :
Informasi kesalahan
Assemblerdirectiven
Format program assembler
Assembler untuk PC (Contoh program)

Pascal
Berisi informasi tentang pembuatan program mikrokontroller dengan menggunakan bahasa pascal.

Penjelasan M68HC11
Berisi informasi tentang :
Register pada prosessor (Akkumulator, Register dan Code Code Register)

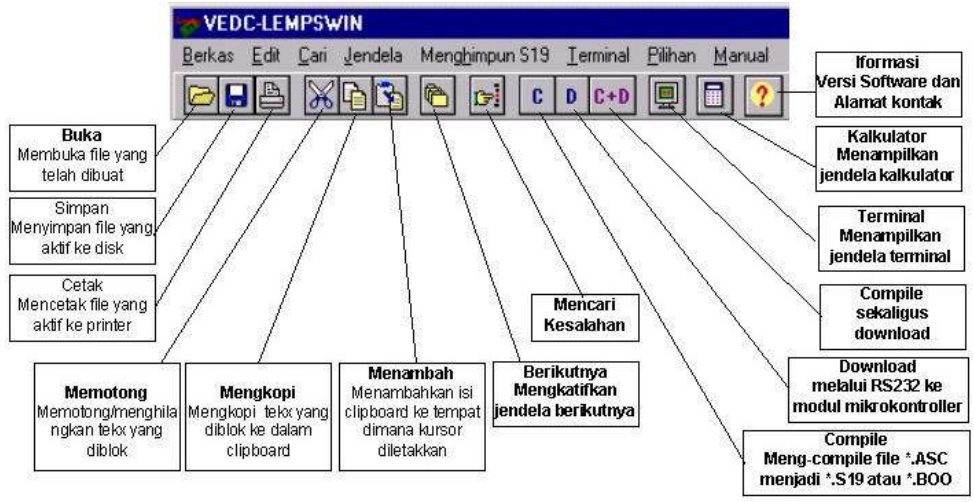
Intruksi percabangan (loncat)
Program pertama
Tabel Instruksi penting

VEDC-LEMPSWIN
Berisi informasi tentang :
Penjelasan umum tentang VEDC LEMPSWIN
Penggunaan kalkulator

Informasi VEDC-LEMPSWIN
Berisi informasi tentang versi software dan alamat kontak.

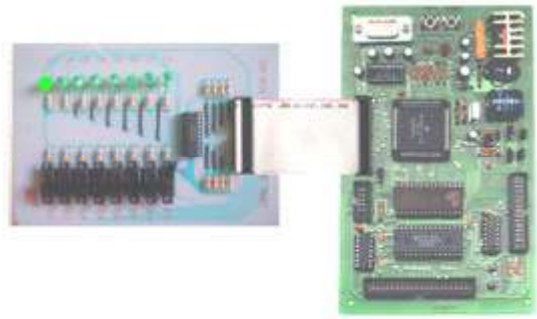
Gambar 8.53 Jendela Informasi VEDCLEMPSWIN

8.14.9. Fungsi Toolbar



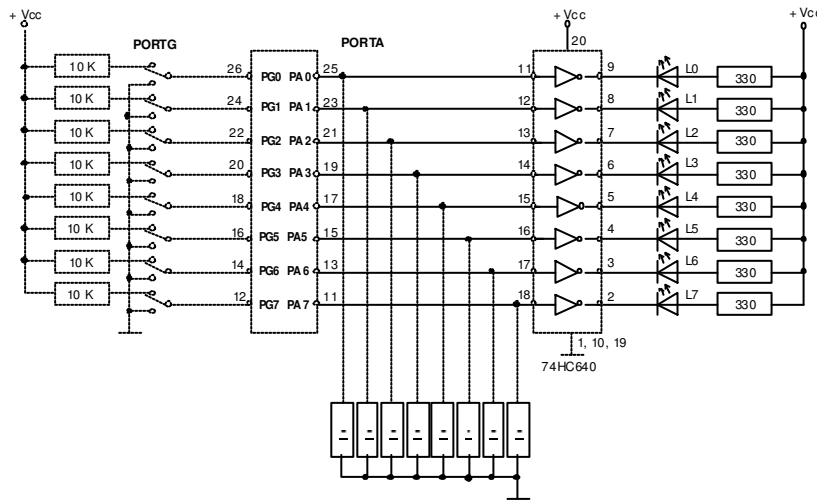
Gambar 8.54 Toolbar VEDCLEMPSWIN

8.14.10. Contoh Pengkodean Program Input Output



Gambar 8.55 Modul Input Output Digital tersambung pada modul mikrokontroller VEDCLEMPS

Berikut ini kita akan mencoba membaca data dari deretan 8 buah saklar pada yang terhubung PORTG dan mengeluarkan data hasil pembacaan itu ke deretan 8 buah LED yang terpasang pada PORTA secara terus menerus.
Opcode diperoleh dengan cara meneerjemahkan dari buku instruksi, dan angka-angka inilah yang diketikkan pada editor software EPROM programmer atau langsung didownload ke modul mikrokontroller. Cara yang demikian amat susah dan tidak mungkin dilakukan untuk program yang panjang. Cara yang paling baik adalah dengan menulis program dalam assembler pada suatu text editor dengan format penulisan yang sudah baku.

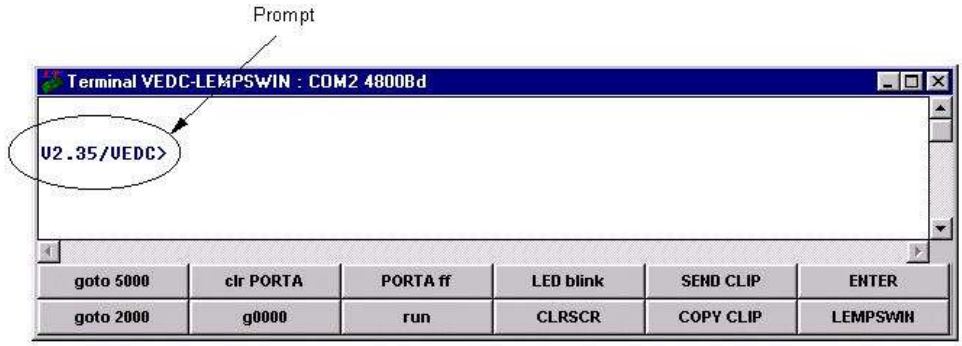


Gambar 8.56 Rangkaian Modul Input Output Digital tersambung pada modul mikrokontroller VEDCLEMPS

Alamat	Opcode				Mnemonic	Keterangan
0000	86	FF			LDAA #FF	Mengisi data langsung #$FF ke dalam Akku A
0002	B7	10	01		STAA DDRA	Mengeluarakan isi Akku A ke DDRA (Adr. $1001)
0005	86	00			LDAA #00	Mengisi data langsung #$00 ke dalam Akku A
0007	B7	10	03		STAA DDRG	Mengeluarakan isi Akku A ke DDRG (Adr $1003)
000A	B6	10	02		LDAA PORTG	Akku A diisi data dari PortG (Adr. $1002)
000D	B7	10	00		STAA PORTA	Isi Akku A diberikan ke PortA (Adr. $1000)
0010	7E	00	0A		JMP $000A	Loncat ke alamat $000A

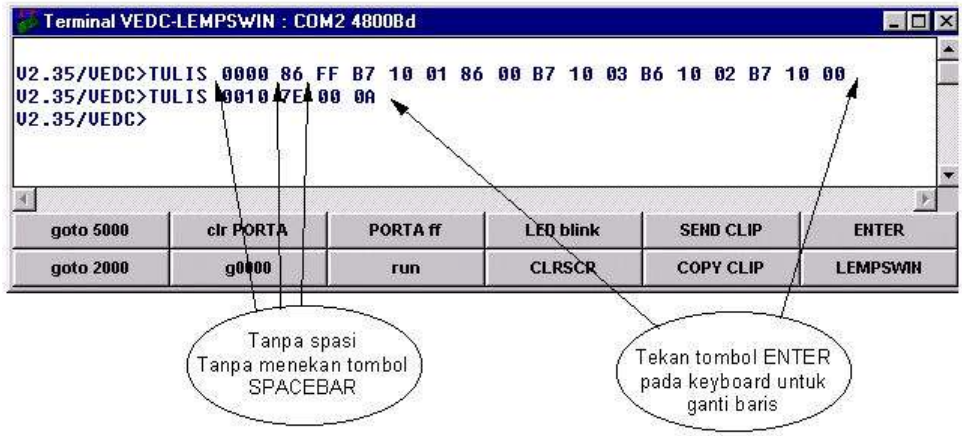
Setelah arah masuk-keluarnya data ditentukan (di-inisialisasi), berikut ini kita akan mencoba membaca data dari deretan 8 buah saklar pada yang terhubung PORTG dan mengeluarkan data hasil pembacaan itu ke deretan 8 buah LED yang terpasang pada PORTA secara terus menerus.

Sambungkanlah modul mikrokontroller VEDCLEMPS dengan kabel RS232 ke komputerdan pasangkan pula powersupply 12 V dc. Kemudian Jalankan program aplkasi windows VEDCLEMPSWIN dan bukalah mode terminal. Perhatikan dan yakinkan bahwa Prompt V2.35/VEDC sudah muncul pada editor mode terminal. Selama prompt belum muncul maka kita tidak dapat menjalakan mikrokontroler. Usahakan pertama kali prompt harus muncul dengan cara menekan tombol reset atau tombol XIRQpada modul mikrokontroller dan jika tombol ENTER pada keyboard ditekan maka pada editor terminal juga akan muncul prompt baru.



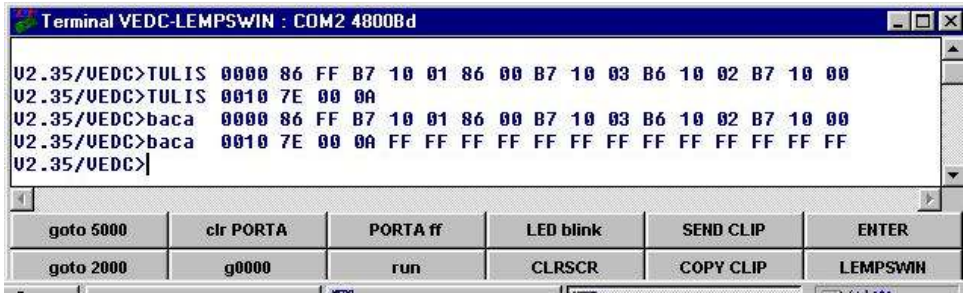
Gambar 8.57 Jendela Terminal VEDCLEMPSWIN

Berikutnya salinlah kode operasi program IN_OUT diatas dengan cara mengetikan kode operasi tersebut dengan bantuan tokens “TULIS” sebagai berikut :



Gambar 8.58 Menulis data RAM pada Jendela Terminal

Untuk melihat apakah data yang sudah kita ketikkan tadi sudah masuk ke RAM dengan alamat awal 0000 atau belum, kita dapat memeriksanya dengan mggunakan tokens “BACA” sebagai berikut :



Gambar 8.59 Membaca data RAM pada Jendela Terminal

Perhatikan apakah data yang ditampilkan sudah benar atau belum, jika belum benar kita dapat memperbaikinya dengan cara menuliskan lagi data yang salah dengan tokon “TULIS” kemudian ketik alamat data yang salah dan selanjutnya ketikkanlah data yang benar kemudian ENTER dan periksalah lagi data baru tersebut.

CATATAN :
Penulisan data yang berupa huruf A,B,C,D,E,F harus dalam bentuk huruf besar (Kapital)

Jika terjadi kesalahan ketik , tombol BACKSPACE tidak berfungsi (tidak dapat dibetulkan),
untuk memperbaikinya tekan tombol ENTER maka akan ganti baris dan ulangi lagi langkah yang salah tersbut.
Tokens BACA,TULIS,GOTO,REGI
bebas menngunakan huruf besar atau kecil.

Apabila program yang ditulis sudah benar, kita dapat menjalankan program tersebut dengan bantuan tokens “GOTO” alamat 0000 sebagai berikut :

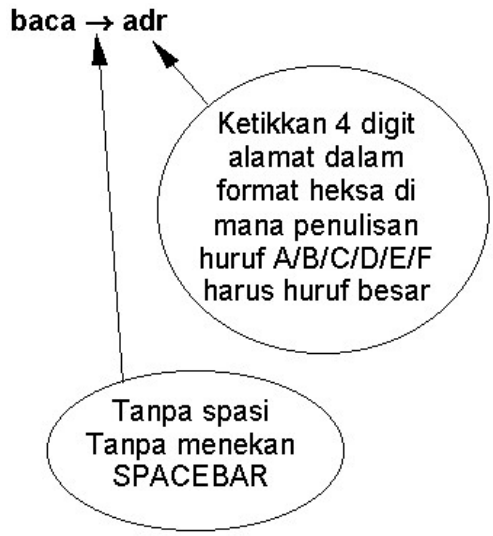
Gambar 8.60 Menjalankan program pada alamat 0000

Mainkanlah deretan saklar pada PORTG dan perhatikan nyala deretan LED pada PORTA apakah sesuai dengan kedudukan saklar, jika saklar ON maka LED menyala dan jika saklar OFF maka LED padam ?

8.14.11. Token VEDCLEMPS EPROM V2.35

BACA

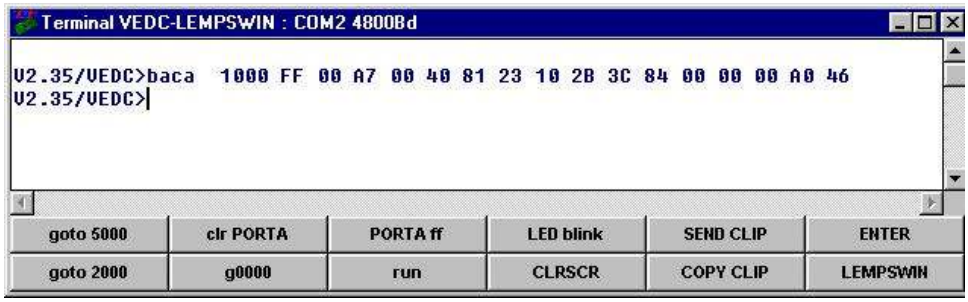
Membaca data dari memori yang dimulai dari 'adr' hingga 15 lokasi memori berikutnya. Setelah menuliskan 4 digit alamat jangan menekan ENTER



Gambar 8.61 Prosedur Baca Token VEDCLEMPS

Contoh :

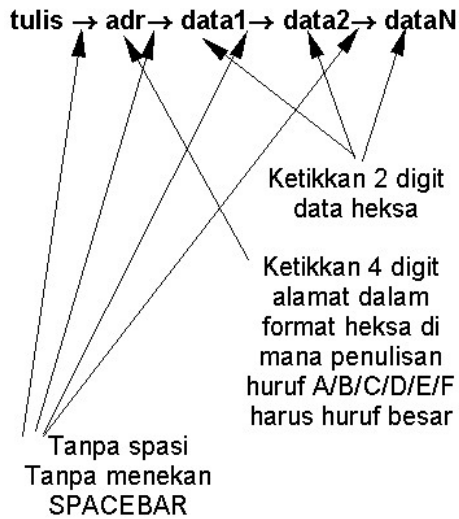
V2.35/VEDC>baca 00FF
V2.35/VEDC>BACA 00FF
V2.35/VEDC>Baca 00FF



Gambar 8.62 Contoh pemakaian token BACA

TULIS

Menulis data pada memori RAM yang dimulai dari alamat 'adr' dan kalau sudah diakhiri dengan menekan tombol ENTER



Gambar 8.63 Prosedur Tulis Token VEDCLEMPS

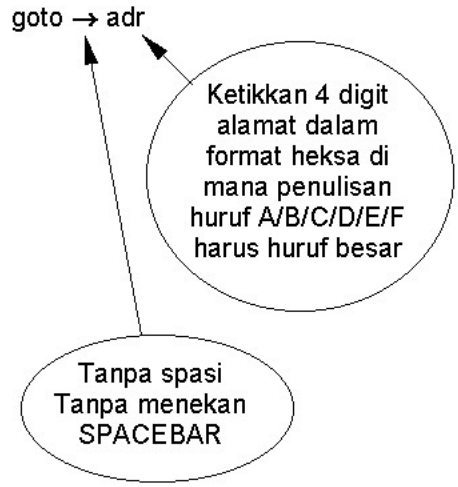
Contoh :

V2.35/VEDC>tulis 1000 FF FF ↵
V2.35/VEDC>TULIS 1000 FF FF ↵
V2.35/VEDC>Tulis 1000 FF FF ↵

Gambar 8.64 Contoh pemakaian token TULIS

GOTO

Menjalankan program yang sudah dibuat dari alamat 'adr'.
Setelah menuliskan 4 digit alamat jangan menekan ENTER

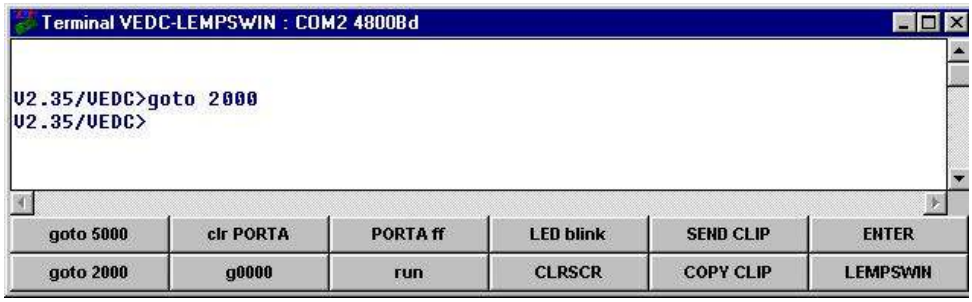


Gambar 8.65 Prosedur GOTO Token VEDCLEMPS

Contoh :

V2.35/VEDC>goto 2000

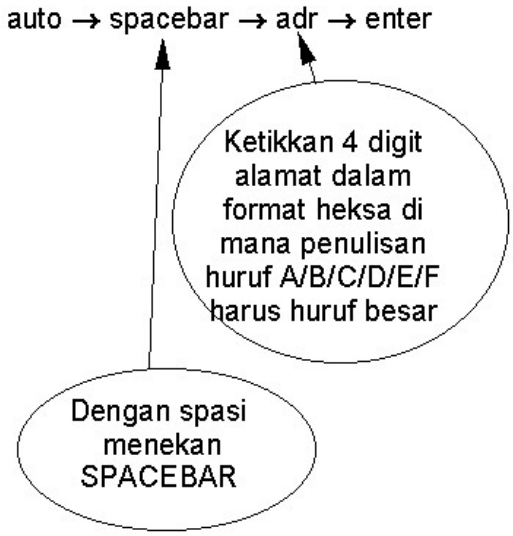
V2.35/VEDC>GOTO 2000
V2.35/VEDC>Goto 2000



Gambar 8.66 Contoh pemakaian token GOTO

AUTO

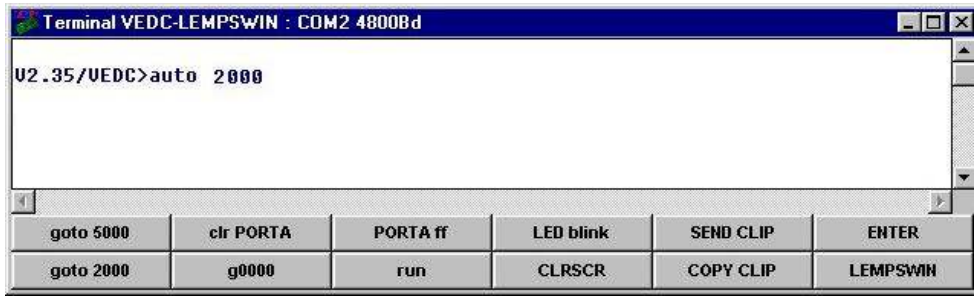
Setelah menjalankan token ini,secara otomatis program akan dijalankan mulai alamat 'adr'.



Gambar 8.67 Prosedur Auto Token VEDCLEMPS

Contoh :

V2.35/VEDC>auto 2000
V2.35/VEDC>AUTO 2000
V2.35/VEDC>Auto 2000



Gambar 8.68 Contoh pemakaian token AUTO

Hati-hati jangan sampai salah ketik pada saat penulisan alamat pada auto setelah menekan spacebar. Penulisan yang salah atau menekan tombol enter atau spacebar lagi menyebabkan program auto menjalankan ke alamat yang salah dan program akan lari serta sulit dihentikan.

Apabila memang telah terjadi kesalahan (salah ketik / tekan tombol), langkah selanjutnya jangan menekan tombol enter, melainkan tekan tombol reset atau XIRQ pada modul mikrokontroller VEDCLEMPS.

8.14.12. Pembuatan Program Dengan Mengunakan Format Assembler

Untuk menulis program assembler pada suatu text editor , susunan penulisan harus diperhatikan. Penulisan dan penempatan instruksi yang tidak mengikuti aturan akan menyebabkan program tersebut tidak dapat di-compile ke bahasa mesin. Proses meng-compile yang sukses akan menghasilkan file dengan extensi *.S19 dan file *.LST. File S19 inilah yang berisi kode operasi (bahasa mesin) yang akan di-download ke mikrokontroller.
Adapun susunan penulisan program assembler adalah sebagai berikut :

Kolom pertama	Kolom kedua	Kolom ketiga	Kolom keempat
Label	**Mnemonic**	**Operand**	**Kommentar**

* Programm pertamaku In_Out
* Isi PortG dikeluarkan ke PortA
* Bagian Pendefinisian ----------

PORTA	equ	$1000	Dengan EQU alamat $1000 = label PORTA
DDRA	equ	$1001	
PORTG	equ	$1002	
DDRG	equ	$1003	
	org	$0000	Alamat awal Program di RAM

* Bagian Inisialisasi ----------

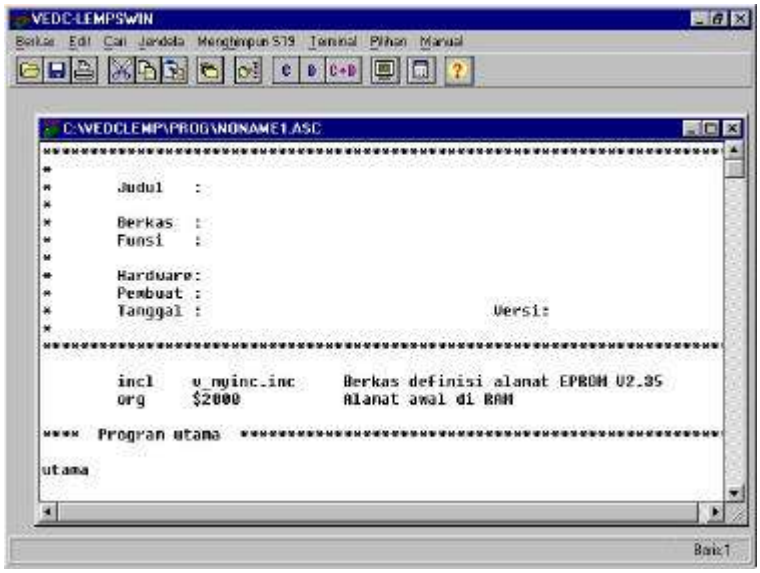
Inisial	ldaa	#$FF	Akku A diisi data #$FF
	staa	DDRA	Isi Akku A dikeluarkan ke DDRA
	ldaa	#$00	Akku A diisi data #$00
	staa	DDRG	Isi Akku A dikeluarkan ke DDRG
* Bagian Program utama ----------			
Lagi	ldaa	PORTG	Membaca PortG dan memasukkannya ke Akku A
	staa	PORTA	Mengeluarkan isi Akku A ke PortA
	jmp	Lagi	Loncat ke label Lagi
* Selesai ----------			
	end		Akhir program

Membuat File Baru

Pilihlah sub menu **Baru LEMPS *.ASC** pada menu **Berkas**

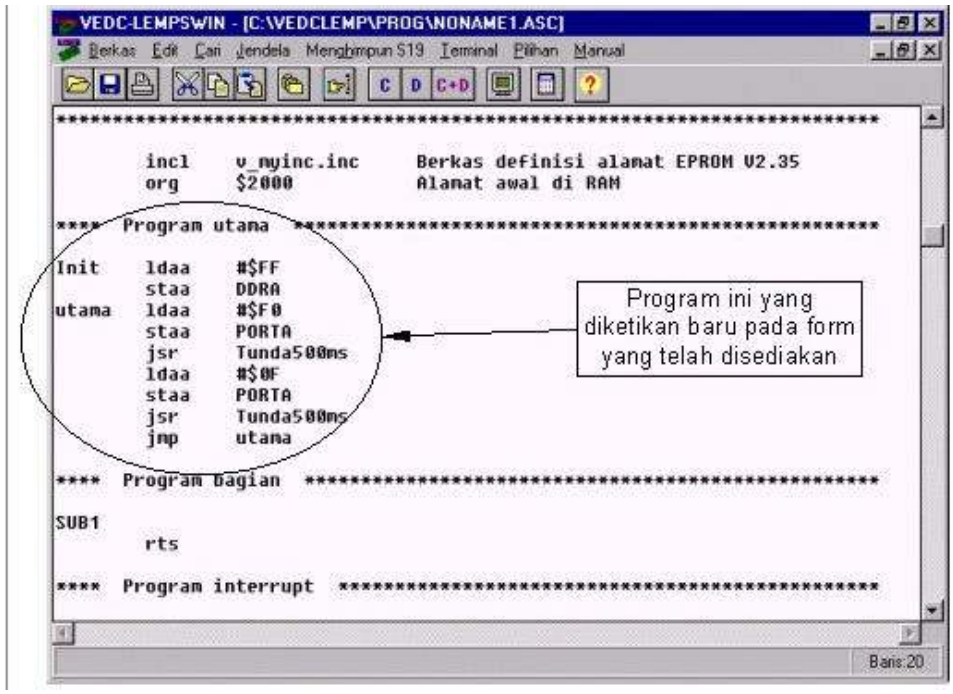
Gambar 8.69 Menu Berkas – Baru Lemps*.ASC

sehingga akan muncul satu jendela baru yaitu editor assembler berisi format urutan penulisan program assembler lengkap dan runtut yang sudah disediakan dengan nama file NONAMA.ASC seperti berikut :



Gambar 8.70 Jendela Editor

Berikutnya salinlah program BLINKER di bawah ini dengan memasukkan ke editor tersebut :



Gambar 8.71 Program blinker pada editor

Menyimpan File

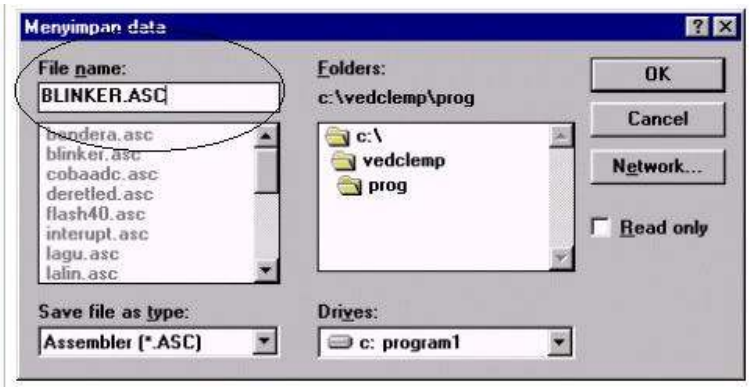
Pilihlah sub menu **Simpan** pada menu **Berkas** sehingga akan muncul satu jendela baru yaitu **Menyimpan data**.

Gambar 8.72 Menu menyimpan file

Atau dengan cara lain yang lebih mudah kita tinggal klick saja pada toolbar dengan gambar disket

Gambar 8.73 Toolbar menyimpan file

Gantilah nama file **NONAME.ASC** yang terdapat pada kotak isian nama file menjadi nama baru **BLINKER.ASC** seperti berikut :



Gambar 8.74 Jendela mengganti nama file

Menghimpun (meng-compile) file

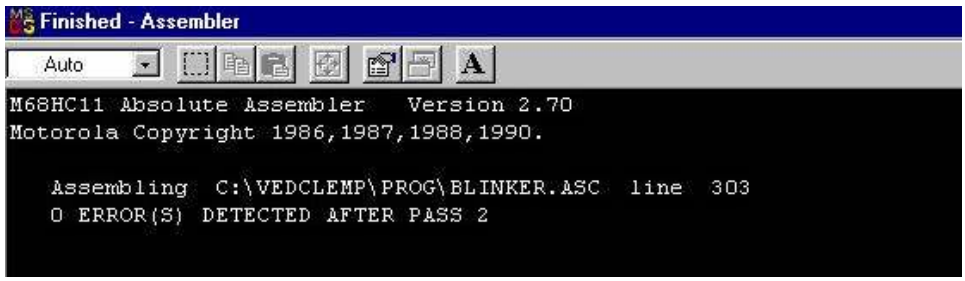
Menghimpun (meng-compile) adalh membuat file baru dengan format S19 atau BOO dari file dalam format assembler. Dengan menghimpun kita akan memperoleh file yang berisi kode operasi dari program yang kita buat dengan bahasa assembler secara otomatis (komputer yang mengerjakan sendiri pengkodean kembali instruksi assembler). File dalam format S19 ini yang akan dikirimkan melalui kabel RS232 ke modul mikrokontroller.

Gambar 8.75 Menu menghimpun file

Untuk menghimpun, pilihlah sub menu **Menghimpun** pada menu **Menghimpun S19** Atau dengan cara lain kita dapat menekan tombol fungsi F9 pada keyboard atau dengan cara lain lagi yaitu dengan meng-klick toolbar dengan gambar icon seperti di bawah ini :

Gambar 8.76 Toolbar menghimpun file

Berikutnya komputer akan memproses compileing dan menampilkan hasilnya pada jendela baru. Pada jendela ini akan muncul informasi apakah proses compileing berhasil atau terjadi kesalahan. Jika berhasil akan diperoleh file baru dalam format S19 atau BOO dan abapila terdapat kesalahan maka tidak akan dihasilkan file tersebut. Banyaknya kesalahan akan ditampilkan pada jendela ini. Informasi kesalahan 0 berarti file berhasil dicompile tanpa ada kesalahan.
Setelah muncul jendela **Finished – Assembler**, tutuplah jendela tersebut, karena setiap kali kita melakukan copileing akan selalu muncul satu jendela satu informasi seperti itu.



Gambar 8.77 Jendela menghimpun

Mencari kesalahan

Gambar 8.78 Menu mencari kesalahan

Apabila pada proses compileing (menghimpun) muncul informasi adanya kesalahan. Sebagai contoh ditunjukkan seperti berikut ini :

Gambar 8.79 Jendela hasil menghimpun

Bahwa telah terjadi satu kesalahan. Kesalahan yang dimaksud tersebut apa kita dapat melihatnya dengan cara memilih sub menu **Cari Kesalahan [^]** pada menu **Cari atau** dengan cara menekan tombol **Control+E** pada keyboard atau dengan cara yang paling cepat dengan meng-klick toolbar dengan gambar icon :

Gambar 8.80 Toolbar mencari kesalahan

Berikutnya akan muncul jendela baru seperti berikut :

Gambar 8.81 Jendela penunjukan kesalahan

Perhatikan bahwa terjadi kesalahan pada perintah **jmp Utama**. Kode kesalahan yang ditunjukkan adalah Error 235. Untuk menmgetahu keslahan apa yang dimaksud, kita dapat melihatnya pada sub menu **Assembler M68HC11** pada menu **Manual** .

Gambar 8.82 Menu manual assembler MC68HC11

Dari manual kita peroleh informasi :
ERROR 235
Tanda(Difinisi) tidak dikenal, Syntaxerror, Format Bilangan
Kita perhatikan bahwa Label Utama tidak diketemukan. Sekarang kita harus melihat dan meneliti kembali program yang kita buat apakah label Utama benar tidak ada ?.

Gambar 8.83 Kesalahan penulisan huruf “U”

Perhatikan bahwa ternyata label **U**tama memang tidak ada yang ada adalah label **u**tama.
Penulisan label harus sama persis berkaitan dengan besarnya huruf yang dipakai. **U** dan **u** adalah tidak sama !
Untuk memperbaikinya samakanlah label yang dipakai, yaitu :

Perintah jmp Utama diganti dengan jmp utama atau label utama diganti dengan Utama
Setelah diperbaiki lakukanlah proses menghimpun lagi sampai diperoleh kesalahan 0

Mengisikan (Download)

Mengisikan adalah mengirim data file S19 melalui kabel RS232 ke modul mikrokontroller.
Download dilakukan dengan cara memilih sub menu **Mengisikan** pada menu **Menghimpun S19**

Gambar 8.84 Menu mengisikan

atau dengan cara menekan tombol fungsi **F8** pada keyboard atau dengan cara yang paling cepat dengan meng-klick toolbar dengan gambar icon :

Gambar 8.85 Toolbar mengisikan

Berikutnya akan muncul jendela baru seperti berikut :

Gambar 8.86 Jendela mengisikan file S19

Kita pilih file S19 yang kita kehendaki untuk diisikan ke mikrokontroller dan selanjutnya jawablah **OK**

Selama proses mengisikan akan muncul tampilan proses mengisikan pada sisi bawah jendela VEDCLEMPS. Tunggulah sampai proses menunjukkan 100 % selesai.

Gambar 8.87 Progressbar selama proses mengisikan

Berikutnya jika proses download berhasil akan muncul jendela baru editor mode terminal dengan informasi nama file dengan disertai data alamat awal program.

Gambar 8.88 Jendela terminal VEDCLemps

Pada prompt V2.35/VEDC> kita dapat menjalankan program dengan cara yang sama seperti sebelumnya yaitu dengan menuliskan token GOTO 2000 atau dengan meng-klick tombol fungsi

Gambar 8.89 Tombol menjalankan program

Menghimpun dan mengisikan (Compile + Download)

Proses menghimpun dan mengisikan (compile + download) dapat kita lakukan sekali jalan saja dengan cara menekan toolbar dengan gambar icon :

Gambar 8.90 Toolbar menghimpun sekaligus mengisikan

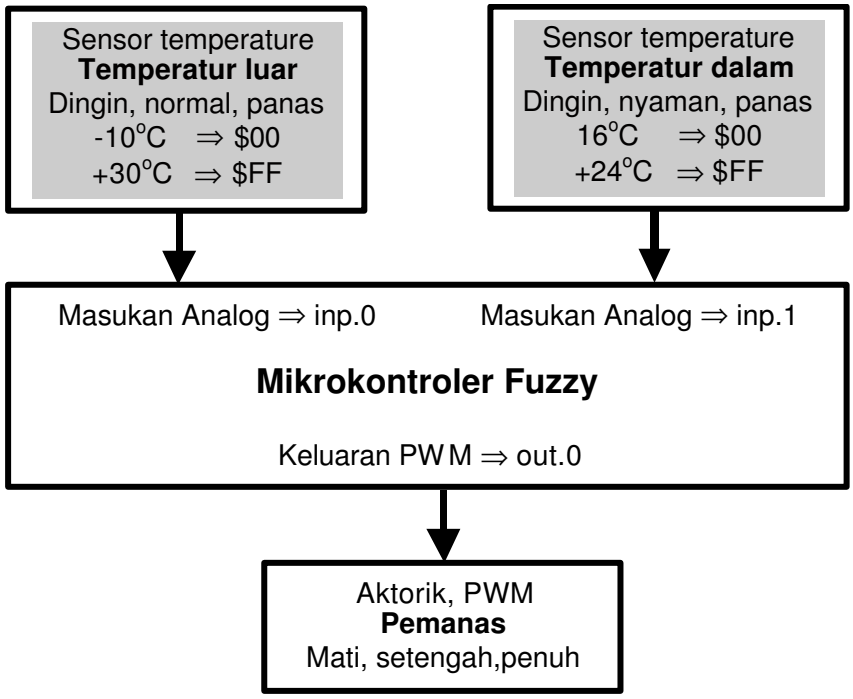
Selanjutnya komputer akan meng-compileing file assembler yang kita buat sekaligus mengisikannya langsung ke mikrokontroller melalui kabel RS232.

8.15. Pemodelan Fuzzy

Pada bagian ini akan diperlihatkan implementasi suatu kontrol temperatur dengan dua masukan dan satu keluaran.
Kontroler fuzzy akan mengukur temperatur diluar ruangan dan temperatur dalam ruangan mengatur pemanas ruangan.
Langkah-langkah implentasi dibagi menjadi tiga tahapan yaitu: Pendefinisan sistem dalam bentuk diagram blok, membuat grafik fungsi keanggotaan dan menetapkan aturan.

8.15.1. Pendefinisian Pengaturan Temperatur Ruangan

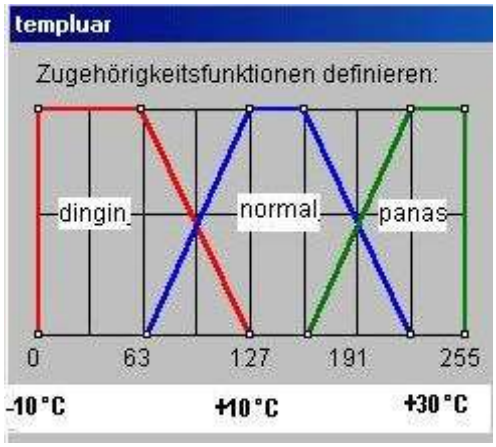
Pada tahap ini ditetapkan nama dan jumlah fungsi keanggotaan (variabel liguistik), lebar nilai masukan dan penetapan port masukan serta keluaran mikrokontroler.



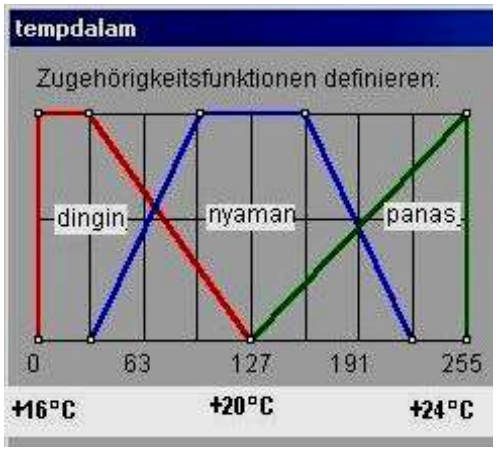
Gambar 8.91 Diagram blok pengatur temperatur ruangan

8.15.2. Membuat grafik fungsi keanggotaan

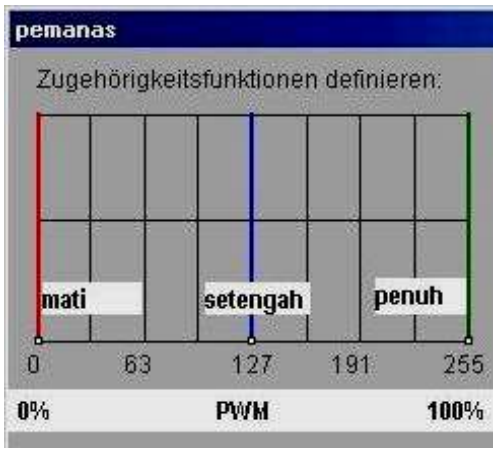
Membuat grafik fungsi keanggotaan untuk masukan dan keluaran. Fungsi keanggotaan masukan dapat berbentuk trapesium maupun segitiga sedangkan untuk fungsi keanggotaan keluaran hanya berbentuk singletons.



Gambar 8.92 Grafik fungsi keanggotaan temperatur luar



Gambar 8.93 Grafik fungsi keanggotaan temperatur dalam



Gambar 8.94 Grafik fungsi keanggotaan pemanas

8.15.3. Menetapkan Aturan Dasar

Menetapkan aturan dasar sesuai kemungkinan yang terjadi dengan banyaknya fungsi keanggotaan masukan dan keluaran., contoh „Jika temperatur luar hangat dan temperatur dalam panas maka kemudian matikan pemanas ruangan. Hubungan variabel lingusistik yang banyak dilakukan dengan operasi AND.

	Temp. luar		Temp dalam		pemanas
IF	Dingin	AND	Dingin	THEN	Senuh
IF	Dingin	AND	nyaman	THEN	Senuh
IF	Dingin	AND	panas	THEN	Setengah
IF	normal	AND	Dingin	THEN	Setengah
IF	normal	AND	nyaman	THEN	Setengah
IF	normal	AND	panas	THEN	Mati
IF	panas	AND	Dingin	THEN	Mati
IF	panas	AND	nyaman	THEN	Mati
IF	panas	AND	panas	THEN	Mati

8.15.4. Implementasi Fuzzy Kontroler Dengan Software FuzzyLemps

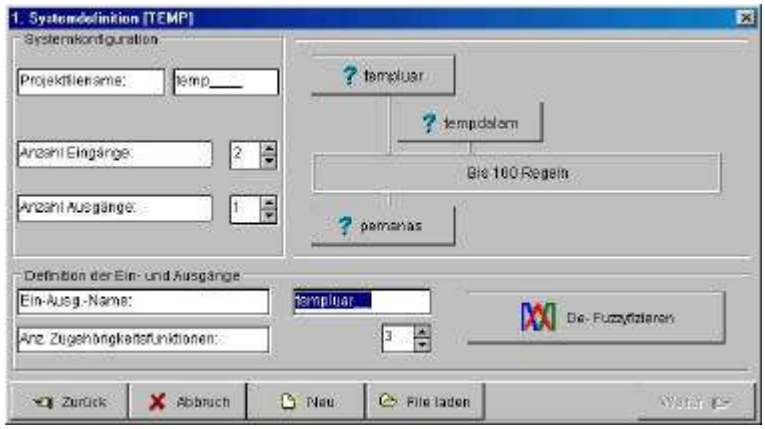
Jalankan program FuzzyLemps dengan cara double-click pada icon seperti tampak pada gambar berikut ini.

Gambar 8.95 Icon Fuzzylemps

Tunggu sampau muncul jendela utama FuzzyLemps

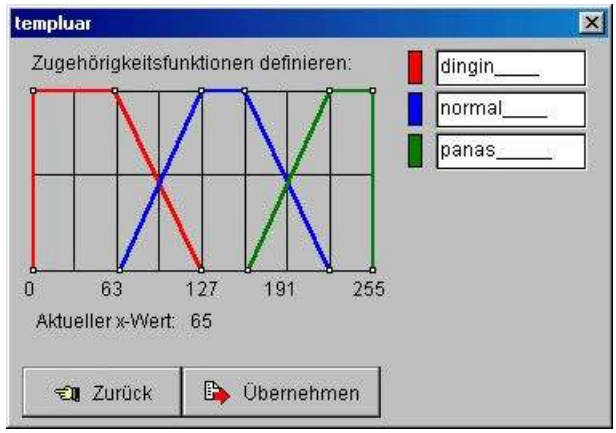
Gambar 8.96 Jendela utama FuzyLemps

Selanjutnya implementasikan diagram blok yang sudah direncanakan kedalam software sebagai berikut

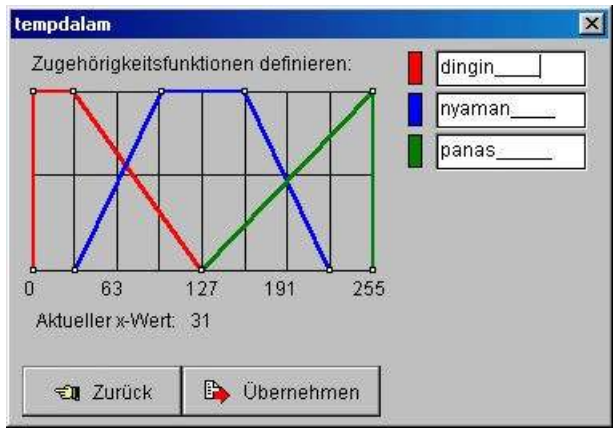


Gambar 8.97 Implementasi diagram blok FuzyLemps

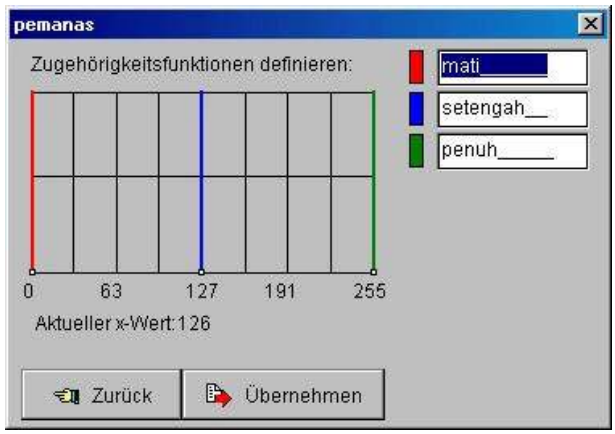
Berikutnya tentukan fungsi keanggotaan untuk semua masukan dan keluaran sesuai dengan rancana pada saat pemodelan.



Gambar 8.98 Implementasi fungsi keanggotaan masukan temperatur luar

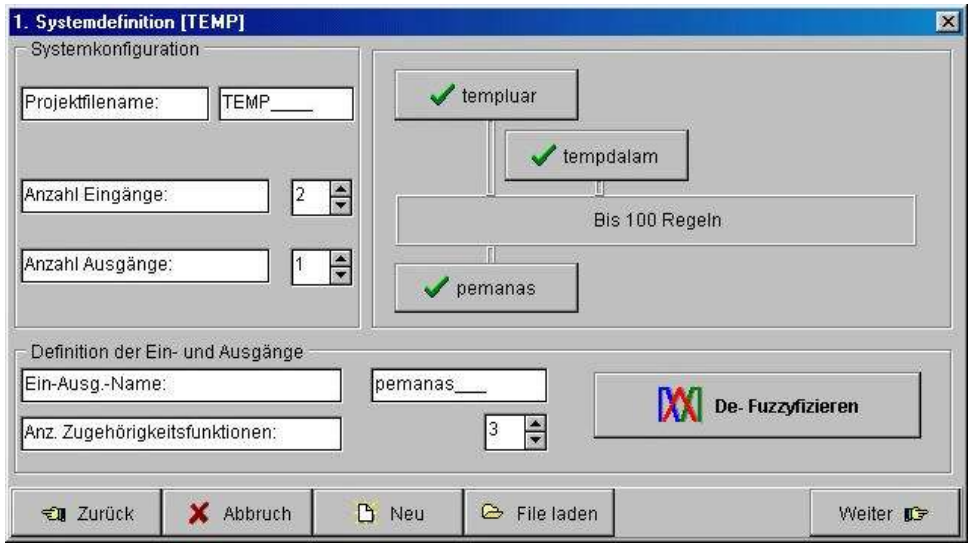


Gambar 8.99 Implementasi fungsi keanggotaan masukan temperatur dalam



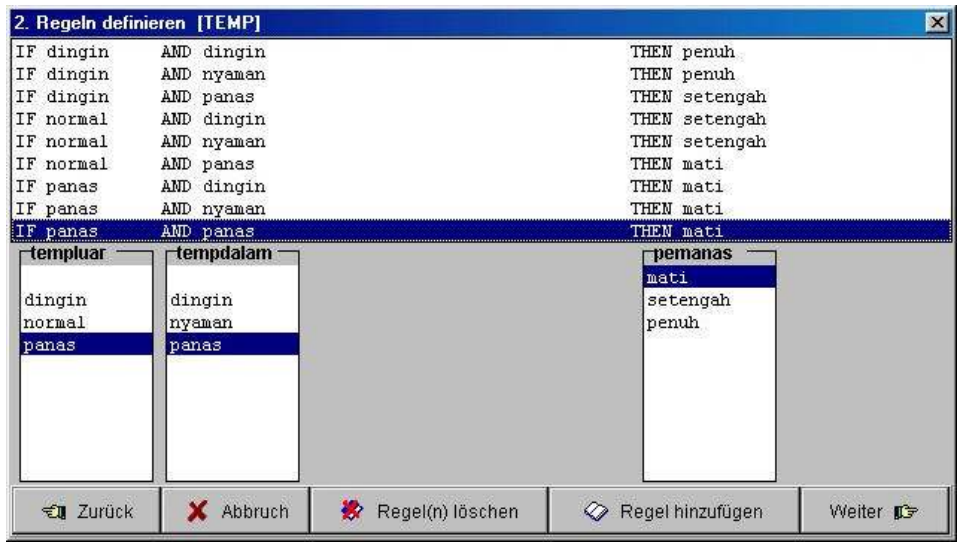
Gambar 8.100 Implementasi fungsi keanggotaan keluaran

Apabila semua fungsi keanggotaan masukan dan keluarn selesai ditentukan , maka diagram blok kontrol akan berubah menjadi seperti pada gambar 3.11. yaitu dengan ditandainya blok masukan maupun keluaran.



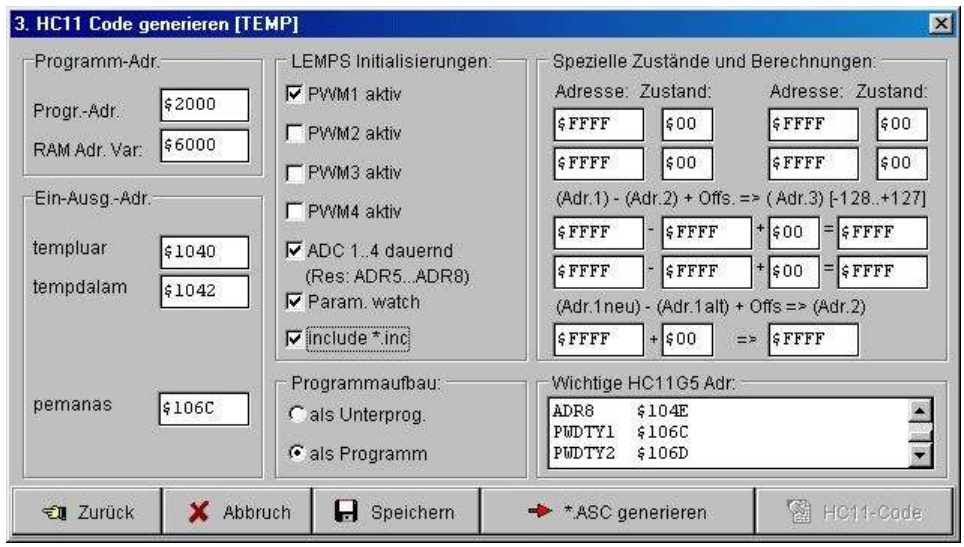
Gambar 8.101 Diagram blok yang telah terisi penuh dengan fungsi keanggotaan

Selanjutnya tekan tombol „Weiter“ untuk mengedit aturan „Rule Base“



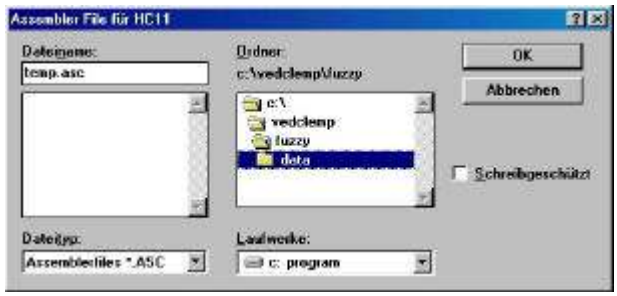
Gambar 8.102 Mengatur aturan sesuai banyaknya kemungkinan

Langkah berikutnya tekan tombol “Weiter” untuk mengatur pengalamatan



Gambar 8.103 Pengaturan alamat pada mikrokontroler

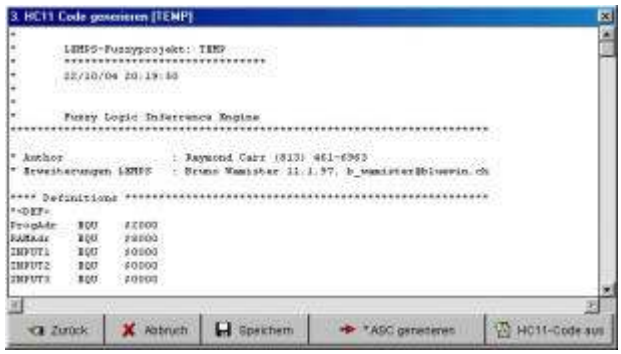
Kemudian simpanlah file fuzzy ini dengan cara menekan tombol “Speichern” dan berilah nama file.



Gambar 8.104 Menyimpan file fuzzy

Berikutnya kita akan membuat file assembler dengan cara menekan tombol “ASC Generieren”.

Gambar 8.105 Konfirmasi pembuatan file ASC

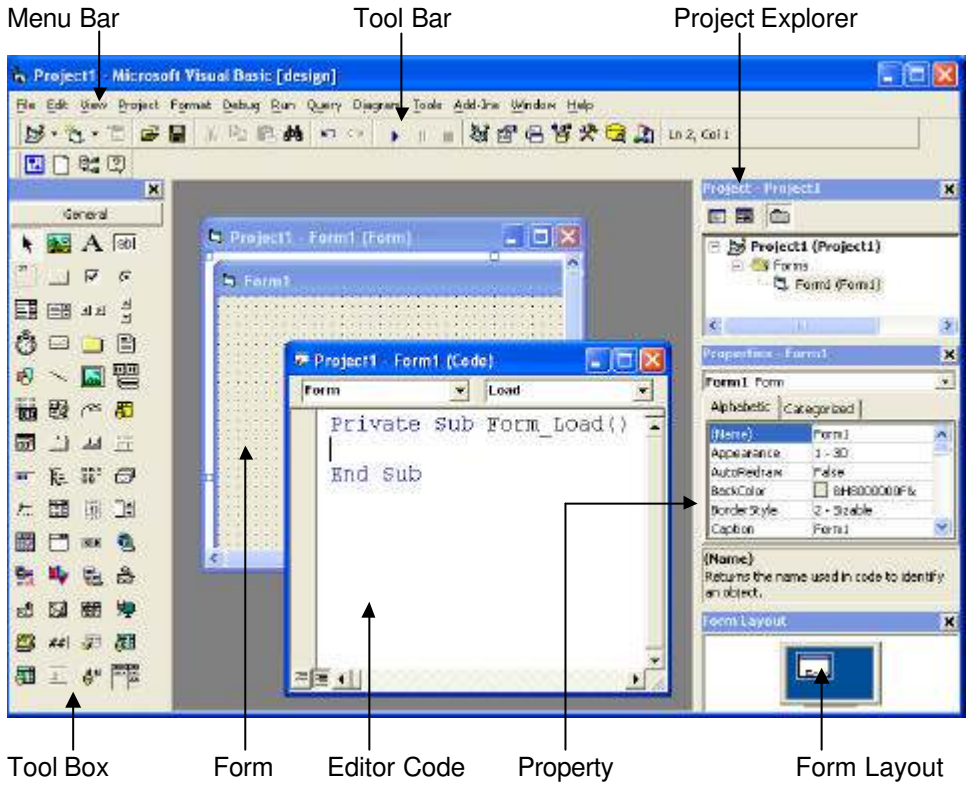


Gambar 8.106 File ASC yang telah dibuat oleh software FuzzyLemps secara otomatis

File ASC ini kemudian dibuka dan dijalankan pada softaware VEDCLEMPS dan didownloadkan ke mikrokontroler.

BAB IX Kontrol Berbasis Komputer

9.1. Mengenal *Integrated Development Environment* (IDE) Visual Basic 6



Gambar 9.1 Desktop Visual Basic

Visual Basic adalah salah satu bahasa pemrograman komputer. Bahasa pemrograman adalah perintah-perintah yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Bahasa pemrograman Visual Basic, yang dikembangkan oleh Microsoft sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu bahasa pemrograman BASIC *(*Beginner's All-purpose Symbolic Instruction Code*)* yang dikembangkan pada era 1950-an. Visual Basic merupakan salah satu Development Tool yaitu alat bantu untuk membuat berbagai macam program komputer, khususnya yang menggunakan sistem operasi Windows. Visual Basic merupakan salah satu bahasa pemrograman komputer yang mendukung object (Object Oriented Programming = OOP)

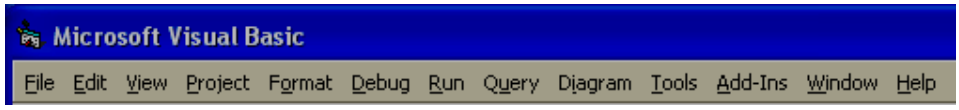
Dalam pemrograman berbasis obyek (OOP), kita perlu memahami istilah object, property, method dan event sebagai berikut :

Object : komponen di dalam sebuah program
Property : karakteristik yang dimiliki object
Method : aksi yang dapat dilakukan oleh object
Event : kejadian yang dapat dialami oleh object

Secara keseluruhan penampilan desktop Visual Basic adalah seperti yang pada Gambar 1.5 yang terdiri atas bagian bagian sebagai berikut :

- Toolbar, shortcut yang dipergunakan untuk membuat perintah
- Menu Bar, menu untuk perintah Visual Basic
- Tool Box, komponen yang dipergunakan untuk membuat form
- Form, bidang yang akan ditampilkan sebagai visual
- Editor Code, bidang tempat menulis program
- Property, daftar setting untuk setiap komponen
- Form Layout, penampakan pada layer
- Project Explorer

9.1.1. Menu Bar



Gambar 9.2 Menu Bar

Menu bar terdiri dari menu File, Edit, View, Project, Format, Debug, Run, Query, Diagram, Tools,Add-Ins, Window dan Help.

9.1.2. Menu Toolbar

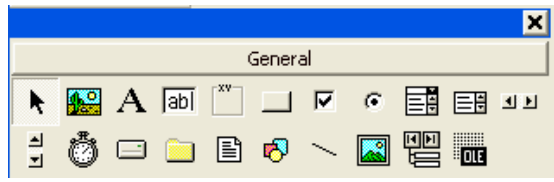
Menu Toolbar merupakan menu berbentuk icon yang berisi perintah. Setiap menu toolbar terdapat juga pada menu utama Visual Basic.

	Add Standard EXE Project
	Add Form
	Menu Editor
	Open Project
	Save Project
	Cut
	Copy
	Paste
	Find
	Undo Typing
	Redo Paste
	Start
	Break
	End
	Project Explorer
	Properties Window
	Form Layout Window
	Object Browser
	Toolbox
	Data View Window
	Visual Component Manager

Gambar 9.3 Menu Toolbar

9.1.3. Menu Toolbox

Setiap pembuatan project baru dan pilihan project adalah Standard EXE maka akan muncul menu toolbox standard seperti pada Gambar 1.4



Gambar 9.4 Menu Toolbox Standard EXE

Pada pembuatan program aplikasi instrumentasi dan kontrol kita tidak mempergunakan menu toolbox standard karena kita masih memerlukan komponen lain untuk keperluan mengakses port juga mengolah serta menyimpan.
Meskipun kita juga masih dapat menambahkan komponen baru sesuai kebutuhan proses kontrol dengan cara menambahkan komponen yang tersedia dalam Visual Component Library dan komponen kontrol OCX, tetapi sebenarnya kita tidak perlu repot dengan hal tersebut melainkan dengan cara membuat project baru menggunakan Wizard yang tersedia untuk keperluan kontrol yaitu VB Enterprise Edition Controls, maka akan muncul toolbox yang dengan komponen lengkap seperti pada gambar berikut :

Gambar 9.5 Menu Toolbox VB Enterprise Edition Controls

	**Pointer** Satu satunya item pada toolbox yang bukan kontrol. Ketika kita memilih pointer, kita hanya dapat mengubah besar atau memindah suatu object yang sudah dipasang pada form.
	**PictureBox** Digunakan untuk menampilkan gambar dan mendudkung untuk operasi grafik serta sebagai container bagi kontrol-kontrol lain.
	**Label** Digunakan untuk menampilkan text tanpa bias diubah oleh pemakai pada saat runtime.
	**Textbox** Digunakan untuk menampilkan text yang dapat diubah oleh pemakai pada saat runtime.
	**Frame** Dipergunakan untuk mengelom-pokkan sekelompok kontrol. Pemakaian kontrol frame yang paling nyata adalah untuk mengelompokkan sejumlah option, seperti yang kita ketahui pada suatu form hanya 1 option yang dapat dipilih setiap saat, hal ini dapat diatasi dengan pemakaian frame, sehingga option dapat dipilih sesuai dengan konteks yang diwakili.
	**CommandButton** Digunakan untuk pelaksanaan suatu perintah yang sudah ditentukan oleh pemakai.
	**CheckBox** Digunakan untuk menampilkan beberapa pilihan yang dapat dipilih lebih dari satu.
	**Option** Digunakan untuk menampilkan beberapa pilihan yang hanya dapat dipilih salah satu dalam suatu form.
	**ComboBox** Jika dibandingkan dengan ListBox, maka ComboBox lebih menghemat pemakaian tempat pada form, dimana hasil pilihan pemakai ditampilkan dalam suatu TextBox yang dapat didrop-down dalam bentuk ListBox. Pada ComboBox pemakai juga dapat mengetik langsung pilihannya, tetapi hal ini tergantung pada Style yang dipergunakan. Jika pada ListBox dimungkinkan pemakai melakukan pilihan ganda, tetapi pada ComboBox hal ini tidak dapat dilakukan.
	**ListBox** Digunakan untuk menampilkan daftar pilihan yang dapat bergeser. Suatu ListBox digunakan jika jumlah pilihan cukup banyak sehingga menjadi tidak efektif kalau menggunakan

	Option atau Check.
	**HscrollBar** Digunakan untuk memudahkan navigasi karena banyaknya informasi dengan cara menggeser ke arah horizontal kiri dan kanan tombol yang tersedia. Fungsi lain adalah sebagai pengatur posisi analog seperti pengaturan kecepatan atau volume.
	**VscrollBar** Digunakan untuk memudahkan navigasi karena banyaknya informasi dengan cara menggeser ke arah vertical atas dan bawah tombol yang tersedia.
	**Timer** Dipergunakan sebagai iven untuk melaksanakan suatu instruksi atau suatu program dalam suatu sekuensial waktu yang dikendaki pemakai secara otomtis.
	**DriveListBox** Dipergunakan agar pemakai dapat memilih dive yang dikehendaki pada saat runtime.
	**DirListBox** Dipergunakan untuk memapilkan direktori dan path pada saat runtime.
	**FileListBox** Dipergunakan untuk memapilkan file pada direktori dan path saat runtime.
	**Shape** Dipergunakan untuk membuat tampilan bentuk lingkaran, kotak atau oval pada form.
	**Line** Dipergunakan untuk menampil-kan visualisasi garis lurus, misalnya untuk membuat tampilan grafik yang dibuat sendiri oleh pemakai.
	**Image** Digunakan untuk menampilkan gambar dalam format bitmaps (∗.bmp), device independent bitmaps (∗.dib), metafiles (∗.wmf), enhanced metafiles (∗.emf), (∗.gif), JPEG compressed files (∗.jpg) dan icons (∗.ico), (∗.cur).
	**Data** Dipergunakan untuk mengakses data yang tersimpan pada database.
	**OLE** Dipergunakan untuk menambah-kan object-object yang dapat masukkan pada form dalam program aplikasi Visual Basic.

	**RemoteData** Dipergunakan untuk mengases data yang tersimpan dalam sumber data ODBC
	**SSTab** SSTab menyediakan sekelompok tab-tab, yang masing-masing kelompok bertindak sebagai container untuk kontrol yang lain. Hanya ada satu tab yang aktif pada saat yang bersamaan.
	**MSFlexGrid** Dipergunakan untuk menampil-kan dan menoperasikan sekum-pulan data seperti misalnya operasi sort, merge, dan memformat table data dalam bentuk strings dan gambar.
	**CommonDialog** Dipergunakan untuk menampilkan standar kotak dialog seprti menyimpan file, membuka file, mengatur setting printer, memilih warna dan bentuk huruf.
	**TabStrip** Dipergunakan untuk mendefinisi-kan banyak halaman pada area yang sama darisuatu window atau kotak dialog.
	**Toolbar** Dipergunakan untuk membuat toolbar pada aplikasi yang kita buat yang dapat diisi dengan tombol-tombol yang dapat berhubungan dengan suatu aplikasi.
	**StatusBar** Digunakan untuk informasi text yang biasanya berupa panel yang terletak pada bagian bawah form.
	**ProgressBar** Dipergunakan sebagai display yang menampilkan suatu kemajuan suatu proses yang sedang dilakukan berupa kotak persegi panjang.
	**TreeView** Dipergunakan untuk menampil-kan informasi secara hierarki.
	**ListWiew** Dipergunakan untuk menampil-kan sejumlah items mengguna-kan satu dari empat pandangan yang berbeda.
	**ImageList** Dipergunakan untuk menyimpan gambar yang masing-masing gambar diberi indeks.
	**Slider** Dipergunakan sebagai masukan atau display dengan cara menggeser penunjuk yang tersedia.

	**ImageCombo** Dipergunakan untuk memilih gambar yang tersedia.
	**Animation** Dipergunakan untuk menampilkan file animasi seperti misalnya file ∗.avi yang tidak bersuara.
	**UpDown** Adalah suatu pasangan tombol dengan dua arah yang dapat dipergunakan untuk menambah atau mengurangi nilai seperti pada scrollbar.
	**MontView** Dipergunakan untuk membuat aplikasi yang menampilkan dan mengatur informasi kalender.
	**DateTimePicker** Dipergunakan untuk mengubah format dan memilih tanggal .
	**FlatScrollBar** Dipergunakan untuk mengatur scrollbar yang dapat dikendalikan oleh mouse dalam dua dimensi.
	**Microsoft Internet Transfer (Inet)** Dipergunakan paling banyak untuk transfer data melalui protocol internet, HyperText Transfer Protocol (HTTP) dan File Transfer Protocol (FTP).
	**RichTextBox** Dipergunakan untuk memasuk-kan dan mengedit text dengan fitur yang lebih baik daripada textbox yang konvensional
	**MSChart** Dipergunakan untuk menampil-kan data dalam bentuk grafik chart.
	**Winsock** Dipergunakan untuk mengakses jaringan server TCP dan UDP.
	**MAPISession** Dipergunakan untuk membuat aplikasi mail Messaging Applica-tion Program Interface (MAPI)
	**MAPIMessages** Dipergunakan untuk memformasi sistim message yang bervariasi
	**Multimedia MCI** Dipergunakan untuk mengatur peralatan multimedia seperti merecord dan playback file multimedia pada Media Control Interface (MCI)
	**PictureClip** Dipergunakan untuk menseleksi suatu area (cropping) dari

	suatu sumber gambar bitmap dan menampilkannya pada suatu form atau picturebox.
	**SysInfo** Dipergunakan untuk merespon pesan sistim yang penting yang dikrimkan ke semua applikasi oleh sistim operasi.
	**MSComm** Dipergunakan untuk komunikasi serial, menerima dan mengeirim data melalui port serial
	**Maskedit** Dipergunakan untuk memasuk-kan data yang terbatas dengan format terntu.
	**DataGrid** Dipergunakan untuk menampil-kan dan memanipulasi sekupulan data series dari baris dan kolom yang mewakili records dan field pada suatu object recordset.
	**DataList** Adalah suatu data-bound list box yang secara otomatis dimasuk-kan dari suatu field ke dalam suatu sumber data dan memperbarui suatu field dari suatu sumber data lain yang bersangkutan.
	**DataCombo** Adalah suatu data-bound combo box yang secara otomatis dimasuk-kan dari suatu field ke dalam suatu sumber data dan memperbarui suatu field dari suatu sumber data lain yang bersangkutan.
	**CoolBar** Suatu CoolBar kontrol berisi sekumpulan Band object yang dipergunakan untuk mengkonfi-gurasi toolbar yang berhubungan dengan suatu form.
	**Adodc** Dipergunakan utntuk membuat dengan cepat suatu hubungan ke database menggunakan Microsoft ActiveX Data Object (ADO)
	**MSHFlexGrid** Microsoft Hierarchical FlexGrid mengatur tampilan dan operasi operasi pada tabular data. Dipergunakan untuk sortir, merge dan memformat table yang berisi strings dan gambar.

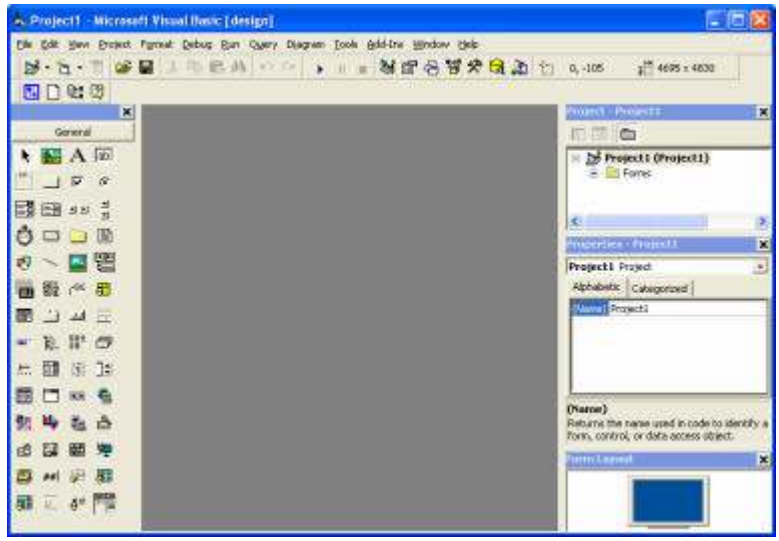
9.1.4. Pembuatan Project

Dalam Visual Basic, pembuatan sebuah program aplikasi harus dikerjakan dalam sebuah project.
Project tersebut berisi kumpulan file-file yang dipergunakan untuk membuat aplikasi. Sebuah project dapat terdiri dari satu file project (.vbp), satu file form untuk setiap form (.frm),satu file data binary untuk setiap form (.frx), satu fileuntuk setiapmodule class (.cls), satu file untuk setiap modul standard (.bas0 satu file resources tunggal (.res). Selain module dan file, beberapa tipe komponen juga dapat dimasukkan ke dalam project, seperti satu atau lebih file yang terdiri dari kontrol ActiveX (.ocx), Insertable object seperti object worksheet Excel, reference yang ditambahkan external ActiveX, ActiveX Designer untuk merancang class pada object serta standard control seperti Command Button. Untuk membuat sebuah program aplikasi, kita terlebih dahulu harus membuat form, baru kemudian membuat file dan modul lain. Setelah seluruh komponen dan kode telah ditulis, langkah selanjutnya adalah membuat project kita menjadi file yang dapat dieksekusi dalam ekstensi EXE. Setiap project biasakan disimpan dalam satu folder tersendiri karena setiap project akan terdiri dari banyak file seperti dijelaskan di atas, yang apabila nama filenya tidak diubah maka file dengan nama yang sama sepeperti defaultnya akan terdindih oleh file yang baru.

9.1.5. Membuat Project Baru

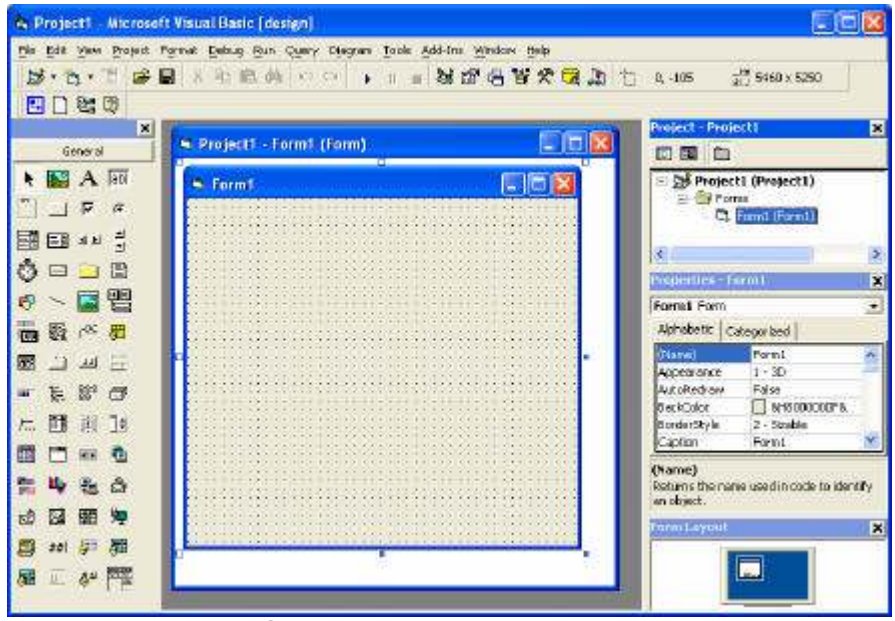
Langkah pertama pembuatan project baru dalam Visual Basic adalah klik menu **File**, pilih sub menu **New Project**, maka akan muncul jendela baru seperti pada Gambar 9.6. Pada jendela tersebut pilih **VB Enterprise Edition Controls** maka selanjutnya muncul form baru seperti Gambar 9.6

Gambar 9.6 Jendela New Project



Gambar 9.7 Desktop Visual Basic untuk project VB Enterprise Edition Controls

Pada project baru VB Enterprise Edition Controls, form tidak langsung muncu dan untuk memunculkannya kita harus mengarahkan mouse ke jendela Project Explorer, kemudian klik **Forms**, maka akan muncul icon file **Form1(Form1).** Selanjutnya double klik pada Form1(Form1) tersebut, maka Form1 akan muncul seperti pada Gambar 9.8.

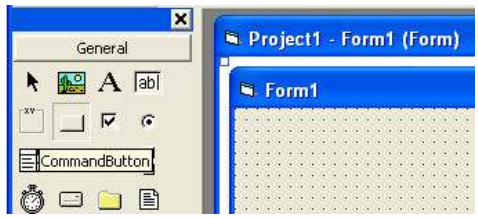


Gambar 9.8 Form dasar

9.1.6. Mengisi Form

Langkah-langkah mengisi form adalah sebagai berikut :

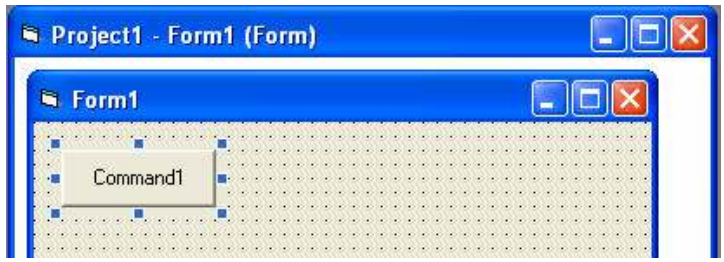
1. Tergantung dari kebutuhan, sebagai contoh kita akan mengisi form yang sudah tersedia dengan sebuah tombol. Klik CommandButton pada Toolbox.



Gambar 9.9 Komponen CommandButton pada Toolbox

Setelah komponen CommandButton di-klik, geser pointer mouse ke bidang Form1 dan pilih lokasi di mana CommandButtun akan diletakkan pada bidang Form1. Perhatikan bahwa pointer mouse akan berubah dari bentuk menjadi +.

2. Dengan pointer +, buat sebuah kotak pada Form1 dengan cara menekan mouse tombol kiri tanpa dilepas sambil membuat kotak sebesar yang kita inginkan. Lepasa tombol mouse setelah mecampai besar yang diinginkan. Pada From1 akan tampak komponen tombol seperti pada Gambar 1.10 dengan nama defaultnya adalah Command1.



Gambar 9.10 Tombol CommandButton pada Form1

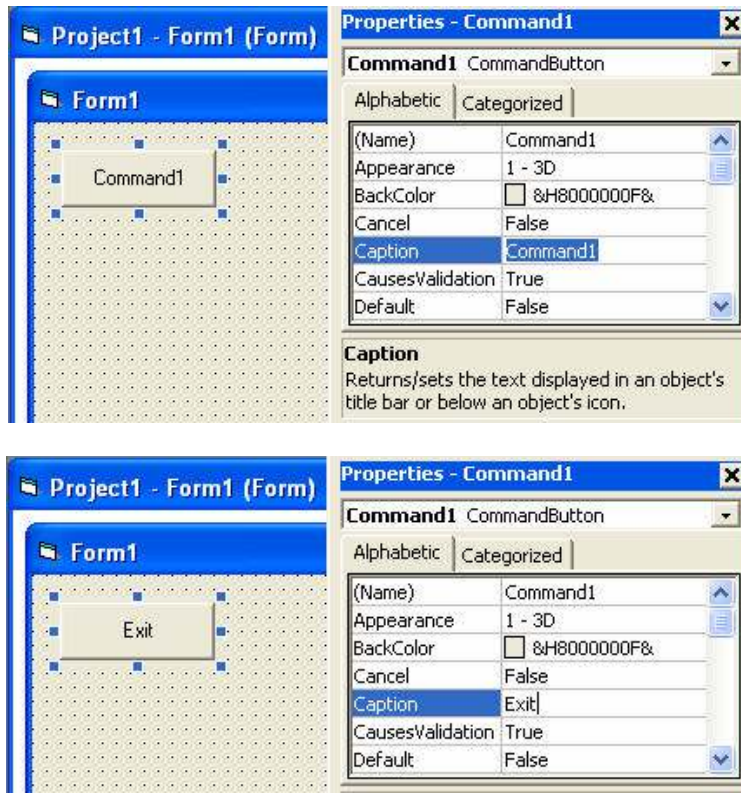
3. Untuk mengatur besarnya tombol dapat dilakukan dengan mengarahkan pointer mouse ke titik titik yang mengitari tombol Command1, tekan tombol kiri mouse, menggeser ke arah yang dikehendaki maka ukuran tombol Comand1 akan mengikuti gerakan mouse dan lepas tombol mouse jika ukuran sudah yang dikehendaki tercpai.

9.1.7. Mengsisi Property

Untuk mengganti tulisan “Command1” pada tombol, arahkan pointer mouse pada komponen tombol pada Form1, klik tombol tersebut, maka jendela Properties di sebelah kanan layer akan aktif untuk pengaturan komponen tombol dengan nama Command1 tersebut.

Pilih jenis property **Caption**, lalu klik pada tulisan “Command1” dan gantilah dengan tulisan “Exit” maka text yang tampil pada tombol Command1 pada Form1 akan berubah menjadi “Exit’.

Selain Caption, semua properti dalam daftar yang terdapat pada jendela properti tersebut dapat diubah atau diatur menurut kebutuhan kita.



Gambar 9.11 Mengubah Properti Caption

9.1.8. Mengetik Kode Program

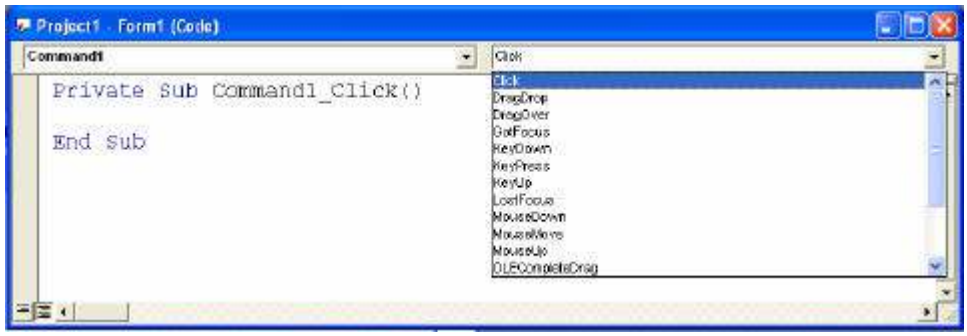
Untuk mengetik program, misalkan apabila tombol Exit pada Form1 bila di-klik akan hilang, maka kita harus menuliskan kode intruksi “End” pada even kejadian bila Command1 di-klik.

Penulisan kode ini dilakukan dengan cara mengarahkan pointer mouse ke object tombol Command1 dan klik tombol kiri mouse maka akan muncul jendela Editor Code seperti tampak pada Gambar 9.12 dan kursor langsung berada di dalam dua statement

```
Private Sub Command1_Click()

End Sub.
```

Atau dengan cara lain yaitu melalui jendela Project Explorer, lalu klik Toolbar View Code. Selanjutnya muncul jendela Editor Code, pilih Object **Command1** dan pilih even **Click** pada Combobox yang tersedia pada bagian atas jendela Editor Code.

Berikutnya pada bidang editor akan muncul dua statement secara otomatis tanpa kita harus mengetik sendiri seperti pada Gambar berikut.



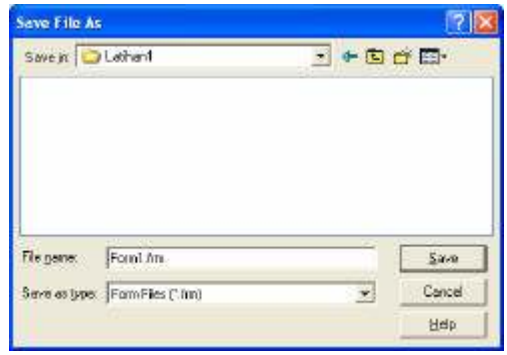
Gambar 9.12 Tempat pengetikan program

Ketikkan kode instruksi “End” pada kursor yang berkedip diantara ke dua statemen tersebut.

```
Private Sub Command1_Click()
End
End Sub
```

9.1.9. Menyimpan Program

Untuk menyimpan program dapat dilakukan dengan dua cara, yaitu dengan menekan toolbar  Save Project atau dengan cara kedua melalui menu **File** – **Save Project As…**.
Selanjutnya akan muncul jendela **Save File As** dengan default nama file Form(.frm) yang mana nama file ini dapat diganti dengan nama lain.



Gambar 9.14 Jendela Save File As

Karena sebuah project akan terdiri dari banyak file, biasakan kita membuat satu folder untuk setiap project dan menyimpan semua file pada pada folder itu. Dalam contoh ini kita membuat folder baru yang bernama Latihan1.

Setelah tombol Save di-klik, pada jendela **Save Project As** akan muncul dengan default nama file Project1.vbp yang dapat pula kita ubah dengan nama yang lain. Simpanlah file froject ini pada folder yang sama.

Gambar 9.15 Jendela Save Project As

9.1.10. Menjalankan dan Menghentikan Program

Untuk menjalankan program tekan toolbar ▶ atau tombol fungsi **F5** atau dengan menu bar **Run** – Start sehingga muncul tampilan Form1 yang memiliki satu object tombol dengan caption "Exit" yang apabika kita tekan akan menghilangkan tampilan Form1.

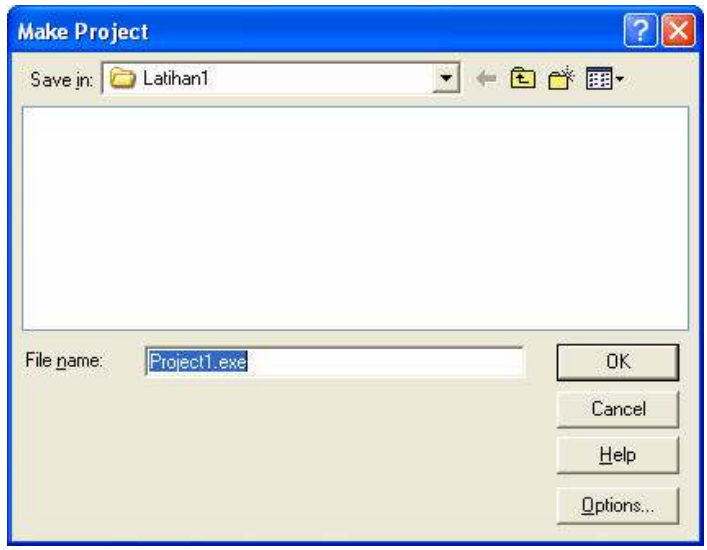
Gambar 9.16 Tampilan program yang sedang jalan

Untuk menghentikan program tekan toolbar ■

Atau dengan menu **Run** – **End** atau dengan menekan kotak silang pada pojok kanan atas form.

9.1.11. Mengkompilasi Program Menjadi File EXE

Untuk mengkompilasi program agar menjadi file yang dapat dieksekusi (dijalankan) secara langsung tanpa melalui Visual Basic, kita pilih menu **File** – **Make Project1.exe…** dan lalu muncul jendela Make Project. Selanjutnya klik tombol OK.



Gambar 9.17 Jendela Make Project

Catatan :

Hasil kompilasi program tergantung jenis file. Bila jenis file-nya .vbp maka hasil kompilasi adalah file ∗.exe. Bila jenis file-nya Component Active X In-Process maka hasil kompilasinya berupa file ∗.dll.
Dan apabila dalam project kita menggunakan komponen ActiveX dari Visual Basic, jika hasil kompilasi itu dijalankan di computer lain yang tidak di-instal Visual Basic dengan versi yang sama maka file ∗.ocx yang dipergunakan juga harus disertakan atau pada satu folder dengan file ∗.exe. Demikian pula jika project kita menggunakan file ∗.dll, maka file ∗.dll tersebut harus ikut di-copy-kan ke dalam folder Windows.

9.1.12. Variabel

Variabel adalah suatu Nama lokasi memori, dipergunakan untuk menyimpan suatu data yang dapat diubah-ubah selama program dijalankan. Setiap variable memiliki sebuah nama unik yang mengidentifikasikannya dalam ruang lingkupnya.
Contoh penulisan deklarasi variable di dalam kode program :

```
Dim data_input As Strings
```

Dalam penamaan suatu variable harus diperhatikan auran-aturanya sebagai berikut :

- Harus diawali dengan huruf.
- Tidak boleh menggunakan spasi. Spasi bisa diganti dengan karakter garis bawah _
- Tidak boleh menggunakan karakter-karakter khusus seperti tanda baca dan sejenisnya : +, -, *, /, <, >
- Tidak boleh menggunakan kata-kata kunci yang sudah dikenal oleh VB seperti : dim, as, string, integer, dll.
- Sebuah variabel hanya dapat menyimpan satu nilai data sesuai dengan type datanya.

Contoh cara mengisi nilai data ke dalam sebuah variabel :

```
data_input  = “A255”
```

Untuk type data tertentu nilai_data harus diapit tanda pembatas. Type data string dibatasi tanda petik ganda :
“data_input”
Type data date dibatasi tanda pagar :
data_input #.

Type data lainnya tidak perlu tanda pembatas.

Sebuah variabel mempunyai ruang-lingkup dan waktu-hidup :

- Variabel global adalah variabel yang dapat dikenali oleh seluruh bagian program. Nilai data yang tersimpan didalamnya akan hidup terus selama program berjalan.
- Variabel lokal adalah variabel yang hanya dikenali oleh satu bagian program saja. Nilai data yang tersimpan didalamnya hanya hidup selama bagian program tersebut dijalankan.

9.1.13. Type Data

Type data adalah atribut suatu suatu variable atau field yang menunjukkan jenis data apa yang dapat dikerjakan.

Boolean, adalah type data yang mempunyai dua nilai True atau False.

Byte, adalah type data single , unsigned , 8 bit (1 byte) angka yang mempunyai nilai 0 – 255 .

Currency, type data angka integer 64 bit (8 byte) dalam skala 10,000 untuk memberi suatu titik angka tetap dengan jumlah 15 digit di sebelah kiri tanda koma dan 4 digit di sebelah kanan. Penyajian ini menyediakan nilai dari minus -922,337,203,685,477.5808 sampai dengan positip 922,337,203,685,477580

Date, type data angka floating-point 64 bit (8 byte) yang mewakili tanggal dalam batas 1 January 100 to 31 December 9999 dan jam mulai 0:00:00 to 23:59:59

Double, type data angka floating-point 64 bit (8 byte) dengan range mulai
-1.79769313486232E308 sampai
-4.94065645841247E-324 untuk nilai negatip dan mulai 4.94065645841247E-324 sampai 1.79769313486232E308 untuk nilai positip

Integer, type data angka 16-bit (2-byte) dengan range mulai dari -32,768 sampai 32,767.

Long, type data angka 32-bit (4-byte) dengan range mulai dari 2,147,483,648 sampai 2,147,483,647

Object, type data alamat 32-bit (4-byte) yang menunjuk suatu object

Single, type data angka 32-bit (4-byte) floating-point, dengan range mulai dari -3.402823E38 sampai -1.401298E-45 untuk nilai negatip dan mulai 1.401298E-45 to 3.402823E38 unutk nilai posistip.

String, type data karakter dengan kode angka mulai 0 sampai 255, yang mana kode 0 – 127 adalah berupa huruf dan symbol sesuai dengan standar keyboard Amerika seperti yang didefinisikan sebagai simbol ASCII dan 128 karaekter dengan kode 128-255 mewakili karakter khusus seperti alpabet internasional, aksen dan simbol mata uang serta pemisah.

Variant, adalah suatu tipe data khusus yang dapat berisi segala macam data kecuali data fixed-lengthString.

9.1.14. Konstanta

Suatu variable yang datanya tidak pernah diubah-ubah dan sering dipergunakan berulang ulang atau karena nilainya sulit untuk diingat, kita dapat menuliskannya sebagai konstanta. Suatu konstanta adalah suatu nama lokasi memory tempat menyimpan data sama seperti variable tetapi isinya tidak pernah diubahiubah selama program berjalan. Ada dua macam konstanta :

- Intrisic atau System-defined konstanta yang disediakan oleh aplikasi atau kontrol.
- Simbolik atau User-defined konstanta adalah dideklarasikan dengan meng-gunakan statement `Const`

Contoh :

```
Const Pi = 3.14
Public Const  D As Integer = 123
Private Const F As Strings = “OK”
Const Tgl = #1/12/2007#
```

9.1.15. Perintah Pengulangan

Loop dipergunakan untuk menjalankan sejumlah baris instruksi berulang-ulang.
Struktur loop yang dikenal dalam Visual Basic adalah :

- Do … Loop
- For … Next
- For Each … Next

Do ... Loop

Digunakan untuk menjalankan suatu blok statemen-statemen berulang-ulang. Ada beberapa variasi-variasi statemen Do...Loop, tetapi masing-masing mengevaluasi suatu kondisi untuk menentukan apakah melanjutkan pengulangan eksekusi atau melanjutnkan statemen berikutnya. Seperti halnya If...Then, kondisi harus berupa suatu nilai atau ekspresi yang mengevaluasi ke False atau True.
Di dalam Do...Loop yang berikut, statemen-statemen dijalankan sepanjang selama kondisi itu adalah True

```
Do While condition
      statements
Loop
```

Ketika Visual Basic menjalankan Do Loop, pertma kali adalah melihat kodisi, jika kondisi False maka semua statemen dalam lokasi loop pengulangan dilewati tidak dijalankan dan keluar dari loop. Tetatpi jika kondisinya True, maka Visual basic menjalankan statemen dan kembali ke statemen Do While dan melihat kondisi lagi.
Konsekuensinya loop dapat menjalankan statement berulang ulang tanpa batas waktu selama kondisinya True.

Contoh :

```
Dim A as Integer
Dim B as Double
A = 0
B = 0
Do While A<100
      B = B + 1
Loop
```

Variasi lain dari statemen Do Loop adalah meletakkan statemen While kondinya setelah Loop, hasilnya adalah statemen pengulangan akan dijalankan minimal satu kali

```
Do
      Statements
Loop While condition
```

Selain dua variasi Do Loop di atas, variasi lain adalah

```
Do Until condition
      statements
Loop
```

Blok statement akan diulang-ulang sampai kondisi terpenuhi, pengulangan berhenti bila kodisi bernilai True

For … Next

Do Loop lebih tepat dipergunakan ketika kita tidak mengetahui berapa banyak suatu statement harus dijalankan. Apabila kita sudah menetapkan suatu statement harus dijalankan sekian kali maka perintah pengulangan yang tepat adalah memakai For Next.
Tidak seperti Do Loop, For Loop mengguna-kan variable yang disebut counter yang mana akan menambah atau mengurangi suatu nilai setiap pengulangan.

```
For counter = start To end [Step increment]
      statements
Next [counter]
```

Argumen start, end dan increment harus berupa angka numeric.
Argumen increment dapat berupa bilangan positip atau negatip. Jika increment adalah bilangan posistip maka start harus lebih kecil atau sama dengan end atau statement tidak akan dijalankan. Jika increment adalah bilangan negative maka statrt hatus lebih besar atau sama dengan end agar statemen dijalankan. Jika Step tidak ditulis maka increment defaultnya adalah 1.
Contoh

```
Private Sub Form_Click ()
   Dim I As Integer
   For i = 0 To Screen.FontCount
      Print Screen.Fonts(i)
   Next
End Sub
```

Kita dapat meletakkan suatu struktutr kontrol dalam struktur kontrol lainnya, yang dikenal dengan istilah Nested. Struktur kontrol pada Visual Basic dapat nested sebanyal level yang kita inginkan.

Meninggalkan suatu Loop

Statement Exit memperbolehkan kita untuk keluar langsung dari suatu Loop. Cara penulisan Exit sangat sederhana dan dapat dituliskan berulang kali dalam suatu Loop, contoh :

```
Sub RandomLoop
  Dim I, MyNum
  Do
    For I = 1 To 1000
      MyNum = Int(Rnd * 100)
      Select Case
        Case 17: MsgBox "C17"
          Exit For
        Case 29: MsgBox "C29"
          Exit Do
        Case 54: MsgBox "C54"
          Exit Sub
        End Select
    Next
  Loop
End Sub
```

9.1.16. Perintah Percabangan

Ada tiga macam bentuk perintah percabangan yang dapat dipilih dalam Visual Basic, yaitu

- If … Then
- If … Then … Else
- Select Case

If … Then

Digunakan untuk menjalankan satu atau lebih statement sesuai kodisi If.

Jika statement hanya satu baris dapat ditulis langsung tanpa menambahkan `End If`, sebagai berikut

```
If condition Then statement
```

Jika statement terdiri lebih dari satu baris maka penulisan statemen adalah sebagai beikut

```
If condition Then
Statements
Statements
End If
```

Condition umumnya adalah perbandingan dan dapat berupa banyak ekspres.Visual Basic menerjemhkan kondisi tersebut sebagai True atau Fasle. Jika Condition adalah True, maka Visual Basic akan menjalakan statement setelah Then.
Contoh

```
If aDate < Now Then aDate = Now
If aDate < Now Then
   aDate = Now
End If
```

If … Then … Else

Jika kita akan memilih salah satu pilihan dari banyak pilihan, maka sebaiknya kita menggunakan If…Then…Else.

```
If condition1 Then
     [statementblock-1]
ElseIf condition2 Then
     [statementblock-2]
Else
     [statementblock-n]
End If
```

Visual Basic akan mengetest condition1, apabila True maka statementblock-1 akan dijalankan, jika hasil test False maka Visual basic akan mengetes condition2, apabila hasil test adalh True maka statementblock-2 akan dijalankan, jika hasil test False maka Visual basic akan menjalankan [statementblock-n].

Select case

Visual Basic menyedian alternative lain dari If Then Else untuk memilih salah satu dari sekian banyak pilihan yang harus dijalankan.
Strukture Select Case bekerja dengan mengetest pertama kali pilihan paling atas dari susunan Select Case, Jika hasil test cocok dengan nilai Case, maka statemenblok yang sesuai dengan Case akan dijalankan.

```
Select Case testexpression
[Case expressionlist1
[statementblock-1]]
[Case expressionlist2
[statementblock-2]]
.
.
.
[Case Else
[statementblock-n]]
End Select
```

Contoh

```
Private Sub mnuCut_Click (Index As Integer)
   Select Case Index
      Case 0
         CopyActiveControl
         ClearActiveControl
      Case 1
         CopyActiveControl
```

```
      Case 2
         ClearActiveControl
      Case 3
         PasteActiveControl
      Case Else
         frmFind.Show
   End Select
End Sub
```

9.1.17. Procedure

Procedure adalah blok kode program yang berisi perintah-perintah untuk mengerjakan tugas tertentu. Bila di dalam kode program yang kita buat ada perintah-perintah untuk melakukan tugas yang sama di beberapa tempat, maka akan lebih baik perintah-perintah tersebut dibuat dalam sebuah procedure. Kemudian, procedure itu bisa di-'panggil' bila diperlukan. Penggunaan procedure sangat menghemat penulisan kode program, karena kode-kode program yang sama dibeberapa tempat cukup dibuat pada satu bagian saja. Selain itu, procedure akan memudahkan perbaikan kode program bila terjadi perubahan atau kesalahan, karena perbaikan cukup dilakukan pada satu bagian saja.
Ada beberapa jenis procedure, yaitu :

- **Procedure Sub,** procedure yang tidak mengembalikan nilai setelah 'tugas'-nya selesai .

  ```
  [Public | Private] Sub name [(arglist)]
      [statements]
      [Exit Sub]
      [statements]
  End Sub
  ```

  Setiap prosedur dipanging, maka statements yang berada antara Sub dan End Sub akan dijalankan.

  Ada dua jenis procedure sub, yaitu :
  - **General Procedure**, procedure yang diaktifkan oleh aplikasi
  - **Event Procedure**, procedure yang diaktifkan oleh system sebagai respon terhadap suatu event.

  Contoh, Sub Tengah yang dapat digunakan untuk menampilkan Form ke tengah layer yang mana x adalah parameter Form yang akan dditampilkan pada tengah layer.

```
Sub Tengah (a As Form)
  x.Top = (Screen.Height) –
  x.Height)/2
  x.Left = (Screen.Width) –
  x. Width)/2
End sub

Private Sub Form_Load()
  Call Tengah(Me)
End sub
```

- **Procedure Function,** procedure yang mengembalikan nilai setelah 'tugas'-nya selesai

  Pada Visual Basic telah tersedia berbagai fungsi bawaan seperti misalnya Sqr, Cos dan Chr. Apabila tidak tersedia fungsi tertentu sesuai keinginan kita, suatu fungsi dapat dibuat sendiri sesuai dengan kebutuhan. Fungsi ini dikenal dengan nama Procedure Function.

```
[Public | Private] Function name [(arglist)]
    [statements]
    [name                =              expression]
    [Exit                                 Function]
    [statements]
    [name                =              expression]
End Function
```

  Contoh fungsi yang mengembalikan nama hari dalam bahasa Indonesia dari suatu masukan x angka beruoa tanggal:

```
Private Sub Form_Load()
x = Hari(Date)
Label1.Caption = x
End Sub

Function Hari(x As Date)
Dim Hrn As String
Select Case Weekday(x)
    Case 1: Hrn = "Minggu"
    Case 2: Hrn = "Senin"
    Case 3: Hrn = "Selasa"
    Case 4: Hrn = "Rabu"
    Case 5: Hrn = "Kamis"
    Case 6: Hrn = "Jumat"
    Case 7: Hrn = "Sabtu"
    Case Else: Bln = "Tidak ada"
```

```
End Select
Hari = Hrn
End Function
```

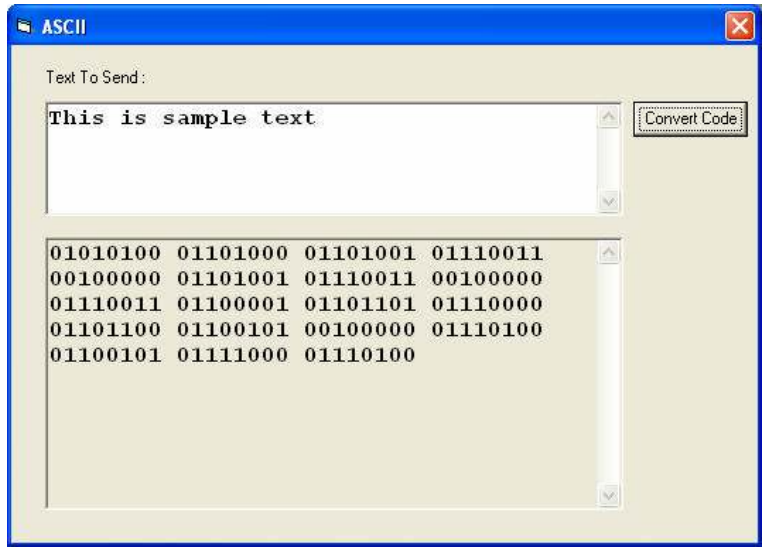
- **Procedure Event,** procedure untuk suatu event pada sebuah object. Digunakan di dalam class module

- **Procedure Property** – procedure untuk mengubah (*let*) atau mengambil (*get*) nilai property pada sebuah object. Digunakan di dalam class module

9.1.18. Latihan Pemrograman

9.1.18.1. Text to Binary

Program ini memperlihatkan bagaimana suatu informasi berupa teks yang dituliskan pada suatu TextBox dapat dikonversi menjadi bilangan biner sesuai kode ASCII
Teks yang akan dikonversi diketik pada TextBox1 yang berwarna putih kemudian dengan menekan tombol "Convert Code" maka akan dihasilkan bilangan binernya pada TextBox2.

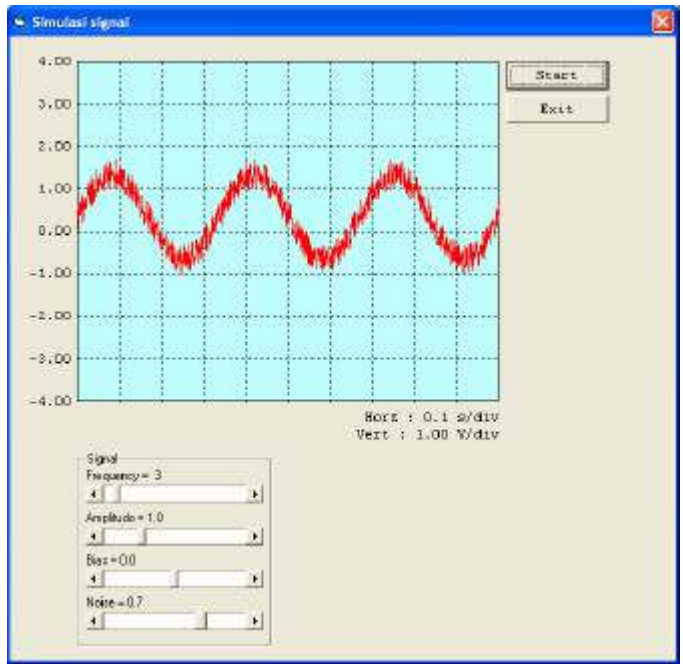


Gambar 9.18 Visualisasi Program Text To Binary

Listing Program

```
Dim l, i, h As Double
Dim t, s, d0, d1, d2, d3, d4, d5, d6, d7  As String
Private Sub Command1_Click()
Text2.Text = ""
l = Len(Text1.Text)
For i = 1 To l
    t = Text1.Text
    s = Mid(t, i, 1)
    h = Asc(s)
    If (h And 1) = 1 Then d0 = "1" Else d0 = "0"
    If (h And 2) = 2 Then d1 = "1" Else d1 = "0"
    If (h And 4) = 4 Then d2 = "1" Else d2 = "0"
    If (h And 8) = 8 Then d3 = "1" Else d3 = "0"
    If (h And 16) = 16 Then d4 = "1" Else d4 = "0"
    If (h And 32) = 32 Then d5 = "1" Else d5 = "0"
    If (h And 64) = 64 Then d6 = "1" Else d6 = "0"
    If (h And 128) = 128 Then d7 = "1" Else d7 = "0"
    Text2.Text = Text2.Text + d7 + d6 + d5 + d4 + d3 +
   d2 + d1 + d0 + " "
Next i
End Sub
```

9.1.18.2. Simulasi Signal



Gambar 9.19 Visualisasi Program Simulasi Sinyal

Program ini untuk mensimulasikan sinyal sinus yang dapat diatur frekuensi, amplitude, bias dan noise dengan mengatur Horisontal Schrollbar.

Listing Program

```
Dim i, UX1, UY1, UX2, UY2, YX1, YY1, YX2, YY2, Ch1,
Ch2 As Double
Dim s1, s2, freq1, freq2, amp1, amp2, bias1, bias2,
noise1, noise2 As Double

Private Sub Check1_Click()
If Check1.Value = 1 Then
    Image1.Visible = False
    Image2.Visible = True
Else
    Image1.Visible = True
    Image2.Visible = False
End If
End Sub

Private Sub Command1_Click()
If Timer1.Enabled = False Then
    Timer1.Enabled = True
    Command1.Caption = "Stop"
    Line23.Visible = True
    If i = 0 Then Form1.Cls
Else
    Timer1.Enabled = False
    Command1.Caption = "Start"
End If
End Sub

Private Sub Command5_Click()
End
End Sub

Private Sub Form_Load()
'inisialisasi
'Seting channel1
HScroll1.Max = 50
HScroll1.Min = 0
HScroll1.Value = 0
HScroll2.Max = 40
HScroll2.Min = 0
```

```
HScroll2.Value = 10
HScroll3.Max = 40
HScroll3.Min = -40
HScroll3.Value = 0
HScroll4.Max = 10
HScroll4.Min = 0
HScroll4.Value = 0
freq1 = HScroll1.Value
amp1 = HScroll2.Value * (1 / 10)
bias1 = HScroll3.Value * (1 / 10)
noise1 = 0.1 * Rnd * HScroll4.Value
Label10.Caption = "Frequency = " + Str(freq1)
Label11.Caption = "Amplitudo = " + Format(amp1, "0.0")
Label12.Caption = "Bias = " + Format(bias1, "0.0")
Label15.Caption = "Noise = " + Format((HScroll4.Value
* 0.1), "0.0")

Command1.Enabled = True
i = 0
UX1 = Shape1.Left
UY1 = Shape1.Top + (Shape1.Height / 2)
YX1 = Shape1.Left
YY1 = Shape1.Top + (Shape1.Height / 2)
Timer1.Interval = 1
Timer1.Enabled = False
Command1.Caption = "Start"
Shape1.Width = Shape1.Height * 10 / 8

'Mengatur garis pada shape
'Garis Horisontal
Line1.X1 = Shape1.Left
Line1.X2 = Shape1.Left + Shape1.Width
Line1.Y1 = Shape1.Top + (1 * (Shape1.Height / 8))
Line1.Y2 = Shape1.Top + (1 * (Shape1.Height / 8))
Line2.X1 = Shape1.Left
Line2.X2 = Shape1.Left + Shape1.Width
Line2.Y1 = Shape1.Top + (2 * (Shape1.Height / 8))
Line2.Y2 = Shape1.Top + (2 * (Shape1.Height / 8))
Line3.X1 = Shape1.Left
Line3.X2 = Shape1.Left + Shape1.Width
Line3.Y1 = Shape1.Top + (3 * (Shape1.Height / 8))
Line3.Y2 = Shape1.Top + (3 * (Shape1.Height / 8))
Line4.X1 = Shape1.Left
Line4.X2 = Shape1.Left + Shape1.Width
Line4.Y1 = Shape1.Top + (4 * (Shape1.Height / 8))
Line4.Y2 = Shape1.Top + (4 * (Shape1.Height / 8))
Line5.X1 = Shape1.Left
```

```
Line5.X2 = Shape1.Left + Shape1.Width
Line5.Y1 = Shape1.Top + (5 * (Shape1.Height / 8))
Line5.Y2 = Shape1.Top + (5 * (Shape1.Height / 8))
Line6.X1 = Shape1.Left
Line6.X2 = Shape1.Left + Shape1.Width
Line6.Y1 = Shape1.Top + (6 * (Shape1.Height / 8))
Line6.Y2 = Shape1.Top + (6 * (Shape1.Height / 8))
Line7.X1 = Shape1.Left
Line7.X2 = Shape1.Left + Shape1.Width
Line7.Y1 = Shape1.Top + (7 * (Shape1.Height / 8))
Line7.Y2 = Shape1.Top + (7 * (Shape1.Height / 8))

'Garis vertikal
Line8.X1 = Shape1.Left + (1 * (Shape1.Width / 10))
Line8.X2 = Shape1.Left + (1 * (Shape1.Width / 10))
Line8.Y1 = Shape1.Top + Shape1.Height
Line8.Y2 = Shape1.Top
Line9.X1 = Shape1.Left + (2 * (Shape1.Width / 10))
Line9.X2 = Shape1.Left + (2 * (Shape1.Width / 10))
Line9.Y1 = Shape1.Top + Shape1.Height
Line9.Y2 = Shape1.Top
Line10.X1 = Shape1.Left + (3 * (Shape1.Width / 10))
Line10.X2 = Shape1.Left + (3 * (Shape1.Width / 10))
Line10.Y1 = Shape1.Top + Shape1.Height
Line10.Y2 = Shape1.Top
Line11.X1 = Shape1.Left + (4 * (Shape1.Width / 10))
Line11.X2 = Shape1.Left + (4 * (Shape1.Width / 10))
Line11.Y1 = Shape1.Top + Shape1.Height
Line11.Y2 = Shape1.Top
Line12.X1 = Shape1.Left + (5 * (Shape1.Width / 10))
Line12.X2 = Shape1.Left + (5 * (Shape1.Width / 10))
Line12.Y1 = Shape1.Top + Shape1.Height
Line12.Y2 = Shape1.Top
Line13.X1 = Shape1.Left + (6 * (Shape1.Width / 10))
Line13.X2 = Shape1.Left + (6 * (Shape1.Width / 10))
Line13.Y1 = Shape1.Top + Shape1.Height
Line13.Y2 = Shape1.Top
Line14.X1 = Shape1.Left + (7 * (Shape1.Width / 10))
Line14.X2 = Shape1.Left + (7 * (Shape1.Width / 10))
Line14.Y1 = Shape1.Top + Shape1.Height
Line14.Y2 = Shape1.Top
Line15.X1 = Shape1.Left + (8 * (Shape1.Width / 10))
Line15.X2 = Shape1.Left + (8 * (Shape1.Width / 10))
Line15.Y1 = Shape1.Top + Shape1.Height
Line15.Y2 = Shape1.Top
Line16.X1 = Shape1.Left + (9 * (Shape1.Width / 10))
Line16.X2 = Shape1.Left + (9 * (Shape1.Width / 10))
```

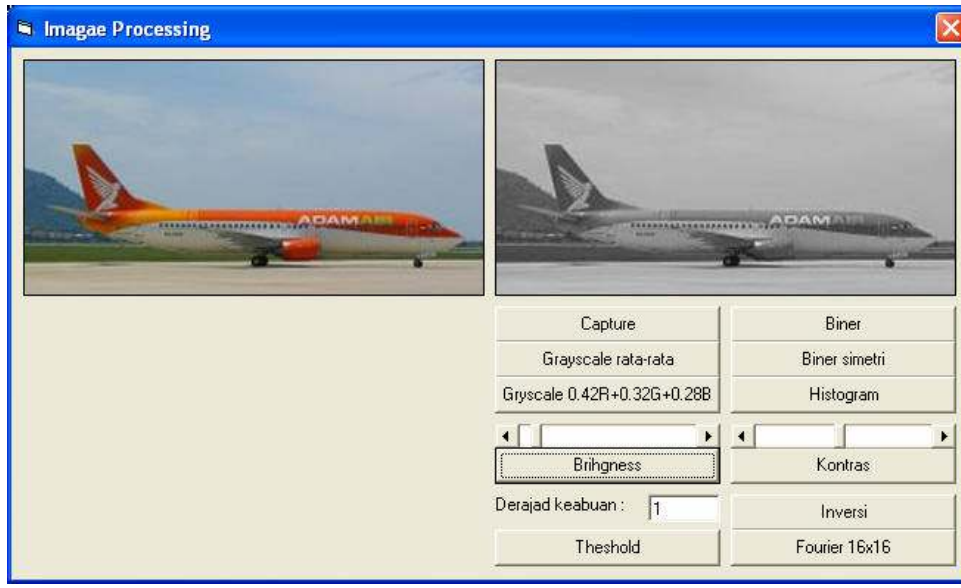
```
Line16.Y1 = Shape1.Top + Shape1.Height
Line16.Y2 = Shape1.Top
Line23.X1 = Shape1.Left
Line23.X2 = Shape1.Left
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height

'Mengatur label skala sumbu y
Label1.Height = 250
Label2.Height = 250
Label3.Height = 250
Label4.Height = 250
Label5.Height = 250
Label6.Height = 250
Label7.Height = 250
Label8.Height = 250
Label9.Height = 250
Label1.Height = 250
Label1.Width = 615
Label2.Width = 615
Label3.Width = 615
Label4.Width = 615
Label5.Width = 615
Label6.Width = 615
Label7.Width = 615
Label8.Width = 615
Label9.Width = 615
Label1.Alignment = 1 ' rata kanan
Label2.Alignment = 1 ' rata kanan
Label3.Alignment = 1 ' rata kanan
Label4.Alignment = 1 ' rata kanan
Label5.Alignment = 1 ' rata kanan
Label6.Alignment = 1 ' rata kanan
Label7.Alignment = 1 ' rata kanan
Label8.Alignment = 1 ' rata kanan
Label9.Alignment = 1 ' rata kanan
Label1.Left = Shape1.Left - 700
Label2.Left = Shape1.Left - 700
Label3.Left = Shape1.Left - 700
Label4.Left = Shape1.Left - 700
Label5.Left = Shape1.Left - 700
Label6.Left = Shape1.Left - 700
Label7.Left = Shape1.Left - 700
Label8.Left = Shape1.Left - 700
Label9.Left = Shape1.Left - 700
Label1.Top = Shape1.Top + (0 * (Shape1.Height / 8)) -
125
```

```
Label2.Top = Shape1.Top + (1 * (Shape1.Height / 8)) -
125
Label3.Top = Shape1.Top + (2 * (Shape1.Height / 8)) -
125
Label4.Top = Shape1.Top + (3 * (Shape1.Height / 8)) -
125
Label5.Top = Shape1.Top + (4 * (Shape1.Height / 8)) -
125
Label6.Top = Shape1.Top + (5 * (Shape1.Height / 8)) -
125
Label7.Top = Shape1.Top + (6 * (Shape1.Height / 8)) -
125
Label8.Top = Shape1.Top + (7 * (Shape1.Height / 8)) -
125
Label9.Top = Shape1.Top + (8 * (Shape1.Height / 8)) -
125
End Sub

Private Sub HScroll1_Change()
freq1 = HScroll1.Value
Label10.Caption = "Frequency = " + Str(freq1)
End Sub

Private Sub HScroll2_Change()
amp1 = HScroll2.Value * (1 / 10)
Label11.Caption = "Amplitudo = " + Format(amp1, "0.0")
End Sub

Private Sub HScroll3_Change()
bias1 = HScroll3.Value * (1 / 10)
Label12.Caption = "Bias = " + Format(bias1, "0.0")
End Sub

Private Sub HScroll4_Change()
noise1 = 0.1 * Rnd * HScroll4.Value
Label15.Caption = "Noise = " + Format((HScroll4.Value
* 0.1), "0.0")
End Sub

Private Sub Timer1_Timer()
'Channel1
freq1 = HScroll1.Value
amp1 = HScroll2.Value * (1 / 10)
bias1 = HScroll3.Value * (1 / 10)
noise1 = 0.1 * Rnd * HScroll4.Value
Label10.Caption = "Frequency = " + Str(freq1)
Label11.Caption = "Amplitudo = " + Format(amp1, "0.0")
```

```
Label12.Caption = "Bias = " + Format(bias1, "0.0")
Label15.Caption = "Noise = " + Format((HScroll4.Value
* 0.1), "0.0")
s1 = (Sin(i * ((360 * freq1) / 1000) * (3.14 / 180)) *
amp1) + bias1 + noise1
Ch1 = s1
Ch2 = 0

'Menampilkan Grafik Ch1
v = (Shape1.Height / 8)
h = ((Shape1.Width / 1000))
UX2 = Shape1.Left + (i * h)
UY2 = Shape1.Top + (Shape1.Height / 2) - (v * Ch1)
If UY2 < Shape1.Top Then UY2 = Shape1.Top
If  UY2  >  Shape1.Top  +  Shape1.Height  Then  UY2  =
Shape1.Top + Shape1.Height
Line (UX1, UY1)-(UX2, UY2), vbRed
'Menampilkan Grafik Ch2
YX2 = Shape1.Left + (i * h)
YY2 = Shape1.Top + (Shape1.Height / 2) - (v * Ch2)
If YY2 < Shape1.Top Then YY2 = Shape1.Top
If  YY2  >  Shape1.Top  +  Shape1.Height  Then  YY2  =
Shape1.Top + Shape1.Height
Line23.X1 = YX2 + 60
Line23.X2 = YX2 + 60
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height
'Line (YX1, YY1)-(YX2, YY2), vbBlue

'Mengirimkan data berikutnya
UX1 = UX2
UY1 = UY2
YX1 = YX2
YY1 = YY2
'Pembacaan data masukan
i = i + 1
If i >= 1000 Then
    Timer1.Enabled = False
    Line23.Visible = False
    UX1 = Shape1.Left
    YX1 = Shape1.Left
    Command1.Caption = "Start"
    i = 0
End If
End Sub
```

9.1.18.3. Image Processing

Program ini memberikan contoh bahwa suatu gambar data diproses kembali sesuai dengan kebutuhan.



Gambar 9.20 Visualisasi Program Image Processing

Listing Program

```
Dim i, j, w, r, g, b, bg, x1, xb, k, k1, k2, fr, xk,
th, a As Double
Dim n1, n2, m1, m2 As Integer
Dim x(400, 400) As Integer
Dim xr(400, 400) As Single

Private Sub Command1_Click()
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       Picture2.PSet (i, j), RGB(r, g, b)
    Next j
Next i
End Sub
```

```
Private Sub Command10_Click()
k = Val(HScroll2.Value) / 10
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (r + g + b) / 3
       xk = k * x1
       Picture2.PSet (i, j), RGB(xk, xk, xk)
    Next j
Next i
End Sub

Private Sub Command11_Click()
m1 = 16: m2 = 16
n1 = 0
For i = 1 To Picture1.ScaleWidth Step 15
    n1 = n1 + 1
    n2 = 0
    For j = 1 To Picture1.ScaleHeight Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       n2 = n2 + 1
       x(n1, n2) = Int((r + g + b) / 3)
       Picture2.PSet (i, j), RGB(x(n1, n2), x(n1, n2),
x(n1, n2))
    Next j
Next i
Picture2.ScaleHeight = m1 + 1
Picture2.ScaleWidth = m2 + 1
For i = 1 To m1
    For j = 1 To m2
        fr = 0
        For k1 = 1 To n1
            For k2 = 1 To n2
                fr = fr + x(k1, k2) * Cos(6.28 * (i *
k1 / m1 + j * k2 / m2))
            Next k2
        Next k1
        w = 255 * Abs(fr) / (n1 * n2)
```

```
        Picture2.Line (i - 0.5, j - 0.5)-(i + 0.5, j +
0.5), RGB(w, w, w), BF
        w = 255 * Abs(fr) / (n1 * n2)
        xr(i, j) = fr
    Next j
Next i
End Sub

Private Sub Command2_Click()
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (r + g + b) / 3
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
End Sub

Private Sub Command3_Click()
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (r + g + b) / 3
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
End Sub

Private Sub Command4_Click()
bg = Val(HScroll1.Value)
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (0.42 * r) + (0.32 * g) + (0.28 * b)
       xb = x1 + bg
```

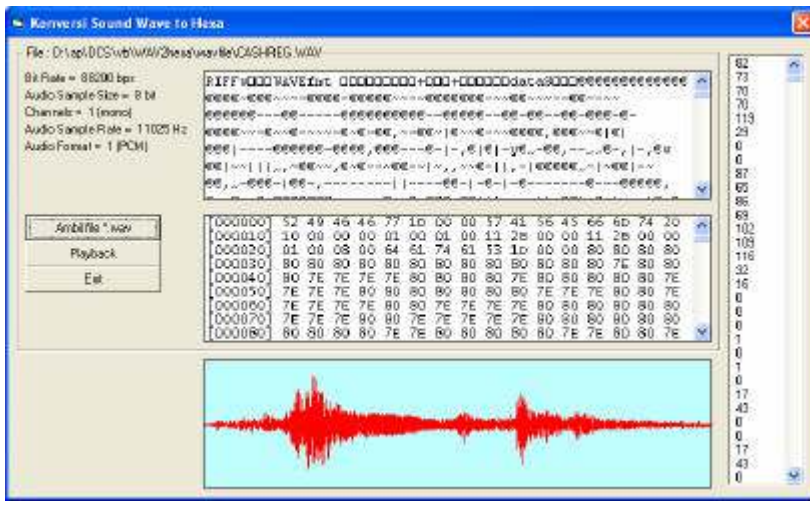
```
       Picture2.PSet (i, j), RGB(xb, xb, xb)
    Next j
Next i
End Sub

Private Sub Command5_Click()
th = Val(Text1)
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (r + g + b) / 3
       a = Int(256 / th)
       x1 = a * Int(x1 / a)
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
End Sub

Private Sub Command6_Click()
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = (r + g + b) / 3
       If x1 < 128 Then x1 = 0 Else x1 = 255
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
End Sub

Private Sub Command7_Click()
Dim xr As Single
Dim wx(400, 300) As Integer
xr = 0: m = 0
For i = 1 To Picture1.Width Step 15
    m = m + 1
    n = 0
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
```

```
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       n = n + 1
       x1 = (r + g + b) / 3
       xr = xr + x1
       wx(m, n) = x1
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
xr = xr / (m * n)
For i = 1 To m
    For j = 1 To n
        If wx(i, j) < xr Then x1 = 0 Else x1 = 255
        Picture2.PSet (15 * (i - 1) + 1, 15 * (j - 1)
+ 1), RGB(x1, x1, x1)
    Next j
Next i
End Sub

Private Sub Command8_Click()
Dim ht, xp As Double
Dim h(256) As Integer
For i = 1 To 256
    h(i) = 0
Next i
For i = 1 To Picture1.Width Step 15
    For j = 1 To Picture1.Height Step 15
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       x1 = Int((r + g + b) / 3)
       h(x1 + 1) = h(x1 + 1) + 1
       Picture2.PSet (i, j), RGB(x1, x1, x1)
    Next j
Next i
ht2 = Picture2.Height
For i = 1 To 256
    xp = 15 * (i - 1) + 1
    Picture2.Line (xp, ht2 - h(i))-(xp, ht2), RGB(255,
0, 0)
Next i
End Sub
```

```
Private Sub Command9_Click()
For i = 1 To Picture1.Width Step 10
    For j = 1 To Picture1.Height Step 10
       w = Picture1.Point(i, j)
       r = w And RGB(255, 0, 0)
       g = Int((w And RGB(0, 255, 0)) / 256)
       b = Int(Int((w And RGB(0, 0, 255)) / 256) /
256)
       Picture2.PSet (Picture1.Width - i, j), RGB(r,
g, b)
    Next j
Next i
End Sub

Private Sub Form_Load()
HScroll1.Max = 127
HScroll1.Min = 0
HScroll1.Value = 0
HScroll2.Max = 20
HScroll2.Min = 1
HScroll2.Value = 10
End Sub
```

9.1.18.4. WAVE PCM Sound File Format

Program ini memperlihatkan bahwa suatu file WAV dapat dilihat datanya dalam bentuk ASCII, data Hexa, data decimal dan grafik bentuk gelombangnya sekaligus disuarakan lagi



Gambar 9.21 Visualisasi Program WAVE PCM sound file format

Listing Program

```
Dim q, v, h, UX1, UY1, UX2, UY2, YX1, YY1, YX2, YY2,
Ch1, Ch2 As Double
Dim lfn, i, dd, l, l1, l2, l3, dt As Double
Dim fn, fn1, fn2, tl, ds, adrs, txt, txt1, txt2, tdin,
tdin2 As String
Dim CZ, Sc1S, NC, SR, BR, BA, BPS, Sc2S As Double
Dim d1, d2, d3, d4, d5 As String
Dim db As Variant
Dim FileNo As Integer
Dim ldin, ibc As Integer

Private Sub Command1_Click()
Command2.Enabled = False
Label1.Caption = ""
Label2.Caption = ""
Label3.Caption = ""
Label4.Caption = ""
Label5.Caption = ""
RichTextBox1.Text = ""
'Membuka File Gambar
CommonDialog1.FileName = "*.wav"
CommonDialog1.Filter = "Semua file (*.*)|*.*|File WAV
(*.wav)|"
CommonDialog1.ShowOpen
FileNo = FreeFile
On Error Resume Next
fn = CommonDialog1.FileName
If InStr(fn, ":") Then
    Form1.Cls
    RichTextBox1.LoadFile (fn)
    db = RichTextBox1.Text
    Frame1.Caption = "File : " + fn
    lfn = Len(RichTextBox1.Text)
    l1 = lfn
    l2 = lfn \ 16
    l3 = lfn Mod 16
    l = 0
    List1.Clear
    List2.Clear
    For x = 1 To (l2)
        tl = ""
        For i = l To (l + 15)
            dd = Asc(Mid(RichTextBox1.Text, i + 1, 1))
            txt = Mid(RichTextBox1.Text, i + 1, 1)
```

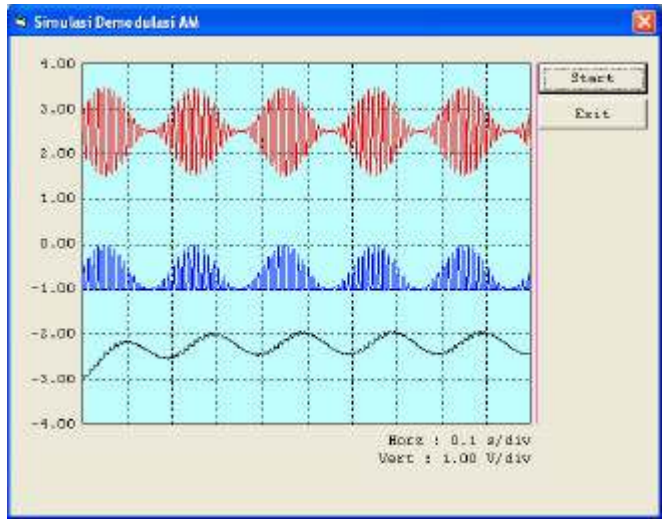
```
            txt1 = txt1 + txt
            ds = Hex(dd)
            List2.AddItem Str(dd)
            If Len(ds) = 1 Then ds = "0" + Hex(dd)
            tl = tl & ds & " "
        Next i
        adrs = Hex(l)
        If Len(adrs) = 1 Then adrs = "00000" + adrs
        If Len(adrs) = 2 Then adrs = "0000" + adrs
        If Len(adrs) = 3 Then adrs = "000" + adrs
        If Len(adrs) = 4 Then adrs = "00" + adrs
        If Len(adrs) = 5 Then adrs = "0" + adrs
        List1.AddItem "[" + adrs + "] " + tl
        l = l + 16
    Next x
    If l3 > 0 Then
        tl = ""
        For i = 1 To l3
            dd = Asc(Mid(RichTextBox1.Text, l + i, 1))
            txt = Mid(RichTextBox1.Text, i + 1, 1)
            txt1 = txt1 + txt
            ds = Hex(dd)
            If Len(ds) = 1 Then ds = "0" + Hex(dd)
            tl = tl & ds & " "
        Next i
        adrs = Hex(l)
        If Len(adrs) = 1 Then adrs = "00000" + adrs
        If Len(adrs) = 2 Then adrs = "0000" + adrs
        If Len(adrs) = 3 Then adrs = "000" + adrs
        If Len(adrs) = 4 Then adrs = "00" + adrs
        If Len(adrs) = 5 Then adrs = "0" + adrs
        List1.AddItem "[" + adrs + "] " + tl
    End If
    List1.AddItem "[Filezise : " + Str(l1) + " Byte]"
    d1 = Mid(List1.List(1), 55, 2)
    d2 = Mid(List1.List(1), 52, 2)
    d3 = Mid(List1.List(1), 49, 2)
    d4 = Mid(List1.List(1), 46, 2)
    d5 = "&h" + d1 + d2 + d3 + d4
     Label1.Caption = "Bit Rate = " + Str(Val(d5) * 8)
+ " bps"
    d1 = Mid(List1.List(2), 19, 2)
    d2 = Mid(List1.List(2), 16, 2)
    d5 = "&h" + d1 + d2
     Label2.Caption = "Audio Sample Size = " + Str(d5)
+ " bit"
    d1 = Mid(List1.List(1), 31, 2)
```

```
    d2 = Mid(List1.List(1), 28, 2)
    d5 = "&h" + d1 + d2
    If (Str(d5) = 1) Then Label3.Caption = "Channels =
" + Str(d5) + " (mono)"
    If (Str(d5) = 2) Then Label3.Caption = "Channels =
" + Str(d5) + " (stereo)"
    If (Str(d5) > 2) Then Label3.Caption = "Channels =
" + Str(d5)
    d1 = Mid(List1.List(1), 43, 2)
    d2 = Mid(List1.List(1), 40, 2)
    d3 = Mid(List1.List(1), 37, 2)
    d4 = Mid(List1.List(1), 34, 2)
    d5 = "&h" + d1 + d2 + d3 + d4
    Label4.Caption = "Audio Sample Rate = " + Str(d5)
+ " Hz"
    d1 = Mid(List1.List(1), 25, 2)
    d2 = Mid(List1.List(1), 22, 2)
    d5 = "&h" + d1 + d2
    If (Str(d5) = 1) Then Label5.Caption = "Audio
Format = " + Str(d5) + " (PCM)"
    If (Str(d5) > 1) Then Label5.Caption = "Audio
Format = " + Str(d5)

    If InStr(Label3.Caption, "mono") Then
    q = 0
    Ch1 = Val(List2.List(q + 44))
    dt = List2.ListCount - 200
    v = (Shape1.Height / 255)
    h = (Shape1.Width / dt)
    UX1 = Shape1.Left + (q * h)
    UY1 = Shape1.Top + (Shape1.Height) - (v * Ch1)
    For q = 1 To (dt)
        Ch1 = Val(List2.List(q + 44))
        UX2 = Shape1.Left + (q * h)
        UY2 = Shape1.Top + (Shape1.Height) - (v * Ch1)
        If UY2 < Shape1.Top Then UY2 = Shape1.Top
        If UY2 > Shape1.Top + Shape1.Height Then UY2 =
Shape1.Top + Shape1.Height
        Line (UX1, UY1)-(UX2, UY2), vbRed
        UX1 = UX2
        UY1 = UY2
    Next q
    Else
        MsgBox "Waweform tidak ditampilkan karena
scope ini hanya dirancang untuk sinyal mono"
    End If
End If
```

```
Command2.Enabled = True
End Sub

Private Sub Command2_Click()
Dim retVal As Long
retVal& = sndPlaySound(fn, SND_ASYNC)
End Sub

Private Sub Command3_Click()
End
End Sub

Private Sub Form_Load()
Command2.Enabled = False
Label1.Caption = ""
Label2.Caption = ""
Label3.Caption = ""
Label4.Caption = ""
Label5.Caption = ""
End Sub
```

9.1.18.5. Simulasi Modulasi-Demodulasi AM

Program ini mensimulasikan modulasi dan demodulasi Sinyal yang berwarna merah adalah sinyal hasil modulasi. Sinyal biru adalah hasil penyearahan oleh dioda detektor. Sinyal hijau adalah hasil filter RC.



Gambar 9.22 Visualisasi Program Simulasi Modulasi–Demodulasi AM

Listing Program

```
Dim i, UX1, UY1, UX2, UY2, YX1, YY1, YX2, YY2, ZX1,
ZY1, ZX2, ZY2 As Double
Dim bias1, bias2, freq1, freq2, freq3, amp1, amp2 As
Double
Dim aCh1, Ch2, s1, s2, s3, s4, am, dm, Unow, Ynow,
Uold, Yold As Double


Private Sub Check1_Click()
If Check1.Value = 1 Then
    Image1.Visible = False
    Image2.Visible = True
Else
    Image1.Visible = True
    Image2.Visible = False
End If
End Sub

Private Sub Command1_Click()
If Timer1.Enabled = False Then
    Timer1.Enabled = True
    Command1.Caption = "Stop"
    Form1.Cls
    i = 0
Else
    Timer1.Enabled = False
    Command1.Caption = "Start"
End If
End Sub

Private Sub Command5_Click()
End
End Sub

Private Sub Form_Load()
'inisialisasi
Command1.Enabled = True
i = 0
UX1 = Shape1.Left
UY1 = Shape1.Top + (Shape1.Height / 2)
YX1 = Shape1.Left
YY1 = Shape1.Top + (Shape1.Height / 2)
ZX1 = Shape1.Left
ZY1 = Shape1.Top + (Shape1.Height / 2)
Uold = 0
```

```
Yold = 0
Timer1.Interval = 1
Timer1.Enabled = False
Command1.Caption = "Start"
Shape1.Width = Shape1.Height * 10 / 8

'Mengatur garis pada shape
'Garis Horisontal
Line1.X1 = Shape1.Left
Line1.X2 = Shape1.Left + Shape1.Width
Line1.Y1 = Shape1.Top + (1 * (Shape1.Height / 8))
Line1.Y2 = Shape1.Top + (1 * (Shape1.Height / 8))
Line2.X1 = Shape1.Left
Line2.X2 = Shape1.Left + Shape1.Width
Line2.Y1 = Shape1.Top + (2 * (Shape1.Height / 8))
Line2.Y2 = Shape1.Top + (2 * (Shape1.Height / 8))
Line3.X1 = Shape1.Left
Line3.X2 = Shape1.Left + Shape1.Width
Line3.Y1 = Shape1.Top + (3 * (Shape1.Height / 8))
Line3.Y2 = Shape1.Top + (3 * (Shape1.Height / 8))
Line4.X1 = Shape1.Left
Line4.X2 = Shape1.Left + Shape1.Width
Line4.Y1 = Shape1.Top + (4 * (Shape1.Height / 8))
Line4.Y2 = Shape1.Top + (4 * (Shape1.Height / 8))
Line5.X1 = Shape1.Left
Line5.X2 = Shape1.Left + Shape1.Width
Line5.Y1 = Shape1.Top + (5 * (Shape1.Height / 8))
Line5.Y2 = Shape1.Top + (5 * (Shape1.Height / 8))
Line6.X1 = Shape1.Left
Line6.X2 = Shape1.Left + Shape1.Width
Line6.Y1 = Shape1.Top + (6 * (Shape1.Height / 8))
Line6.Y2 = Shape1.Top + (6 * (Shape1.Height / 8))
Line7.X1 = Shape1.Left
Line7.X2 = Shape1.Left + Shape1.Width
Line7.Y1 = Shape1.Top + (7 * (Shape1.Height / 8))
Line7.Y2 = Shape1.Top + (7 * (Shape1.Height / 8))

'Garis vertikal
Line8.X1 = Shape1.Left + (1 * (Shape1.Width / 10))
Line8.X2 = Shape1.Left + (1 * (Shape1.Width / 10))
Line8.Y1 = Shape1.Top + Shape1.Height
Line8.Y2 = Shape1.Top
Line9.X1 = Shape1.Left + (2 * (Shape1.Width / 10))
Line9.X2 = Shape1.Left + (2 * (Shape1.Width / 10))
Line9.Y1 = Shape1.Top + Shape1.Height
Line9.Y2 = Shape1.Top
Line10.X1 = Shape1.Left + (3 * (Shape1.Width / 10))
```

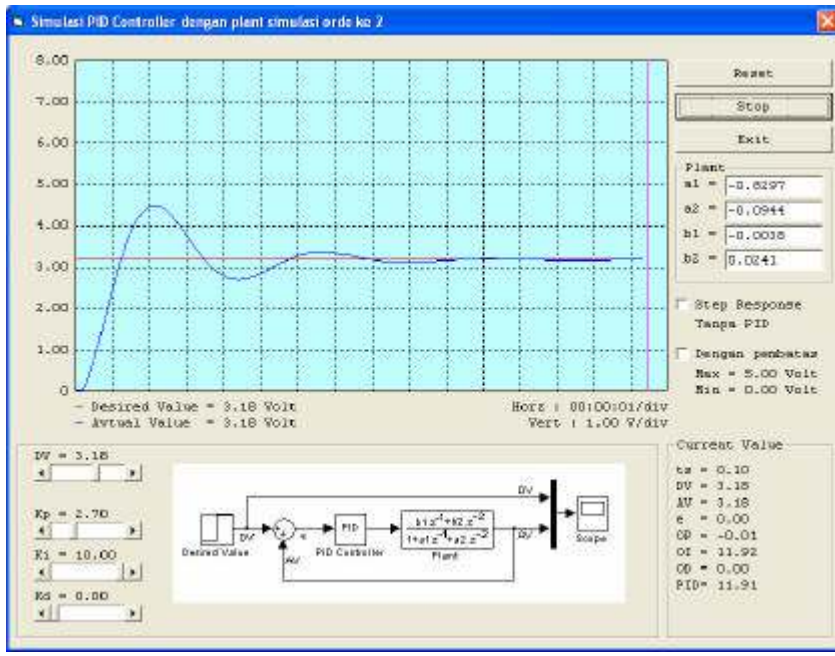
```
Line10.X2 = Shape1.Left + (3 * (Shape1.Width / 10))
Line10.Y1 = Shape1.Top + Shape1.Height
Line10.Y2 = Shape1.Top
Line11.X1 = Shape1.Left + (4 * (Shape1.Width / 10))
Line11.X2 = Shape1.Left + (4 * (Shape1.Width / 10))
Line11.Y1 = Shape1.Top + Shape1.Height
Line11.Y2 = Shape1.Top
Line12.X1 = Shape1.Left + (5 * (Shape1.Width / 10))
Line12.X2 = Shape1.Left + (5 * (Shape1.Width / 10))
Line12.Y1 = Shape1.Top + Shape1.Height
Line12.Y2 = Shape1.Top
Line13.X1 = Shape1.Left + (6 * (Shape1.Width / 10))
Line13.X2 = Shape1.Left + (6 * (Shape1.Width / 10))
Line13.Y1 = Shape1.Top + Shape1.Height
Line13.Y2 = Shape1.Top
Line14.X1 = Shape1.Left + (7 * (Shape1.Width / 10))
Line14.X2 = Shape1.Left + (7 * (Shape1.Width / 10))
Line14.Y1 = Shape1.Top + Shape1.Height
Line14.Y2 = Shape1.Top
Line15.X1 = Shape1.Left + (8 * (Shape1.Width / 10))
Line15.X2 = Shape1.Left + (8 * (Shape1.Width / 10))
Line15.Y1 = Shape1.Top + Shape1.Height
Line15.Y2 = Shape1.Top
Line16.X1 = Shape1.Left + (9 * (Shape1.Width / 10))
Line16.X2 = Shape1.Left + (9 * (Shape1.Width / 10))
Line16.Y1 = Shape1.Top + Shape1.Height
Line16.Y2 = Shape1.Top
Line23.X1 = Shape1.Left
Line23.X2 = Shape1.Left
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height

'Mengatur label skala sumbu y
Label1.Height = 250
Label2.Height = 250
Label3.Height = 250
Label4.Height = 250
Label5.Height = 250
Label6.Height = 250
Label7.Height = 250
Label8.Height = 250
Label9.Height = 250
Label1.Height = 250
Label1.Width = 615
Label2.Width = 615
Label3.Width = 615
Label4.Width = 615
```

```
Label5.Width = 615
Label6.Width = 615
Label7.Width = 615
Label8.Width = 615
Label9.Width = 615
Label1.Alignment = 1 ' rata kanan
Label2.Alignment = 1 ' rata kanan
Label3.Alignment = 1 ' rata kanan
Label4.Alignment = 1 ' rata kanan
Label5.Alignment = 1 ' rata kanan
Label6.Alignment = 1 ' rata kanan
Label7.Alignment = 1 ' rata kanan
Label8.Alignment = 1 ' rata kanan
Label9.Alignment = 1 ' rata kanan
Label1.Left = Shape1.Left - 700
Label2.Left = Shape1.Left - 700
Label3.Left = Shape1.Left - 700
Label4.Left = Shape1.Left - 700
Label5.Left = Shape1.Left - 700
Label6.Left = Shape1.Left - 700
Label7.Left = Shape1.Left - 700
Label8.Left = Shape1.Left - 700
Label9.Left = Shape1.Left - 700
Label1.Top = Shape1.Top + (0 * (Shape1.Height / 8)) -
125
Label2.Top = Shape1.Top + (1 * (Shape1.Height / 8)) -
125
Label3.Top = Shape1.Top + (2 * (Shape1.Height / 8)) -
125
Label4.Top = Shape1.Top + (3 * (Shape1.Height / 8)) -
125
Label5.Top = Shape1.Top + (4 * (Shape1.Height / 8)) -
125
Label6.Top = Shape1.Top + (5 * (Shape1.Height / 8)) -
125
Label7.Top = Shape1.Top + (6 * (Shape1.Height / 8)) -
125
Label8.Top = Shape1.Top + (7 * (Shape1.Height / 8)) -
125
Label9.Top = Shape1.Top + (8 * (Shape1.Height / 8)) -
125
End Sub

Private Sub Timer1_Timer()
freq1 = 5   'signal
bias1 = 0.5
amp1 = 0.5
```

```
s1 = (Sin(i * ((360 * freq1) / 1000) * (3.14 / 180)) *
amp1) + bias1
freq2 = 90  'carrier
bias2 = 0
amp2 = 1
s2 = (Sin(i * ((360 * freq2) / 1000) * (3.14 / 180)) *
amp2) + bias2
s3 = (s1 * s2) + 2.5 'Amplitudo Modulation +bias 2
volt
If s3 >= 2.5 Then s4 = s3 - 2.5 Else s4 = 0
Unow = s4
Ynow = (0.99 * Yold) + (0.00995 * Uold)

Ch1 = s3
Ch2 = s4 - 1
Ch3 = (Ynow * 5) - 3
'Menampilkan Grafik Ch1
v = (Shape1.Height / 8)
h = ((Shape1.Width / 1000))
UX2 = Shape1.Left + (i * h)
UY2 = Shape1.Top + (Shape1.Height / 2) - (v * Ch1)
If UY2 < Shape1.Top Then UY2 = Shape1.Top
If  UY2  >  Shape1.Top  +  Shape1.Height  Then  UY2  =
Shape1.Top + Shape1.Height
Line (UX1, UY1)-(UX2, UY2), vbRed
'Menampilkan Grafik Ch2
YX2 = Shape1.Left + (i * h)
YY2 = Shape1.Top + (Shape1.Height / 2) - (v * Ch2)
If YY2 < Shape1.Top Then YY2 = Shape1.Top
If  YY2  >  Shape1.Top  +  Shape1.Height  Then  YY2  =
Shape1.Top + Shape1.Height
Line (YX1, YY1)-(YX2, YY2), vbBlue
'Menampilkan Grafik Ch3
ZX2 = Shape1.Left + (i * h)
ZY2 = Shape1.Top + (Shape1.Height / 2) - (v * Ch3)
If ZY2 < Shape1.Top Then ZY2 = Shape1.Top
If  ZY2  >  Shape1.Top  +  Shape1.Height  Then  ZY2  =
Shape1.Top + Shape1.Height
Line23.X1 = YX2 + 60
Line23.X2 = YX2 + 60
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height
Line (ZX1, ZY1)-(ZX2, ZY2), vbMaron
'Mengirimkan data berikutnya
UX1 = UX2
UY1 = UY2
YX1 = YX2
```

```
YY1 = YY2
ZX1 = ZX2
ZY1 = ZY2
Uold = Unow
Yold = Ynow
'Pembacaan data masukan
i = i + 1
If i >= 1000 Then
    Timer1.Enabled = False
    Command1.Caption = "Start"
    'Form1.Cls
    'i = 0
    UX1 = Shape1.Left
    UY1 = Shape1.Top + (Shape1.Height / 2)
    YX1 = Shape1.Left
    YY1 = Shape1.Top + (Shape1.Height / 2)
    Uold = 0
    Yold = 0
End If
End Sub
```

9.1.18.6. Simulasi Kontrol PID



Gambar 9.23 Visualisasi Program Simulasi Kontrol PID

Listing Program

```
Dim a1, a2, b1, b2 As Double
Dim ts, du, dt, DV, AV, Kp, Ki, Kd, POut, eio As
Double
Dim eiold, IOut, IOutold, edo, edoold, DOut, PIDOut As
Double
Dim i, UX1, UY1, UX2, UY2, YX1, YY1, YX2, YY2, OutPID
As Double
Dim Unow, Um1, Um2, Ynow, Ym1, Ym2, Yp As Double
Dim d1, d2, d3, d4, d5, d6 As String

Private Sub Check1_Click()
If Check1.Value = 1 Then
    Image1.Visible = False
    Image2.Visible = True
Else
    Image1.Visible = True
    Image2.Visible = False
End If
End Sub

Private Sub Command1_Click()
If Timer1.Enabled = False Then
    Timer1.Enabled = True
    Command1.Caption = "Stop"
Else
    Timer1.Enabled = False
    Command1.Caption = "Start"
End If
End Sub

Private Sub Command2_Click()
'inisialisasi
i = 0
ts = 0.1
Timer1.Interval = 100
Timer1.Enabled = False
Command1.Caption = "Start"
DV = HScroll1.Value * (1 / 100)
Kp = HScroll2.Value * (1 / 100)
Ki = HScroll3.Value * (1 / 100)
Kd = HScroll4.Value * (1 / 100)
eiold = 0
IOutold = 0
edoold = 0
edo = 0
```

```
e = 0
Ym1 = 0
Ym2 = 0
Um1 = 0
Um2 = 0
Ynow = 0
Unow = 0
Yp = 0
PIDOut = 0
Label15.Caption = "DV = " + Format(DV, "#0.00 Volt")
Label16.Caption = "Kp = " + Format(Kp, "#0.00")
Label17.Caption = "Ki = " + Format(Ki, "#0.00")
Label18.Caption = "Kd = " + Format(Kd, "#0.00")
Form1.Cls
UX1 = Shape1.Left
UY1 = Shape1.Top + Shape1.Height
YX1 = Shape1.Left
YY1 = Shape1.Top + Shape1.Height
Line23.X1 = Shape1.Left
Line23.X2 = Shape1.Left
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height
End Sub
Private Sub Command5_Click()
End
End Sub

Private Sub Form_Load()
'inisialisasi
Image1.Visible = True
Image2.Visible = False
Text1.Text = "-0.8297"
Text2.Text = "-0.0944"
Text3.Text = "-0.0038"
Text4.Text = "0.0241"
'Text1.Text = "0.9048"
'Text2.Text = "0"
'Text3.Text = "0.09516"
'Text4.Text = "0"
Command1.Enabled = True
Command2.Enabled = True
i = 0
ts = 0.1
UX1 = Shape1.Left
UY1 = Shape1.Top + Shape1.Height
YX1 = Shape1.Left
YY1 = Shape1.Top + Shape1.Height
```

```
Timer1.Interval = 100
Timer1.Enabled = False
Command1.Caption = "Start"
HScroll1.Max = 500
HScroll1.Min = 0
HScroll2.Max = 1000
HScroll2.Min = 0
HScroll3.Max = 1000
HScroll3.Min = 0
HScroll4.Max = 1000
HScroll4.Min = 0
HScroll1.Value = 0
HScroll2.Value = 0
HScroll3.Value = 0
DV = HScroll1.Value * (1 / 100)
Kp = HScroll2.Value * (1 / 100)
Ki = HScroll3.Value * (1 / 100)
Kd = HScroll4.Value * (1 / 100)
eiold = 0
IOutold = 0
edoold = 0
PIDOut = 0
Um2 = 0
Um1 = 0
Unow = 0
Ym2 = 0
Ym1 = 0
Ynow = 0
AV = Ynow
Label15.Caption = "DV = " + Format(DV, "#0.00 Volt")
Label16.Caption = "Kp = " + Format(Kp, "#0.00")
Label17.Caption = "Ki = " + Format(Ki, "#0.00")
Label18.Caption = "Kd = " + Format(Kd, "#0.00")
'Mengatur garis pada shape
'Garis Horisontal
Line1.X1 = Shape1.Left
Line1.X2 = Shape1.Left + Shape1.Width
Line1.Y1 = Shape1.Top + (1 * (Shape1.Height / 8))
Line1.Y2 = Shape1.Top + (1 * (Shape1.Height / 8))
Line2.X1 = Shape1.Left
Line2.X2 = Shape1.Left + Shape1.Width
Line2.Y1 = Shape1.Top + (2 * (Shape1.Height / 8))
Line2.Y2 = Shape1.Top + (2 * (Shape1.Height / 8))
Line3.X1 = Shape1.Left
Line3.X2 = Shape1.Left + Shape1.Width
Line3.Y1 = Shape1.Top + (3 * (Shape1.Height / 8))
```

```
Line3.Y2 = Shape1.Top + (3 * (Shape1.Height / 8))
Line4.X1 = Shape1.Left
Line4.X2 = Shape1.Left + Shape1.Width
Line4.Y1 = Shape1.Top + (4 * (Shape1.Height / 8))
Line4.Y2 = Shape1.Top + (4 * (Shape1.Height / 8))
Line5.X1 = Shape1.Left
Line5.X2 = Shape1.Left + Shape1.Width
Line5.Y1 = Shape1.Top + (5 * (Shape1.Height / 8))
Line5.Y2 = Shape1.Top + (5 * (Shape1.Height / 8))
Line6.X1 = Shape1.Left
Line6.X2 = Shape1.Left + Shape1.Width
Line6.Y1 = Shape1.Top + (6 * (Shape1.Height / 8))
Line6.Y2 = Shape1.Top + (6 * (Shape1.Height / 8))
Line7.X1 = Shape1.Left
Line7.X2 = Shape1.Left + Shape1.Width
Line7.Y1 = Shape1.Top + (7 * (Shape1.Height / 8))
Line7.Y2 = Shape1.Top + (7 * (Shape1.Height / 8))

'Garis vertikal
Line8.X1 = Shape1.Left + (1 * (Shape1.Width / 16))
Line8.X2 = Shape1.Left + (1 * (Shape1.Width / 16))
Line8.Y1 = Shape1.Top + Shape1.Height
Line8.Y2 = Shape1.Top
Line9.X1 = Shape1.Left + (2 * (Shape1.Width / 16))
Line9.X2 = Shape1.Left + (2 * (Shape1.Width / 16))
Line9.Y1 = Shape1.Top + Shape1.Height
Line9.Y2 = Shape1.Top
Line10.X1 = Shape1.Left + (3 * (Shape1.Width / 16))
Line10.X2 = Shape1.Left + (3 * (Shape1.Width / 16))
Line10.Y1 = Shape1.Top + Shape1.Height
Line10.Y2 = Shape1.Top
Line11.X1 = Shape1.Left + (4 * (Shape1.Width / 16))
Line11.X2 = Shape1.Left + (4 * (Shape1.Width / 16))
Line11.Y1 = Shape1.Top + Shape1.Height
Line11.Y2 = Shape1.Top
Line12.X1 = Shape1.Left + (5 * (Shape1.Width / 16))
Line12.X2 = Shape1.Left + (5 * (Shape1.Width / 16))
Line12.Y1 = Shape1.Top + Shape1.Height
Line12.Y2 = Shape1.Top
Line13.X1 = Shape1.Left + (6 * (Shape1.Width / 16))
Line13.X2 = Shape1.Left + (6 * (Shape1.Width / 16))
Line13.Y1 = Shape1.Top + Shape1.Height
Line13.Y2 = Shape1.Top
Line14.X1 = Shape1.Left + (7 * (Shape1.Width / 16))
Line14.X2 = Shape1.Left + (7 * (Shape1.Width / 16))
Line14.Y1 = Shape1.Top + Shape1.Height
Line14.Y2 = Shape1.Top
```

```
Line15.X1 = Shape1.Left + (8 * (Shape1.Width / 16))
Line15.X2 = Shape1.Left + (8 * (Shape1.Width / 16))
Line15.Y1 = Shape1.Top + Shape1.Height
Line15.Y2 = Shape1.Top
Line16.X1 = Shape1.Left + (9 * (Shape1.Width / 16))
Line16.X2 = Shape1.Left + (9 * (Shape1.Width / 16))
Line16.Y1 = Shape1.Top + Shape1.Height
Line16.Y2 = Shape1.Top
Line17.X1 = Shape1.Left + (10 * (Shape1.Width / 16))
Line17.X2 = Shape1.Left + (10 * (Shape1.Width / 16))
Line17.Y1 = Shape1.Top + Shape1.Height
Line17.Y2 = Shape1.Top
Line18.X1 = Shape1.Left + (11 * (Shape1.Width / 16))
Line18.X2 = Shape1.Left + (11 * (Shape1.Width / 16))
Line18.Y1 = Shape1.Top + Shape1.Height
Line18.Y2 = Shape1.Top
Line19.X1 = Shape1.Left + (12 * (Shape1.Width / 16))
Line19.X2 = Shape1.Left + (12 * (Shape1.Width / 16))
Line19.Y1 = Shape1.Top + Shape1.Height
Line19.Y2 = Shape1.Top
Line20.X1 = Shape1.Left + (13 * (Shape1.Width / 16))
Line20.X2 = Shape1.Left + (13 * (Shape1.Width / 16))
Line20.Y1 = Shape1.Top + Shape1.Height
Line20.Y2 = Shape1.Top
Line21.X1 = Shape1.Left + (14 * (Shape1.Width / 16))
Line21.X2 = Shape1.Left + (14 * (Shape1.Width / 16))
Line21.Y1 = Shape1.Top + Shape1.Height
Line21.Y2 = Shape1.Top
Line22.X1 = Shape1.Left + (15 * (Shape1.Width / 16))
Line22.X2 = Shape1.Left + (15 * (Shape1.Width / 16))
Line22.Y1 = Shape1.Top + Shape1.Height
Line22.Y2 = Shape1.Top
Line23.X1 = Shape1.Left
Line23.X2 = Shape1.Left
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height

'Mengatur label skala sumbu y
Label1.Height = 250
Label2.Height = 250
Label3.Height = 250
Label4.Height = 250
Label5.Height = 250
Label6.Height = 250
Label7.Height = 250
Label8.Height = 250
Label9.Height = 250
```

```
Label1.Height = 250
Label1.Width = 615
Label2.Width = 615
Label3.Width = 615
Label4.Width = 615
Label5.Width = 615
Label6.Width = 615
Label7.Width = 615
Label8.Width = 615
Label9.Width = 615
Label1.Alignment = 1 ' rata kanan
Label2.Alignment = 1 ' rata kanan
Label3.Alignment = 1 ' rata kanan
Label4.Alignment = 1 ' rata kanan
Label5.Alignment = 1 ' rata kanan
Label6.Alignment = 1 ' rata kanan
Label7.Alignment = 1 ' rata kanan
Label8.Alignment = 1 ' rata kanan
Label9.Alignment = 1 ' rata kanan
Label1.Left = Shape1.Left - 700
Label2.Left = Shape1.Left - 700
Label3.Left = Shape1.Left - 700
Label4.Left = Shape1.Left - 700
Label5.Left = Shape1.Left - 700
Label6.Left = Shape1.Left - 700
Label7.Left = Shape1.Left - 700
Label8.Left = Shape1.Left - 700
Label9.Left = Shape1.Left - 700
Label1.Top = Shape1.Top + (0 * (Shape1.Height / 8)) -
125
Label2.Top = Shape1.Top + (1 * (Shape1.Height / 8)) -
125
Label3.Top = Shape1.Top + (2 * (Shape1.Height / 8)) -
125
Label4.Top = Shape1.Top + (3 * (Shape1.Height / 8)) -
125
Label5.Top = Shape1.Top + (4 * (Shape1.Height / 8)) -
125
Label6.Top = Shape1.Top + (5 * (Shape1.Height / 8)) -
125
Label7.Top = Shape1.Top + (6 * (Shape1.Height / 8)) -
125
Label8.Top = Shape1.Top + (7 * (Shape1.Height / 8)) -
125
Label9.Top = Shape1.Top + (8 * (Shape1.Height / 8)) -
125
End Sub
```

```
Private Sub HScroll1_Change()
DV = HScroll1.Value * (1 / 100)
Label15.Caption = "DV = " + Format(DV, "#0.00 Volt")
End Sub

Private Sub HScroll2_Change()
Kp = HScroll2.Value * (1 / 100)
Label16.Caption = "Kp = " + Format(Kp, "#0.00")
End Sub

Private Sub HScroll3_Change()
Ki = HScroll3.Value * (1 / 100)
Label17.Caption = "Ki = " + Format(Ki, "#0.00")
End Sub

Private Sub HScroll4_Change()
Kd = HScroll4.Value * (1 / 100)
Label18.Caption = "Kd = " + Format(Kd, "#0.00")
End Sub

Private Sub Timer1_Timer()
'PID Calculation
DV = HScroll1.Value * (1 / 100)
Kp = HScroll2.Value * (1 / 100)
Ki = HScroll3.Value * (1 / 100)
Kd = HScroll4.Value * (1 / 100)
e = DV - AV
POut = e * Kp
eio = e * Ki
IOut = IOutold + (ts * eio)
edo = e * Kd
du = edo - edoold
dt = ts
DOut = du / dt
PIDOut = POut + IOut + DOut
If Check2.Value = 1 Then
    If PIDOut > 5 Then PIDOut = 5
    If PIDOut < 0 Then PIDOut = 0
End If

'Plant parameter
a1 = Val(Text1.Text)
a2 = Val(Text2.Text)
b1 = Val(Text3.Text)
b2 = Val(Text4.Text)
```

```
If Check1.Value = 1 Then
    Unow = DV
    Ynow = (-1 * (a1 * Ym1)) - (a2 * Ym2) + (b1 * Um1)
+ (b2 * Um2)
Else
    Unow = PIDOut
    Ynow = (-1 * (a1 * Ym1)) - (a2 * Ym2) + (b1 * Um1)
+ (b2 * Um2)
End If

Label10.Caption = "Desired Value = " + Format(DV,
"#0.00 Volt")
Label11.Caption = "Avtual Value   = " + Format(AV,
"#0.00 Volt")
Label15.Caption = "DV = " + Format(DV, "#0.00 Volt")
Label16.Caption = "Kp = " + Format(Kp, "#0.00")
Label17.Caption = "Ki = " + Format(Ki, "#0.00")
Label18.Caption = "Kd = " + Format(Kd, "#0.00")
Label25.Caption = "DV = " + Format(DV, "#0.00")
Label24.Caption = "AV = " + Format(AV, "#0.00")
Label23.Caption = "e  = " + Format(e, "#0.00")
Label22.Caption = "OP = " + Format(POut, "#0.00")
Label26.Caption = "OI = " + Format(IOut, "#0.00")
Label27.Caption = "OD = " + Format(DOut, "#0.00")
Label28.Caption = "PID= " + Format(PIDOut, "#0.00")
Label29.Caption = "ts = " + Format(ts, "#0.00")
'Menampilkan hasil keluaran

'Menampilkan Grafik U
v = (Shape1.Height / 8)
h = ((Shape1.Width / 160))
UX2 = Shape1.Left + (i * h)
UY2 = Shape1.Top + Shape1.Height - (v * DV)
If UY2 < Shape1.Top Then UY2 = Shape1.Top
If  UY2  >  Shape1.Top  +  Shape1.Height  Then  UY2  =
Shape1.Top + Shape1.Height
Line (UX1, UY1)-(UX2, UY2), vbRed
'Menampilkan Grafik Y
YX2 = Shape1.Left + (i * h)
YY2 = Shape1.Top + Shape1.Height - (v * Ynow)
If YY2 < Shape1.Top Then YY2 = Shape1.Top
If  YY2  >  Shape1.Top  +  Shape1.Height  Then  YY2  =
Shape1.Top + Shape1.Height
Line23.X1 = YX2 + 60
Line23.X2 = YX2 + 60
Line23.Y1 = Shape1.Top
Line23.Y2 = Shape1.Top + Shape1.Height
```

```
Line (YX1, YY1)-(YX2, YY2), vbBlue

'Mengirimkan data berikutnya
UX1 = UX2
UY1 = UY2
YX1 = YX2
YY1 = YY2
eiold = eio
IOutold = IOut
edoold = edo
Ym2 = Ym1
Ym1 = Ynow
Um2 = Um1
Um1 = Unow
AV = Ynow
'Pembacaan data masukan
i = i + 1
If i >= 160 Then
    Form1.Cls
    i = 0
    UX1 = Shape1.Left
    UY1 = Shape1.Top + Shape1.Height
    YX1 = Shape1.Left
    YY1 = Shape1.Top + Shape1.Height
End If
End Sub
```

9.2. Peralatan Input Output

Personal Computer (PC) memiliki arsitektur yang terbuka sehingga memungkinkan penggunaan komputer menjadi sangat luas untuk berbagai keperluan dan pengolahan data baik data internal maupun data eksternal.

Pada masa lalu peralatan input output tersebut berupa input output card yang dipasang pada slot-slot ISA bus yang terdapat pada motherboard komputer. Pada masa sekarang produk komputer tidak lagi dilengkapi dengan slot-slot expansi ISA bus melainkan dengan slot-slot PCI yang jarak antar pin-pin sangat dekat sehingga tidak memungkinkan untuk membuat input output card sendiri dengan menyolder komponen dalam bentuk Surface Mountain device yang kecil menggunkaan tangan dan solder konvensional, melainkan harus meng-gunakan mesin solder SMD

Seiring dengan perkembangan teknologi, kecenderungan pemakaian komputer jenis Laptop semakin meningkat dibandingan dengan pemakaian personal computer (PC) dan peralatan input output card tidak mungkin lagi dipasang pada laptop. Oleh sebab itu memenuhi kebutuhan perlatan input output yang dapat dipergunakan pada komputer dan laptop maka peralatan input output harus berada diluar sistim komputer (external) dan model penyambungan datanya dapat menyesuaikan dengan komunikasi port yang tersedia pada kebanyakan komputer misalnya RC232 atau USB.

Kelebihan dari sistim peralatan input output external ini adalah kepraktisan pemakaiannya tidak perlu membuka cover computer untuk memasang card melainkan cukup dengan menghubung-kan peralatan input output ini ke port RS232 atau USB computer.

Gambar 9.24 Peralatan Input Output

Pada bagian ini, peralatan yang akan dihahas adalah tampak seperti pada Gambar 9.24 yang memiliki input output untuk sinyal digital maupun analog dan juga dilengkapi dengan power supply yang penjelasan lengkapnya adalah sebagai berikut :

Input Digital : PORTG, 8 saluran

Output Digital : PORTA, 8 saluran

Input Analog

Jumlah Channel : 8 Channel s
Resulusi ADC : 8 bits
Tegangan input : DC 0 s.d. + 5,10 Volt DC

Output Analog

Jumlah Channel	:	2 Channels
Resulusi ADC	:	8 bits
Tegangan output	:	-10 s.d. + 10 Volt DC
Arus maksimal	:	2 A

Output Power Supply

Tegangan Output	:	- 12 Volt DC Fixed, 2 A + 12 Volt DC Fixed, 2 A + 5 Volt DC Fixed, 2 A GND
Tegangan Line	:	220 Volt AC 50 Hz.
Komunikasi Data	:	RS232
Processor	:	MC68HC11F1

Parameter Komunikasi

Transmission rate	:	4800 baud
Character coding	:	8 bit ASCII
Parity	:	None
Stop Bits	:	1

9.3. Mengakses Port Serial

Pada umumnya suatu komputer (PC) menyediakan dua macam saluran serial yaitu Standard Communication Port RS232 (COM) dan Universal Serial Bus (USB)

9.3.1. Mengakses Standard Communication Port RS232

COM Port RS232 pada komputer biasanya berupa socket DB9 male seperti tampak pada Gambar 9.25

Gambar 9.25 COM Port RS232

Susunan pin adalah sebagai berikut

Tabel 9.1 Konfigurasi Pin RS-232

Nomor Pin	Signal	
1	DCD	Data Carrier Detect
2	RxD	Received Data
3	TxD	Transmitted Data
4	DTR	Data Terminal Ready
5	GND	Signal Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

Untuk mengakses COM Port, Visual Basic menyediakan komponen kontrol yaitu Microsoft Comm Control yang dapat ditambahkan sebagai komponen baru pada Toobox Standard dengan icon gambar telepon.

Kontrol MSComm menyediakan fasilitas komunikasi serial yang meliputi pengiriman dan penerimaan data melalui port serial dengan berbagai properti sebagai berikut :

CommPort

Dipergunakan untuk memilih COM, misalnya COM1, COM2. Nilai antara 1 sampai 16. CommPort harus sudah disetting sebelum membuka port.

Contoh pemakaian :

```
MSComm1.CommPort = 1
```

DTREnable

Menentukan apakah dimungkinkan jalur Data Terminal Ready (DTR) selama komunikasi. Sinyal DTR biasanya dikirimoleh komputer ke modem untuk menunjukkan bahwa komputer telah siap untuk menerima datangnya transmisi.

Contoh pemakaian :

```
MSComm1.DTREnable = True
```

EOFEnable

Menentukan apakah isyarat End Of File dimungkinkan

Contoh pemakaian :

```
MSComm1.EOFEnable = False
```

Handshaking

Handshaking mengacu pada protocol komunikasi internal, yang mana data ditransfer dari port perangkat keras ke buffer penerima. Manakala sustu karakter data tiba di port serial, alat komunikasi harus memindahkannya ke dalam buffer penerima sedemikian rupa segingga program dapat membaca data yang dating tersebut. Jika tidak ada buffer penerima dan program diharapkan untuk membaca setiap karakter yang dating secara langsung dari perangkat keras, mungkin data akan hilang disebabkan karakter dating dengan cepat. Protokol handshaking menjamin data tidak hilang selama buffer bekerja. Jika data dating pada port dengan cepat, maka alat komunikasi akan memindahkan data ke dalam buffer penerima.

Data dalam bilangan integer.
Pilihan untuk protocol handshaking adalah :

0 = comNone
(Default) No handshaking.

1 = comXOn/XOff
XON/XOFF handshaking.

2 = comRTS
RTS/CTS (Request To Send / Clear To Send) handshaking

3 = comRTSXOnXOff
Both Request To Send and XON/XOFF handshaking.

Contoh pemakaian :

```
MSComm1.Handshaking = False
```

InBufferSize

Menntukan besarnya kapasitas memeory buffer penerima.
Contoh pemakaian :

```
MSComm1.InBufferSize = 1024
```

InputLen

Membaca jumlah karakter yang masuk dalam buffer penerima
Contoh pemakaian :

```
Dim D1 As Integer
D1 = MSComm1.InputLen
```

InputMode

Menentukan mode masukan

0 = cominputModeText
1 = cominputModeBinary

Contoh pemakaian :
```
MSComm1.InputMode = 0
```

NullDiscard
Menentukan pengosongan buffer masukan
Contoh pemakaian :
```
MSComm1.NullDiscard = False
```

OutBufferSize
Menentukan besarnya kapasitas memory buffer pengirim
Contoh pemakaian :
```
MSComm1.OutBufferSize = 512
```

ParityReplace
Membaca apakah parity berubah

RThreshold
Menetapkan dan mngembalikan banyaknya karakter ke penerima sebelum kontrol MSCOmm menetapkan property CommEvent ke comEvReceive dab menghasilkan even OnComm.
Setting berupa bilangan integer yang menetapkan banyaknya karakter yang akan diterima sebelum menghasilkan event OnComm
Contoh pemakaian :
```
MSComm1.RThreshold = 0
```

RTSEnable
Menentukan apakah memungkinkan jalur Request To Send (RTS). Biasanya sinyal RTS yang minta ijin untuk untuk mengirim data dikirim ke komputer oleh modem.
Contoh pemakaian :
```
MSComm1.RTSEnable = False
```

Settings
Dipergunakan untuk mengatur parameter baud rate, parity, data bit dan stop bit dengan format string sebagai berikut :
“B,P,D,S”
yang mana :
B = baud rate
P = parity
D = jumlah data bit
S = jumlah stop bit
Contoh pemakaian :
```
MSComm1.Setting = “9600,N,8,1”
```

SThreshold
Setting berupa bilangan integer yang menetapkan banyaknya karakter yang akan dikirim sebelum menghasilkan event OnComm
Contoh pemakaian :
```
MSComm1.SThreshold = 0
```

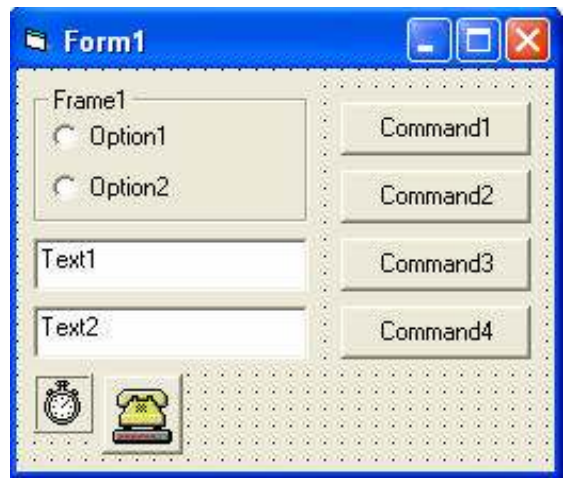
Input
Dipergunakan untuk membaca masukan yang diterima
Contoh pemakaian :
```
Dim D1 As String
D1 = MSComm1.Input
```

Output
Dipergunakan untuk mengirim data keluaran
Contoh pemakaian :
```
MSComm1.Output = "Hallo"
```
Adapun event yang efektif pada kontrol MSComm adalah

OnComm
Event ini terjadi ketika nilai property CommEvent berubah, yang mengindikasikan terjadinya event komunikasi atau terjadinya kesalahan.

Berikut ini adalah contoh program yang berfungsi untuk mengirim dan menerima data melalui port serial RS232.

Komponen yang diperlukan adalah 1 buah Form, 1 buah Frame, 2 buah Option, 2 buah TextBox, 4 buah CommandButton, 1 buah Timer, 1 buah MSComm



Gambar 9.26 Layout komponen pada Form1

Keterangan :

Option dan Option2 diubah captionnya menjadi COM1 dan COM2, dipergunakan untuk memilih COM Port.

Tombol Command1 diubah captionnya menjadi “Connect” untuk menyambungkan program ke COM Port yang dipilih, apabila COM Port yang dipilih tidak tersedia atau sudah dipakai oleh aplikasi lain maka akan muncul pesan bahwa “COM tidak bisa dipakai , ganti COM yang lain”.
Jika COM Port dapat dipakai maka tombol “Connect” ini tidak boleh ditekan lagi dan harus dimatikan karena perintah untuk membuka port yang sama hanya boleh dijalankan sekali. Selain itu selama tersambung ke COM, juga harus mematikan pilihan COM serta menghidupkan tombol “Disconnect” dan tombol “Send”
Jika koneksi ke COM tidak berhasil, maka Option1 dan Option2 harus enable kembali untuk mengganti pilihan COM dan tombol “Connect” harus hidup kembali dan mematikan tombol “Disconnect” dan tombol “Send”

Tombol caption CommandButton2 diganti dengan “Disconnect” dan berfungsi untuk menutup koneksi ke COM. Jika tombol ini ditekan, maka koneksi ke COM akan ditutup, mematikan tombol “Disconnect” dan tombol “Send” dan meng-enable-kan tombol “Connect” serta pilhan COM1 dan COM2

TextBox1 dipergunakan untuk memasukkan text yang akan dikirim ke COM. Jika sambungan ke COM sudah di-connect dan tombol “Send” enable, maka denmgan meng-klik tombol “Send” , text pada TextBox1 akan langsung dikirim.

TextBox2 dipergunakan untuk menampilkan text yang masuk diterima oleh COM. Jika ada text masuk pada buffer (MSComm1.InbufferCount>0) maka isi buffer diambil dan ditampilkan ke TextBox2.
Karena berfungsi sebagai penampil, maka TextBox2 harus tidak bias ditulisi. Untuk itu TextBox2 harus kunci (dilocked).

Pada program kirim terima text (RXTX) ini, MSComm1 diatur dengan baud rate 9400, Parity None, Data bit 8 dan Stop bit 1.

Tombol caption CommandButton4 diubah menjadi “Exit” yang berfungsi untuk keluar dari program ini.

Selengkapnya listing program ini ditampilkan sebagai berikut :

```
Dim d1 As String
```

```
Private Sub Command1_Click()
'Membuka port komunikasi
On Error Resume Next
MSComm1.PortOpen = True
If Err Then
   MsgBox "COM" + Str(MSComm1.CommPort) + " tidak bisa
   dipakai, ganti COM yang lain"
    Command1.Enabled = True
    Command2.Enabled = False
    Command3.Enabled = False
    Option1.Enabled = True
    Option2.Enabled = True
    Exit Sub
End If
Command1.Enabled = False
Command2.Enabled = True
Command3.Enabled = True
Option1.Enabled = False
Option2.Enabled = False
End Sub
```

```
Private Sub Command2_Click()
    MSComm1.PortOpen = False
    Command1.Enabled = True
    Command2.Enabled = False
    Command3.Enabled = False
    Option1.Enabled = True
    Option2.Enabled = True
End Sub
```

```
Private Sub Command3_Click()
MSComm1.Output = Text1.Text
End Sub
```

```
Private Sub Command4_Click()
End
End Sub
```

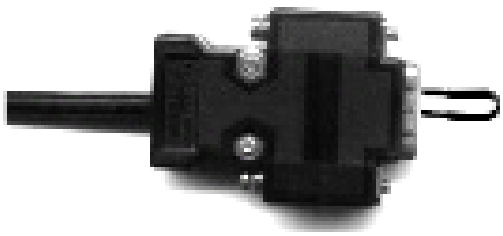
```
Private Sub Form_Load()
Form1.Caption = "RXTX"
Frame1.Caption = "COMPort"
Option1.Caption = "COM1"
```

```
Option2.Caption = "COM2"
Command1.Caption = "Connect"
Command2.Caption = "Disconnect"
Command3.Caption = "Send"
Command4.Caption = "Exit"
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"
Option1.Value = True
Text1.Text = ""
Text2.Text = ""
Text2.Locked = True
Text2.Appearance = 0
Command2.Enabled = False
Command3.Enabled = False
Timer1.Interval = 1
Timer1.Enabled = False
End Sub
```

```
Private Sub Option1_Click()
MSComm1.CommPort = 1
End Sub
```

```
Private Sub Option2_Click()
MSComm1.CommPort = 2
End Sub
```

```
Private Sub Timer1_Timer()
If MSComm1.InBufferCount > 0 Then
    d1 = MSComm1.Input
    Text2.Text = d1
End If
End Sub
```

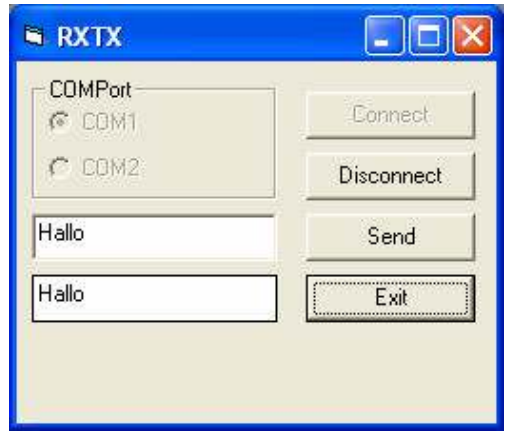


Gambar 9.27 Hubung Singkat Pin No. 2 (TxD) dan Pin No. 3 (RxD)

Untuk mencoba program ini, sambungkanlah kabel RS232 dengan stecker DB9. Kemudian hubung singkatlah pin nomor 2 (TxD) dan pin nomor 3 (RxD).

Jika Tombol “Send” ditekan maka text pada TextBox1 akan keluar dari pin Txt dan memalui kabel diterima pin RxD kemudian hasilnya ditampilkan pada TextBox2. Jika tidak ada gangguan maka text yang diterima harus sama seperti teks yang dikirim.
Selanjutnya, cob alepas kabel hubung singkat tersebut dan coba lakukan pengiriman teks yang lain. Hasilnya TextBox2 tidak akan muncul text baru.

Hasil jalannya program kirim terima teks RXTX ini tampak seperti pada Gambar 9.28 berikut ini.



Gambar 9.28 Hasil jalannya program

9.3.2. Menggunakan Port USB

Pada komputer atau laptop yang diproduksi atau keluaran terakhir sekarang ini biasanya hanya memiliki satu buah COM Port atau bahkan sama sekali tidak memiliki sambungan COM Port dan LPT. Untuk melayani komunikasi data hanya disediakan Universal Serial Bus (USB). Melalui USB inilah sambungan ke peralatan input output harus disambungkan
Untuk mengakses data serialdari peralatan input output yang bekerjanya menggunakan RS232 diperlukan adanya peralatan tambahan berupa kabel yang disebut USB to RS232 Converter.

Gambar 9.29 USB to RS232 Converter

Di pasaran telah tersedia berbagai macam USB to RS232 converter ini yang dapat disambungkan ke port USB pada komputer atau Laptop. Dan selanjutnya menginstal software driver pada Windows XP dengan cara sebagai berikut

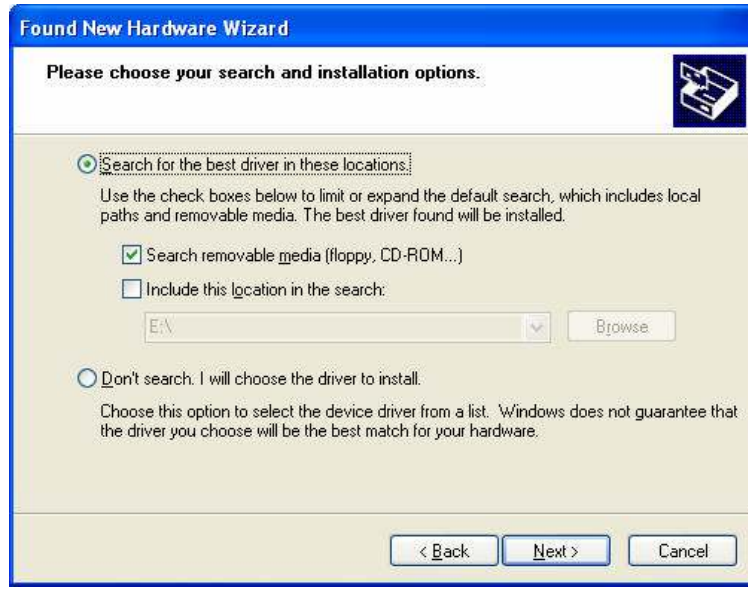
1. Pasang kabel USB to RS232 Converter pada salah satu USB port yang tersedia pada computer atau laptop. Tunggu sampai pesan seperti pada Gambar 1.30 muncul.

Gambar 9.30 Pesan Found New Hardware

2. Double-click pada pesan tersebut dan ikuti petunjuk berikut ini sampai finish

Gambar 9.31 Welcome to the Found New hardware Wizard

Pilihlah Install from a list or specific location (Advanced), kemudian tekan tombol **Next >**



Gambar 9.32 Please choose your search and installation options

Pilih seperti pada gambar di atas lalu tekan tombol **Next >**

Gambar 9.33 Please wait while the wizard searches….

Tunggu sampai tombol **Next >** enable dan berikutnya tekan tombol **Next >**

Gambar 9.34 Pesan peringatan

Apabila muncul **pesan seperti diatas, lanjutkan saja proses instalasi dengan menekan tombol Continue Anyway**

Gambar 9.35 Please wait while the wizard installs the software

Tunggu sampai tombol **Next >** enable dan berikutnya tekan tombol **Next >**

Setelah proses instalasi software selesai akan muncul pesan seperti gambar di bawah, senjutnya tekan tombol **Finish**

Gambar 9.36 Completing the Found New Hardware Wizard

Berikutnya akan muncul pesan bahwa hardware baru telah terinstall dan siap dipergunakan sebagai berikut

Gambar 9.37 Pesan new hardware telah terinstall dan siap dipergunakan

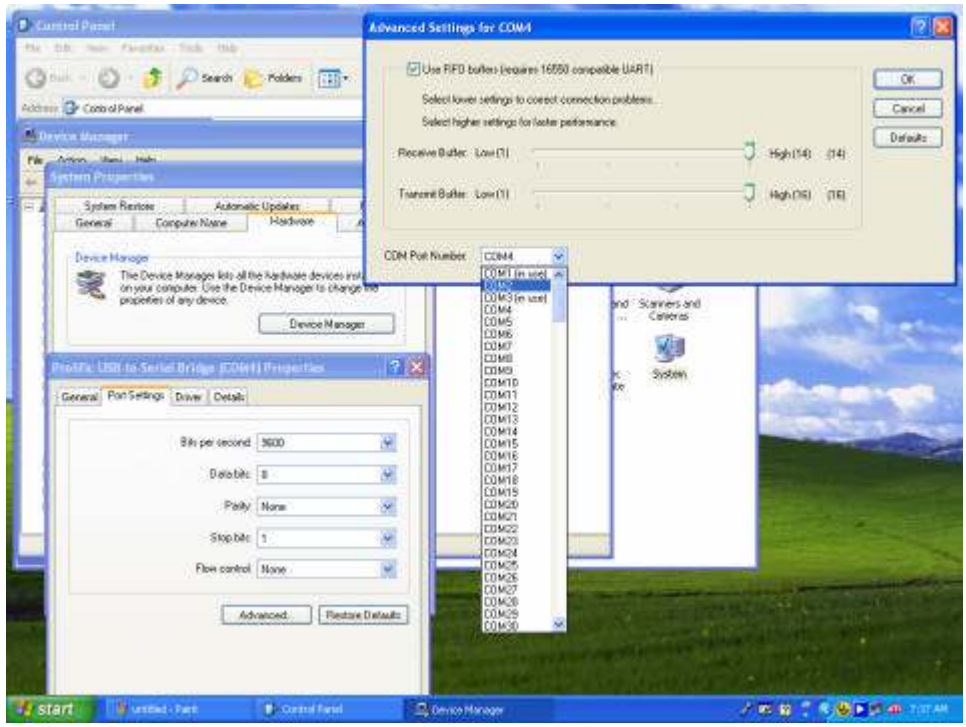
Berikutnya kita harus melihat pada COM berapa kabel USB to RS232 Converter kita tersambung. Ini penting untuk diketahui karena pada saat pembuatan software aplikasi untuk mengakses COM port harus diketahui berapa Port number-nya.
Lakukan langkah-langkah berikut ini untuk melihat Port number atau juga dapat dilakukan perubahan nomot port sesuai pilihan yang tersedia sepanjang nomor port tersebut tidak dipakai oleh aplikasi lain.

Click tombol **Start** pada Task bar Windows XP, pilih **Control Panel**, pilih System, pilih **Hardware**, pilih **Device Manager**, kemudian akan muncul informasi seperti pada Gambar 1.37

Gambar 9.38 Informasi hardware

Lihat pada bagian Port(COM & LPT), tampak informasi bahwa USB to RS232 telah tersambung pada COM4, **Prolific USB-toSerial Bridge(COM4)**

Untuk menubah Port number dapat dilakukan dengan cara double-click pada **Prolific USB-toSerial Bridge(COM4)** atau dengan cara lain click-right buttons mouse dan plih **Properties** dan akan muncul jendela baru **Prolific USB-toSerial Bridge(COM4) Properties,** selanjutnya pilih **Port Settings**, pilih **Advanced** maka akan muncul jendela baru **Advanced Settings for COM4** dan pilihlah COM Port Number dengan cara meng-click ComboBox maka akan muncul Port number yang tersedia dan mana saja yang udah terpakai. Pilihlah Port Number yang belum terpakai selanjutnya tekan tombol **OK** dan tutup semua jendela yang muncul.



Gambar 9.39 Jendela mengubah COM Port Number

9.4. Implementasi Pemrograman Untuk Aplikasi Kontrol Melalui Port Serial

Untuk mengimplementasikan berbagai aplikasi kontrol menggunakan komputer atau Laptop dengan software Visual Basic, kita memerlukan peralatan beserta kelengkapnnya sebagai berikut :

Analog-Digital I/O

Kabel input output digital dengan stecker DB25
Male – female

Kabel serial RS232 dengan stecker DB9
Male – female

Kabel converter USB to RS232 dan CD driver

Kabel input output analog dengan
BNC – Stecker banana 8 mm

Kabel jumper dengan stecker banana 8 mm

Kabel ke line
AC 220V

Input output test

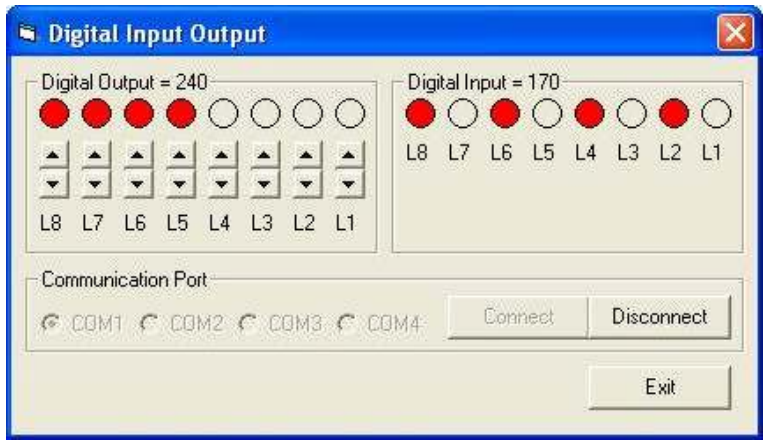
Gambar 9.40 Peralatan Analog-Digital I/O beserta kelengkapannya

9.4.1. Input Output Digital

Fungsi :
Membaca masukan digital dari deretan saklar PORTG dan menampilkan hasil pembacaan dengan visualisasi LED dan teks.
Membaca data deretan saklar pada Form dan Mengeluarkan data tersebut ke PORTA.

Peralatan :
Kabel USB to RS232 Converter, Kabel RS232, Kabel input output digital DB25 dan Modul Input output test

Gambar 9.41 Rangkaian Percobaan Input Output Digital



Gambar 9.42 Visualisasi Program Input Output Digital

Listing program :

```
Dim dout, din As String
Dim saklar, lampu As Double
Dim d0, d1, d2, d3, d4, d5, d6, d7 As Double
```

```
Private Sub Command1_Click()
End
End Sub
```

```
Private Sub Command2_Click()
On Error Resume Next
MSComm1.PortOpen = True
If Err Then
    MsgBox "COM" + Str(MSComm1.CommPort) + " tidak
bisa dipakai, ganti COM yang lain"
    MSComm1.PortOpen = False
    Command2.Enabled = True
    Command3.Enabled = False
    VScroll1.Enabled = False
    VScroll2.Enabled = False
    VScroll3.Enabled = False
    VScroll4.Enabled = False
    VScroll5.Enabled = False
    VScroll6.Enabled = False
    VScroll7.Enabled = False
    VScroll8.Enabled = False
    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
```

```
    Option4.Enabled = True
    Timer1.Enabled = False
    Exit Sub
End If
    MsgBox    "Device    connected    to    COM"    +
Str(MSComm1.CommPort)
    Timer1.Enabled = True
    Command2.Enabled = False
    Command3.Enabled = True
    VScroll1.Enabled = True
    VScroll2.Enabled = True
    VScroll3.Enabled = True
    VScroll4.Enabled = True
    VScroll5.Enabled = True
    VScroll6.Enabled = True
    VScroll7.Enabled = True
    VScroll8.Enabled = True
    Option1.Enabled = False
    Option2.Enabled = False
    Option3.Enabled = False
    Option4.Enabled = False
End Sub
```

```
Private Sub Command3_Click()
    MSComm1.PortOpen = False
    Command2.Enabled = True
    Command3.Enabled = False
    VScroll1.Enabled = False
    VScroll2.Enabled = False
    VScroll3.Enabled = False
    VScroll4.Enabled = False
    VScroll5.Enabled = False
    VScroll6.Enabled = False
    VScroll7.Enabled = False
    VScroll8.Enabled = False
    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
    Option4.Enabled = True
    Timer1.Enabled = False
End Sub
```

```
Private Sub Form_Load()
Frame1.Caption = "Digital Output"
Frame2.Caption = "Digital Input"
Form1.Caption = "Digital Input Output"
```

```
Command2.Enabled = True
Command3.Enabled = False
Option1.Value = True
MSComm1.CommPort = 1
MSComm1.Settings = "4800,n,8,1"
Timer1.Interval = 100
Timer1.Enabled = False
VScroll1.Max = 0
VScroll1.Min = 1
VScroll2.Max = 0
VScroll2.Min = 1
VScroll3.Max = 0
VScroll3.Min = 1
VScroll4.Max = 0
VScroll4.Min = 1
VScroll5.Max = 0
VScroll5.Min = 1
VScroll6.Max = 0
VScroll6.Min = 1
VScroll7.Max = 0
VScroll7.Min = 1
VScroll8.Max = 0
VScroll8.Min = 1
VScroll1.Enabled = False
VScroll2.Enabled = False
VScroll3.Enabled = False
VScroll4.Enabled = False
VScroll5.Enabled = False
VScroll6.Enabled = False
VScroll7.Enabled = False
VScroll8.Enabled = False
End Sub
```

```
Private Sub Option1_Click()
MSComm1.CommPort = 1
End Sub
```

```
Private Sub Option2_Click()
MSComm1.CommPort = 2
End Sub
```

```
Private Sub Option3_Click()
MSComm1.CommPort = 3
End Sub
```

```
Private Sub Option4_Click()
```

```
MSComm1.CommPort = 4
End Sub

Private Sub Timer1_Timer()
'Membaca saklar dan mengirimkan datanya ke RS232
d0 = VScroll1.Value * 1
d1 = VScroll2.Value * 2
d2 = VScroll3.Value * 4
d3 = VScroll4.Value * 8
d4 = VScroll5.Value * 16
d5 = VScroll6.Value * 32
d6 = VScroll7.Value * 64
d7 = VScroll8.Value * 128
saklar = d0 + d1 + d2 + d3 + d4 + d5 + d6 + d7
If (saklar And 1) = 1 Then Shape11.FillColor = &HFF&
Else Shape11.FillColor = &H8000000F
If (saklar And 2) = 2 Then Shape12.FillColor = &HFF&
Else Shape12.FillColor = &H8000000F
If (saklar And 4) = 4 Then Shape13.FillColor = &HFF&
Else Shape13.FillColor = &H8000000F
If (saklar And 8) = 8 Then Shape14.FillColor = &HFF&
Else Shape14.FillColor = &H8000000F
If (saklar And 16) = 16 Then Shape15.FillColor = &HFF&
Else Shape15.FillColor = &H8000000F
If (saklar And 32) = 32 Then Shape16.FillColor = &HFF&
Else Shape16.FillColor = &H8000000F
If (saklar And 64) = 64 Then Shape17.FillColor = &HFF&
Else Shape17.FillColor = &H8000000F
If (saklar And 128) = 128 Then Shape18.FillColor =
&HFF& Else Shape18.FillColor = &H8000000F
Frame1.Caption = "Digital Output = " & Format(saklar,
"000")
dout = "G" + Format(saklar, "000")
MSComm1.Output = dout
'Membaca status lampu dan memvisualkannya
din = MSComm1.Input
Frame2.Caption = "Digital Input = " & Mid(din, 2, 3)
If Mid(din, 1, 1) = "g" Then
    lampu = Val(Mid(din, 2, 3))
    If (lampu And 1) = 1 Then
        Shape1.FillColor = &HFF&
        lamp = lamp Or &H1
    Else
        Shape1.FillColor = &H8000000F
        lamp = lamp And &HFE
    End If
    If (lampu And 2) = 2 Then
```

```
    Shape2.FillColor = &HFF&
    lamp = lamp Or &H2
Else
    Shape2.FillColor = &H8000000F
    lamp = lamp And &HFD
End If
If (lampu And 4) = 4 Then
    Shape3.FillColor = &HFF&
    lamp = lamp Or &H4
Else
    Shape3.FillColor = &H8000000F
    lamp = lamp And &HFB
End If
If (lampu And 8) = 8 Then
    Shape4.FillColor = &HFF&
    lamp = lamp Or &H8
Else
    Shape4.FillColor = &H8000000F
    lamp = lamp And &HF7
End If
If (lampu And 16) = 16 Then
    Shape5.FillColor = &HFF&
    lamp = lamp Or &H10
Else
    Shape5.FillColor = &H8000000F
    lamp = lamp And &HEF
End If
If (lampu And 32) = 32 Then
    Shape6.FillColor = &HFF&
    lamp = lamp Or &H20
Else
    Shape6.FillColor = &H8000000F
    lamp = lamp And &HDF
End If
If (lampu And 64) = 64 Then
    Shape7.FillColor = &HFF&
    lamp = lamp Or &H40
Else
    Shape7.FillColor = &H8000000F
    lamp = lamp And &HBF
End If
If (lampu And 128) = 128 Then
    Shape8.FillColor = &HFF&
    lamp = lamp Or &H80
Else
    Shape8.FillColor = &H8000000F
    lamp = lamp And &H7F
```

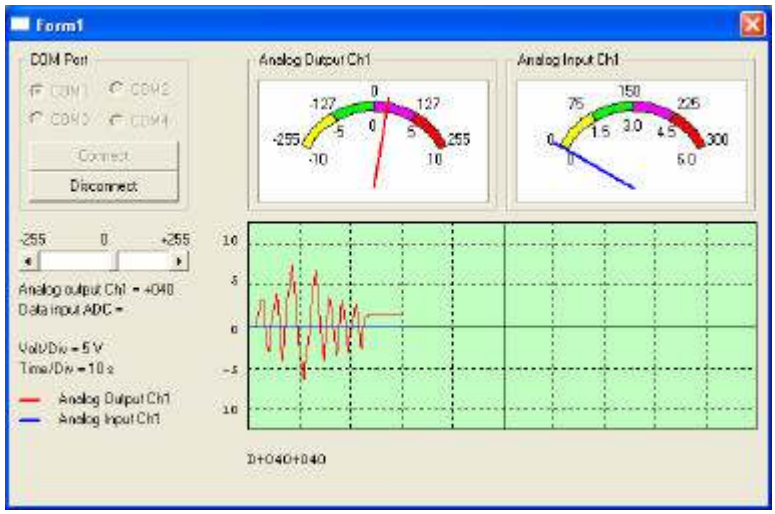
```
    End If
End If
End Sub
```

9.4.2. Input Output Analog

Fungsi :
Membaca masukan tegangan analog mulai dari 0 Volt sampai 5.10 Volt DC dari sumber tegangan DC variable dan menampilkan hasil pembacaan dengan visualisasi Voltmeter analog dan teks serta grafik. Tegangan analog masuk memalui input analog Channel1.
Mengeluarkan tegangan analog mulai dari – 10 Volt sampai dengan + 10 Volt ke output analog Channel 1 yang terhubung ke Voltmeter Digital. Pengaturan tegangan output dilakukan dengan menggeser-geser Horzontal Scrollbar pada Form.

Peralatan :
Kabel USB to RS232 Converter, Kabel RS232, Kabel input output analog dengan BNC – Stecker banana 8 mm, Sumber tegangan DC variable 0 s.d. 5.10 Volt dan Multimeter digital

Gambar 9.43 Rangkaian Percobaan Input Output Analog



Gambar 9.44 Visualisasi Program Input Output Analog

Listing program :

```
Dim i, gain, Xa, Xb, dx, X2, UX1, UX2, UY1, UY2, YX1,
YX2, YY1, YY2 As Integer
Dim x, d1, d2 As String
Dim Unow, Ynow, din, garisnull As Double
Dim sd, sr, A, B, R As Double
Dim adc1i, dac1i, dac2i As Integer
Dim adc1s, dac1s, dac2s, dacout As String
```

```
Private Sub Command1_Click()
MSComm1.PortOpen = True
Timer1.Enabled = True
HScroll1.Enabled = True
Command1.Enabled = False
Command2.Enabled = True
Option1.Enabled = False
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
dout = HScroll1.Value
MSComm1.Output = dacout          'mengirimkan data DAC
MSComm1.Output = "A0"            'membaca data ADC
End Sub
```

```
Private Sub Command2_Click()
MSComm1.PortOpen = False
Timer1.Enabled = False
```

```
HScroll1.Enabled = False
Command1.Enabled = True
Command2.Enabled = False
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True
End Sub
```

```
Private Sub Form_Load()
'Settings MSComm
MSComm1.CommPort = 1
MSComm1.Settings = "4800,n,8,1"
Option1.Value = True

'Tampilan jarum meter1
dac1i = 0
dac2i = 0
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
If dac1i < 0 Then dac1s = "-" & Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = "-" & Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
Label34.Caption = dacout
Line17.BorderWidth = 2
Line18.BorderColor = vbRed
sd = (dac1i * ((150 - 30) / 510)) + 90
If sd > 180 Then sd = 180
If sd < 0 Then sd = 0
sr = Sin(sd / 57.3)
R = 1100
A = sr * R
B = Sqr((R * R) - (A * A))
If sd <= 90 Then Line17.X1 = (1000 - B) + 400
If ((sd > 90) And (sd <= 180)) Then Line17.X1 = 1400 +
B
Line17.Y1 = (1000 - A) + 400
Line17.X2 = 1400
Line17.Y2 = 1400

'Tampilan jarum meter2
adc1i = 0
Line18.BorderWidth = 2
Line18.BorderColor = vbBlue
sd = (adc1 * ((150 - 30) / 300)) + 30
If sd > 180 Then sd = 180
```

```
If sd < 0 Then sd = 0
sr = Sin(sd / 57.3)
R = 1100
A = sr * R
B = Sqr((R * R) - (A * A))
If sd <= 90 Then Line18.X1 = (1000 - B) + 400
If ((sd > 90) And (sd <= 180)) Then Line18.X1 = 1400 +
B
Line18.Y1 = (1000 - A) + 400
Line18.X2 = 1400
Line18.Y2 = 1400

'Inisial value
HScroll1.Max = 255
HScroll1.Min = -255
HScroll1.Value = 0
Timer1.Interval = 100
Timer1.Enabled = False
HScroll1.Enabled = False
Label1.Caption = "Data output DAC = " + Format(dac1i,
"000")
Label23.Caption = "Data input ADC = " + Format(din,
"000")
Command2.Enabled = False

'Pengaturan tampilan grafik
Label2.Caption = "Volt/Div = 5 V"
Label3.Caption = "Time/Div = 10 s"
Form1.Cls
gain = 1000
Xa = Shape1.Left
Xb = Shape1.Left + Shape1.Width
garisnull = Shape1.Top + (Shape1.Height / 2)
UX1 = Xa
UY1 = garisnull
YX1 = UX1
YY1 = UY1
dx = (Xb - Xa) / 1000
i = 0
Shape1.Left = UX1
Shape1.Width = Xb - Xa
Shape1.Height = 2.5 * gain
Shape1.Top = UY1 - (Shape1.Height / 2)
sold = 0
snow = 0
f = 0
Line1.X1 = Shape1.Left
```

```
Line1.Y1 = Shape1.Top + (Shape1.Height / 2)
Line1.X2 = Shape1.Left + Shape1.Width
Line1.Y2 = Shape1.Top + (Shape1.Height / 2)
Line2.X1 = Shape1.Left
Line2.Y1 = Shape1.Top + 0.25 * gain
Line2.X2 = Shape1.Left + Shape1.Width
Line2.Y2 = Shape1.Top + 0.25 * gain
Line3.X1 = Shape1.Left
Line3.Y1 = Shape1.Top + 0.75 * gain
Line3.X2 = Shape1.Left + Shape1.Width
Line3.Y2 = Shape1.Top + 0.75 * gain
Line4.X1 = Shape1.Left
Line4.Y1 = Shape1.Top + 1.75 * gain
Line4.X2 = Shape1.Left + Shape1.Width
Line4.Y2 = Shape1.Top + 1.75 * gain
Line5.X1 = Shape1.Left
Line5.Y1 = Shape1.Top + 2.25 * gain
Line5.X2 = Shape1.Left + Shape1.Width
Line5.Y2 = Shape1.Top + 2.25 * gain
Line6.X1 = Shape1.Left + (1 * (Shape1.Width / 10))
Line6.Y1 = Shape1.Top
Line6.X2 = Shape1.Left + (1 * (Shape1.Width / 10))
Line6.Y2 = Shape1.Top + Shape1.Height
Line7.X1 = Shape1.Left + (2 * (Shape1.Width / 10))
Line7.Y1 = Shape1.Top
Line7.X2 = Shape1.Left + (2 * (Shape1.Width / 10))
Line7.Y2 = Shape1.Top + Shape1.Height
Line8.X1 = Shape1.Left + (3 * (Shape1.Width / 10))
Line8.Y1 = Shape1.Top
Line8.X2 = Shape1.Left + (3 * (Shape1.Width / 10))
Line8.Y2 = Shape1.Top + Shape1.Height
Line9.X1 = Shape1.Left + (4 * (Shape1.Width / 10))
Line9.Y1 = Shape1.Top
Line9.X2 = Shape1.Left + (4 * (Shape1.Width / 10))
Line9.Y2 = Shape1.Top + Shape1.Height
Line10.X1 = Shape1.Left + (5 * (Shape1.Width / 10))
Line10.Y1 = Shape1.Top
Line10.X2 = Shape1.Left + (5 * (Shape1.Width / 10))
Line10.Y2 = Shape1.Top + Shape1.Height
Line11.X1 = Shape1.Left + (6 * (Shape1.Width / 10))
Line11.Y1 = Shape1.Top
Line11.X2 = Shape1.Left + (6 * (Shape1.Width / 10))
Line11.Y2 = Shape1.Top + Shape1.Height
Line12.X1 = Shape1.Left + (7 * (Shape1.Width / 10))
Line12.Y1 = Shape1.Top
Line12.X2 = Shape1.Left + (7 * (Shape1.Width / 10))
Line12.Y2 = Shape1.Top + Shape1.Height
```

```
Line13.X1 = Shape1.Left + (8 * (Shape1.Width / 10))
Line13.Y1 = Shape1.Top
Line13.X2 = Shape1.Left + (8 * (Shape1.Width / 10))
Line13.Y2 = Shape1.Top + Shape1.Height
Line14.X1 = Shape1.Left + (9 * (Shape1.Width / 10))
Line14.Y1 = Shape1.Top
Line14.X2 = Shape1.Left + (9 * (Shape1.Width / 10))
Line14.Y2 = Shape1.Top + Shape1.Height
End Sub

Private Sub Option1_Click()
MSComm1.CommPort = 1
End Sub
```

```
Private Sub Option2_Click()
MSComm1.CommPort = 2
End Sub
```

```
Private Sub Option3_Click()
MSComm1.CommPort = 3
End Sub
```

```
Private Sub Option4_Click()
MSComm1.CommPort = 4
End Sub
```

```
Private Sub Timer1_Timer()
'Mengeluarkan data ke DAC
dac1i = HScroll1.Value
dac2i = HScroll1.Value
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
If dac1i < 0 Then dac1s = Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
Label34.Caption = dacout
MSComm1.Output = dacout          'mengirimkan data DAC
MSComm1.Output = "A0"            'membaca data ADC
Label1.Caption = "Analog output Ch1 = " + dac1s

'Membaca data masukan ADC
If MSComm1.InBufferCount > 0 Then
    d1 = MSComm1.Input
    d2 = Mid(d1, 2, 3)
    adc1i = Val(d2)
    Label23.Caption = "Analog input Ch1 = " + d2
```

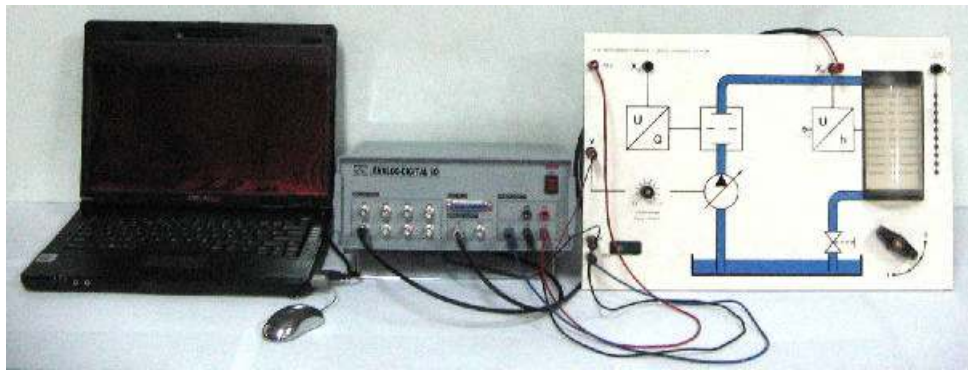
```
End If

'Tampilan jarum meter1
Line17.BorderWidth = 2
Line17.BorderColor = vbRed
sd = (dac1i * ((150 - 30) / 510)) + 90
If sd > 180 Then sd = 180
If sd < 0 Then sd = 0
sr = Sin(sd / 57.3)
R = 1100
A = sr * R
B = Sqr((R * R) - (A * A))
If sd <= 90 Then Line17.X1 = (1000 - B) + 400
If ((sd > 90) And (sd <= 180)) Then Line17.X1 = 1400 +
B
Line17.Y1 = (1000 - A) + 400
Line17.X2 = 1400
Line17.Y2 = 1400

'Tampilan jarum meter2
Line18.BorderWidth = 2
Line18.BorderColor = vbBlue
sd = (adc1i * ((150 - 30) / 300)) + 30
If sd > 180 Then sd = 180
If sd < 0 Then sd = 0
sr = Sin(sd / 57.3)
R = 1100
A = sr * R
B = Sqr((R * R) - (A * A))
If sd <= 90 Then Line18.X1 = (1000 - B) + 400
If ((sd > 90) And (sd <= 180)) Then Line18.X1 = 1400 +
B
Line18.Y1 = (1000 - A) + 400
Line18.X2 = 1400
Line18.Y2 = 1400

'Menggambar 2 grafik dalam satu layar dengan data
masukan Unow dan Ynow
Unow = dac1i / 255
Ynow = adc1i / 510
UX2 = Xa + (i * dx)
UY2 = garisnull - (Unow * gain)
If UY2 < Shape1.Top Then UY2 = Shape1.Top
Line (UX1, UY1)-(UX2, UY2), vbRed
YX2 = Xa + (i * dx)
YY2 = garisnull - (Ynow * gain)
If YY2 < Shape1.Top Then YY2 = Shape1.Top
```

```
If YY2 > Shape1.Top + Shape1.Height Then YY2 =
Shape1.Top + Shape1.Height
Line (YX1, YY1)-(YX2, YY2), vbBlue
UX1 = UX2
UY1 = UY2
YX1 = YX2
YY1 = YY2
Uold = Unow
Yold = Ynow
sold = snow
i = i + 1
If i > 1000 Then
   i = 0
   Form1.Cls
   UX1 = Xa
   UY1 = garisnull
   YX1 = UX1
   YY1 = UY1
End If
End Sub
```

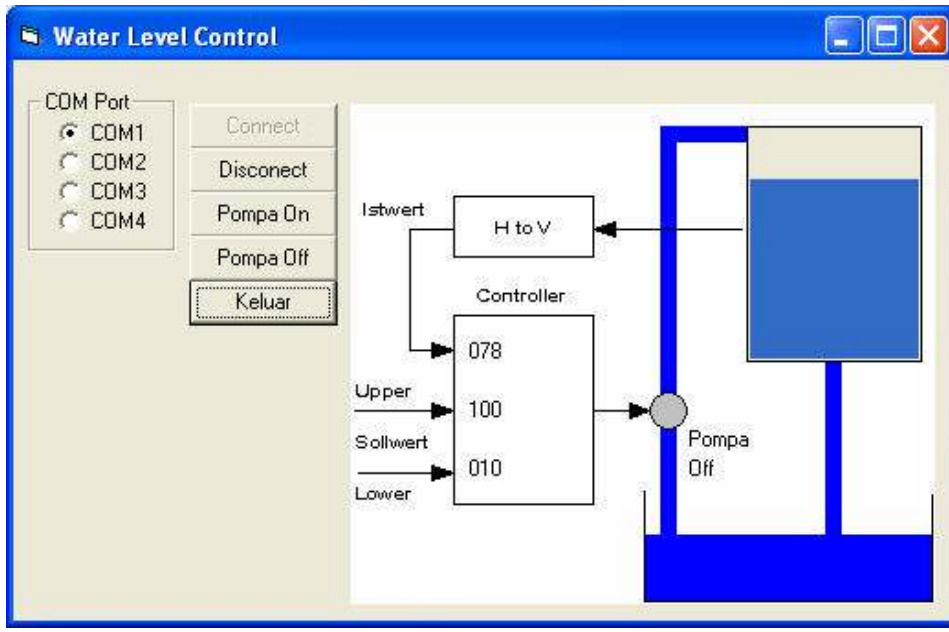
9.4.3. Water Level Control

Fungsi :
Mengatur level air dalam tangki. Masukan berupa sensor level analog yang dipasang pada tangki dengan tegangan keluaran sensor sebesar 0 volt sampai 5.10 Volt yang mewakili level air mulai 0 % sampai dengan 100 %. Ouput level sensor ini terhubung ke Channel2.
Output Analog Chanel1 mengeluarkan tegangan +10 Volt yang diberikan ke pompa air DC untuk memompa air dari sumber ke dalam tangki.
Mati hidupnya pompa air diatur secara oto matis, yaitu apabila level air mencapai batas atas maka pompa mati dan bila mencapai batas bawah pompa akan hidup. Batasatas diset = 100 % dan batas bawah diset = 10 %.
Pada panel disediakan pula tombol On/Off manual jika dikehendaki pengaturan pompa secara manual.
Visualisasi berupa sistim pengaturan level air seperti pada trainer.

Peralatan :
Kabel USB to RS232 Converter, Kabel RS232, Kabel input output analog BNC – banan 8 mm, Kabel jumper banan 8 mm dan Modul Water Level Control



Gambar 9.47 Rangkaian Percobaan Water Level Control



Gambar 9.48 Visualisasi Program Water Level Control

Listing program :

```
Dim comport As Integer
Dim d1, d2, d3, d4, d5, dout As String
Dim d2d, upper, lower As Double
```

```
Private Sub Command1_Click()
If MSComm1.PortOpen = False Then MSComm1.PortOpen =
True
dout = "D+000+000"
Timer1.Enabled = True
```

```
Command1.Enabled = False
Command2.Enabled = True
Command4.Enabled = True
Command5.Enabled = True
MSComm1.Output = "A4"
MSComm1.Output = dout
End Sub
```

```
Private Sub Command2_Click()
If MSComm1.PortOpen = True Then
    dout = "D+000+000"
    MSComm1.Output = dout
    MSComm1.PortOpen = False
    Timer1.Enabled = False
    Command1.Enabled = True
    Command2.Enabled = False
    Command4.Enabled = False
    Command5.Enabled = False
End If
End Sub
```

```
Private Sub Command3_Click()
If MSComm1.PortOpen = True Then MSComm1.PortOpen =
False
End
End Sub
```

```
Private Sub Command4_Click()
If d2d >= 100 Then
    dout = "D+000+000"
    MsgBox "Pompa dimatikan, Tangki sudah penuh"
Else
    Timer1.Enabled = False
    dout = "D+255+000"
    MSComm1.Output = dout
    Shape1.FillColor = &HFF&
    Label8.Caption = "On"
    Timer1.Enabled = True
End If
End Sub
```

```
Private Sub Command5_Click()
Timer1.Enabled = False
dout = "D+000+000"
MSComm1.Output = dout
Shape1.FillColor = &HC0C0C0
```

```
Label8.Caption = "Off"
Timer1.Enabled = True
End Sub
```

```
Private Sub Form_Load()
MSComm1.Settings = "4800,n,8,1"
Form1.Caption = "Water Level Control"
Command2.Enabled = False
Command4.Enabled = False
Command5.Enabled = False
Timer1.Interval = 100
Timer1.Enabled = False
MSComm1.CommPort = 1
Label5.Caption = ""
upper = 100
lower = 10
Label1.Caption = Format(upper, "000")
Label2.Caption = Format(lower, "000")
End Sub
```

```
Private Sub Option1_Click(Index As Integer)
If MSComm1.PortOpen = True Then MSComm1.PortOpen =
False
MSComm1.CommPort = 1
End Sub
```

```
Private Sub Option2_Click(Index As Integer)
If MSComm1.PortOpen = True Then MSComm1.PortOpen =
False
MSComm1.CommPort = 2
End Sub
```

```
Private Sub Option3_Click(Index As Integer)
If MSComm1.PortOpen = True Then MSComm1.PortOpen =
False
MSComm1.CommPort = 3
End Sub
```

```
Private Sub Option4_Click(Index As Integer)
If MSComm1.PortOpen = True Then MSComm1.PortOpen =
False
MSComm1.CommPort = 4
End Sub
```

```
Private Sub Timer1_Timer()
If MSComm1.InBufferCount > 0 Then
```

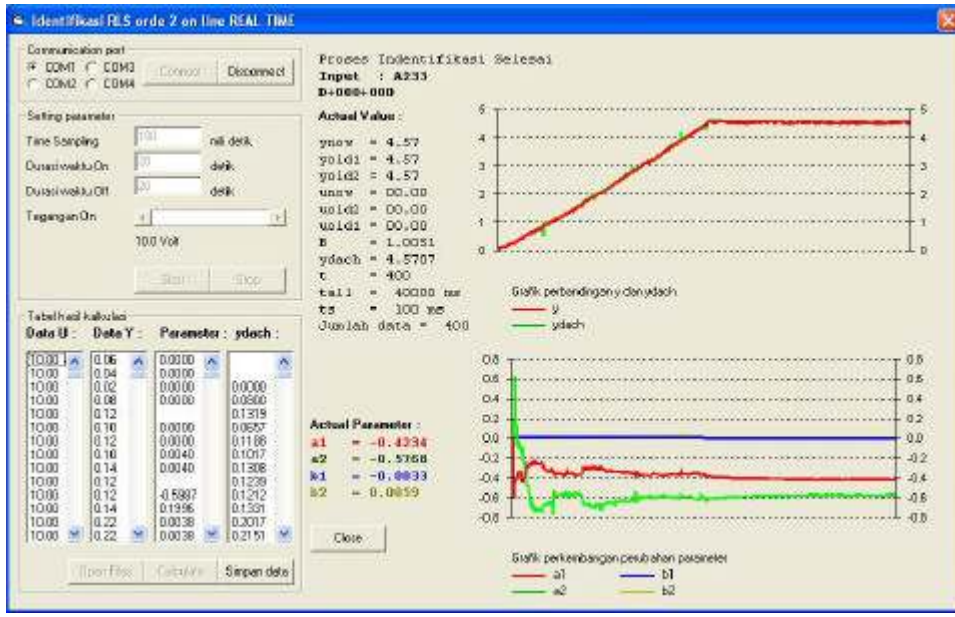
```
    d1 = MSComm1.Input
    d2 = Mid(d1, 2, 3)
    d2d = Val(d2) * (100 / 255)
    ProgressBar1.Value = d2d
    Label5.Caption = Format(d2d, "000")
End If
If d2d >= upper Then
    dout = "D+000+000"
    MSComm1.Output = dout
    Shape1.FillColor = &HC0C0C0
    Label8.Caption = "Off"
End If
If d2d <= lower Then
    dout = "D+255+000"
    MSComm1.Output = dout
    Shape1.FillColor = &HFF&
    Label8.Caption = "On"
End If
MSComm1.Output = "A4"
End Sub
```

9.4.4. Idenfikasi Plant

Fungsi :
Mengidentifikasi suatu plant dengan membaca data masukan analog dari Channel1 dan mngeluarkan tegangan kontrol ke output DAC Channel1 dengan variasi tegangan –10 Volt sampai dengan +10 Volt.
Pengaturan waktu On/Off serta teganganinput step dapat diatur.
Hasil identifikasi disimpan dalam suatu file dan ditampilkan dalam bentuk grafik dan teks

Peralatan :
Kabel USB to RS232 Converter, Kabel RS232, Kabel input output analog BNC – banan 8 mm, Kabel jumper banan 8 mm dan Plant (Water Level Control atau Kontrol Kecepatan Motor DC)

Gambar 9.49 Rangkaian Percobaan Identifikasi



Gambar 9.50 Visualisasi Program Identifikasi

Listing program :

```
Dim thetadach, a1, a2, b1, b2 As Single
Dim I, I11, I12, I13, I14, I21, I22, I23, I24, I31,
I32, I33, I34, I41, I42, I43, I44 As Single
Dim P, P11, P12, P13, P14, P21, P22, P23, P24, P31,
P32, P33, P34, P41, P42, P43, P44 As Single
Dim H, H11, H12, H13, H14, H21, H22, H23, H24, H31,
H32, H33, H34, H41, H42, H43, H44 As Single
```

```
Dim C, C11, C12, C13, C14, C21, C22, C23, C24, C31,
C32, C33, C34, C41, C42, C43, C44 As Single
Dim D, D11, D12, D13, D14, D21, D22, D23, D24, D31,
D32, D33, D34, D41, D42, D43, D44 As Single
Dim psi, ynow, yold1, yold2, unow, uold1, uold2 As
Single
Dim A, A11, A21, A31, A41 As Single
Dim B As Single
Dim gamma, gamma11, gamma21, gamma31, gama41 As Single
Dim ydach As Single
Dim error As Single
Dim hit, r, U, Y, dout As Double
Dim d1, d2, d3 As String
Dim ts, ton, toff, tall As Double
Dim ibc, FileNo, Counter, k, q As Integer
Dim Ps, GetValues() As String
Dim dac1i, dac2i As Double
Dim dacout As String
```

```
Private Sub Command1_Click()
If MSComm1.PortOpen = True Then
    MSComm1.Output = "P000"
    MSComm1.PortOpen = False
End If
End
End Sub
```

```
Private Sub Command2_Click()
ts = Val(Text1.Text)
'interval ts dalam milidetik
If ts < 100 Then
    MsgBox "Time sampling minimal 100 mili detik"
    Exit Sub
End If
ton = Val(Text2.Text)
If ton < 4 Then
    MsgBox "Lama on minimal 4 detik"
    Exit Sub
End If
toff = Val(Text3.Text)
If toff < 4 Then
    MsgBox "Lama off minimal 4 detik"
    Exit Sub
End If
hit = 0
a1 = 0
```

```
a2 = 0
b1 = 0
b2 = 0
List1.Clear
List2.Clear
List3.Clear
List4.Clear
List3.AddItem Format(a1, "###0.0000")
List3.AddItem Format(a2, "###0.0000")
List3.AddItem Format(b1, "###0.0000")
List3.AddItem Format(b2, "###0.0000")
List3.AddItem ""
List4.AddItem ""
List4.AddItem ""

tall = (ton * 1000) + (toff * 1000)
'tall dalam mili detik
Timer1.Interval = ts
Command2.Enabled = False
Command3.Enabled = True
Command7.Enabled = False
Command8.Enabled = False
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
HScroll1.Enabled = False
Label9.Caption  =  "Tunggu  sedang  persiapan  proses
identifikasi ... "
dac1i = HScroll1.Value
dac2i = 0
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
If dac1i < 0 Then dac1s = Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
dout = dac1i * (10 / 255)
Label41.Caption = Format(dout, "00.0") & " Volt"
MSComm1.Output = dacout
MSComm1.Output = "A0"
Label36.Caption = "Output : " + dacout
Label37.Caption = "tall  = " + Str(tall) + " ms"
Label38.Caption = "ts    = " + Str(ts) + " ms"
Label39.Caption = "Jumlah data = " + Str((tall / ts))
Timer1.Enabled = True
End Sub
```

```
Private Sub Command3_Click()
Label9.Caption = ""
dac1i = 0
dac2i = 0
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
If dac1i < 0 Then dac1s = Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
dout = dac1i * (10 / 255)
MSComm1.Output = dacout
Label36.Caption = dacout
Command3.Enabled = False
Command2.Enabled = True
Command7.Enabled = True
Command8.Enabled = True
Timer1.Enabled = False
Text1.Enabled = True
Text2.Enabled = True
Text3.Enabled = True
HScroll1.Enabled = True
End Sub
```

```
Private Sub Command4_Click()
Command2.Enabled = True
Command5.Enabled = True
Command4.Enabled = False
MSComm1.PortOpen = True
End Sub
```

```
Private Sub Command5_Click()
Timer1.Enabled = False
MSComm1.Output = "D+000+000"
MSComm1.PortOpen = False
Command5.Enabled = False
Command4.Enabled = True
End Sub
```

```
Private Sub Command6_Click()
CommonDialog1.FileName = "DataU.m"
CommonDialog1.Filter = "*.m"
CommonDialog1.DialogTitle = "Menyimpan Data U"
CommonDialog1.ShowSave
FileNo = FreeFile
On Error Resume Next
Open CommonDialog1.FileName For Output As FileNo
```

```
If Err Then
   Exit Sub
End If
Open CommonDialog1.FileName For Output As FileNo
Close FileNo
Open CommonDialog1.FileName For Append As FileNo
    For k = 0 To List1.ListCount
        Print #FileNo, List1.List(k)
    Next k
Close FileNo
'*********************************
CommonDialog1.FileName = "DataY.m"
CommonDialog1.Filter = "*.m"
CommonDialog1.DialogTitle = "Menyimpan Data Y"
CommonDialog1.ShowSave
FileNo = FreeFile
On Error Resume Next
Open CommonDialog1.FileName For Output As FileNo
If Err Then
   Exit Sub
End If
Open CommonDialog1.FileName For Output As FileNo
Close FileNo
Open CommonDialog1.FileName For Append As FileNo
    For k = 0 To List2.ListCount
        Print #FileNo, List2.List(k)
    Next k
Close FileNo
'*********************************
CommonDialog1.FileName = "Theta.m"
CommonDialog1.Filter = "*.m"
CommonDialog1.DialogTitle   =   "Menyimpan   Parameter
Theta"
CommonDialog1.ShowSave
FileNo = FreeFile
On Error Resume Next
Open CommonDialog1.FileName For Output As FileNo
If Err Then
   Exit Sub
End If
Open CommonDialog1.FileName For Output As FileNo
Close FileNo
Open CommonDialog1.FileName For Append As FileNo
        Print #FileNo, Str(a1)
        Print #FileNo, Str(a2)
        Print #FileNo, Str(b1)
        Print #FileNo, Str(b2)
```

```
Close FileNo
End Sub
```

```
Private Sub Command7_Click()
a1 = 0
a2 = 0
b1 = 0
b2 = 0
List3.Clear
List4.Clear
List3.AddItem Format(a1, "###0.0000")
List3.AddItem Format(a2, "###0.0000")
List3.AddItem Format(b1, "###0.0000")
List3.AddItem Format(b2, "###0.0000")
List3.AddItem ""
List4.AddItem ""
List4.AddItem ""
'Membuka file
CommonDialog1.DialogTitle = "Open Data input file *.m"
CommonDialog1.DefaultExt = "*.m"
CommonDialog1.FileName = "*.m"
CommonDialog1.Filter = "*.m"
CommonDialog1.ShowOpen
FileNo = FreeFile
Counter = 0
On Error Resume Next
Open CommonDialog1.FileName For Input As FileNo
If Err Then
    Exit Sub
End If
'Mengambil data
Do Until EOF(FileNo)
    Counter = Counter + 1
    ReDim Preserve GetValues(Counter)
    Line Input #FileNo, GetValues(Counter)
Loop
Close FileNo
'Memasukkan data ke List1
List1.Clear
For k = 1 To (Counter - 1)
    List1.AddItem GetValues(k)
Next k
'Membuka file
CommonDialog1.DialogTitle = "Open Data output file
*.m"
CommonDialog1.DefaultExt = "*.m"
```

```
CommonDialog1.FileName = "*.m"
CommonDialog1.Filter = "*.m"
CommonDialog1.ShowOpen
FileNo = FreeFile
Counter = 0
On Error Resume Next
Open CommonDialog1.FileName For Input As FileNo
If Err Then
    Exit Sub
End If
'Mengambil data
Do Until EOF(FileNo)
    Counter = Counter + 1
    ReDim Preserve GetValues(Counter)
    Line Input #FileNo, GetValues(Counter)
Loop
Close FileNo
'Memasukkan data ke List1
List2.Clear
For k = 1 To (Counter - 1)
    List2.AddItem GetValues(k)
Next k
Label39.Caption    =    "Jumlah    data    =    "    +
Str(List1.ListCount)
End Sub
```

```
Private Sub Command8_Click()
MSChart1.Enabled = False
MSChart2.Enabled = False
'monitor
Label1.Caption = "yold1 = "
Label2.Caption = "yold2 = "
Label3.Caption = "ynow  = "
Label4.Caption = "uold1 = "
Label5.Caption = "uold2 = "
Label6.Caption = "unow  = "
Label7.Caption = "B     = "
Label8.Caption = "ydach = "
Label9.Caption = " "
Label10.Caption = "a1   = "
Label11.Caption = "a2   = "
Label12.Caption = "b1   = "
Label13.Caption = "b2   = "
Label19.Caption = "t     = "
Label20.Caption = "Input"
Label36.Caption = "Output"
```

```
Label14.Caption = "Data U :"
Label15.Caption = "Data Y :"
Label16.Caption = "Parameter :"
Label18.Caption = "ydach :"
Label21.Caption = "Actual Value :"
Label22.Caption = "Actual Parameter :"
MSChart1.chartType = 3
MSChart2.chartType = 3
MSChart1.ColumnCount = 1
MSChart2.ColumnCount = 1
MSChart1.RowCount = 1
MSChart2.RowCount = 1

'inisial value
a1 = 0
a2 = 0
b1 = 0
b2 = 0
'menampilkan parameter thetadach
List3.Clear
List4.Clear
List3.AddItem Format(a1, "###0.0000")
List3.AddItem Format(a2, "###0.0000")
List3.AddItem Format(b1, "###0.0000")
List3.AddItem Format(b2, "###0.0000")
List3.AddItem ""
List4.AddItem ""
List4.AddItem ""

alfa = 100000
I11 = 1
I12 = 0
I13 = 0
I14 = 0
I21 = 0
I22 = 1
I23 = 0
I24 = 0
I31 = 0
I32 = 0
I33 = 1
I34 = 0
I41 = 0
I42 = 0
I43 = 0
I44 = 1
P11 = I11 * alfa
```

```
P12 = I12 * alfa
P13 = I13 * alfa
P14 = I14 * alfa
P21 = I21 * alfa
P22 = I22 * alfa
P23 = I23 * alfa
P24 = I24 * alfa
P31 = I31 * alfa
P32 = I32 * alfa
P33 = I33 * alfa
P34 = I34 * alfa
P41 = I41 * alfa
P42 = I42 * alfa
P43 = I43 * alfa
P44 = I44 * alfa
I = 2
r = 0
hit = 0
Label39.Caption    =    "Jumlah    data    =    "    +
Str(List1.ListCount)
'================================

For I = 0 To List1.ListCount
If I > 3 Then
    yold1 = Val(List2.List(I - 2))
    yold2 = Val(List2.List(I - 3))
    ynow = Val(List2.List(I - 1))
    uold1 = Val(List1.List(I - 1))
    uold2 = Val(List1.List(I - 2))
    unow = Val(List1.List(I - 1))
    'menghitung A
    A11 = (P11 *  -yold1) + (P12 *  -yold2) + (P13 *
   uold1) + (P14 * uold2)
    A21 = (P21 *  -yold1) + (P22 *  -yold2) + (P23 *
   uold1) + (P24 * uold2)
    A31 = (P31 *  -yold1) + (P32 *  -yold2) + (P33 *
   uold1) + (P34 * uold2)
    A41 = (P41 *  -yold1) + (P42 *  -yold2) + (P43 *
   uold1) + (P44 * uold2)
    'menghitung B
    B = (-yold1 *  A11) + (-yold2 * A21) + (uold1 *
   A31) + (uold2 * A41) + 1
    'menghitung gamma
    gamma11 = A11 / B
    gamma21 = A21 / B
    gamma31 = A31 / B
    gamma41 = A41 / B
```

```
 'menghitung ydach
 ydach = (-yold1 * a1) + (-yold2 * a2) + (uold1 *
b1) + (uold2 * b2)
 error = ynow - ydach
 'menghitung thetadach
 a1 = a1 + (gamma11 * error)
 a2 = a2 + (gamma21 * error)
 b1 = b1 + (gamma31 * error)
 b2 = b2 + (gamma41 * error)
 'menampilkan parameter thetadach
 List3.AddItem Format(a1, "###0.0000")
 List3.AddItem Format(a2, "###0.0000")
 List3.AddItem Format(b1, "###0.0000")
 List3.AddItem Format(b2, "###0.0000")
 List3.AddItem ""
 List4.AddItem Format(ydach, "###0.0000")
 'monitor
 Label1.Caption = "yold1 = " + List2.List(I - 2)
 Label2.Caption = "yold2 = " + List2.List(I - 3)
 Label3.Caption = "ynow  = " + List2.List(I - 1)
 Label4.Caption = "uold1 = " + List1.List(I - 2)
 Label5.Caption = "uold2 = " + List1.List(I - 3)
 Label6.Caption = "unow  = " + List1.List(I - 1)
 Label7.Caption  =  "B          =  "  +  Format(B,
"###0.0000")
 Label8.Caption  =  "ydach  =  "  +  Format(ydach,
"###0.0000")
  Label9.Caption = "Tunggu sedang melakukan proses
identifikasi ... "
 Label10.Caption  =  "a1       =  "  +  Format(a1,
"###0.0000")
 Label11.Caption  =  "a2       =  "  +  Format(a2,
"###0.0000")
 Label12.Caption  =  "b1       =  "  +  Format(b1,
"###0.0000")
 Label13.Caption  =  "b2       =  "  +  Format(b2,
"###0.0000")
 'menghitung P
 C11 = gamma11 * -yold1
 C12 = gamma11 * -yold2
 C13 = gamma11 * uold1
 C14 = gamma11 * uold2
 C21 = gamma21 * -yold1
 C22 = gamma21 * -yold2
 C23 = gamma21 * uold1
 C24 = gamma21 * uold2
 C31 = gamma31 * -yold1
```

```
 C32 = gamma31 * -yold2
 C33 = gamma31 * uold1
 C34 = gamma31 * uold2
 C41 = gamma41 * -yold1
 C42 = gamma41 * -yold2
 C43 = gamma41 * uold1
 C44 = gamma41 * uold2
 D11 = I11 - C11
 D12 = I12 - C12
 D13 = I13 - C13
 D14 = I14 - C14
 D21 = I21 - C21
 D22 = I22 - C22
 D23 = I23 - C23
 D24 = I24 - C24
 D31 = I31 - C31
 D32 = I32 - C32
 D33 = I33 - C33
 D34 = I34 - C34
 D41 = I41 - C41
 D42 = I42 - C42
 D43 = I43 - C43
 D44 = I44 - C44
  H11 = (D11 * P11) + (D12 * P21) + (D13 * P31) +
(D14 * P41)
  H12 = (D11 * P12) + (D12 * P22) + (D13 * P32) +
(D14 * P42)
  H13 = (D11 * P13) + (D12 * P23) + (D13 * P33) +
(D14 * P43)
  H14 = (D11 * P14) + (D12 * P24) + (D13 * P34) +
(D14 * P44)
  H21 = (D21 * P11) + (D22 * P21) + (D23 * P31) +
(D24 * P41)
  H22 = (D21 * P12) + (D22 * P22) + (D23 * P32) +
(D24 * P42)
  H23 = (D21 * P13) + (D22 * P23) + (D23 * P33) +
(D24 * P43)
  H24 = (D21 * P14) + (D22 * P24) + (D23 * P34) +
(D24 * P44)
  H31 = (D31 * P11) + (D32 * P21) + (D33 * P31) +
(D34 * P41)
  H32 = (D31 * P12) + (D32 * P22) + (D33 * P32) +
(D34 * P42)
  H33 = (D31 * P13) + (D32 * P23) + (D33 * P33) +
(D34 * P43)
  H34 = (D31 * P14) + (D32 * P24) + (D33 * P34) +
(D34 * P44)
```

```
     H41 = (D41 * P11) + (D42 * P21) + (D43 * P31) +
    (D44 * P41)
     H42 = (D41 * P12) + (D42 * P22) + (D43 * P32) +
    (D44 * P42)
     H43 = (D41 * P13) + (D42 * P23) + (D43 * P33) +
    (D44 * P43)
     H44 = (D41 * P14) + (D42 * P24) + (D43 * P34) +
    (D44 * P44)
     P11 = H11
     P12 = H12
     P13 = H13
     P14 = H14
     P21 = H21
     P22 = H22
     P23 = H23
     P24 = H24
     P31 = H31
     P32 = H32
     P33 = H33
     P34 = H34
     P41 = H41
     P42 = H42
     P43 = H43
     P44 = H44
End If
Next I
'grafik theta
If List3.ListCount / 5 < 5 Then
    MsgBox "Data tidak cukup untuk dikalkulasi"
    Exit Sub
End If
MSChart1.RowCount = List3.ListCount / 5
MSChart1.ColumnCount = 4
For r = 1 To List3.ListCount / 5
    'MSChart1.RowLabel = "t" + Str(r)
    MSChart1.Row = r
    MSChart1.Column = 1
    MSChart1.Data = List3.List((r * 5) - 5)
    MSChart1.Row = r
    MSChart1.Column = 2
    MSChart1.Data = List3.List((r * 5) - 4)
    MSChart1.Row = r
    MSChart1.Column = 3
    MSChart1.Data = List3.List((r * 5) - 3)
    MSChart1.Row = r
    MSChart1.Column = 4
    MSChart1.Data = List3.List((r * 5) - 2)
```

```
Next r
'grafik perbandingan y dan ydach
MSChart2.RowCount = (List3.ListCount / 5) - 3
MSChart2.ColumnCount = 2
For r = 1 To ((List3.ListCount / 5) - 3)
    MSChart2.Row = r
    'MSChart2.RowLabel = "t" + Str(r)
    MSChart2.Column = 1
    MSChart2.Data = List2.List(r + 1)
    MSChart2.Row = r
    MSChart2.Column = 2
    MSChart2.Data = List4.List(r + 1)
Next r
Label9.Caption = "Proses Indentifikasi Selesai"
End Sub
```

```
Private Sub Form_Load()
MSChart1.Enabled = False
MSChart2.Enabled = False
'monitor
Label1.Caption = "yold1 = "
Label2.Caption = "yold2 = "
Label3.Caption = "ynow  = "
Label4.Caption = "uold1 = "
Label5.Caption = "uold2 = "
Label6.Caption = "unow  = "
Label7.Caption = "B     = "
Label8.Caption = "ydach = "
Label9.Caption = " "
Label10.Caption = "a1   = "
Label11.Caption = "a2   = "
Label12.Caption = "b1   = "
Label13.Caption = "b2   = "
Label19.Caption = "t     = "
Label20.Caption = "Input"
Label36.Caption = "Output"
Command2.Enabled = False
Command3.Enabled = False
Command5.Enabled = False
Command6.Enabled = False
HScroll1.Max = 255
HScroll1.Min = -255
HScroll1.Value = 0
dac1i = HScroll1.Value
dac2i = 0
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
```

```
If dac1i < 0 Then dac1s = Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
dout = dac1i * (10 / 255)
Label41.Caption = Format(dout, "00.0") & " Volt"
MSComm1.CommPort = 1
hit = 0
Frame1.Caption = "Communication port"
Command4.Caption = "Connect"
Command5.Caption = "Disconnect"

Form1.Caption = "Identifikasi RLS orde 2 on line REAL
TIME"
Label14.Caption = "Data U :"
Label15.Caption = "Data Y :"
Label16.Caption = "Parameter :"
Label18.Caption = "ydach :"
Label21.Caption = "Actual Value :"
Label22.Caption = "Actual Parameter :"
Command1.Caption = "Close"
Command2.Caption = "Start"
Command3.Caption = "Stop"
Timer1.Enabled = False
MSChart1.chartType = 3
MSChart2.chartType = 3
MSChart1.ColumnCount = 1
MSChart2.ColumnCount = 1
MSChart1.RowCount = 1
MSChart2.RowCount = 1

'inisial value
a1 = 0
a2 = 0
b1 = 0
b2 = 0
'menampilkan parameter thetadach
List3.AddItem Format(a1, "###0.0000")
List3.AddItem Format(a2, "###0.0000")
List3.AddItem Format(b1, "###0.0000")
List3.AddItem Format(b2, "###0.0000")
List3.AddItem ""
List4.AddItem ""
List4.AddItem ""

alfa = 100000
I11 = 1
```

```
I12 = 0
I13 = 0
I14 = 0
I21 = 0
I22 = 1
I23 = 0
I24 = 0
I31 = 0
I32 = 0
I33 = 1
I34 = 0
I41 = 0
I42 = 0
I43 = 0
I44 = 1
P11 = I11 * alfa
P12 = I12 * alfa
P13 = I13 * alfa
P14 = I14 * alfa
P21 = I21 * alfa
P22 = I22 * alfa
P23 = I23 * alfa
P24 = I24 * alfa
P31 = I31 * alfa
P32 = I32 * alfa
P33 = I33 * alfa
P34 = I34 * alfa
P41 = I41 * alfa
P42 = I42 * alfa
P43 = I43 * alfa
P44 = I44 * alfa
I = 2
r = 0
End Sub
```

```
Private Sub HScroll1_Change()
dac1i = HScroll1.Value
dac2i = 0
If dac1i >= 0 Then dac1s = "+" & Format(dac1i, "000")
If dac1i < 0 Then dac1s = Format(dac1i, "000")
If dac2i >= 0 Then dac2s = "+" & Format(dac2i, "000")
If dac2i < 0 Then dac2s = Format(dac2i, "000")
dacout = "D" & dac1s & dac2s
dout = dac1i * (10 / 255)
Label41.Caption = Format(dout, "00.0") & " Volt"
End Sub
```

```
Private Sub Option1_Click(Index As Integer)
MSComm1.CommPort = 1
End Sub
```

```
Private Sub Option2_Click(Index As Integer)
MSComm1.CommPort = 2
End Sub
```

```
Private Sub Option3_Click(Index As Integer)
MSComm1.CommPort = 3
End Sub
```

```
Private Sub Option4_Click(Index As Integer)
MSComm1.CommPort = 4
End Sub
```

```
Private Sub Timer1_Timer()
Label19.Caption = "t     =" + Str(hit)
If hit <= (ton * 1000) / ts Then
    dac1i = HScroll1.Value
    dac2i = 0
    If dac1i >= 0 Then dac1s = "+" & Format(dac1i,
   "000")
    If dac1i < 0 Then dac1s = Format(dac1i, "000")
    If dac2i >= 0 Then dac2s = "+" & Format(dac2i,
   "000")
    If dac2i < 0 Then dac2s = Format(dac2i, "000")
    dacout = "D" & dac1s & dac2s
    dout = dac1i * (10 / 255)
    U = dout
    MSComm1.Output = dacout
    MSComm1.Output = "A0"
    Label36.Caption = dacout
    List1.AddItem Format(dout, "00.00")
Else
    dac1i = 0
    dac2i = 0
    If dac1i >= 0 Then dac1s = "+" & Format(dac1i,
   "000")
    If dac1i < 0 Then dac1s = Format(dac1i, "000")
    If dac2i >= 0 Then dac2s = "+" & Format(dac2i,
   "000")
    If dac2i < 0 Then dac2s = Format(dac2i, "000")
    dacout = "D" & dac1s & dac2s
    dout = dac1i * (10 / 255)
```

```
    MSComm1.Output = dacout
    MSComm1.Output = "A0"
    Label36.Caption = dacout
    List1.AddItem Format(dout, "00.00")
End If
```

```
'membaca data masukan dari plant
d1 = MSComm1.Input
Label20.Caption = "Input  : " + d1
d2 = Mid(d1, 1, 1)
If d2 = "A" Then
    d3 = Mid(d1, 2, 3)
    Y = Val(d3) * (5 / 255)
    List2.AddItem Format(Y, "#0.00")
Else
    d3 = "000"
    Y = Val(d3) * (5 / 255)
    List2.AddItem Format(Y, "#0.00")
End If
'================================
I = hit
If I > 3 Then
    yold1 = Val(List2.List(I - 2))
    yold2 = Val(List2.List(I - 3))
    ynow = Val(List2.List(I - 1))
    uold1 = Val(List1.List(I - 1))
    uold2 = Val(List1.List(I - 2))
    unow = Val(List1.List(I - 1))
    'menghitung A
     A11 = (P11 *  -yold1) + (P12 *  -yold2) + (P13 *
   uold1) + (P14 * uold2)
     A21 = (P21 *  -yold1) + (P22 *  -yold2) + (P23 *
   uold1) + (P24 * uold2)
     A31 = (P31 *  -yold1) + (P32 *  -yold2) + (P33 *
   uold1) + (P34 * uold2)
     A41 = (P41 *  -yold1) + (P42 *  -yold2) + (P43 *
   uold1) + (P44 * uold2)
    'menghitung B
     B = (-yold1 *  A11) + (-yold2 * A21) + (uold1 *
   A31) + (uold2 * A41) + 1
    'menghitung gamma
    gamma11 = A11 / B
    gamma21 = A21 / B
    gamma31 = A31 / B
    gamma41 = A41 / B
    'menghitung ydach
```

```
 ydach = (-yold1 * a1) + (-yold2 * a2) + (uold1 *
b1) + (uold2 * b2)
 error = ynow - ydach
 'menghitung thetadach
 a1 = a1 + (gamma11 * error)
 a2 = a2 + (gamma21 * error)
 b1 = b1 + (gamma31 * error)
 b2 = b2 + (gamma41 * error)
 'menampilkan parameter thetadach
 List3.AddItem Format(a1, "###0.0000")
 List3.AddItem Format(a2, "###0.0000")
 List3.AddItem Format(b1, "###0.0000")
 List3.AddItem Format(b2, "###0.0000")
 List3.AddItem ""
 List4.AddItem Format(ydach, "###0.0000")
 'monitor
 Label1.Caption = "yold1 = " + List2.List(I - 2)
 Label2.Caption = "yold2 = " + List2.List(I - 3)
 Label3.Caption = "ynow  = " + List2.List(I - 1)
 Label4.Caption = "uold1 = " + List1.List(I - 2)
 Label5.Caption = "uold2 = " + List1.List(I - 3)
 Label6.Caption = "unow  = " + List1.List(I - 1)
 Label7.Caption  =  "B          =  "  +  Format(B,
"###0.0000")
 Label8.Caption  =  "ydach  =  "  +  Format(ydach,
"###0.0000")
 Label9.Caption = "Tunggu sedang melakukan proses
identifikasi ... "
 Label10.Caption  =  "a1      =  "  +  Format(a1,
"###0.0000")
 Label11.Caption  =  "a2      =  "  +  Format(a2,
"###0.0000")
 Label12.Caption  =  "b1      =  "  +  Format(b1,
"###0.0000")
 Label13.Caption  =  "b2      =  "  +  Format(b2,
"###0.0000")
 'menghitung P
 C11 = gamma11 * -yold1
 C12 = gamma11 * -yold2
 C13 = gamma11 * uold1
 C14 = gamma11 * uold2
 C21 = gamma21 * -yold1
 C22 = gamma21 * -yold2
 C23 = gamma21 * uold1
 C24 = gamma21 * uold2
 C31 = gamma31 * -yold1
 C32 = gamma31 * -yold2
```

```
 C33 = gamma31 * uold1
 C34 = gamma31 * uold2
 C41 = gamma41 * -yold1
 C42 = gamma41 * -yold2
 C43 = gamma41 * uold1
 C44 = gamma41 * uold2
 D11 = I11 - C11
 D12 = I12 - C12
 D13 = I13 - C13
 D14 = I14 - C14
 D21 = I21 - C21
 D22 = I22 - C22
 D23 = I23 - C23
 D24 = I24 - C24
 D31 = I31 - C31
 D32 = I32 - C32
 D33 = I33 - C33
 D34 = I34 - C34
 D41 = I41 - C41
 D42 = I42 - C42
 D43 = I43 - C43
 D44 = I44 - C44
  H11 = (D11 * P11) + (D12 * P21) + (D13 * P31) +
(D14 * P41)
  H12 = (D11 * P12) + (D12 * P22) + (D13 * P32) +
(D14 * P42)
  H13 = (D11 * P13) + (D12 * P23) + (D13 * P33) +
(D14 * P43)
  H14 = (D11 * P14) + (D12 * P24) + (D13 * P34) +
(D14 * P44)
  H21 = (D21 * P11) + (D22 * P21) + (D23 * P31) +
(D24 * P41)
  H22 = (D21 * P12) + (D22 * P22) + (D23 * P32) +
(D24 * P42)
  H23 = (D21 * P13) + (D22 * P23) + (D23 * P33) +
(D24 * P43)
  H24 = (D21 * P14) + (D22 * P24) + (D23 * P34) +
(D24 * P44)
  H31 = (D31 * P11) + (D32 * P21) + (D33 * P31) +
(D34 * P41)
  H32 = (D31 * P12) + (D32 * P22) + (D33 * P32) +
(D34 * P42)
  H33 = (D31 * P13) + (D32 * P23) + (D33 * P33) +
(D34 * P43)
  H34 = (D31 * P14) + (D32 * P24) + (D33 * P34) +
(D34 * P44)
```

```
    H41 = (D41 * P11) + (D42 * P21) + (D43 * P31) +
   (D44 * P41)
    H42 = (D41 * P12) + (D42 * P22) + (D43 * P32) +
   (D44 * P42)
    H43 = (D41 * P13) + (D42 * P23) + (D43 * P33) +
   (D44 * P43)
    H44 = (D41 * P14) + (D42 * P24) + (D43 * P34) +
   (D44 * P44)
    P11 = H11
    P12 = H12
    P13 = H13
    P14 = H14
    P21 = H21
    P22 = H22
    P23 = H23
    P24 = H24
    P31 = H31
    P32 = H32
    P33 = H33
    P34 = H34
    P41 = H41
    P42 = H42
    P43 = H43
    P44 = H44
End If

If hit >= Int(((tall) / ts)) Then
    Timer1.Enabled = False
    'grafik theta
    'Label17.Caption  =  "Jumlah  data  theta  :  "  +
   Str(List3.ListCount / 5)
    MSChart1.RowCount = List3.ListCount / 5
    MSChart1.ColumnCount = 4
    For r = 1 To List3.ListCount / 5
        MSChart1.RowLabel = "t" + Str(r)
        MSChart1.Row = r
        MSChart1.Column = 1
        MSChart1.Data = List3.List((r * 5) - 5)
        MSChart1.Row = r
        MSChart1.Column = 2
        MSChart1.Data = List3.List((r * 5) - 4)
        MSChart1.Row = r
        MSChart1.Column = 3
        MSChart1.Data = List3.List((r * 5) - 3)
        MSChart1.Row = r
        MSChart1.Column = 4
        MSChart1.Data = List3.List((r * 5) - 2)
```

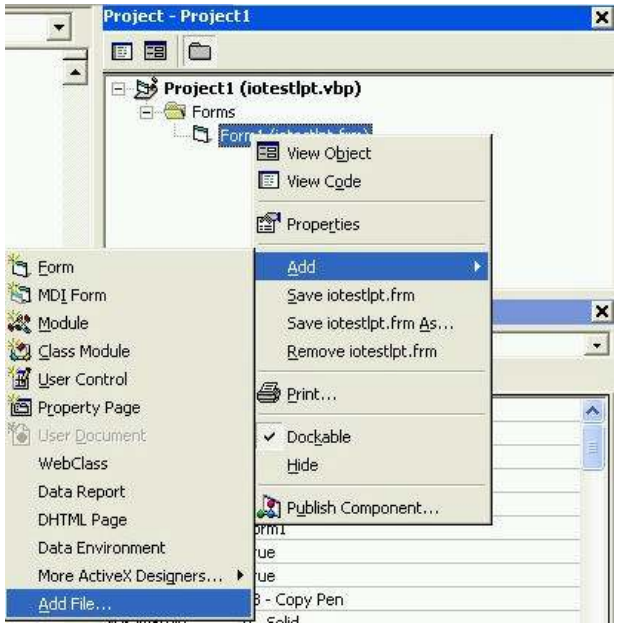
```
    Next r
    'grafik perbandingan y dan ydach
        MSChart2.RowCount = (List3.ListCount / 5) - 3
        MSChart2.ColumnCount = 2
        For r = 1 To ((List3.ListCount / 5) - 3)
            MSChart2.Row = r
            MSChart2.RowLabel = "t" + Str(r)
            MSChart2.Column = 1
            MSChart2.Data = List2.List(r + 1)
            MSChart2.Row = r
            MSChart2.Column = 2
            MSChart2.Data = List4.List(r + 1)
        Next r
        Label9.Caption    =    "Proses    Indentifikasi
      Selesai"
        Command2.Enabled = False
        Command3.Enabled = False
        Command6.Enabled = True
End If
'================================
hit = hit + 1
End Sub
```

9.5. Mengakses Port Paralel

Apabila implementasi kontrol yang akan kita buat menngunnakan komputer dan bukan Laptop, maka ada dua macam pilihan interfacing yang mungkin dilakukan dengan menggunakan port parallel, yaitu :

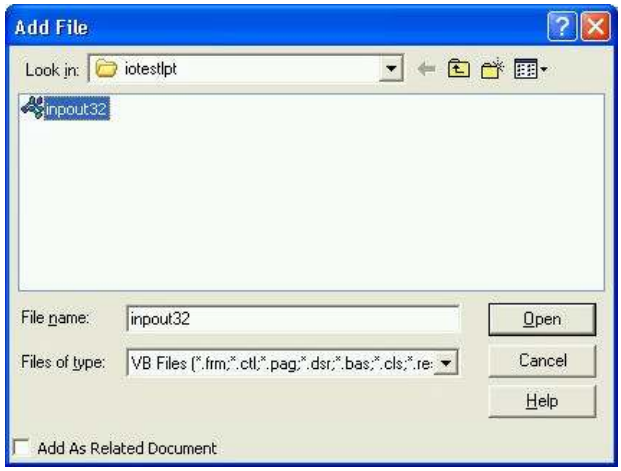
- Menggunakan Custuom Input Output Card yang dipasang pada slot ekspansi PCI Bus atau ISA Bus.
- Menggunakan saluran printer LPT

Untuk mengakses kedua macam saluran parallel tersebut, Visual Basic memerlukan file Dinamically Linked Libraries **inpout32.dll** dan file module **inpout32.bas** yang kedua file tersebut dapat didownload dari beberapa situs internet secara gratis.
Copykanlah file inpout32.dll dan inpout32.bas ke dalam folder project kita dan ke dalam folder c:\windows\system32.
Selanjutnya ketika sudah membuka Visual Basic dan membuat project baru, tambahkanlah modul inpout32.bas ke dalam project dengan cara arahkan pointer mouse ke jendela Project Explorer, lalu klik kana mouse dan pilih menu **Add** seperti tampak pada gambar berikut :



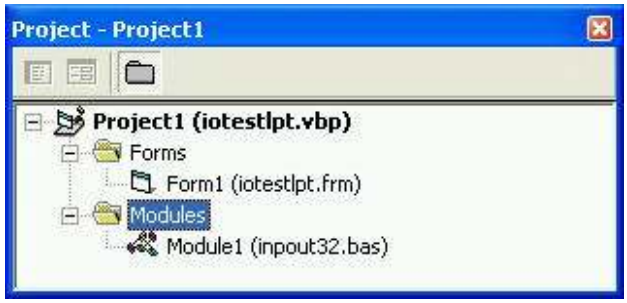
Gambar 9.51 Menu pada Project Explorer

Kemudian akan muncul pilihan lagi, pilihlah **Add Files ...** dan selanjutnya akan muncul kotak dialog seperti pada Gambar 9.52



Gambar 9.52 Kotak Dialog Add File

Pada kotak dialog ini sarilah file inpout32.bas yang sebelumnya telah kita copikan pada folder project kita. Kemudian tekan tombol **Open**, dan sebagai hasilnya pada jendela **Project Explorer** akan muncul tambahan Module yaitu Module1(inpout32.bas) seperti tampak pada Gambar 1.53



Gambar 9.53 Kotak Dialog Add File

Module inpout32.bas ini berisi deklarasi fungsi mengakses port parallel dengan cara pemakan sebagai berikut :

Untuk mengeluarkan data ke port parallel, kita menggunakan instruksi :

```
Out Alamat, DataOut
```

Contoh mengeluarkan data heksa &HFF ke port parallel dengan alamat &H378

```
Dim Alamat, DataOut As Integer
Alamat = &H378
DataOut = &HFF
Out Alamat, DataOut
```

Untuk membaca data masukan dari port parallel, kita menggunakan instruksi :

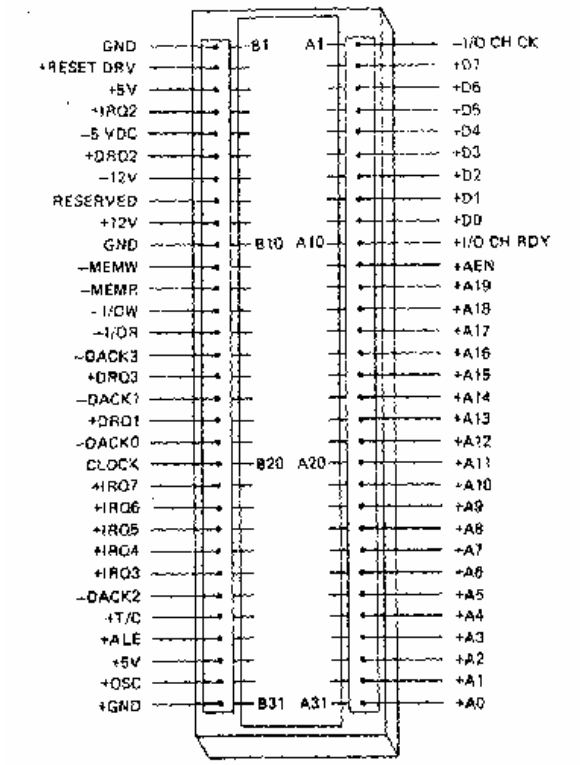
```
In (Alamat)
```

Contoh membaca data dari port parallel dengan alamat &H379

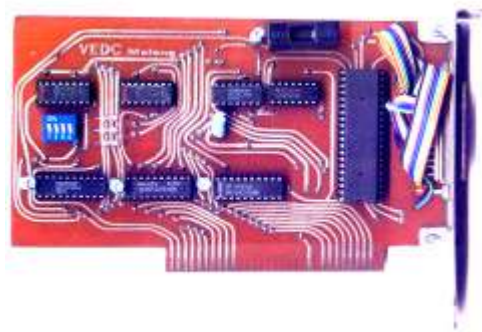
```
Dim Alamat, DataIn As Integer
Alamat = &H379
DataIn = In (Alamat)
```

9.5.1. Slot Expansi

Berikut ini adalah gambar slot expansi yang terdapat pada computer



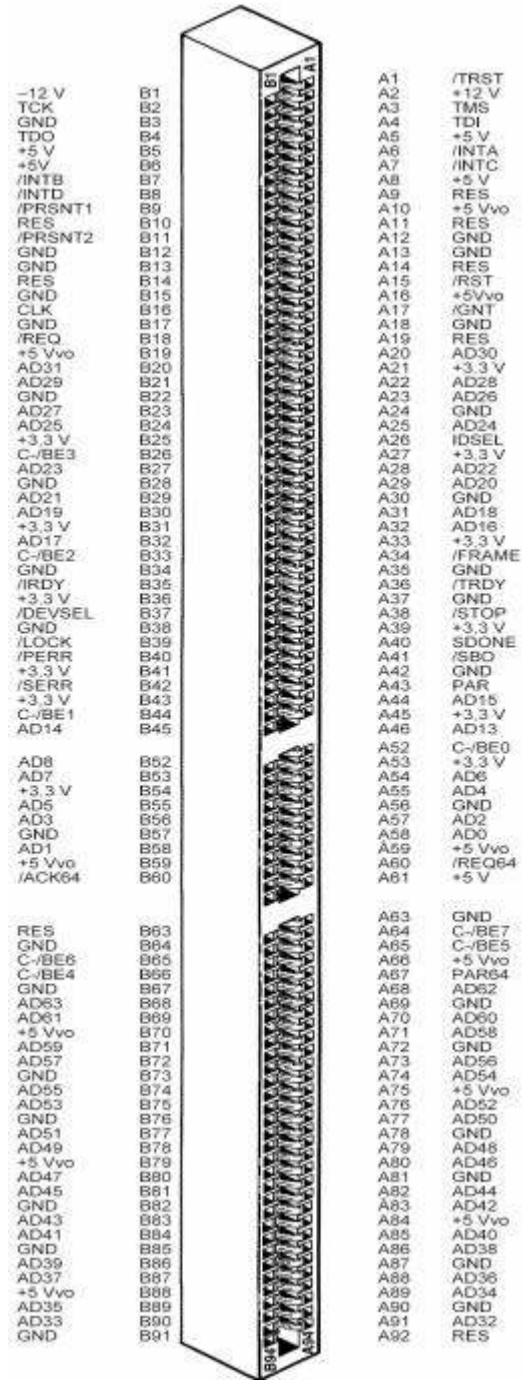
Gambar 9.54 Slot Ekspansi ISA Bus



Gambar 9.55 Input Output Card

Input Output Card seperti tampak pada Gambar 9.55 dapat dipasang pada ISA Bus ini, tetapi pada masa sekarang ISA bus sudah tidak ditemukan pada motherboard computer keluaran terbaru dan sebagai

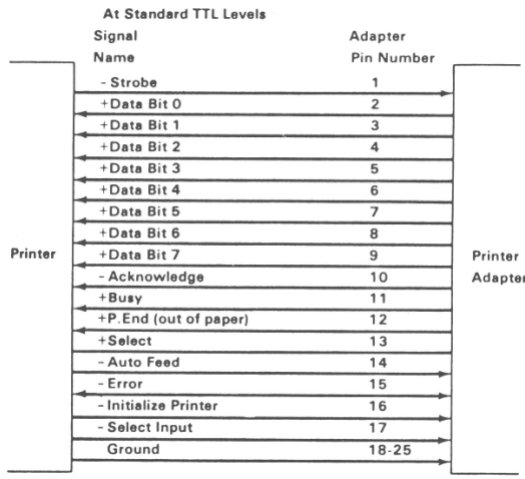
gantinya adalah slot ekspansi mengunakan PCI Bus seperti tampak pada Gambar 9.56



Gambar 9.56 Slot Ekspansi PCI Bus

9.5.2. LPT

Selain dipergunakan untuk saluran printer, port pada LPT dapat juga diakses untuk keperluan lain. Pemakaian LPT sangat menguntungkan karena kita tidak perlu membuka kotak CPU dan tidak pula diperlukan IO card. Custom hardware yang akan dikontrol dengan PC langsung disambungkan ke saluran LPT.



Gambar 9.57 Pin LPT

Ada tiga port yang tersedia pada saluran LPT, yaitu :

Port keluaran $378

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2

Port masukan $379

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin 11	Pin 10	Pin 12	Pin 13	Pin 15	-	-	-

Port masukan $37A

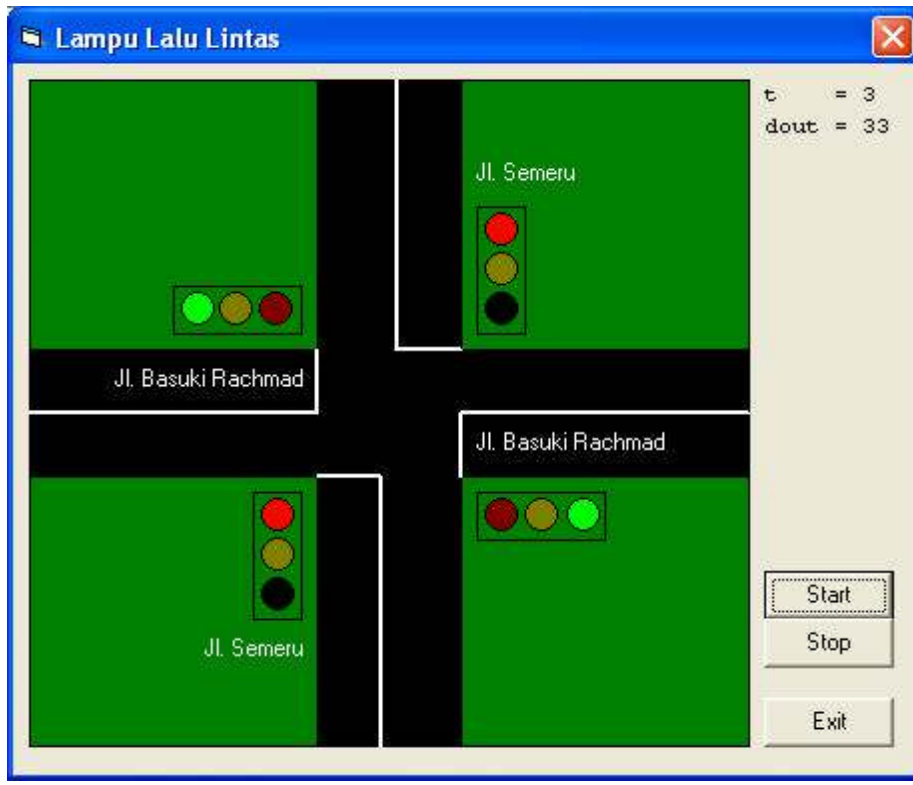
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	Pin 17	Pin16	Pin 14	Pin 1

Untuk mendapatkan 8 bit masukan kita harus mengkombinasikan 5 bit masukan dari port $379 dan 3 bit masukan port $37A.
Dari kombonasi masukan yang demikian, terdapat beberapa pin yang memiliki masukan terbalik (inverting) dan supaya diperoleh data masukan 8 bit siap pakai yang , maka kita terlebih dahulu harus mengkonversi data masukan tersebut sebagai berikut :

```
Dim DataIn As Integer
Dim Data379, Data37A As Integer

Data379 = In (&H379)
Data37A = In (&H379)
DataIn = (( Data379 Xor &H80) And &HF8 ) + (( Data37A
Xor &H3) And &H7 )
```

9.6. Implementasi Pemrograman Untuk Aplikasi Kontrol Melalui Port Paralel LPT

9.6.1. Kontrol Lampu Lalu Lintas



Gambar 9.58 Visualisasi Program Kontrol Lampu Lalu Lintas

Tabel kebenaran :

Lampu								Data Heksa	Waktu Penyalaan
		Jl. Basuki Rachmad			Jl. Semeru				
		H2	K2	M2	H1	K1	M1		
Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2		
0	0	1	0	0	0	0	1	&H21	5 detik
0	0	0	1	0	0	0	1	&H11	3 detik
0	0	0	0	1	1	0	0	&H0C	5 detik
0	0	0	0	1	0	1	0	&H0A	3 detik

Siklus waktu (t) dalam detik :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
dout = &H21					dout = &H11			dout = &H0C					dout = &H0A			

Listing Program :

```
Dim h378, h379, h37a As Integer
Dim t, dout As Double
```

```
Private Sub Command1_Click()
Timer1.Enabled = True
End Sub
```

```
Private Sub Command2_Click()
dout = &H0
Timer1.Enabled = False
End Sub
```

```
Private Sub Command3_Click()
dout = &H0
Out &H378, dout
End
End Sub
```

```
Private Sub Form_Load()
Timer1.Enabled = False
Timer1.Interval = 1000
dout = &H0
```

```
t = 0
End Sub
```

```
Private Sub Timer1_Timer()
If t >= 0 And t <= 4 Then dout = &H21
If t >= 5 And t <= 7 Then dout = &H11
If t >= 8 And t <= 12 Then dout = &HC
If t >= 13 And t <= 15 Then dout = &HA

Label5.Caption = "t    =" + Str(t)
If dout < &H10 Then Label6.Caption = "dout = H0" +
Hex(dout) Else Label6.Caption = "dout = H" + Hex(dout)

Out &H378, dout
'Jl. Bromo
If (dout And 1) = 1 Then m1(0).FillColor = &HFF& Else
m1(0).FillColor = &H80&
If (dout And 2) = 2 Then k1(0).FillColor = &HFFFF&
Else k1(0).FillColor = &H8080&
If (dout And 4) = 4 Then h1(0).FillColor = &HFF00&
Else h1(0).FillColor = &H0&
If (dout And 1) = 1 Then m1(1).FillColor = &HFF& Else
m1(1).FillColor = &H80&
If (dout And 2) = 2 Then k1(1).FillColor = &HFFFF&
Else k1(1).FillColor = &H8080&
If (dout And 4) = 4 Then h1(1).FillColor = &HFF00&
Else h1(1).FillColor = &H0&

'Jl. Semeru
If (dout And 8) = 8 Then m2(0).FillColor = &HFF& Else
m2(0).FillColor = &H80&
If (dout And 16) = 16 Then k2(0).FillColor = &HFFFF&
Else k2(0).FillColor = &H8080&
If (dout And 32) = 32 Then h2(0).FillColor = &HFF00&
Else h2(0).FillColor = &H0&
If (dout And 8) = 8 Then m2(1).FillColor = &HFF& Else
m2(1).FillColor = &H80&
If (dout And 16) = 16 Then k2(1).FillColor = &HFFFF&
Else k2(1).FillColor = &H8080&
If (dout And 32) = 32 Then h2(1).FillColor = &HFF00&
Else h2(1).FillColor = &H0&
t = t + 1
If t = 16 Then t = 0
End Sub
```

File Module :
Inpout32.bas

```
'Inp and Out declarations for direct port I/O
'in 32-bit Visual Basic 4 programs.

Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As
Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal
Value As Integer)
```

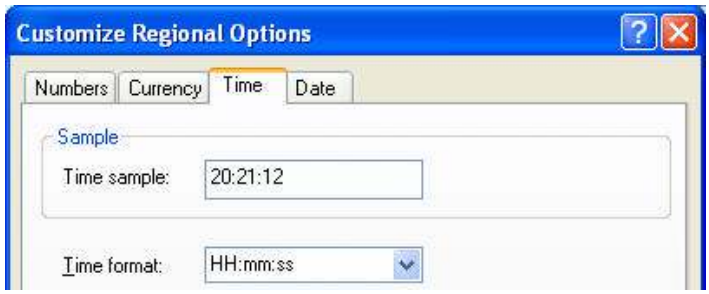
9.6.2. Kontrol Bel Sekolah Otomatis

Program berikut ini adalah membunyikan bel berupa suara terompet yang tersimpan dengan nama file ***"Trumpet1.wav"*** pada

- Setiap hari selasa jam 07:00:00
- Setiap hari selasa jam 08:00:00

Menyalakan lampu setiap hari pada jam 18:00:00
Dan mematikan lampu setiap hari pada jam 05:00:00
Lampu tersambung pada port LPT pin 6
Data utntuk menghidupkan lampu :
`Out &H378,&H10`
Data utntuk mematikan lampu :
`Out &H378,&H0`

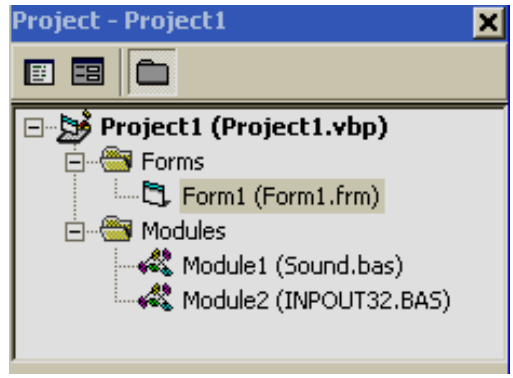
Jangan lupa bahwa seting jam pada control panel seperti format yang tampak pada Gambar berikut :



Gambar 9.59 Setting Format Jam Program Kontrol Bel Sekolah Otomatis

Gambar 9.60 Visualisasi Program Kontrol Bel Sekolah Otomatis

Selain mengeluarkan sinyal kontrol ke port LPT yang nantinya akan tersambung ke state relay Lampu Teras, program ini juga mengeluarkan suara bel sekolah yang dikeluarkan dari sound card ke Power Amplifier melalui line out loudspeaker,untuk itu project ini ditambahi dua file module pada yaitu Module1 (Sound.bas) dan Module2 (INPOUT32.BAS)



Gambar 9.61 Project Explorer Program Kontrol Bel Sekolah Otomatis

Pada project ini file suara bel sekolah yang akan dibunyikan juga harus tersedia dalam folder project, sehingga keseluruhan file yang harus ada dalam folder ini adalah seperti gambar beikut :

Gambar 9.62 File Program Kontrol Bel Sekolah Otomatis

Listing Program :

```
Dim tgl, hari, jam As Double
Dim retVal As Long
```

```
Private Sub Command1_Click()
Out &H378, &H10
End Sub
```

```
Private Sub Command2_Click()
Out &H378, &H0
End Sub
```

```
Private Sub Form_Load()
Timer1.Interval = 1000
End Sub
```

```
Private Sub Timer1_Timer()
hari = Weekday(Date)
jam = Val(Format(Time, "hhmmss"))
If hari = 1 Then Label1.Caption = "Minggu"
If hari = 2 Then Label1.Caption = "Senin"
If hari = 3 Then Label1.Caption = "Selasa"
If hari = 4 Then Label1.Caption = "Rabu"
If hari = 5 Then Label1.Caption = "Kamis"
If hari = 6 Then Label1.Caption = "Jumat"
If hari = 7 Then Label1.Caption = "Sabtu"
Label2.Caption = Date
Label3.Caption = Time

'bel
If hari = 2 And jam = 70000 Then
    retVal& = sndPlaySound(App.Path & "\Trumpet1.wav",
SND_ASYNC)
End If
If hari = 2 And jam = 80000 Then
    retVal& = sndPlaySound(App.Path & "\Trumpet1.wav",
SND_ASYNC)
End If

'lampu
If jam = 180000 Then Out &H378, &H10
If jam = 50000 Then Out &H378, &H10
End Sub
```

Listing Program
File Module1 : Sound.bas

```
Declare Function sndPlaySound Lib "winmm.dll" _
        Alias "sndPlaySoundA" (ByVal lpszSoundName _
        As String, ByVal uFlags As Long) As Long

Public Const SND_ASYNC = &H1
Public Const SND_LOOP = &H8
Public Const SND_NODEFAULT = &H2
Public Const SND_NOSTOP = &H10
Public Const SND_SYNC = &H0
```

Listing Program
File Module2 : INPOUT32.BAS

```
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As
Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal
Value As Integer)
```

1.6.3 Kontrol Posisi Gerak Lurus 1 Axis

Informasi data interface

PORTA (LPT &H378) : Output

			Lampu	Motor Step			
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
				L1	L2	L3	L4

Data Motor full step 1.8 °
Ke kanan ke arah posisi sensor 1 : &H08, &H04, &H02, &H01
Ke kiri ke arah posisi sensor 2 : &H01, &H02, &H04, &H08

Menyalakan lampu : &H10
Mematikan lampu : &H00

PORTB (LPT &H379) : Input

Proximity Sensor							
kiri	kanan						
S1	S2						
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0

S1	S2	Data Desimal
0	0	0
0	1	64
1	0	128
1	1	192

Data Sensor

Sensor aktif high,
jika ada logam maka outputnya = 1,
tidak ada logam outputnya = 0

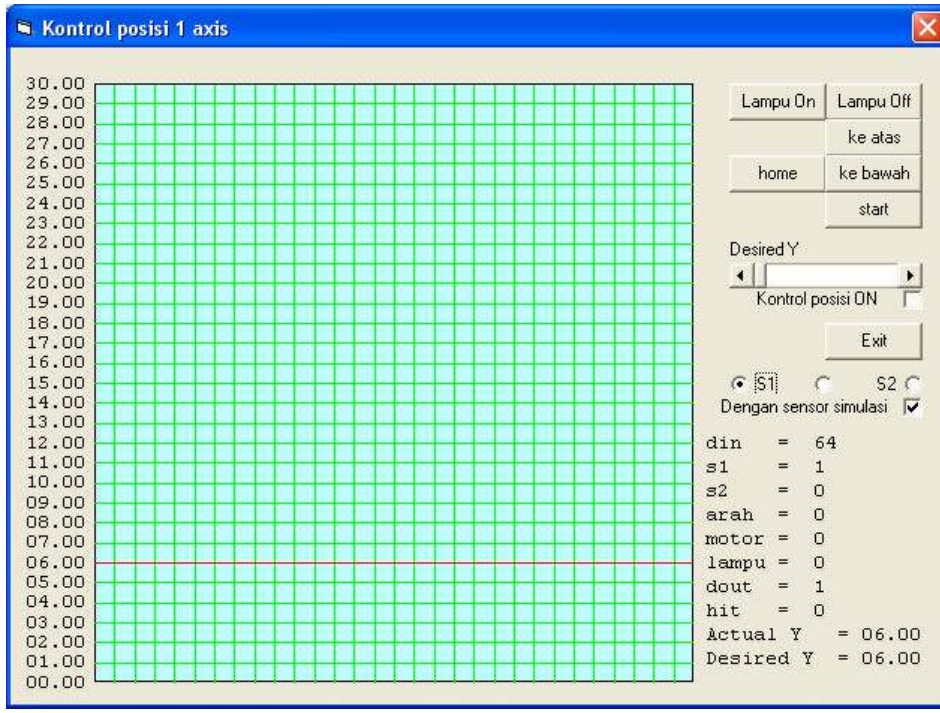
Data counter pada posisi paling kiri (home)
= 0 → cm 6,0
Data counter pada posisi paling kanan
= 16411step → cm 30,6

Total panjang gerakan = (30,6 – 6,0) → 24,6 cm
Resolusi gerakan per step
= 246 mm / 16411 step → 0.015 mm/step

Data pin konector DB9

Motor step					Proximity sensor			
pin 1	pin 2	pin 3	pin 4	pin 5	pin 6	pin 7	pin 8	pin 9
Hitam L1	Kuning L2	Biru dan Merah (+Vcc)	Krem L3	Orange L4	Biru Sensor1 Biru Sensor2 (GND)	Coklat sensor1 Coklat sensor2 (+Vcc)	Hitam Sensor2 (Output)	Hitam Sensor1 (Output)

Gambar 9.63 Rangkaian Percobaan Kontrol Posisi Gerak Lurus 1 Axis



Gambar 9.64 Visualisasi Program Kontrol Posisi Gerak Lurus 1 Axis

Listing Program

```
Dim h378, h379, h37a, din, dout, dm, c, f As Integer
Dim hit, pos, soll, ist As Double
Dim i, a, s1, s2, m, l, sit, sih, sy, y As Integer
```

```
Private Sub Check2_Click()
If Option2.Value = True Then din = 0
End Sub
```

```
Private Sub Command1_Click()
dout = &H0
Out &H378, dout
End
End Sub
```

```
Private Sub Command2_Click()
a = 1
End Sub
```

```
Private Sub Command3_Click()
If m = 1 Then
    m = 0
    Command3.Caption = "Start"
Else
    m = 1
    Command3.Caption = "Stop"
End If
End Sub
```

```
Private Sub Command4_Click()
a = 2
End Sub
```

```
Private Sub Command5_Click()
l = 1
dout = dout Or &H10
Out &H378, dout
End Sub
```

```
Private Sub Command6_Click()
l = 0
dout = dout And &HEF
Out &H378, dout
End Sub
```

```
Private Sub Command8_Click()
m = 1
a = 1
End Sub
```

```
Private Sub Form_Load()
'inisialisasi
Command1.Enabled = True
i = 0
UX1 = Shape1.Left
UY1 = Shape1.Top + (Shape1.Height / 2)
YX1 = Shape1.Left
YY1 = Shape1.Top + (Shape1.Height / 2)

Timer1.Interval = 1
Shape1.Width = 6000
Shape1.Height = 6000
'garis sumbu y
Line3.X1 = Shape1.Left
Line3.X2 = Shape1.Left + Shape1.Width
Line3.Y1 = Shape1.Top + Shape1.Height
Line3.Y2 = Line3.Y1
Line3.BorderColor = vbRed
'Mengatur garis pada shape
'Garis Horisontal
For i = 0 To 28
    Line1(i).BorderStyle = 1
    Line1(i).BorderColor = vbGreen
    Line1(i).X1 = Shape1.Left
    Line1(i).X2 = Shape1.Left + Shape1.Width
    Line1(i).Y1   =   Shape1.Top   +   ((i   +   1)   *
(Shape1.Height / 30))
    Line1(i).Y2   =   Shape1.Top   +   ((i   +   1)   *
(Shape1.Height / 30))
Next i
'Garis Vertikal
For i = 0 To 28
    Line2(i).BorderStyle = 1
    Line2(i).BorderColor = vbGreen
    Line2(i).X1   =   Shape1.Left   +   ((i   +   1)   *
(Shape1.Width / 30))
    Line2(i).X2   =   Shape1.Left   +   ((i   +   1)   *
(Shape1.Width / 30))
    Line2(i).Y1 = Shape1.Top + Shape1.Height
    Line2(i).Y2 = Shape1.Top
```

```
Next i

'Mengatur label skala sumbu y
For i = 0 To 30
    Label1(i).BackStyle = 0
    Label1(i).Caption = Format(30 - (i), "00.00")
    Label1(i).Height = 250
    Label1(i).Alignment = 1 ' rata kanan
    Label1(i).Left = Shape1.Left - 700
    Label1(i).Top = Shape1.Top + (i * (Shape1.Height /
30)) - 125
Next i
c = 0
Timer1.Interval = 1
dm = 1
dout = dm
Out &H378, dout
a = 0
l = 0
f = 0
hit = 0
Option2.Value = True
din = 0
End Sub
```

```
Private Sub Option1_Click()
din = 64
End Sub
```

```
Private Sub Option2_Click()
din = 0
End Sub
```

```
Private Sub Option3_Click()
din = 128
End Sub
```

```
Private Sub Timer1_Timer()
'membaca data masukan sensor1 dan sensor2
If Check2.Value = 1 Then
    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
Else
    din = ((Inp(&H379) And &HF8) Xor &H80) And &HC0
    Option1.Enabled = False
```

```
    Option2.Enabled = False
    Option3.Enabled = False
End If
If din = 192 Then
    s1 = 1
    s2 = 1
End If
If din = 128 Then
    s1 = 0
    s2 = 1

End If
If din = 64 Then
    s1 = 1
    s2 = 0
    hit = 0
End If
If din = 0 Then
    s1 = 0
    s2 = 0
End If

'menampilkan status
L1.Caption = "din   = " + Str(din)
L2.Caption = "s1    = " + Str(s1)
Label3.Caption = "s2    = " + Str(s2)
Label4.Caption = "arah  = " + Str(a)
Label5.Caption = "motor = " + Str(m)
Label6.Caption = "lampu = " + Str(l)
Label7.Caption = "dout  = " + Str(dout)
Label8.Caption = "hit   = " + Str(hit)
Label9.Caption  =  "Actual  Y      =  "  +  Format(pos,
"00.00") + " cm"
Label10.Caption  =  "Desired  Y    =  "  +  Format(soll,
"00.00") + " cm"

If (Check1.Value = 1) Then
    Command2.Enabled = False
    Command4.Enabled = False
    Command5.Enabled = False
    Command6.Enabled = False
    Command8.Enabled = False
    If  (Val(Format(soll,  "0.00"))  >  Val(Format(pos,
"0.00"))) And (s2 = 0) Then
        'm = 1
        a = 2
        Command6_Click
```

```
    End If
    If  (Val(Format(soll,  "0.00"))  <  Val(Format(pos,
"0.00"))) And (s1 = 0) Then
        'm = 1
        a = 1
        Command6_Click
    End If
    If  (Val(Format(soll,  "0.00"))  =  Val(Format(pos,
"0.00"))) Then
        m = 0
        a = 0
        Command5_Click
        Command3.Caption = "Start"
    End If
Else
    Command2.Enabled = True
    Command4.Enabled = True
    Command5.Enabled = True
    Command6.Enabled = True
    Command8.Enabled = True
End If

'menggerakkan motor ke arah sensor 1 (ke kiri)
If (m = 1) And (a = 1) And (s1 = 0) And (dm = 1) And
(f = 0) Then
    dm = 2
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit - 1
    f = 1
End If
If (m = 1) And (a = 1) And (s1 = 0) And (dm = 2) And
(f = 0) Then
    dm = 4
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit - 1
    f = 1
End If
If (m = 1) And (a = 1) And (s1 = 0) And (dm = 4) And
(f = 0) Then
    dm = 8
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit - 1
    f = 1
End If
```

```
If (m = 1) And (a = 1) And (s1 = 0) And (dm = 8) And
(f = 0) Then
    dm = 1
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit - 1
    f = 1
End If

'menggerakkan motor ke arah sensor 2 (ke kanan)
If (m = 1) And (a = 2) And (s2 = 0) And (dm = 1) And
(f = 0) Then
    dm = 8
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit + 1
    f = 1
End If
If (m = 1) And (a = 2) And (s2 = 0) And (dm = 2) And
(f = 0) Then
    dm = 1
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit + 1
    f = 1
End If
If (m = 1) And (a = 2) And (s2 = 0) And (dm = 4) And
(f = 0) Then
    dm = 2
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit + 1
    f = 1
End If
If (m = 1) And (a = 2) And (s2 = 0) And (dm = 8) And
(f = 0) Then
    dm = 4
    dout = (dout And &HF0) Or dm
    Out &H378, dout
    hit = hit + 1
    f = 1
End If

pos = (60 + (hit * 0.015)) / 10
soll = HScroll1.Value / 100

'menampilkan garis sumbu Y
```

```
sit = Shape1.Top
sih = Shape1.Height
ymax = 3000
sy = ((sit + sih) - sit) / ymax
y = pos * 100
If y < 0 Then y = 0
If y > 3000 Then y = 3000
Line3.Y1 = (sit + sih) - (y * sy)
Line3.Y2 = Line3.Y1
f = 0
End Sub
```

File Module INPOUT32.BAS :

```
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As
Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal
Value As Integer)
```

DAFTAR PUSTAKA

Beuth, Klaus,"*Elektronik 4 Digitaltechnik*", Vogel-Buchverlag, Wuerzburg, 1982ac

"*Elektronika Daya*", Gunadarma

ELWE, "*Lehrsysteme Leistungelektronik*".

Europalehrmittel, "*Fachkunde Information Elektronik*", Verlag Stuttgart hal 13/14.

Horn / Nur Lesson plan 51520104 PPPGT Malang 1988.

Horn / Rizal, Lesson Plan 51510102.

Horn / Sutrisno Lesson plan 52520203 PPPGT Malang 1988.

ITB, Polyteknik Mekanik Swiss, "Teknik Listrik Terpakai", hal 39 – 47

Kamajaya, "*Fisika 1*", Ganeqa Exact, Bandung, 1994.

MC68HC11F1 Technical Data, Motorola Inc., Arizona, 1990

MC68HC11F1 Programming Reference Guide, Motorola Inc., Arizona

M68HC11 Reference Manual, Motorola Inc., Arizona, 2002

MC68HC11F1 Technical Summary 8-Bit Microcontroller,Motorola Inc., Arizona, 1997

M. Affandi Agus Ponijo, "*Pengetahuan Dasar Teknik Listrik*"

M. Affandi Agus Ponijo, "*Pengetahuan Dasar Teknik Listrik*"

"*Microprocessor and Microcomputer*", ITT Fachlergänge, Pforzheim, 1979

Nur / Supr , Lesson plan 51520101, PPPG Teknologi Malang, 1988.

Ogata, K(1997). "*Teknik Kontrol Automatik*". Jilid 1. Erlangga: Jakarta

PEDC, *"Ilmu Listrik*", Bandung, PEDC, 1981, hal 127-129,PT Gunung Agung, 1981.

Pflaum, Richard. "*Elektronik IVA Leistungelektronik (lehrbuch) Werner Dzieia*", Verlag
KG Munchen

Pitowarno, E.(2006). "*Robotika Disain, Kontrol, Dan Kecerdasan Buatan*". Andi: Yogyakarta

Schmidt, Walf Deiter (1997). "*Sensor Schaltungs Technik*". Vogel (Wurzburg). Germany

Sugihartono, Drs.(1996). "*Dasar-dasar Kontrol Pneumatik*". Tarsito: Bandung

Suma'mur P.K, Msc, Dr ; "*Keselamatan kerja dan Pencegahan kecelakaan*" ;

Stielew, Roth, Prof, Dr, Ing. *"Institutfur Leistungselektronik und Elektrische Antriebe*"

W. Ernest, "*Elektrotecnik*", FranKfruft, Sauerlaender 1982, hal 13,14, 15-17.

Wil Helm Benz, "*Tabellen Buch Elektronik*", Kohl & Noltemeter & 10, Frankfurt, 1989

Uma'mur P.K, Msc, Dr ; "*Keselamatan kerja dan Pencegahan Kecelakaan*"

http://www.bbc.co.uk/schools/gcsebitesize/design/systemscontrol/pneumaticsrev1.shtml. (12.01.2008)

http://64.78.42.182/sweethaven/MechTech/hydraulics01/modulemain.asp? whichMod=0100. (12.01.2008)

en.wikipedia.org/wiki/**Brushless_DC**_electric_**motor** - 40k. (12.01.2008)

DAFTAR ISTILAH

AC Choper
ADC
adder
aktuator
akumulator
alkali
alternatif current
ALU
amper
amplifier
amplitudo
AND
anoda
aritmatika
arus cerat
arus listrik
arus pusar
asam nitrat
assembler
atom
band pas filter
basis
beda potensial
bias
biner
biner
biner
bit
BJT
boolean
break down
bus
caption
carry
celcius.
chip
clock
coil
common
coulomb
counter
CPU
crowbaring
cut-off
daya listrik
DC Choper
dekoder
demodulasi
density
depletion Layer
depolarisator
desimal
destilasi
DIAC
diagram bode
diamater
dielektrikum
difusi
digital
dimmer
dioda
dioda Schottky
dioda varactor
dioda zener
dipole
direct current
diskrit
double
download
drain
efisiensi
ekivalen

LAMPIRAN B.2

ekonomis
elekrolisa
elektro statis
elektroda
elektrolit
elektron
elektronika daya
emitor
EXOR
Fahrenheit.
farad
fase
ferro magnetik
filter
flag
flip-flop
flow chart
fluks
form
forward bias
foto cell
frekuensi
frekuensi modulasi
fungsi
fuzzy
fuzzylemps
galvanis
gate
gaya gerak listrik
gaya gerak magnet
generator
HandShaking
henry
hertz
hidrolika
high pass filter
histerisis
hole
horse power
hukum Kirchhoff
hukum ohm
IGBT
Impendansi
indeks
induksi
induktansi
induktor
input
input pembalik
instrumentasi
integer
ionisasi
ISA
isolasi
jembatan Wheatstone
jendela
joule
Joule
junction
kapasitansi
kapasitas panas
katoda
katup
kelvin.
kimiawi
kode program
koefisien
kolektor
kondensator
konduksi
konfigurasi
konstanta
kontrol
kontroler
konveksi
konversi
konverter
korona

korosi
kursor
LDR
loop
loop
low pass filter
LPT
LSI
medan magnit
mekanik
memori
mikrokomputer
mikrokontroller
mneumonic
modulasi
MOS
MOSFET
motor stepper
muatan listrik
multiplekser
Neutron
NOT
offset
ohm
oksidasi
op-amp
op-code
OR
orde dua
orde satu
oscilator
osilasi
osilator
osiloskop
output
overflow
parallel
parity
pengalamatan
penghantar
penyearah
penyulutan
penyulutan
permitivitas listrik
pewaktu
phasa
piston
plant
plasma
pneumatik
pointer
pointer
polarisasi
polaritas
pop
port
port
potensial barier
potensiometer
potensiometer
power supply
PPI
PROM
prosedur
Proton
pulsa
push
PWM
radiasi
radiator
radioaktif
RAM
Reamur.
register
register
reluktansi
RePROM
resonansi

LAMPIRAN B.4

reverse bias
ROM
root locus
RS232
satu fase
sekuensial
semi penghantar
semikonduktor
semikonduktor
sensor
seri
servo
siemens
silicon
silikon
sinyal
sinyal
Source
stack
statement
stator
string
subrutin
tabel kebenaran
tahanan
tahanan jenis
tegangan
Tegangan Knee
tegangan listrik
temperatur
tesla
thermocouple
thyristor
tiga fase
timer
Titik Q
Toleransi
Transistor
transistor
transistor bipolar,
Transkonduktansi
transportasi
triac
Unijunction Transistor
unipolar
usaha listrik
USB
valensi
variabel
variant
visual basic
volt
watt
weber

ISBN 978-979-060-089-8
ISBN 978-979-060-092-8

Buku ini telah dinilai oleh Badan Standar Nasional Pendidikan (BSNP) dan telah dinyatakan layak sebagai buku teks pelajaran berdasarkan Peraturan Menteri Pendidikan Nasional Nomor 45 Tahun 2008 tanggal 15 Agustus 2008 tentang Penetapan Buku Teks Pelajaran yang Memenuhi Syarat Kelayakan untuk digunakan dalam Proses Pembelajaran.