BAB III

LANDASAN TEORI

3.1 *Programmable Logic Controller (PLC)*

Programmable Logic Controller (PLC) adalah komputer elektronik yang mudah digunakan (*user friendly*) yang memiliki fungsi kendali untuk berbagai tipe dan tingkat kesulitan yang beraneka ragam. Definisi *Programmable Logic Controller* menurut Capiel (1982) adalah *"Sistem elektronik yang beroperasi secara digital dan didesain untuk pemakaian di lingkungan industri, dimana sistem ini menggunakan memori yang dapat diprogram untuk penyimpanan secara internal instruksi-instruksi yang mengimplementasikan fungsi-fungsi spesifik seperti logika, urutan, perwaktuan, pencacahan dan operasi aritmatik untuk mengontrol mesin atau proses melalui modul-modul I/O digital maupun analog"*. Berdasarkan namanya konsep PLC adalah sebagai berikut:

a. *Programmable*, menunjukkan kemampuan dalam hal memori untuk menyimpan program yang telah dibuat yang dengan mudah diubah-ubah fungsi atau kegunaannya.
b. *Logic*, menunjukkan kemampuan dalam memproses input secara aritmatik dan *logic* (ALU), yakni melakukan operasi membandingkan, menjumlahkan, mengalikan, membagi, mengurangi, negasi, *AND*, *OR*, dan lain sebagainya.
c. *Controller*, menunjukkan kemampuan dalam mengontrol dan mengatur proses sehingga menghasilkan output yang diinginkan.

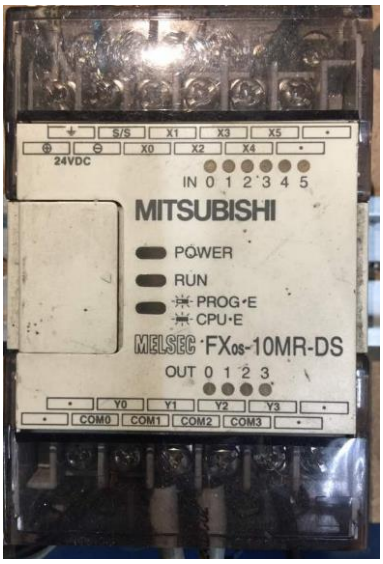
PLC ini dirancang untuk menggantikan suatu rangkaian *relay* sekuensial

dalam suatu sistem kontrol. Selain dapat di program, alat ini juga dapat dikendalikan, dan dioperasikan oleh orang yang tidak memiliki pengetahuan di bidang pengoperasian komputer secara khusus. PLC ini memiliki bahasa pemrograman yang mudah dipahami dan dapat dioperasikan bila program yang telah dibuat dengan menggunakan *software* yang sesuai dengan jenis PLC yang digunakan sudah dimasukkan. Alat ini bekerja berdasarkan input-input yang ada dan tergantung dari keadaan pada suatu waktu tertentu yang kemudian akan meng-*ON* atau meng-*OFF*kan *output-output*. 1 menunjukkan bahwa keadaan yang diharapkan terpenuhi sedangkan 0 berarti keadaan yang diharapkan tidak terpenuhi. PLC juga dapat diterapkan untuk pengendalian sistem yang memiliki output banyak. Secara umum fungsi PLC adalah sebagai berikut:

a. *Sequential Control*. PLC memproses *input* sinyal biner menjadi *output* yang digunakan untuk keperluan pemrosesan teknik secara berurutan (*sequential*), disini PLC menjaga agar semua langkah dalam proses sekuensial berlangsung dalam urutan yang tepat.
b. *Monitoring Plant*. PLC secara terus menerus memonitor status suatu sistem (misalnya temperatur, tekanan, tingkat ketinggian) dan mengambil tindakan yang diperlukan sehubungan dengan proses yang dikontrol (misalnya nilai sudah melebihi batas) atau menampilkan pesan tersebut pada operator.

Prinsip kerja sebuah PLC adalah menerima sinyal masukan proses yang dikendalikan lalu melakukan serangkaian instruksi logika terhadap sinyal masukan tersebut sesuai dengan program yang tersimpan dalam memori lalu menghasilkan sinyal keluaran untuk mengendalikan aktuator atau peralatan

lainnya. (Budi, 2013)



Gambar 3. 1 PLC FX0s-10MR-DS

3.1.1 Spesifikasi PLC FXos-10MR-DS

Specifications			
Elektrical Data			
Max. number inputs/outputs			10
Power Supply	AC range (+10%,-15%)		-
	Frequency at AC		-
	DC range (+10%,-15%)		24 V DC
Max. input apparent power			4 W
Inrush Current at ON	100 V AC		-
	200 V AC		-
	24 V DC		60 A/1.5 ms
Allowable momentary power failure time			5 ms
External current supply (24 V DC)			-
Inputs			
Integrated Inputs			6
Min. Current for Logical 1		mA	4.5
Max. Current for Logical 0		mA	1.5
Response Time		ms	10 ms (at time of shipping), adjustable from 0 - 15 ms in steps of 1 ms
Outputs			
Integrated Outputs			4
Output		Art	Relay
Switching Voltage (max)		V	<264 V AC, <30V DC, for transistor version : 5-30V DC
Max. Output Current	Per Output	A	2.5
	per group*	A	-
Max.Switching Current	Inductive load		80 VA
	Lamp load	W	100
Response time		ms	10
Life of Contact (Switching times)			3000000 at 20VA, 1000000at 35 VA, 200000 at 80 VA
Mechanical data			
Weight		kg	0.3
Dimensions (W x H x D)		mm	60 x90 x 47

Gambar 3. 2 Spesifikasi PLC FXos-10MR-DS

3.1.2 Spesifikasi Pemrograman PLC FXos-10MR-DS

System data	
Program data	
Program memory	800 steps EEPROM (internal)
Program language	Stepladder instructions, instruction list
Program execution	Periodical execution of the stored program
Program protection	Password protection with 3 protection levels
Number of instructions	20 sequence instructions, 2 step ladder instructions, 35 applied instructions
Cycle period	1.6 – 3.5 µs/log. instruction
Operands	
Internal relays	512, 16 buffered
Special relays	56
Step relays	64
Timers	56 (100 ms), with 24 to be switched to 10 ms
External setpoint entry via potentiometer	1
Counter	16 inputs, 16 Bit
High speed counter inputs	4 counter inputs, 32 Bit
Data register	32, 16 Bit
Filer register	—
Index register	2, 16 Bit
Special register	27, 16 Bit
Pointer	64
Nesting operands	8
Interrupt inputs	4
Constants	16 / 32 Bit

Gambar 3. 3 Spesifikasi Pemrograman PLC FXos-10MR-DS

3.1.3 GX Developer

GX Developer mendukung MELSEC *Instruction List* (IL), MELSEC *Ladder Diagram* (LD) dan MELSEC *Sequential Function Chart* (SFC). Pada Program GX Developer menungkinkan pengguna mengubah program IL ke program LD atau sebaliknya pada saat menjalankan program.

3.2 Pemrograman PLC

Untuk menjalankan suatu PLC diperlukan program untuk mengatur jalannya *input* ataupun *output* dari *port* PLC. Macam–macam bahasa pemrograman PLC menurut standart IEC61131-3 dapat dikelompokkan menjadi:

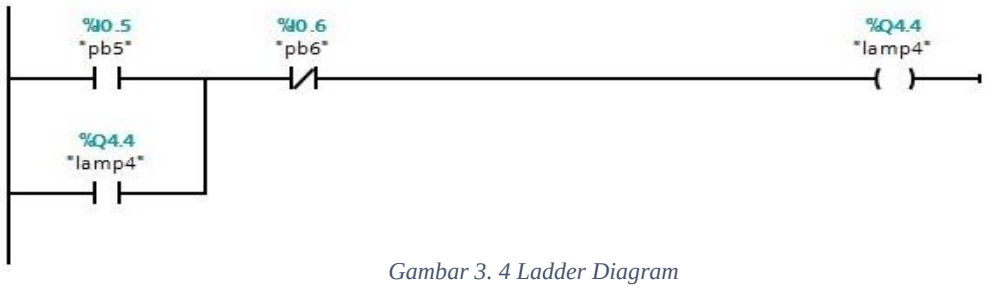
1. Representasi gambar atau simbol
    a. *Ladder Diagram* (LAD)
    b. Diagram blog fungsional (FBD)
    c. Urutan *chart* fungsi (sekuensial fungsional *chart* / FSC)
2. Tabel perintah
    a. Daftar instruksi (stetment list / STL)
    b. Teks terstruktur (ST)

Macam-macam Bahasa program yang ditetapkan oleh (International Electrotecnic Comminssion) IEC61131-3 adalah sebagai berikut:

3.2.1 *Ladder Diagram*

Ladder diagram adalah bahasa pemrograman yang yang dibuat dari persamaan fungsi logika dan fungsi-fungsi lain berupa pemrosesan data atau fungsi waktu dan pencacahan. *Ladder Diagram* terdiri dari susunan kontak-kontak dalam satu *group* perintah secara horizontal dari kiri ke kanan, dan terdiri dari banyak *group* perintah secara verikal. Contoh dari *Ladder Diagram* ini adalah: kontak *normaly open*, kontak *normaly close*, *output coil*, pemindahan data garis vertikal paling kiri dan paling kanan dia sumsikan sebagai fungsi tegangan, bila fungsi dari *group* perintah menghubungkan dua garis vertikal tersebut maka rangkaian perintah akan bekerja. Ladder Diagram memiliki beberapa macam dasar teori, yaitu :

a. Bahasa *Ladder Diagram*



Gambar 3. 4 Ladder Diagram

1. *Contact NO (Normally Open)*

Contact Normally Open adalah kondisi dimana saat kontak tersebut tidak ditekan/mati maka kontak tersebut dalam kondisi tidak terhubung/putus. Sebaliknya, saat kontak tersebut ditekan/bekerja maka kontak tersebut dalam kondisi terhubung. *Contact NO* ditunjukkan pada "pb5".

2. *Contact NC (Normally Close)*

Contact Normally Close adalah kondisi dimana saat kontak tersebut tidak ditekan/mati maka kontak tersebut dalam kondisi terhubung. Sebaliknya, saat kontak tersebut ditekan/bekerja maka kontak tersebut dalam kondisi tidak terhubung/putus. *Contact NC* ditunjukkan pada "pb6".

3. *Bus Bar*

Bus Bar adalah terminal yang akan dilalui arus listrik apabila dinyalakan , jadi akan melewati komponen yang ada pada *ladder* tersebut. *Bus Bar* sendiri ditunjukkan pada Garis vertikal hitam dan garis akhir pada "lamp4" disebelah kiri yang menunjukkan simbol kutub (+) menuju ke

kutub (-) ,yang berada disebelah kanan .

Pada dasarnya untuk membuat program *ladder diagram* adalah dengan menghubungkan *bus bar* sisi kiri ke *bus bar* sisi kanan sesuai dengan kondisi dan instruksi yang diinginkan untuk dikerjakan oleh unit PLC dalam menjalankan perintah ke mesin yang dikontrolnya. Jalur operasi kerja itu bisa dibagi dalam 2 bagian, yaitu :

- Sisi Kiri merupakan sisi pengkondisian, dimana biasanya terdiri dari rangkaian simbol kontak NO dan/atau NC, baik yang berasal dari *switch input* langsung ataupun dari *switch internal relay* hasil operasi perintah kerja dalam program yang bersangkutan.
- Sisi Kanan merupakan sisi perintah kerja, dimana biasanya berupa simbol *relay* dan bisa dipasang sebagai *output* langsung ataupun berupa *internal relay, timer, counter* dan operasi-operasi lainnya.

Jadi bilamana kondisi-kondisi yang ada di sisi kiri bisa dalam keadaan terhubung semua, maka arus listrik kutub (+) dari *bus bar* kiri akan mengalir dan menghidupkan operasi kerja di sisi kanan yang menempel dengan listrik kutub (-) di *bus bar* kanan.

4. *Input*

Input merupakan masukan dari luar PLC, baik dari *Switch, Sensor, Relay, Timer, Potentiometer* ataupun peralatan listrik yang lain, yang secara fisik ada di rangkaian listrik dari mesin, yang dihubungkan ke unit *Input* PLC, bisa berupa *digital input* maupun *analog input*. Biasanya

dilambangkan dengan kontak *NO* atau *NC* yang berfungsi sebagai syarat untuk berlakunya suatu operasi yng kita inginkan. Input ini biasanya dilambangkan dengan huruf *I (input*=Inggris) atau *E (eingang*=Jerman) atau *X* (Jepang) atau mungkin yang lain, tergantung dari jenis PLC dan bahasa pabrik pembuatnya. *Input* bisa ditunjukkan melalui pb5 dan pb6.

5. *Ouput*

Output merupakan hasil keluaran dari PLC, yang mana bisa berupa *digital output* maupun *analog output*, yang bisa langsung dihubungkan kerangkaian listrik yang lain di mesin tersebut melalui *unit Output PLC*. Output ini biasanya dilambangkan dengan huruf *O (output*=Inggris) atau *A (ausgang*=Jerman) atau *Y (Jepang)* atau mungkin yang lain, tergantung dari jenis PLC dan bahasa pabrik pembuatnya. *Output* disini ditunjukkan dengan gambar "*Coil*" simbol dibawah lamp4.

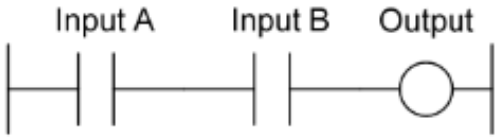
6. *Internal Relay*

Internal relay merupakan *relay* memori dari PLC itu sendiri, dimana bisa berupa *relay, timer, counter*, atau operasi-operasi logika yang lain. Seperti *Input* dan *Output*, simbol-simbol dari *internal relay* ini cukup beragam dan berbeda antara pabrikan yang satu dengan yang lain. Bukan hanya itu, jenis fungsinyapun juga bisa berbeda satu dengan yang lain. Jika kita lihat gambar 3.4 ada simbol %id dan %iQ ini merupakan alamat *memory* dari plc , dimana %id0.5 berarti kita menggunakan alamat input nomer 5 , dan %iQ4.4 berarti kita menggunakan alamat 4 nomer 4 .

b. Gerbang Logika

1. Logika *AND*

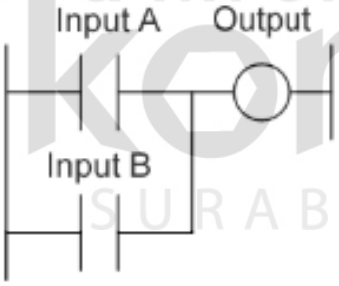
Gerbang *AND* pada sebuah Ladder Diagram diperlihatkan pada Gambar 3.5. Untuk menghasilkan *Output ON* (logika 1) maka *Input* A dan *Input* B harus dalam keadaan *ON*.



Gambar 3. 5 Logika AND

2. Logika *OR*

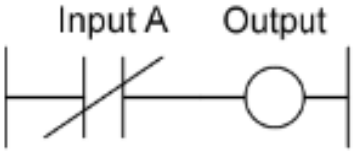
Sistem gerbang *OR* pada sebuah *Ladder Diagram* diperlihatkan pada Gambar 3.6. Untuk menghasilkan *Output ON* (logika 1) maka *Input* A atau *Input* B (atau keduanya)dalam keadaan *ON*.



Gambar 3. 6 Logika OR

3. Logika *NOT*

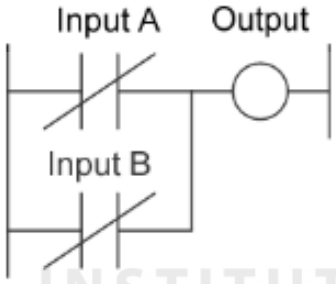
Sistem gerbang *NOT* pada sebuah diagram tangga diperlihatkan pada Gambar 3.7 *Output* akan bernilai *ON* justru jika *input* A sedang tidak aktif (*OFF* atau logika 0). *Input* A disini dikatakan sebagai kontak *normally closed (NC).*



Gambar 3. 7 Logika NOT

4. Logika *NAND*

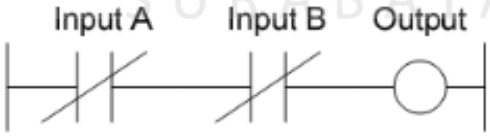
Gambar 3.8 memperlihatkan sebuah *Ladder Diagram* yang mengimplementasikan sebuah gerbang logika *NAND.*



Gambar 3. 8 Logika NAND

5. Logika *NOR*

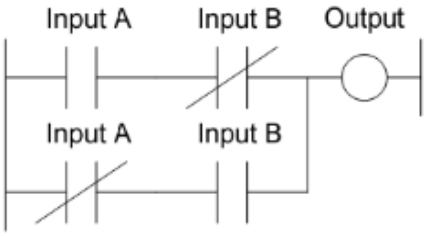
Gambar 3.9 memperlihatkan sebuah *Ladder Diagram* yang mengimplementasikan sebuah gerbang logika *NOR.*



Gambar 3. 9 Logika NOR

6. Logika *XOR*

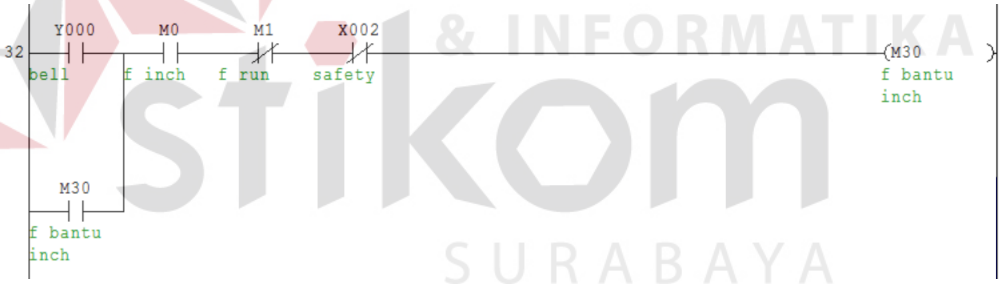
Gambar 3.10 memperlihatkan sebuah *Ladder Diagram* yang mengimplementasikan sebuah gerbang logika *XOR.*



Gambar 3. 10 Logika XOR

c. Fungsi *Latching* (Pengunci)

Seringkali terdapat situasi-situasi di mana output harus tetap berada dalam keadaan hidup meskipun *input* telah terputus. Istilah rangkaian *latching* (pengunci) dipergunakan untuk rangkaian-rangkaian yang mampu mempertahankan dirinya sendiri (*self-maintaining*), dalam artian bahwa setelah dihidupkan, rangkaian akan mempertahankan kondisi ini hingga *input* lainnya diterima.

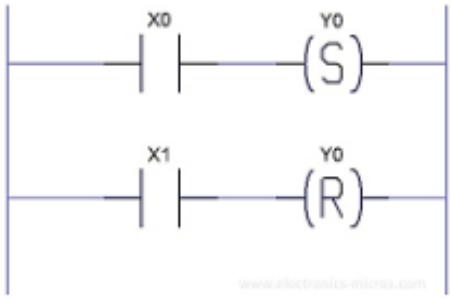


Gambar 3. 11 Latching (Penguncian)

SET – instruksi ini mengubah status pada sebuah bit menjadi *ON* ketika kondisi eksekusi juga bernilai *ON*. Apabila kondisi berubah menjadi menjadi *OFF*, status bit ini tetap *ON*.

RESET – berkebalikan dengan *SET*, instruksi ini akan mengubah status sebuah bit menjadi *OFF* ketika kondisi eksekusi *ON*. Ketika kondisi

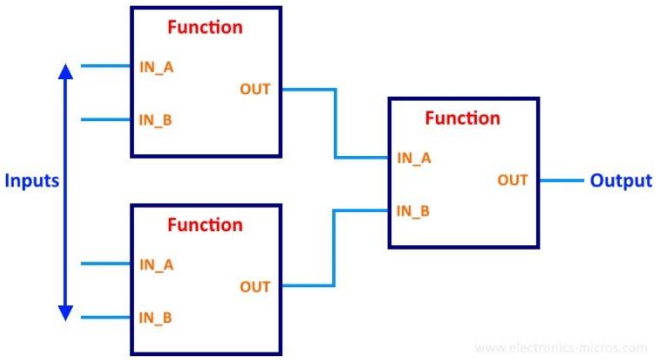
eksekusi berubah menjadi *OFF*, status bit tidak berubah (tetap *OFF*). Simbol instruksi *SET-RESET* ditunjukkan oleh Gambar 3.12.



Gambar 3. 12 Fungsi SET dan RESET

3.2.2 *Function Block Diagram* (FB/FBD)

Function Block diagram adalah suatu fungsi-fungsi logika yang disederhanakan dalam gambar blok dan dapat dihubungkan dalam suatu fungsi atau digabungkan dengan fungsi blok lain. Seperti SFC, FBD adalah bahasa grafis yang memungkinkan pemrograman dalam bahasa lain (tangga, daftar instruksi, atau teks terstruktur) yang akan bersarang di dalam FBD. Dalam FBD, program muncul sebagai blok elemen yang "dihubungkan" bersama-sama dengan cara yang menyerupai diagram rangkaian. FBD yang paling berguna dalam aplikasi yang melibatkan tingkat tinggi informasi/data *flow* antara komponen kontrol, seperti kontrol proses.



Gambar 3. 13 Function Block Diagram (FB/FBD)

3.2.3 Statement List (STL)

STL adalah bahasa program jenis tingkat rendah mirip dengan Bahasa *Assembly*. Intruksi yang dibuat berupa susunan sederhana menuju ke *operand* yang berupa alamat atau register.

```
0001 STEP 1                    (1)
0002  IF                       I0.1
0003  THEN   LOAD              V7
0004           TO              TP0
0005           WITH            HSC
0006         LOAD              V5
0007           TO              TP1
0008           WITH            HSC
0009         LOAD              V9
0010           TO              TP2

0011           WITH            HSC
0012
```

Gambar 3. 14 Statement List (STL)

3.2.3 *Structure Text* (ST) atau *Structure Language* (SCL)

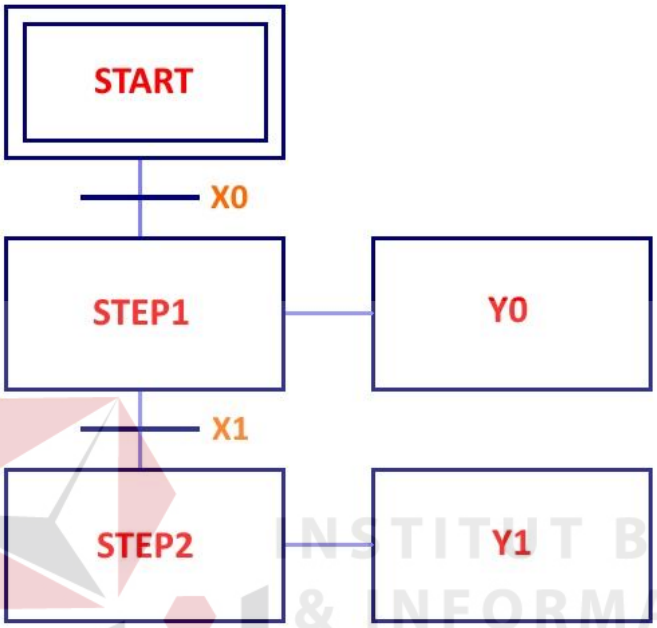
Teks terstruktur merupakan bahasa tingkat tinggi yang dapat memproses sistem logika ataupun alogaritma dan memungkinkan pemrosesan sistem lain. Perintah umumnya menggunakan *IF…THEN…ELSE, WHILE…DO, REPEAT…UNTIL* dll.

```
F8:10 := 0;
WHILE (N7:0 < 5) DO
   F8:10 := F8:10 + F8:[N7:0];
   N7:0 := N7:0 + 1;
END_WHILE;
```

Gambar 3. 15 Structure Text (ST) atau Structure Language (SCL)

3.2.4 Sequential Function Chart *(SFC)*

Bahasa Program yang dibuat dan disimpan dalam *chart*. Bagian-bagian *chart* memiliki fungsi urutan langkah, transisi dan percabangan. Tiap step memiliki status proses dan bisa terdiri dari struktur yang berurutan.



Gambar 3. 16 Sequential Function Chart (SFC)