Kegiatan Belajar 2

Teknik Pemrograman PLC

Tujuan Pembelajaran :

1. Mahasiswa dapat merancang program kendali PLC sederhana
2. Mahasiswa dapat memasukkan program ke dalam PLC
3. Mahasiswa mampu mengidentifikasi kebenaran program

A. Unsur-Unsur Program

Program kendali PLC terdiri atas tiga unsur yaitu : alamat, instruksi, dan operand. Alamat adalah nomor yang menunjukkan lokasi, instruksi, atau data dalam daerah memori. Instruksi harus disusun secara berurutan dan menempatkannya dalam alamat yang tepat sehingga seluruh instruksi dilaksanakan mulai dari alamat terendah hingga alamat tertinggi dalam program.

Instruksi adalah perintah yang harus dilaksanakan PLC. PLC hanya dapat melaksanakan instruksi yang ditulis menggunakan ejaan yang sesuai. Oleh karena itu, pembuat program harus memperhatikan tata cara penulisan instruksi.

Operand adalah nilai berupa angka yang ditetapkan sebagai data yang digunakan untuk suatu instruksi. *Operand* dapat dimasukkan sebagai konstanta yang menyatakan nilai angka nyata atau merupakan alamat data dalam memori.

B. Bahasa Pemrograman

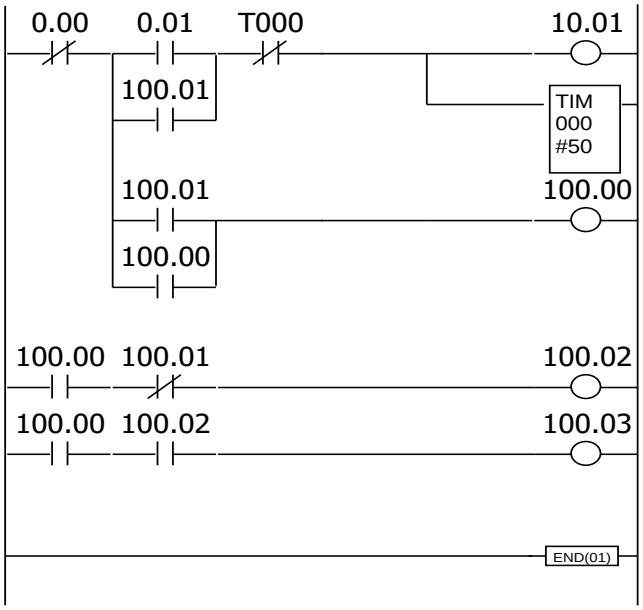
Program PLC dapat dibuat dengan menggunakan beberapa cara yang disebut bahasa pemrograman. Bahasa pemrograman tersebut antara lain : diagram ladder, kode mneumonik, diagram blok fungsi, dan teks terstruktur. Beberapa merk PLC hanya mengembangkan program diagram ladder dan kode mneumonik.

1. Diagram Ladder

Diagram ladder terdiri atas sebuah garis vertikal di sebelah kiri yang disebut bus bar, dengan garis bercabang ke kanan yang disebut rung. Sepanjang garis instruksi, ditempatkan kontak-kontak yang mengendalikan/mengkondisikan instruksi lain di sebelah kanan. Kombinasi logika

kontak-kontak ini menentukan kapan dan bagaimana instruksi di sebelah kanan dieksekusi.

Contoh :



Gambar 6. Contoh Diagram Ladder

Terlihat dari gambar di atas bahwa garis instruksi dapat bercabang kemudian menyatu kembali. Sepasang garus vertikal disebut kontak (kondisi). Ada dua kontak, yaitu kontak NO (*Normally Open*) yang digambar tanpa garis diagonal dan kontak NC (*Normally Closed*) yang digambar dengan garis diagonal. Angka di atas kontak menunjukkan *bit operand*.

2. Kode *Mneumonik*

Kode *mneumonik* memberikan informasi yang sama persis seperti halnya diagram ladder. Sesungguhnya, program yang disimpan di dalam memori PLC dalam bentuk mneumonik, bahkan meskipun program dibuat dalam bentuk diagram ladder. Oleh karena itu, memahami kode mneumonik itu sangat penting. Berikut ini contoh program *mneumonik*:

Tabel 8. Contoh Program *Mneumonik*

Alamat	**Instruksi**	***Operand***
00000	LD	HR 01

Alamat	Instruksi	*Operand*
00001	AND	0.01
00002	OR	0.02
00003	LD NOT	0.03
00004	OR	0.04
00005	AND LD	
00006	MOV(21)	
		0.00
		DM 00
00007	CMP(20)	
		DM 00
		HR 00

C. Struktur Daerah Memori

Program pada dasarnya adalah pemrosesan data dengan berbagai instruksi pemrograman. Data disimpan dalam daerah memori PLC..

Data yang merupakan operand suatu instruksi dialokasikan sesuai dengan jenis datanya. Tabel di bawah ini ditunjukkan daerah memori PLC CP1E-N40DRA sebagai berikut :

Tabel 9. Daerah Memori PLC CP1E-N40DRA.

Daerah Data		Words	Bit
Common Input Output (CIO)	Input	0 dan 1	0.00 – 0.11
	Output	100 dan 101	100.00 – 100.07
	Kerja (internal)	200 – 231	200.00 – 231.15
TR (*Temporarilly Relay*)			TR0 – TR7
Timer/counter		TC0 – TC7	

D. Instruksi Pemrograman

Terdapat banyak instruksi untuk memprogram PLC, tetapi tidak semua instruksi dapat digunakan pada semua model PLC. Instruksi pemrograman dapat dikelompokkan sebagai berikut :

Klasifikasi menurut pengkodean mneumonik :

1. Instruksi dasar
2. Instruksi khusus

Klasifikasi menurut kelompok fungsi

1. Instruksi sisi kiri (ladder)
2. Instruksi sisi kanan

Klasifikasi menurut kelompok fungsi

1. Instruksi ladder
2. Instruksi kendali bit
3. Instruksi timer/ counter
4. Instruksi geser bit
5. Instruksi sub routine
6. Instruksi ekspansi

1. Instruksi Diagram Ladder

Instruksi diagram ladder adalah instruksi sisi kiri yang mengkondisikan instruksi lain di sisi kanan. Pada program diagram ladder instruksi ini disimbolkan dengan kontak-kontak seperti pada rangkaian kendali elektromagnet.

Instruksi diagram ladder terdiri atas enam instruksi ladder dan dua instruksi blok logika. Instruksi blok logika adalah instruksi yang digunakan untuk menghubungkan bagian yang lebih kompleks.

Instruksi *LOAD* dan *LOAD NOT*

Instruksi *LOAD* dan *LOAD NOT* menentukan kondisi eksekusi awal, oleh karena itu, dalam diagram ladder disambung ke bus bar sisi kiri. Tiap instruksi memerlukan satu baris kode mneumonik. Kata “instruksi” mewakili sembarang instruksi lain yang dapat saja instruksi sisi kanan yang akan dijelaskan kemudian.

Diagram Ladder

Mneumonik

Alamat	Instruksi	Operand
00000	LD	0.00
00001	Instruksi	
00002	LD NOT	0.01
00003	Instruksi	

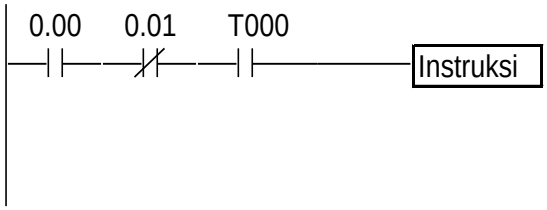
Gambar 7. Penggunaan Instruksi LOAD dan LOAD NOT

Jika hanya ada satu kontak seperti contoh di atas, kondisi eksekusi pada sisi kanan akan *ON* jika kontaknya *ON.* Untuk instruksi *LD* yang kontaknya *NO,* kondisi eksekusinya akan *ON* jika IR 0.00 ON; dan untuk instruksi *LD NOT* yang kontaknya *NC*, akan *ON* jika IR 0.01 *OFF..*

Instruksi *AND* dan *AND NOT*

Jika dua atau lebih kontak disambung seri pada garis yang sama, kontak pertama berkait dengan instruksi *LOAD* atau *LOAD NOT* dan sisanya adalah instruksi *AND* atau *AND NOT*.

Diagram Ladder



Mneumonik

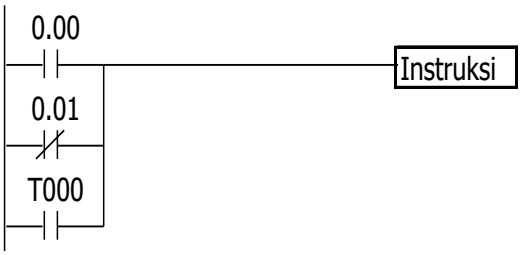
Alamat	Instruksi	Operand
00000	LD	0.00
00001	AND NOT	0.01
00002	AND	T000
00003	Instruksi	

Gambar 8. Penggunaan Instruksi *AND* dan *AND NOT*

Instruksi *OR* dan *OR NOT*

Jika dua atau lebih kontak terletak pada dua instruksi terpisah dan disambung paralel, kontak pertama mewakili instruksi *LOAD* atau *LOAD NOT* dan sisanya mewakili instruksi *OR* atau *OR NOT*.

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR NOT	0.01
00002	OR	T000
00003	Instruksi	

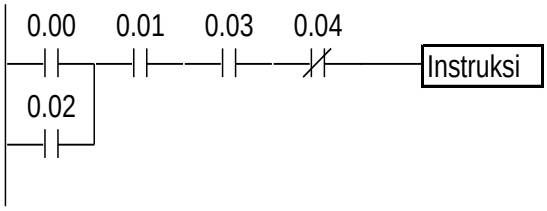
Gambar 9. Penggunaan Instruksi *OR* dan OR *NOT*

Instruksi akan mempunyai kondisi eksekusi ON jika salah satu di antara tiga kontak *ON,* yaitu saat IR 0.00 ON, saat IR 0.01 *OFF*, atau saat IR 0.03 ON.

Kombinasi Instruksi *AND* dan *OR*

Jika instruksi *AND* dan *OR* dikombinasikan pada diagram yang lebih rumit, mereka dapat dipandang secara individual di mana tiap instruksi menampilkan operasi logika pada kondisi eksekusi dan status bit operand.

Diagram Ladder



Mneumonik

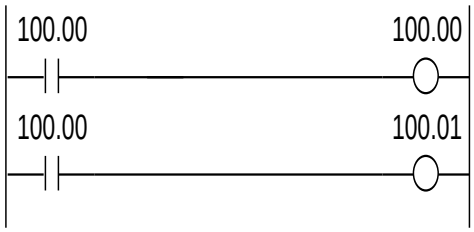
Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR	0.02
00002	AND	0.01
00003	AND	0.03
00004	AND NOT	0.04
00005	Instruksi	

Gambar 10. Kombinasi Instruksi AND dan OR

Di sini AND terletak di antara statur IR 0.00 dan status IR 0.01 untuk menentukan kondisi eksekusi dengan meng-OR-kan status IR 0.02. Hasil operasi ini menentukan kondisi eksekusi dengan meng-AND-kan status IR 0.03 yang selanjutnya menentukan kondisi eksekusi dengan meng-AND-kan kebalikan status IR 0.04.

2. Instruksi *OUT* dan *OUT NOT*

Cara paling sederhana untuk meng-O*UTPUT*-kan kombinasi kondisi eksekusi adalah dengan meng-*OUTPUT*-kan langsung menggunakan instruksi *OUTPUT* dan *OUTPUT NOT*. Istruksi ini digunakan untuk mengendalikan status bit operand sesuai dengan kondisi eksekusi. Dengan instruksi *OUTPUT*, *bit operand* akan *ON* selama kondisi eksekusinya *ON* dan akan *OFF* selama kondisi eksekusinya OFF. Dengan instruksi *OUTPUT NOT*, bit operand akan ON selama kondisi eksekusinya *OFF* dan akan *OFF* selama kondisi eksekusinya *ON.*



Alamat	Instruksi	Operand
00000	LD	0.00
00001	OUT	100,00
00002	LD	0.01
00003	OUT NOT	100.01

Gambar 11. Penggunaan Instruksi *OUTPUT* dan *OUTPUT NOT*

Pada contoh di atas, IR 100.00 akan ON jika IR 0.00 ON dan IR 100.01 akan OFF selama IR 0.01 ON. Di sini IR 0.00 dan IR 0.01 merupakan bit input dan IR 100.00 dan IR 100.01 merupakan bit output yang ditetapkan untuk peralatan yang dikendalikan PLC.

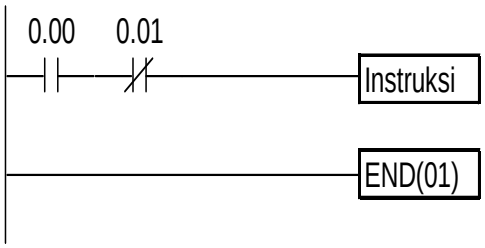
3. Instruksi *END(01)*

Instruksi terakihir yang diperlukan untuk melengkapi suatu program adalah instruksi *END.* Saat PLC menscan program, ia mengeksekusi semua instruksi hingga instruksi *END* pertama sebelum kembali ke awal program dan memulai eksekusi lagi

Nomor yang mengikuti instruksi *END* dalam kode mneumonik adalah kode fungsinya, yang digunakan saat memasukkan instruksi ke dalam PLC menggunakan konsol pemrogram.

Instruksi END tidak memerlukan operand dan tidak boleh ada kontak ditempatkan pada garis instruksi yang sama. Jika dalam program tidak ada instruksi *END*, program tersebut tidak akan dieksekusi.

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD	0.00
00001	AND NOT	0.01
00002	Instruksi	
00003	END(01)	

Gambar 12. Penggunaan Instruksi *END(01)*

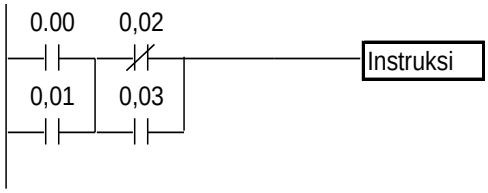
4. Instruksi Blok Logika

Jika rangkaian logika tidak dapat diwujudkan dengan instruksi *AND, AND NOT, OR*, atau *OR NOT* saja, maka perlu menggunakan instruksi blok logika. Perbedaannya adalah bahwa instruksi *AND, AND NOT, OR*, dan *OR NOT* mengkombinasikan antar kondisi eksekusi dengan suatu bit operand, sedangkan instruksi blok logika yang terdiri dari instruksi *AND LOAD* dan *OR LOAD* mengkombinasikan kondisi eksekusi dengan kondisi eksekusi terakhir yang belum digunakan. Instruksi blok logika tidak diperlukan dalam program diagram ladder, tetapi diperlukan hanya pada program mneumonik.

Instruksi *AND LOAD*

Instruksi *AND LOAD* meng-*AND*-kan kondisi eksekusi yang dihasilkan oleh dua blok logika.

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR	0.01
00002	LD NOT	0.02
00003	OR	0.03
00004	AND LD	
00005	Instruksi	

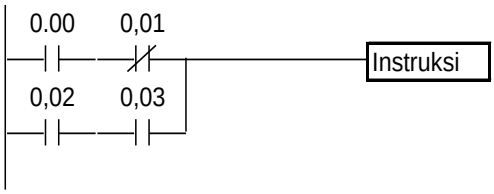
Gambar 13. Penggunaan Instruksi *AND LOAD*

Instruksi *OR LOAD*

Instruksi *OR LOAD* meng-OR-kan kondisi eksekusi yang dihasilkan oleh dua blok logika.

Diagram di bawah ini memerlukan instruksi *OR LOAD* antara blok logika atas dan blok logika bawah. Kondisi eksekusi akan dihasilkan untuk instruksi pada sisi kanan, baik saat IR 0.00 ON dan IR 0.01 OFF, atau saat IR 0.02 dan IR 0.03 keduanya ON.

Diagram Ladder



Mneumonik

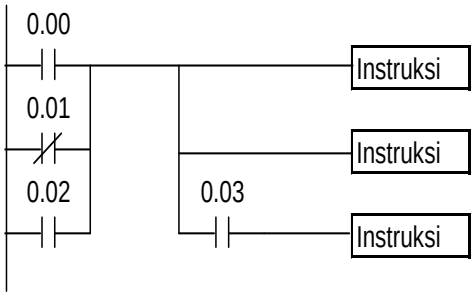
Alamat	Instruksi	Operand
00000	LD	0.00
00001	AND NOT	0.01
00002	LD	0.02
00003	ND	0.03
00004	OR LD	
00005	Instruksi	

Gambar 14. Penggunaan Instruksi OR LOAD

a. Mengkode Instruksi Sisi Kanan Ganda

Jika terdapat lebih dari satu instruksi sisi kanan dengan kondisi eksekusi yang sama, masing-masing dikode secara berurutan mengikuti kondisi eksekusi terakhir pada garis instruksi. Pada contoh di bawah ini, garis instruksi terakhir berisi satu kontak lagi yang merupakan instruksi *AND* terhadap IR 0.03.

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR NOT	0.01
00002	OR	0.02
00003	Instruksi 1	
00004	Instruksi 2	
00005	AND	
00006	Instruksi 3	

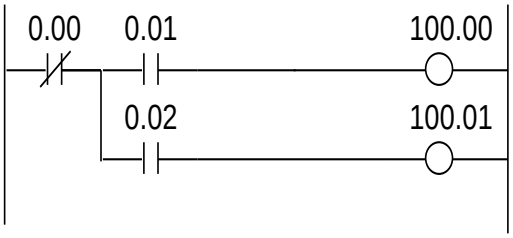
Gambar 15. Mengkode Instruksi Sisi Kanan Ganda

5. Penggunaan *Bit TR*

Bit TR (Temporarily Relay) digunakan untuk mempertahankan kondisi eksekusi pada garis instruksi *bercabang.* Hal ini dipertahankan karena garis instruksi dieksekusi menuju ke instruksi sisi kanan sebelum kembali ke titik cabang untuk mengeksekusi instruksi lainnya. Jika ada kontak pada garis instruksi setelah titik cabang, kondisi eksekusi untuk instruksi yang pertama tidak sama dengan kondisi pada titik cabang sehingga untuk mengeksekusi instruksi berikutnya menggunakan kondisi eksekusi titik cabang dan kontak lain setelah titik cabang tersebut.

Jika program dibuat dalam bentuk diagram ladder, tidak perlu memperhatikan bit TR karena bit TR hanya relevan pada pemrograman bentuk mneumonik.

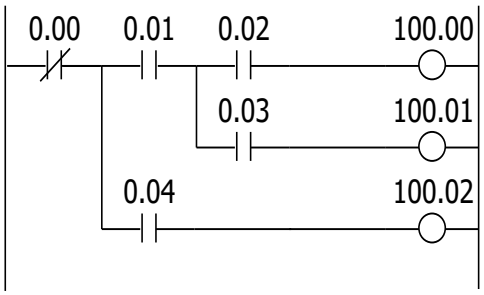
Terdapat delapan bit TR, yaitu TR0 sampai dengan TR7 yang dapat digunakan untuk mempertahankan kondisi eksekusi sementara..



Alamat	Instruksi	Operand
00000	LD NOT	0.00
00001	OUT	TR0
00002	AND	0.01
00003	OUT	100.00
00004	LD NOT	TR0
00005	AND	0.02
00006	OUT	100.01

Gambar 16. Penggunaan *Bit TR*

Contoh berikut ini menunjukkan penggunaan dua bit TR yaitu TR0 dan TR1 pada sebuah program.



Alamat	Instruksi	Operand
00000	LD NOT	0.00
00001	OUT	TR0
00002	AND	0.01
00003	OUT	TR1
00004	AND	0.02
00005	OUT	100.00
00006	LD NOT	TR1
00007	AND	0.03
00008	OUT	100.01
00009	LD NOT	TR0
00010	AND	0.04
00011	OUT	100.02

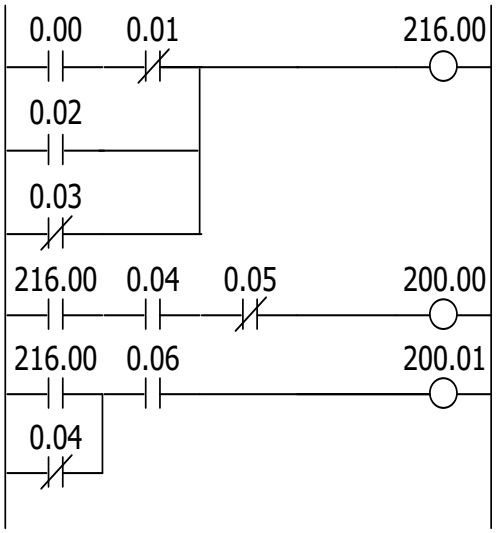
Gambar 17. Penggunaan Dua Bit TR

6. Penggunaan Bit Kerja (*Internal Relay*)

Bit kerja tidak ditransfer dari atau ke dalam PLC. Semua bit pada daerah IR yang tidak dialokasikan sebagai bit input/output dan bit pada daerah AR (*Auxilary Relay*) dapa digunakan sebagai bit kerja. Bit input/output dan bit yang dialokasikan untuk keperluan tertentu tidak dapat digunakan sebagai bit kerja.

Jika mengalami kesulitan pada pemrograman suatu program pengendalian pertimbangan pertama harus diberikan pada bit kerja untuk menyederhanakan program.

Bit kerja sering digunakan sebagai operand untuk salah satu instruksi *OUTPUT, OUTPUT NOT, DIFERENTIATE UP*, *DIFERENTIATE DOWN*, dan *KEEP*, kemudian digunakan sebagai kondisi yang menentukan bagaimana instruksi lain dieksekusi. Bit kerja juga dapat digunakan untuk menyederhanakan program saat kombinasi kondisi tertentu digunakan berulang-ulang. Pada contoh berikut ini IR 0.00, IR 0.01, IR 0.02, dan IR 0.03 dikombinasikan pada blok logika yang menyimpan kondisi eksekusinya sebagai status IR 216.00. Kemudian IR 216.00 dikombinasikan dengan kontak lain untuk menentukan kondisi output untuk IR 200.00 dan IR 200.01.



Alamat	Instruksi	Operand
00000	LD	0.00
00001	AND NOT	0.01
00002	OR	0.02
00003	OR NOT	0.03
00004	OUT	216.00
00005	LD	216.00
00006	AND	0.04
00007	AND NOT	0.05
00008	OUT	200.00
00009	LD	216.00
00010	OR NOT	0.04
00011	AND	0.06
00012	OUT	200.01

Gambar 18. Penggunaan Bit Kerja

7. Instruksi *Timer*

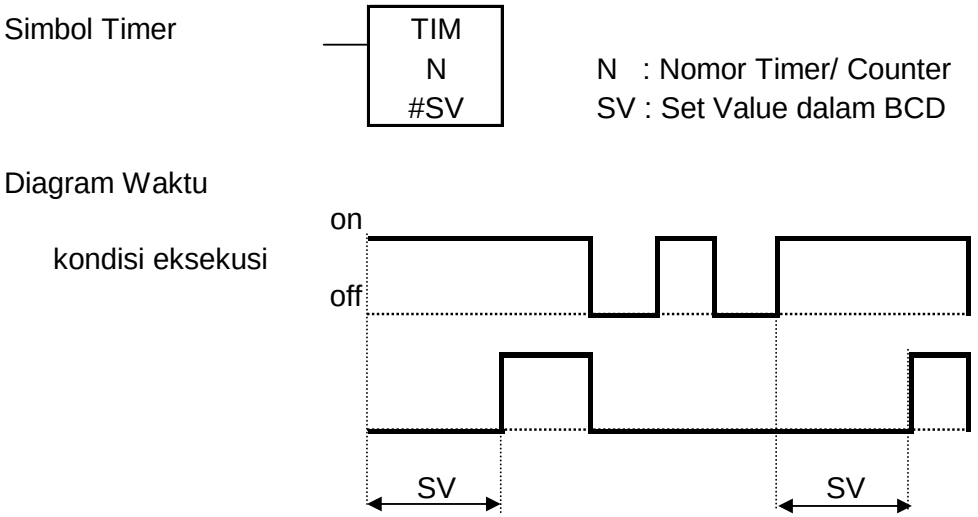
Instruksi Timer digunakan untuk operasi tunda waktu. Ia memerlukan dua operand yang terletak pada dua baris instruksi, yaitu baris pertama untuk nomor timer dan yug kedua untuk settig waktu (SV = *Set Value*). Meskipun demikian, instruksi Timer terletak dalam satu alamat.

Nomor Timer dipakai bersama untuk nomor *Counter*. Nomor *Timer/ Counter* hanya boleh digunakan sekali. Maksudnya, sekali nomor Timer/ Counter telah digunakan, ia tidak boleh digunakan untuk instruksi *Timer/ Counter* yang lain. Tetapi, nomor timer sebagai operand suatu kontak dapat digunakan sebanyak yang diperlukan.

Banyaknya nomor *Timer/ Counter* bergantung kepada tipe PLC. Misalnya, PLC OMRON CPM1A, *terdapat 128 nomor*, yaitu dari 000 sampai dengan 127. tidak diperlukan awalan apapun untuk menyatakan nomor timer. Tetapi, jika nomor timer sebagai operand suatu kontak harus diberi awalan TIM.

SV dapat berupa konstanta atau alamat *channel/ words*. Jika *channel* daerah IR sebagai unit input dimasukkan sebagai alamat *channel*, unit input ini harus disambung sedemikian sehingga SV dapat diset dari luar. *Timer/ Counter*

yang disambung dengan cara ini hanya dapat diset dari luar dalam mode *MONITOR* atau *RUN*. Semua SV, termasuk yang diset dari luar harus dalam BCD (*Binary Coded Decimal)*, yaitu bilangan desimal yang dikode biner. Penulisan SV harus diawali dengan tanda #.



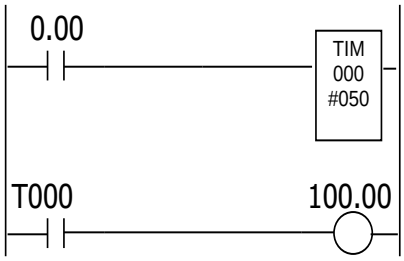
Gambar 19. Diagram Waktu Instruksi Timer

Timer bekerja saat kondisi eksekusinya beralih ke on dan direset (ke SV) saat kondisi eksekusinya beralih ke off. Jika kondisi eksekusi lebh lama daripada SV, *completion flag*, yaitu tanda yang menunjukkan hitungan waktu telah berakhir, tetap on hingga *Timer* direset. Timer akan reset jika trletak pada bagian program interlock saat kondisi eksekusi instruksi interlock (IL) off, dan saat terjadi pemutusan daya. Jika dikehendaki timer tidak reset oleh dua keadaan tersebut, maka bit pulsa clock pada daerah SR untuk mencacah *Counter* yang menghasilkan Timer menggunakan instruksi *Counter.*

SV mempunyai harga antara 0000 sampai dengan 9999 (BCD) dalam satuan deci-detik. Jadi, misalnya menghendaki 10 detik, maka nilai SV harus 100. Jika SV dinyatakan tidak dalam BCD, akan muncul pesan kesalahan.

Di bawah ini diberikan program-program penerapan timer.

a. Tunda on (1)



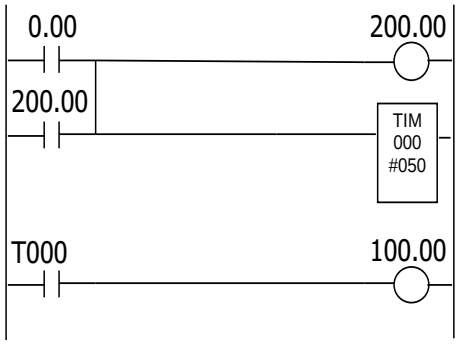
Alamat	Instruksi	Operand
00000	LD	0.00
00001	TIM	0,00
		#50
00002	LD	T000
00003	OUT	100.00

Gambar 20. Program Tunda *On*

Jika kondisi eksekusi timer (hanya ditentukan oleh kontak 0.00) on, maka timer aktif. Lima detik kemudian (completion flag timer on) kontak TIM 000 on hingga selanjutnya output 10.00 on. Jika lama kontak 0.00 on lebih pendek daripada SV, maka completion flag tetap off dan output 100.00 juga tetap off.

Agar dapat aktif meskipun kontak 0.00 hanya on sesaat, gunakan bit kerja untuk mengendalikan timer secara tidak langsung seperti ditunjukkan pada program berikut ini.

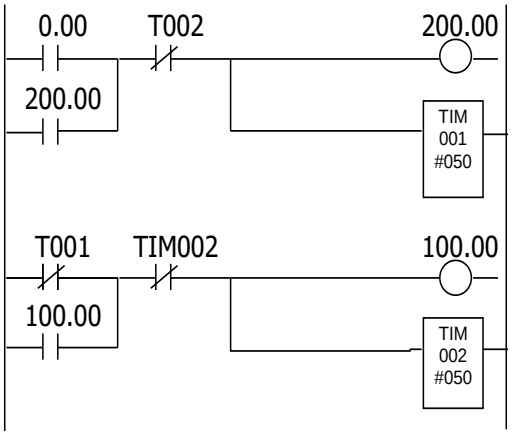
b. Tunda *on* (2)



Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR	200.00
00002	OUT	200.00
00003	TIM	000
		#050
00004	LD	T000
00005	OUT	100.00

Gambar 21. Program Tunda On (2)

c. Tunda *on* dan *off*



Alamat	Instruksi	Operand
00000	LD	0.00
00001	OR	200.00
00002	AND NOT	T002
00003	OUT	200.00
00004	TIM	001
		#050
00005	LD NOT	T001
00006	OR	100.00
00007	AND NOT	T002
00008	OUT	100.00
00009	TIM	002
		#050

Gambar 22. Program Tunda *On & Off*

3. Peringatan dalam Pemrograman

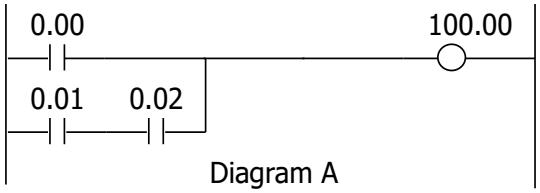
Untuk mengurangi kemungkinan terjadinya kesalahan dalam merancang program kendali, perlu diingat hal-hal sebagai berikut :

a. Jumlah kondisi (kontak) yang digunakan seri atau paralel dan juga banyaknya perulangan penggunaan suatu bit tak terbatas sepanjang kapasitas memori PLC tidak dilampaui.

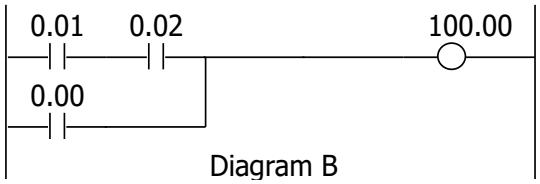
b. Diantara dua garis instruksi tidak boleh ada kondisi yang melintas secara vertikal.

c. Tiap garis instruksi harus memiliki sedikitnya satu kondisi yang menentukan eksekusi instruksi sisi kanan, kecuali untuk instruksi *END*(01), *ILC*(03) dan *JME*(05).

d. Dalam merancang diagram ladder harus memperhatikan kemungkinan instruksi yang diperlukan untuk memasukannya. Misalnya, pada gambar A di bawah ini diperlukan instruksi *OR LOAD*. Hal ini dapat dihindari dengan menggambar ulang diagram ladder seperti gambar B.



Alamat	Instruksi	Operand
00000	LD	0.00
00001	LD	0.01
00002	AND	0.02
00003	OR LD	
00004	OUT	100.00



Alamat	Instruksi	Operand
00000	LD	0.01
00001	AND	0.02
00002	OR	0.00
00003	OUT	100.00

Gambar 23. Penyederhanaan Program Logika

4. Eksekusi program

Saat eksekusi program, PLC men-scan program dari atas ke bawah, mengecek semua kondisi, dan mengeksekusi semua instruksi. Instruksi harus ditempatkan dengan tepat, misalnya data yang dikehendaki dipindahkan ke words sebelum words tersebut digunakan sebagai operand instruksi. Ingat bahwa garis instruksi berakhir pada instruksi terminal sisi kanan, setelah itu baru mengeksekusi garis instruksi bercabang ke instruksi terminal yang lain.

Eksekusi program semata-mata merupakan salah satu tugas yang dilakukan oleh PLC sebagai bagian dari waktu siklus.

5. Langkah-langkah pembuatan program

Untuk membuat program kendali PLC ditempuh melalui langkah-langkah sistematis sebagi berikut :

a. Menguraikan urutan kendali

Pembuatan program diawali dengan penguraian urutan kendali. menggunakan kalimat-kalimat logika, gambar-gambar, diagram waktu, atau bagan alir (flow chart).

b. Menetapkan bit operand untuk peralatan input/ output.

Bit operand untuk peralatan input/ output mengacu pada daerah memori PLC yang digunakan.

Jumlah bit oprand yang tersedia bergantung kepada tipe PLC yang dispesifikasikan menurut jumlah input-outputnya. Perbandingan jumlah bit input dan output pada umumnya 3 : 2.

Tabel 10. Contoh Daerah Memori PLC OMRON CP1E-N40DRA.

Daerah Data		Words	Bit
Common Input Output (CIO)	Input	0 dan 1	0.00 – 0.11
	Output	100 dan 101	10.00 – 10.07
	Kerja (internal)	200 – 231	200.00 – 231.15
TR (Temporarilly Relay)			TR0 – TR7
Timer/counter		TC0 – TC7	

c. Membuat program kendali

Program kendali PLC dapat dibuat dengan diagram ladder jika menggunakan komputer atau kode mneumonik jika menggunakan konsol.

6. Program Kendali Motor

Contoh :

- Operasi motor satu arah putaran
- Operasi motor dua arah putaran
- Operasi motor dua kecepatan
- Operasi motor start bintang segitiga
- Operasi beberapa motor kendali kerja berurutan

1. Program Kendali Motor Satu arah Putaran

a. Urutan Kendali Motor

Jika tombol Start ditekan, motor berputar searah jarum jam, dan jika kemudian tombol Start dilepaskan[1)], motor tetap berputar dalam arah yang sama. Jika tombol Stop ditekan, motor berhenti berputar.

b. Penetapan *Bit I/O*

Tabel 11. Penetapan Bit I/O pada Kendalai Motor Satu Arah Putaran

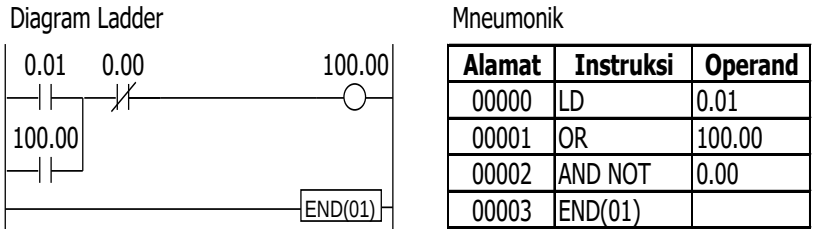
No	Alat input/output	*Bit operand*	Fungsi
1	Tombol Stop	0.00	Menghentikan operasi motor
2	Tombol Start	0.01	Menjalankan motor
3	Kontaktor[2)]	100.00	Menghubungkan motor ke jaringan

Keterangan :

1) Kecuali untuk operasi yang sangat khusus, secara umum operasi menjalankan motor adalah dengan menekan tombol Start dan jika kemudian tombol ini dilepas motor akan tetap berputar. Maka, selanjutnya untuk menjalankan motor cukup disebutkan dengan menekan tombol *Start* saja.

2) Motor berdaya kecil dapat disambung langsung ke PLC. Tetapi, untuk motor berdaya cukup dengan arus nominal diatas kemampuan PLC harus menggunakan kontaktor sebagai penghubung motor ke jaringan.

3) Program Kendali PLC



Alamat	Instruksi	Operand
00000	LD	0.01
00001	OR	100.00
00002	AND NOT	0.00
00003	END(01)	

Gambar 24. Program Kendali Motor Satu Arah Putaran

2. Program Kendali Motor Dua Arah Putaran

a. Urutan Kendali Motor

Jika tombol *Forward* (FWD) ditekan, motor berputar searah jarum jam dan jika yang ditekan tombol *Reverse* (REV), motor berputar berlawanan arah jarum jam. Tombol *STOP* digunakan untuk menghentikan operasi motor setia saat.

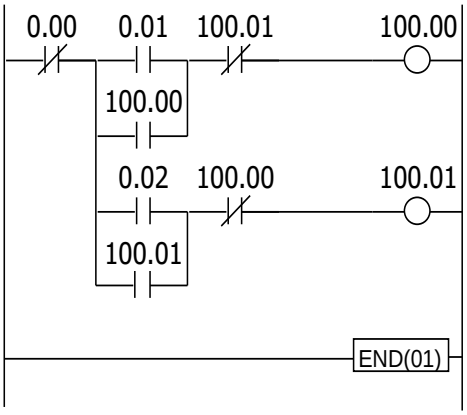
b. Penetapan Bit I/O

Tabel 12. Penetapan Bit I/O pada Kendali Motor Dua Arah Putaran

No	Alat input/output	*Bit operand*	Fungsi
1	Tombol Stop	0.00	Menghentikan operasi motor
2	Tombol Fwd	0.01	Menjalankan motor searah jarum jam
3	Tombol Rev	0.02	Menjalankan motor berlawanan arah jarum jam
4	Kontaktor K1	100.00	Kontaktor putaran searah jarum jam
5	Kontaktor K2	100.01	Kontaktor putaran berlawanan arh jarum jam

a. Program Kendali PLC

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD NOT	0.00
00001	OUT	TR0
00002	LD	0.01
00003	OR	100.00
00004	AND LD	
00005	AND NOT	100.01
00006	OUT	10
00007	LD	TR0
00008	LD	0.02
00009	OR	100.01
00010	AND LD	
00011	AND NOT	100.00
00012	OUT	100.01
00013	END(01)	

Gambar 25 Program Kendali Motor Dua Arah Putaran

3. Program Kendali Motor Dua Kecepatan

a. Urutan Kendali Motor

Jika tombol *LOW* ditekan, motor berputar dalam kecepatan rendah, dan jika kemudian tombol *High* ditekan motor berputar dalam kecepatan tinggi. Motor tidak dapat distart langsung pada kecepatan tinggi dan pada kecepatan tinggi motor tidak dapat dipindahkan ke kecepatan rendah. Tombol Stop untuk menghentikan operasi motor.

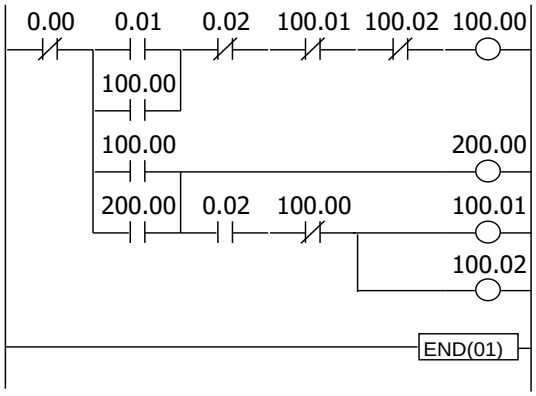
b. Penetapan *Bit I/O*

Tabel 13. Penetapan Bit I/O pada kendali Motor Dua Kecepatan

No	Alat input/output	*Bit operand*	Fungsi
1	Tombol Stop	0.00	Menghentikan operasi motor
2	Tombol Low Speed	0.01	Menjalankan motor kecepatan rendah
3	Tombol High Speed	0.02	Menjalankan motor kecepatan tinggi
4	Kontaktor K1	100.00	Kontaktor kecepatan rendah
5	Kontaktor K2	100.01	Kontaktor kecepatan tinggi
6	Kontaktor K3	100.00	Kontaktor kecepatan tinggi

c. Program Kendali PLC

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD NOT	0.00
00001	OUT	TR0
00002	LD	0.01
00003	OR	100.01
00004	AND LD	
00005	AND NOT	0.02
00006	AND NOT	100.01
00007	AND NOT	100.02
00008	OUT	100.00
00009	LD	TR0
00010	LD	100.00
00011	OR	200.00
00012	AND LD	
00013	OUT	200.00
00014	AND	0.02
00015	AND NOT	100.00
00016	OUT	100.01
00017	OUT	100.02
00018	END(01)	

Gambar 26. Program Kendali Motor Dua Kecepatan

4. Program Kendali Motor Sistem Start Bintang Segitiga

a. Urutan Kendali Motor

Jika tombol Start ditekan, motor berputar dalam sambungan bintang. Lima detik kemudian, motor berputar dalam sambungan segitiga. Tombol Stop untuk menghentikan operasi motor setiap saat.

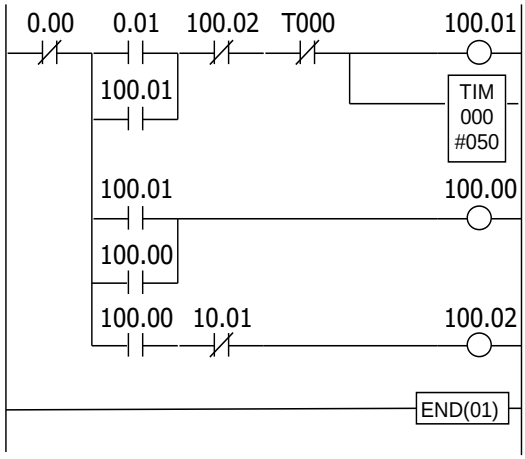
b. Penetapan *Bit I/O*

Tabel 14. Penetapan Bit I/O pada Kendali Motor Star/Segitiga

No	Alat input/output	*Bit operand*	Fungsi
1	Tombol Stop	0.00	Menghentikan operasi motor
2	Tombol Start	0.01	Menjalankan motor
3	Kontaktor K1	100.00	Kontaktor utama
4	Kontaktor K2	100.01	Kontaktor bintang
5	Kontaktor K3	100.02	Kontaktor segitiga

c. Program Kendali PLC

Diagram Ladder



Mneumonik

Alamat	Instruksi	Operand
00000	LD NOT	0.00
00001	OUT	TR0
00002	LD	0.01
00003	OR	100.01
00004	AND LD	
00005	AND NOT	100.02
00006	AND NOT	T000
00007		#050
00008	OUT	100.01
00009	LD	TR0
00010	LD	100.01
00011	OR	100.00
00012	AND LD	
00013	OUT	100.00
00014	LD	TR0
00015	AND	100.00
00016	AND NOT	100.01
00017	OUT	100.02
00018	END(01)	

Gambar 27. Program Kendali Motor Start Bintang Segitiga.

Rangkuman

1. Program kendali PLC terdiri atas tiga unsur yaitu alamat, instruksi dan operand.
2. Program PLC dapat dibuat dengan diagram ladder atau kode mneumonik. Pemilihan tipe program ditentukan oleh alat pemrogram yang akan digunakan.
3. Untuk dapat membuat program kendali PLC, pemrogram harus memahami struktur daerah memori PLC yang akan digunakan. Daerah memori PLC berbeda-beda sesuai dengan tipe PLC.
4. Memahami instruksi pemrograman memegang peranan paling penting dalam pembuatan program kendali. Terdeapat banyak sekali instruksi pemrograman, tetapi tidak semua instruksi dapat duterapkan pada semua tipe PLC.
5. Setiap program selalu diawali dengan instruksi LOAD dan diakhiri dengan instruksi END. Tanpa instruksi END program tidak dapat dieksekusi.
6. Program dieksekusi dengan menscan mulai dari alamat terendah hingga ke alamat tertinggi yaitu instruksi END. Pada diagram ladder ini berarti program dikesekusi mulai dari atas ke bawah bila garis instruksi bercabang, dan kemudian ke kanan hingga mengeksekusi instruksi sisi kanan.
7. Pembuatan program PLC harus dilakukan secara sistematis, yaitu mendeskripsikan sistem kendali, menetapkan operand untuk alat input/ output, baru membuat program.
8. Banyak sekali variasi program kendali motor sebagai penggerak mesin. Tetapi, untuk operasi motor induksi, suatu motor yang paling banyak digunakan sebagai penggerak mesin, secara prinsip hanya ada beberapa operasi motor yaitu operasi motor satu arah putaran, operasi dua arah putaran, operasi dua kecepatan, operasi dengan start bintang segitiga, operasi berurutan dan operasi bergantian.