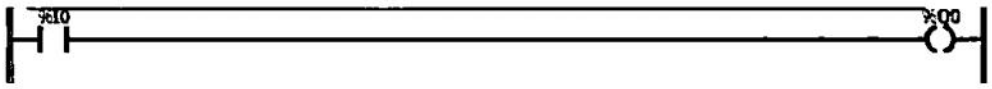
BAB IV

PEMBAHASAN

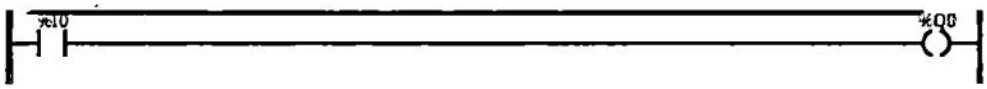
4.1 Program *Input-Output* Dasar

Berikut program *ladder input-output* dasar yang disimulasikan pada Classic Ladder.



Gambar 4.1. Kondisi Awal Program *Input-Output* Dasar

Pada kondisi awal, terlihat bahwa rangkaian tidak aktif. Kontak %I0 bernilai 0 atau tidak aktif, sehingga *coil* %Q0 bernilai 0 atau tidak aktif.

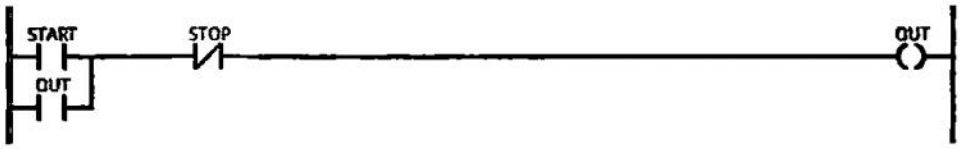


Gambar 4.2. Program *Input-Output* Dasar dengan %I0 yang Aktif

Gambar di atas menunjukkan bahwa ketika kontak %I0 diaktifkan atau bernilai 1, *coil* %Q0 akan aktif atau bernilai 1. Dengan demikian rangkaian tersebut menjadi aktif.

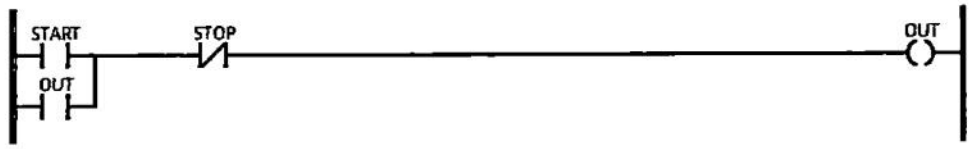
4.2 Program *Latching*

Berikut program *ladder latching* yang disimulasikan pada Classic Ladder.



Gambar 4.3. Kondisi Awal Program *Latching*

Pada kondisi awal, terlihat bahwa rangkaian tidak aktif. Tombol START juga tidak aktif atau bernilai 0.



Gambar 4.4. Program *Latching* dengan Tombol *START* yang Aktif

Gambar di atas menunjukkan bahwa ketika tombol *START* aktif atau bernilai 1, *coil OUT* akan aktif. Selain itu kontak *OUT* pengunci juga aktif atau bernilai 1.

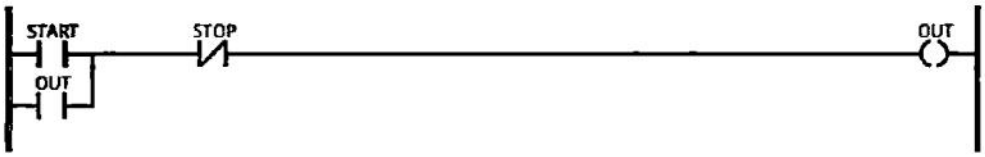
Pada program ini penulis menggunakan nama simbol berikut untuk menggantikan nama variabel alamat kontak dan *coil* yang digunakan.

Symbols names

Variable	Symbol name	Comment
%I0	START	PB
%I1	STOP	PB
%Q0	OUT	LED

Gambar 4.5. Nama-nama Simbol yang Digunakan

Adapun saat tombol *START* dinonaktifkan kembali atau bernilai 0, rangkaian akan tetap aktif. Hal ini dikarenakan kontak *OUT* pengunci tetap aktif dan menjaga keaktifan *coil OUT*. Gambar berikut menunjukkan hal ini.

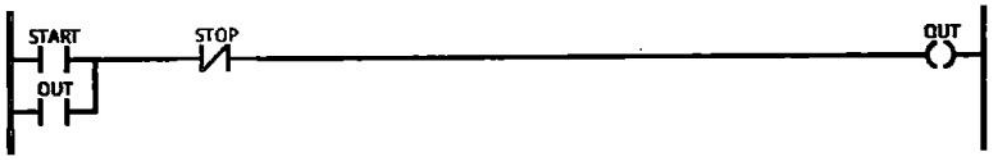


Gambar 4.6. Program *Latching* dengan Tombol *START* Inaktif atau Bernilai 0

Pada saat tombol *STOP* diaktifkan atau bernilai 1, rangkaian akan nonaktif. Hal ini terjadi karena tombol *STOP* merupakan kontak *normally close*. Gambar berikut menunjukkan hal ini.

Gambar 4.7. Program *Latching* dengan Tombol *STOP* Aktif atau Bernilai 1

Pada gambar berikut, tombol *STOP* dikembalikan kondisinya menjadi bernilai 1. Tidak terdapat perubahan pada rangkaian.

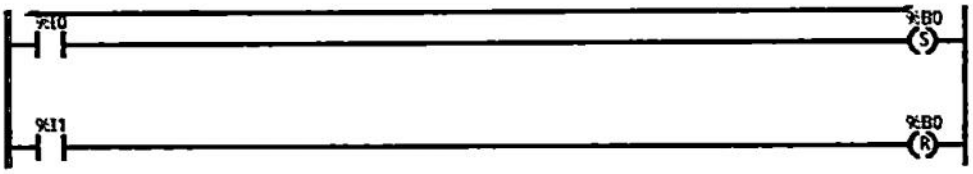


Gambar 4.8. Program *Latching* dengan Tombol *STOP* Inaktif atau Bernilai 0

Catatan: Tombol *START* dan tombol *STOP* dibuat untuk merepresentasikan *pushbutton START* dan *STOP*.

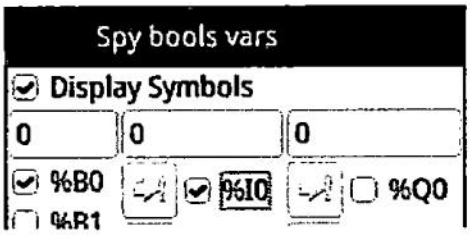
4.3 Program *SET - RESET*

Berikut contoh simulasi dari program *ladder* dengan *coil* khusus, yaitu *SET* dan *RESET*.



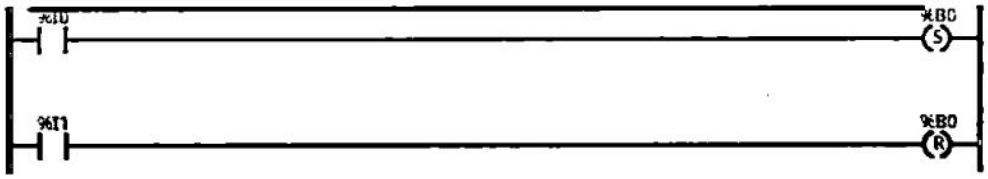
Gambar 4.9. Kondisi Awal Program *SET-RESET*

Pada kondisi awal, terlihat bahwa semua kontak tidak aktif dan semua *coil* juga aktif. Gambar tersebut pun menunjukkan rangakaian yang tidak aktif.



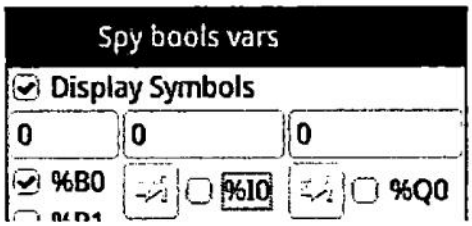
Gambar 4.10. Kondisi Variabel Setelah %I0 Diaktifkan

Selanjutnya dilakukan pengaktifan atau pengubahan nilai ke 1 pada kontak %I0. Hal ini menyebabkan *coil SET* %B0 aktif. Kondisi ini dapat dilihat pada gambar berikut.



Gambar 4.11. Program *SET-RESET* dengan %I0 Aktif

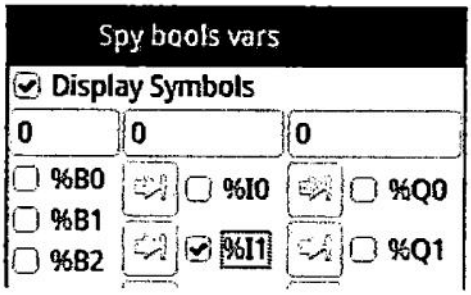
Dengan aktifnya *coil SET* %B0, memori internal %B0 akan bernilai 1. Nilai ini akan tetap, walaupun kontak %I0 dinonaktifkan. Hal ini ditunjukkan oleh gambar berikut.



Gambar 4.12. Kondisi Variabel Setelah %I0 Dinoaktifkan

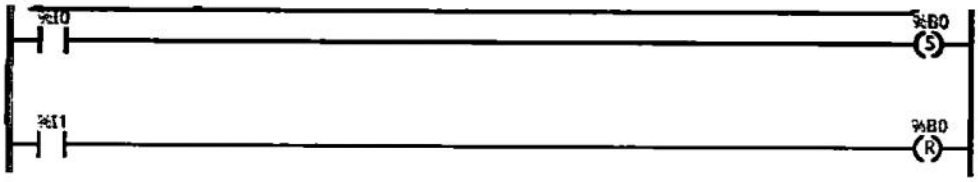
Setelah itu, dilakukan pengaktifan kontak %I1 yang tersambung dengan *coil RESET* %B0. Dengan pengaktifan *RESET* tersebut, memori internal %B0 akan bernilai 0 kembali. Nilai 0 ini akan tetap, walaupun kontak %I1 dinonaktifkan kembali.

Gambar berikut menunjukkan status dari variabel %B0 saat kontak %I1 diaktifkan atau bernilai 1.



Gambar 4.13. Kondisi Variabel Setelah %I1 Diaktifkan

Sementara gambar berikut menunjukkan kondisi pengaktfan kontak I=%I1 yang menyebabkan *coil RESET* %B0 aktif.



Gambar 4.14. Program *SET-RESET* dengan %I1 Aktif

Catatan: Kontak %I0 dan %I1 pada program tersebut biasa dianggap sebagai *pushbutton*.

4.4 Program Gerbang Logika Dasar

4.4.1 AND

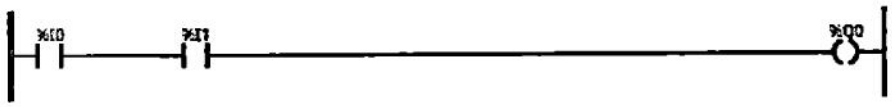
Program gerbang logika *AND* dibuat secara sederhana dengan melibatkan dua buah kontak *normally open* dan sebuah *coil*. Kontak yang terlibat adalah %I0 dan %I1, sedangkan *coil* yang terlibat adalah %Q0. Gambar berikut memperlihatkan kondisi awal dari rangkaian program ladder untuk gerbang logika *AND* pada Classic Ladder.

Gambar 4.15. Kondisi Awal Program *Ladder AND*

Kondisi awal tersebut memperlihatkan %I0 dan %I1 yang tidak aktif.

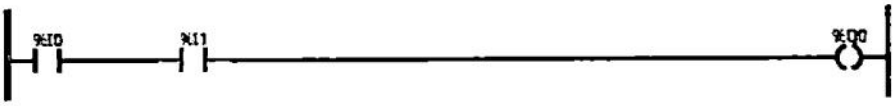
Output dari rangkaian tersebut bernilai 0 atau salah.

Ketika hanya %I0 yang diaktifkan, *output* dari rangkaian bernilai 0 atau salah. Hal ini dapat dilihat pada gambar berikut.



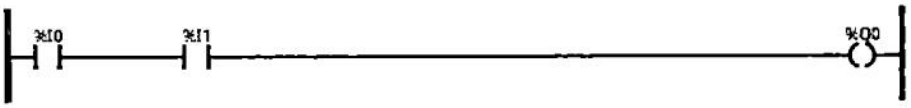
Gambar 4.16. Program Ladder *AND* dengan *Input* I0 Aktif

Saat %I1 saja yang diaktifkan, *output* dari rangkaian juga bernilai 0 atau salah. Hal ini dapat terlihat pada gambar berikut.



Gambar 4.17. Program *Ladder AND* dengan *Input* I1 Aktif

Saat kedua *input* baik %I0 maupun %I1 diaktifkan, *output* rangkaian bernilai 1 atau benar. Hal ini ditunjukkan oleh gambar berikut dimana *output* %Q0 aktif.



Gambar 4.18. Program *Ladder AND* dengan Kedua *Input* Aktif dan *Output* Aktif

Oleh karena itu, dapat dibuat sebuah tabel kebenaran berikut yang mewakili rangkaian tersebut.

Tabel 4.1. Tabel Kebenaran Program *AND*

%I0	%I1	%Q0
0	0	0
1	0	0
0	1	0
1	1	1

Tabel tersebut sesuai dengan tabel kebenaran dari gerbang logika *AND*.

4.4.2 *OR*

Program gerbang logika *OR* dibuat secara sederhana dengan melibatkan dua buah kontak *normally open* dan sebuah *coil*. Kontak yang terlibat adalah %I0 dan %I1, sedangkan *coil* yang terlibat adalah %Q0. Gambar berikut memperlihatkan kondisi awal dari rangkaian program *ladder* untuk gerbang logika *OR* pada Classic Ladder.

Gambar 4.19. Kondisi Awal Program *Ladder OR*

Pada kondisi awal tersebut kontak %I0 dan kontak %I1 tidak aktif atau bernilai 0. *Output* dari rangkaian juga bernilai 0. Hal ini ditunjukkan oleh tidak aktifnya *coil* %Q0.

Saat kontak %I0 aktif, *output* rangkaian akan bernilai 1. Hal ini ditunjukkan oleh aktifnya *coil* %Q0 sebagai berikut.

Gambar 4.20. Program *Ladder OR* dengan *Input* %I0 Aktif

Ketika kontak %I1 aktif, *output* rangkaian akan bernilai 1. Hal ini ditunjukkan oleh aktifnya *coil* %Q0 sebagai berikut.

Gambar 4.21. Program *Ladder OR* dengan *Input* %I1 Aktif

Saat kedua kontak aktif (%I0 dan %I1), *output* rangkaian akan bernilai 1. Hal ini ditunjukkan oleh aktifnya *coil* %Q0 sebagai berikut.

Gambar 4.22. Program *Ladder OR* dengan Kedua *Input* Aktif

Dengan demikian dapat dibuat sebuah tabel kebenaran sebagai berikut.

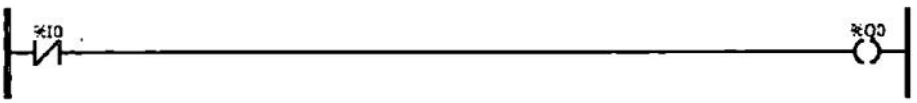
Tabel 4.2. Tabel Kebenaran Program *Ladder OR*

%I0	%I1	%Q0
0	0	0
1	0	1
0	1	1
1	1	1

Tabel tersebut menunjukkan bahwa program *ladder* tersebut sesuai dengan logika gerbang *OR*.

4.4.3 *NOT*

Program gerbang logika *NOT* melibatkan satu *input* dan satu *output*. *Input* yang digunakan berupa kontak *normally close* yang beralamat %I0 sedangkan *output* yang digunakan berupa *coil* yang beralamat %Q0. Gambar berikut menunjukkan kondisi awal dari rangkaian program.

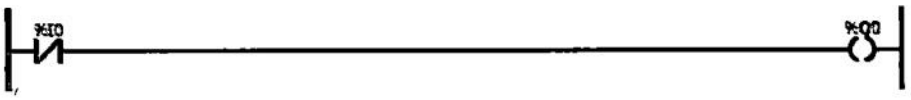


Gambar 4.23. Program *NOT* dengan *Input* Bernilai 0

Pada gambar tersebut, kontak %I0 bernilai 0. Namun dikarenakan kontak tersebut *normally close*, jadi *output* akan aktif. Dapat terlihat bahwa *coil* %Q0 aktif.

Sementara gambar berikut menunjukkan kondisi saat kontak %I0 diberi

nilai 1. *Output* rangkaian akan bernilai 0 atau tidak aktif. Hal ini ditunjukkan oleh tidak aktifnya *coil* %Q0.



Gambar 4.24. Program *NOT* dengan *Input* Bernilai 1

Oleh karena itu dapat dibuat sebuah tabel kebenaran dari program *ladder* tersebut sebagai berikut.

Tabel 4.3. Tabel Kebenaran Program *Ladder NOT*

%I0	%Q0
0	1
1	0

Tabel tersebut menunjukkan bahwa program *ladder* tersebut relevan dengan logika dari gerbang *NOT*.

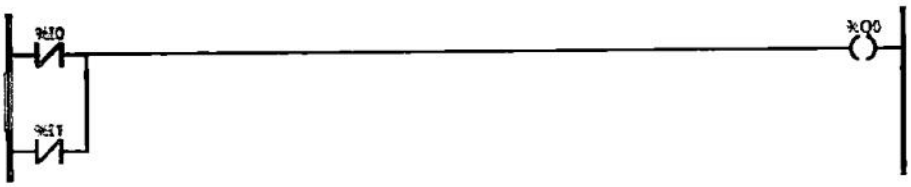
4.4.4 *AND NOT*

Program gerbang logika *AND NOT* atau *NOT AND* melibatkan dua *input* dan satu *output*. *Input* yang digunakan adalah kontak *normally close* yang beralamat %I0 dan %I1, sedangkan *output* yang digunakan adalah *coil* beralamat %Q0. Gambar-gambar berikut menunjukkan simulasi program tersebut yang dilakukan pada Classic Ladder.

Gambar 4.25. Kondisi Awal Program *NOT AND*

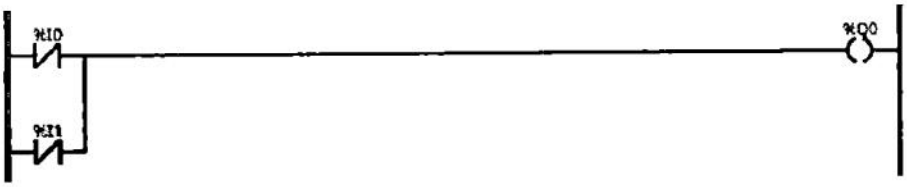
Pada kondisi awal di mana kedua *input*(kontak %I0 dan %I1) memiliki nilai 0, *output* rangkaian bernilai 1 atau aktif. Hal ini ditunjukkan oleh aktifnya *coil* %Q0.

Saat kontak %I0 diberi nilai 1, *output* rangkaian akan bernilai 1 atau aktif. Hal ini ditunjukkan oleh aktifnya *coil* %Q0.



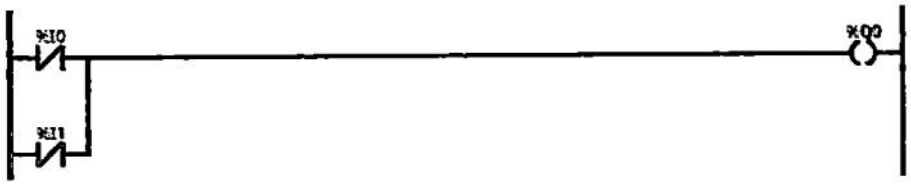
Gambar 4.26. Program *NOT AND* dengan *Input* %I0 Bernilai 1

Saat kontak %I1 diberi nilai 1, *output* rangkaian akan bernilai 1 atau aktif. Hal ini ditunjukkan oleh *coil* %Q0 yang aktif pada gambar berikut.



Gambar 4.27. Program *NOT AND* dengan *Input* %I1 Bernilai 1

Adapun saat kedua kontak bernilai 1, *output* rangkaian akan bernilai 0 atau tidak aktif. Hal ini ditunjukkan oleh *coil* %Q0 yang aktif pada gambar berikut.



Gambar 4.28. Program *NOT AND* dengan Kedua *Input*(%I0 dan %I1) Bernilai 1

Oleh karena itu dapat dibuat sebuah tabel kebenaran dari program tersebut sebagai berikut.

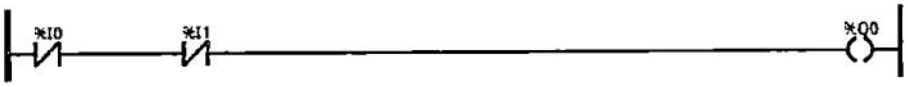
Tabel 4.4. Tabel Kebenaran Program *Ladder NOT AND*

%I0	%I1	%Q0
0	0	1
1	0	1
0	1	1
1	1	0

Tabel tersebut menunjukkan bahwa program tersebut relevan terhadap logika gerbang *NOT AND*.

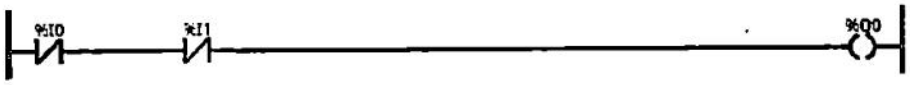
4.4.5 *OR NOT*

Program gerbang logika *OR NOT* atau *NOT OR* melibatkan dua *input* dan satu *output* yang disusun seri. *Input* yang digunakan adalah kontak *normally close* beralamat %I0 dan %I1. *Output* yang digunakan adalah *coil* beralamat %Q0. Berikut gambar-gambar yang menunjukkan simulasi dari program ini.



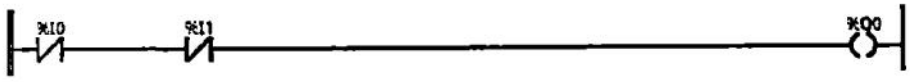
Gambar 4.29. Kondisi Awal Program *NOT OR*

Pada kondisi awal di mana semua *input* (kontak %I0 dan %I1) bernilai 0, *output* dari rangkaian bernilai 1. Hal ini terjadi dikarenakan kontak jenis *normally close*. Aktifnya *output* ditandai dengan aktifnya *coil* %Q0.



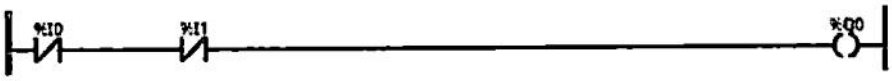
Gambar 4.30. Program *NOT OR* dengan *Input* %I0 Bernilai 1

Pada gambar tersebut terlihat bahwa saat kontak %I0 bernilai 1, *output* rangkaian akan bernilai 0. Hal ini ditunjukkan oleh tidak aktifnya *coil* %Q0.



Gambar 4.31. Program *NOT OR* dengan *Input* %I1 Bernilai 1

Pada gambar tersebut dapat dilihat bahwa saat kontak %I1 diberi nilai 1, *output* rangkaian akan bernilai 0 atau tidak aktif. Hal ini ditunjukkan oleh tidak aktifnya *coil* %Q0.



Gambar 4.32. Program *NOT OR* dengan Kedua *Input* (%I0 an %I1) Bernilai 0

Pada gambar tersebut terlihat bahwa saat kedua kontak (%I0 dan %I1) bernilai 1, *output* rangkaian akan bernilai 0 atau tidak aktif. Hal ini ditunjukkan oleh *coil* %Q0 yang tidak aktif.

Dengan demikian dapat dibuat sebuah tabel kebenaran dari program tersebut sebagai berikut.

Tabel 4.5. Tabel Kebenaran Program *Ladder NOT OR*

%I0	%I1	%Q0
0	0	1
1	0	0
0	1	0
1	1	0

Tabel tersebut menunjukkan bahwa program tersebut relevan dengan logika gerbang *NOT OR*.

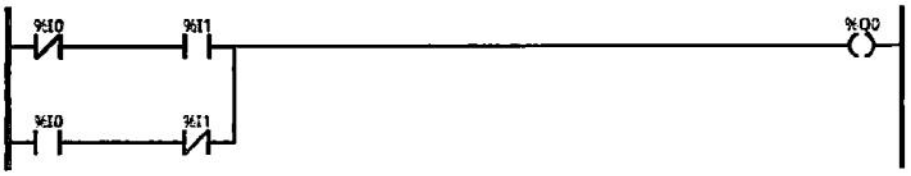
4.4.6 EXOR

Program gerbang logika EXOR melibatkan 4 buah kontak sebagai *input* dan 1 buah *coil* sebagai *output*. 2 buah kontak dihubungkan secara seri yang terdiri dari 1 kontak normally close beralamat %I0 dan 1 kontak normally open beralamat %I1. Sementara 2 buah kontak lainnya memiliki alamat yang sama tetapi dengan jenis yang berkebalikan. 2 kontak pertama dan dua kontak ke-2 dihubungkan secara paralel ke *coil output*.

Gambar 4.33. Kondisi Awal Program *Ladder* EXOR

Pada kondisi awal tersebut, semua kontak *input* bernilai 1. Hal ini berarti bahwa kontak-kontak tersebut masih berada pada kondisi normalnya. Adapun

terlihat bahwa *output* dari rangkaian tersebut bernilai 0 atau tidak aktif.

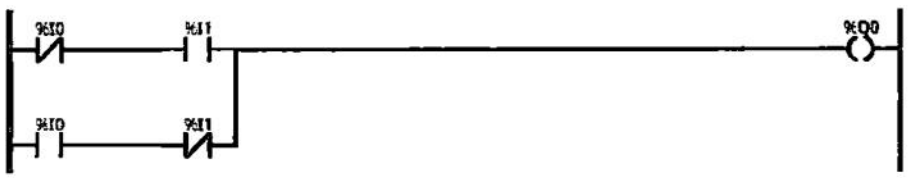


Gambar 4.34. Program *Ladder* EXOR dengan *Input* %I0 Bernilai 1

Pada gambar di atas terlihat bahwa *output* rangkaian bernilai 1 atau aktif. Hal ini dikarenakan terdapat *input* dari kontak %I0 pada bagian bawah yang aktif.

Gambar 4.35. Program *Ladder* EXOR dengan *Input* %I1 Bernilai 1

Pada gambar di atas dapat dilihat bahwa *output* rangkaian bernilai 1 atau aktif. Hal ini dikarenakan kontak %I1 yang bernilai 1 dan mengaktifkan *input*.



Gambar 4.36. Program *Ladder* EXOR dengan *Input* %I0 dan %I1 Bernilai 1

Pada gambar di atas terlihat bahwa *output* rangkaian bernilai 0. Hal ini dikarenakan kedua kontak yang sama-sama bernilai 1 dan mengubah kontak-kontak *Normally Close* menjadi terbuka, sehingga tidak terdapat *input*.

Dengan demikian dapat dibuat sebuah tabel kebenaran dari program *ladder* tersebut sebagai berikut.

Tabel 4.6. Tabel Kebenaran Program *Ladder* EXOR

%I0	%I1	%Q0
0	0	0
1	0	1
0	1	1
1	1	0

Tabel tersebut menunjukkan bahwa program *ladder* tersebut relevan dengan logika gerbang EXOR.

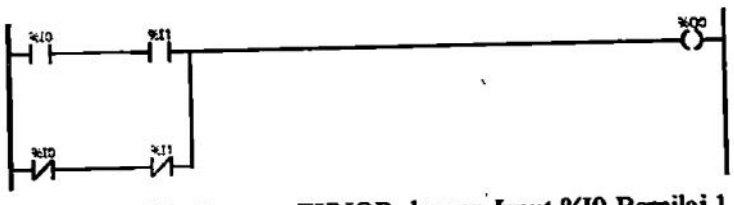
4.4.7 EXNOR

Program gerbang logika EXNOR melibatkan 4 buah kontak dan sebuah *coil*. 2 kontak *Normally Open* dihubungkan seri pada satu baris dan 2 kontak *Normally Closed* dihubungkan pada baris lainnya. Kedua baris tersebut dihubungkan secara paralel ke *output*.

Gambar 4.37. Kondisi Awal Program *Ladder* EXNOR

Kondisi awal tersebut memperlihatkan *output* yang aktif pada saat semua kontak bernilai 0. Kedua kontak *normally close* adalah yang berperan dalam hal ini.

Pada gambar berikut terdapat sebuah alamat yang diubah nilainya menjadi 1. Alamat tersebut adalah %I0.



Gambar 4.38. Program EXNOR dengan *Input* %I0 Bernilai 1

Tabel 4.7. Tabel Kebenaran Program *Ladder* EXNOR

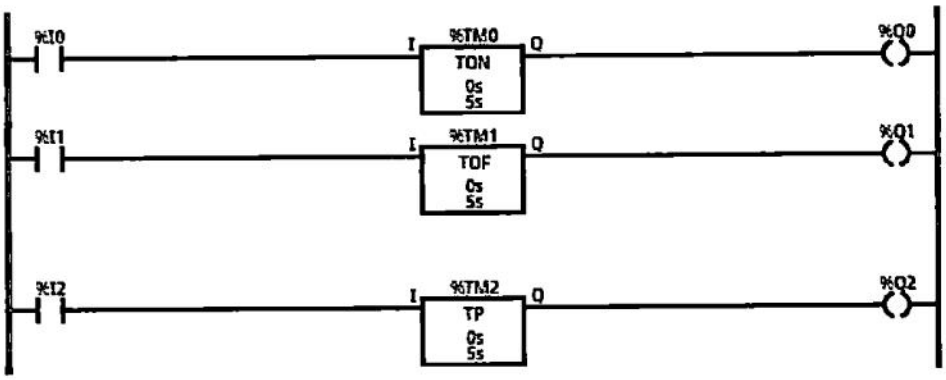
%I0	%I1	%Q0
0	0	1
1	0	0
0	1	0
1	1	1

Tabel tersebut menunjukkan bahwa program *ladder* tersebut relevan dengan logika gerbang EXNOR.

4.5 Program *Timer*

Program *timer* ini mencakup semua jenis/mode *timer* yang terdapat pada Classic Ladder. *Timer* pada Classic Ladder terdiri dari TON(*Timer On Delay*), TOF(*Timer Off Delay*), dan TP(*Timer Pulse*). Adapun *timer* yang digunakan adalah *timer* versi baru, yaitu *timer* IEC.

Timer-timer tersebut ditempatkan dalam satu *rung* yang sama. Masing-masing *timer* diatur untuk memiliki *preset* 5 detik. Setiap *timer* memiliki kontak *input* dan *coil*/kumparan *output* sendiri yang berbeda. Berikut adalah gambar kondisi awal dari *timer-timer* tersebut.

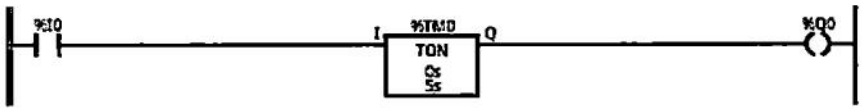


Gambar 4.41. Kondisi awal dari *Timer-timer* yang Digunakan

Di bawah ini akan dibahas penggunaan setiap jenis *timer* pada Classic Ladder. Simulasi juga dilakukan pada *timer-timer* ini.

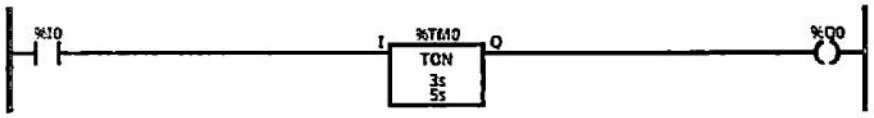
4.5.1 TON

TON(*Timer On Delay*) merupakan *timer* yang outputnya tertunda selama waktu tertentu. Pada gambar-gambar berikut terdapat simulasi dari program *ladder* yang melibatkan TON.



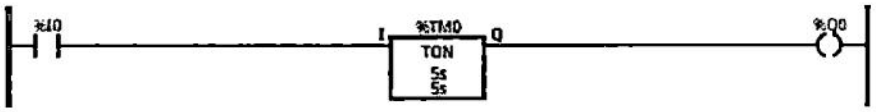
Gambar 4.42. Kondisi Awal dari TON

Ketika kontak %I0 diaktifkan, TON akan memulai menghitung waktu dari angka 0. Gambar berikut menunjukkan waktu TON yang berjalan.



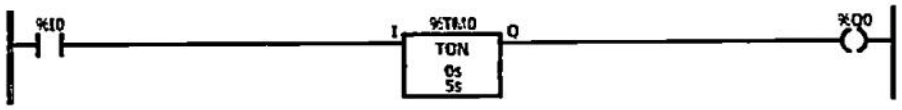
Gambar 4.43. TON dengan Waktu yang Berjalan

Saat waktu dari TON tersebut mencapai nilai *preset*-nya, *output* dari TON akan aktif. Dengan demikian, *coil* %Q0 akan aktif. Hal ini ditunjukkan oleh gambar berikut.



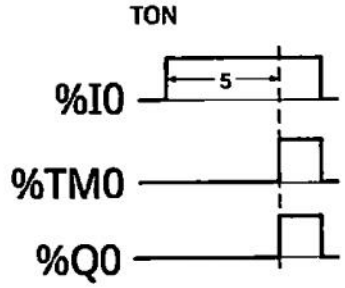
Gambar 4.44. TON dengan Waktu Mencapai *Preset*

Ketika kontak %I0 dinonaktifkan, maka *input* TON juga menjadi tidak aktif dan TON mengalami *reset*. Waktu aktual TON kembali kembali ke 0 dan output TON menjadi tidak aktif, sehingga *coil* Q0 juga menjadi tidak aktif. Hal ini menunjukkan bahwa *input* TON tidak termasuk dalam kategori *one-shot*. Berikut gambar yang menunjukkan hal tersebut.



Gambar 4.45. TON yang Kembali ke Kondisi Awal, Setelah Mengalami *Reset*

Dengan demikian dapat dibuat sebuah *timing diagram* sebagai berikut untuk program *timer on delay* tersebut.

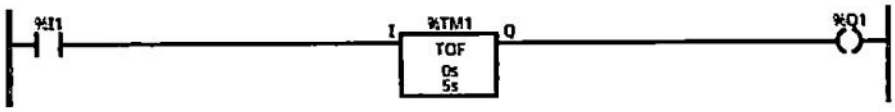


Gambar 4.46. *Timing Diagram* Program TON

Timing diagram tersebut menunjukkan bahwa TON tersebut sesuai dengan teori tentang *timer on delay*.

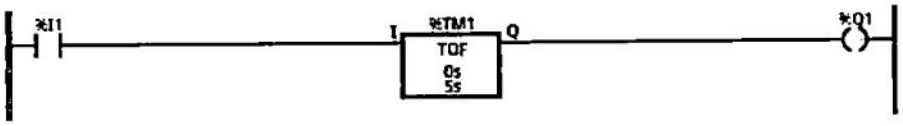
4.5.2 TOF

TOF(*Timer Off Delay*) merupakan *timer* yang menghitung waktu penundaan penonaktifan dari sebuah *output*, setelah sebelumnya aktif. Gambar-gambar berikut menunjukkan penggunaan TOF pada Classic Ladder.



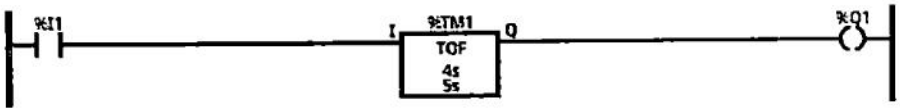
Gambar 4.47. Kondisi Awal Program *Ladder* TOF

Pada gambar di atas, terlihat kondisi awal dari program yang melibatkan TOF. Sementara gambar berikut menunjukkan TOF yang diberi *input* dari kontak %I1.



Gambar 4.48. TOF dengan *Input ON* dan *Output ON*

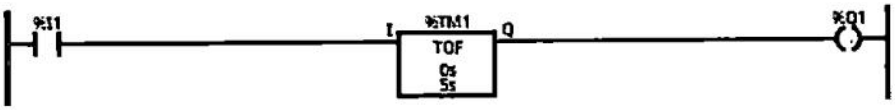
Dapat dilihat pada gambar di atas bahwa saat *input* TOF aktif, *output* TOF akan aktif secara langsung. Pada tahap selanjutnya, *input* TOF dinonaktifkan. Berikut gambar yang dapat menunjukkan fenomena yang terjadi pada tahap selanjutnya.



Gambar 4.49. TOF dengan *Input OFF* dan *Output ON*

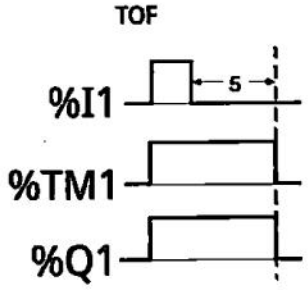
Gambar di atas menunjukkan bahwa saat *input* TOF dinonaktifkan, *output* TOF tetap aktif. Kemudian waktu akan berjalan mundur dari *preset* ke 0.

Ketika waktu aktual TOF mencapai angka 0, *output* TOF akan menjadi tidak aktif. Hal ini ditunjukkan oleh gambar berikut.



Gambar 4.50. TOF dengan *Input OFF* dan *Output OFF*, Setelah Waktu Selesai

Dengan demikian dapat dibuat sebuah *timing diagram* untuk program *timer off delay* tersebut sebagai berikut.

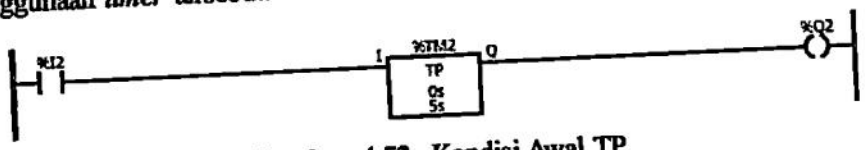


Gambar 4.51. *Timing Diagram* Program TOF

Timing diagram tersebut menunjukkan bahwa TOF tersebut sesuai dengan teori tentang *timer off delay*.

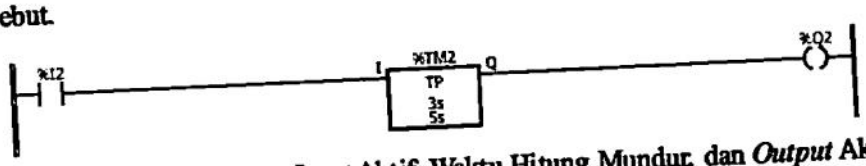
4.5.3 TP

TP(*Pulse Timer*) merupakan *timer* yang bekerja dengan *input* yang sejenis dengan *counter*, yaitu *one shot*. *Timer* ini berfungsi untuk mengaktifkan *output* dalam rentang waktu tertentu. Ini berarti bahwa *output timer* ini akan aktif sejak pulsa diterima sampai waktu habis. Gambar-gambar berikut menunjukkan penggunaan *timer* tersebut.



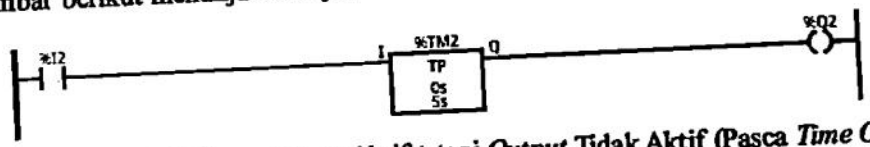
Gambar 4.52. Kondisi Awal TP

Pada kondisi awal, TP memiliki nilai aktual 0 detik. Namun, saat *input* diaktifkan dengan kontak %I2 sebagai pemicunya, nilai aktual TP menjadi 5 dan TP memulai hitung mundur ke angka 0. Pada saat yang sama, *output* dari TP aktif dan mengaktifkan *coil* %Q2. Berikut gambar yang dapat menunjukkan hal tersebut.



Gambar 4.53. TP dengan *Input* Aktif, Waktu Hitung Mundur, dan *Output* Aktif

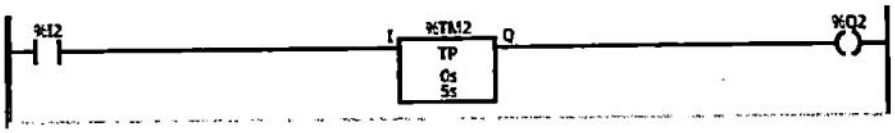
Saat waktu mencapai angka 0 detik, *output* TP akan dinonaktifkan. Gambar berikut menunjukkannya.



Gambar 4.54. TP dengan *Input* Aktif tetapi *Output* Tidak Aktif (Pasca *Time Out*)

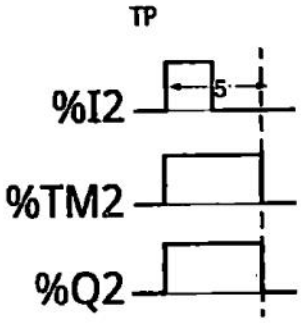
Sementara gambar berikut menunjukkan bahwa *input* yang dinonaktifkan

kembali, tidak berdampak apapun terhadap kondisi TP. Hal ini akan sama saja walaupun *input* dinonaktifkan kembali saat waktu hitung mundur belum mencapai nilai 0.



Gambar 4.55. TP dengan *Input* Tidak Aktif dan *Output* Tidak Aktif

Dengan demikian dapat dibuat sebuah *timing diagram* untuk program *timer pulse* tersebut sebagai berikut.



Gambar 4.56. *Timing Diagram* Program TP

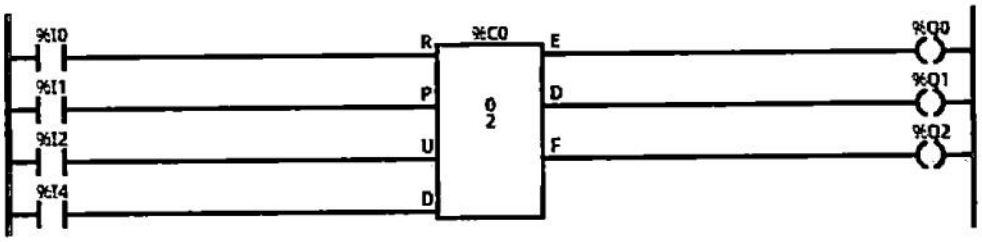
Timing diagram tersebut menunjukkan bahwa TP tersebut sesuai dengan teori tentang *timer pulse*.

4.6 Program *Counter*

Program *counter* ini mencakup semua *input* dan *output* dari sebuah *counter* pada Classic Ladder. *Input* yang dimaksudkan adalah R(Mengatur Ulang/*Reset*), P(*Preset*), U(Hitung Maju/*Up Count*), D(Hitung Mundur/*Down Count*). *Output* yang dimaksudkan adalah E(*Underflow*), D(Selesai/*Done*), dan F(*Overflow*).

Counter yang penulis gunakan memiliki angka *preset* 2. Masing-masing *input* sudah tersambung dengan kontak sebagai pemicunya, sedangkan

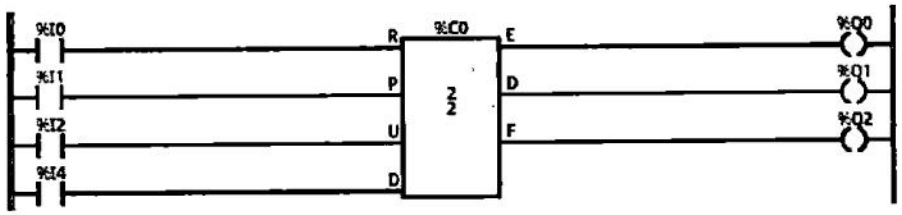
masing-masing *output* tersambung dengan *coil*/kumparan. Adapun gambar berikut merupakan kondisi awal dari *counter* dalam *software* Classic Ladder yang akan dibahas fungsi-fungsi *input* dan *output*-nya.



Gambar 4.57. Rangkaian *Ladder* Program *Counter* (Kondisi Awal)

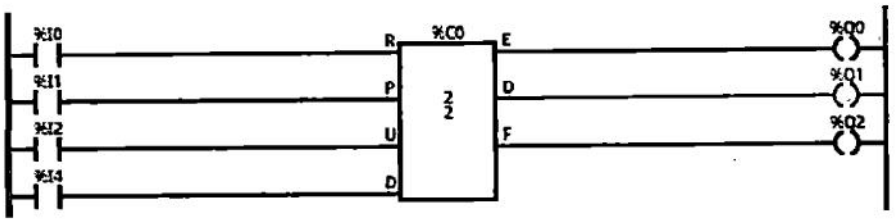
4.6.1 *PRESET*

Preset merupakan nilai patokan dari sebuah *counter* di mana *counter* tersebut menyelesaikan perhitungannya. Dalam artian lain, ketika nilai *preset* tercapai, maka *output* D(*Done*) yang menandakan *counter* telah selesai menghitung akan aktif. Saat *input* P(*Preset*) pada sebuah *counter* diberi masukan atau diaktifkan, maka nilai aktual *counter* tersebut akan menjadi nilai *preset*-nya. Hal ini terlepas dari nilai yang dimiliki oleh *counter* tersebut sebelumnya. Simulasi pada *software* Classic Ladder berikut menunjukkan hal tersebut.



Gambar 4.58. *Counter* dengan *Input* P(*Preset*) Aktif yang Mengaktifkan *Output* D dan *Output* Q1.

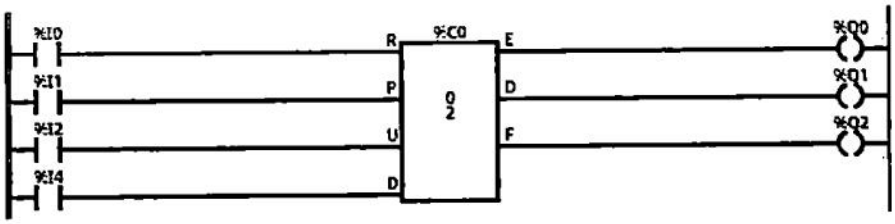
Walaupun setelah itu *input* pada P(*Preset*) dinonaktifkan kembali, nilai aktual dari *counter* tersebut tetap akan sama dengan *preset*-nya. Jadi *input* ini dapat dikategorikan sebagai *input one shot*.



Gambar 4.59. *Counter* dengan *Input* P(*Preset*) Nonaktif Kembali, *Output* D dan *Output* Q1 Tetap Aktif

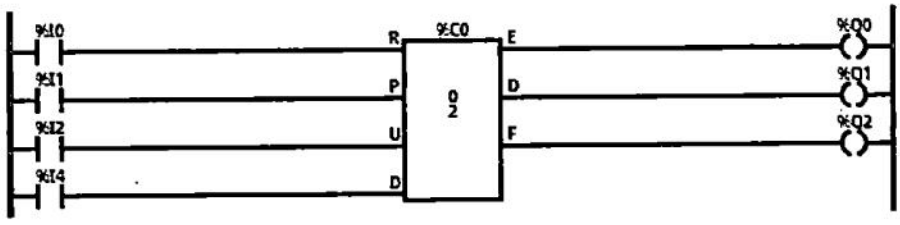
4.6.2 *RESET*

Reset merupakan tindakan atau operasi pengatural ulang dan pengembalian ke kondisi awal *counter*. Pada gambar berikut dapat dilihat simulasi pengaktifan *input* R(*Reset*) pada sebuah *counter*.



Gambar 4.60. *Counter* dengan *Input* R(*Reset*) Aktif

Pada gambar tersebut, terlihat bahwa saat *input* R(*Reset*) diaktifkan, nilai aktual dari *counter* tersebut secara langsung kembali ke kondisi awalnya atau nilai 0. Hal ini terlepas dari kondisi aktual nilai *counter* sebelumnya. Kontak I0 menjadi pemicu *input* R tersebut. Pada gambar berikut, dapat dilihat penonaktifan *input* R setelah diaktifkan.



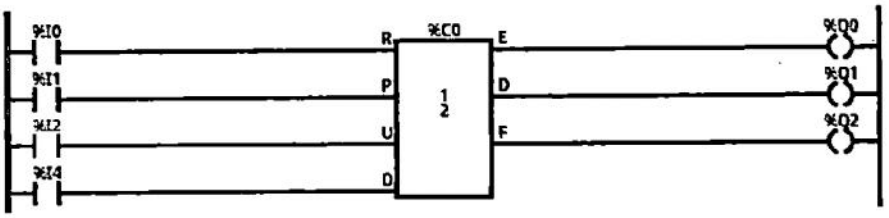
Gambar 4.61. *Counter* dengan *Input* R(*Reset*) yang Dinonaktifkan Kembali

Gambar tersebut memperlihatkan bahwa *input* R yang dinonaktifkan

kembali pasca pengaktifannya, tidak memberikan efek apapun terhadap kondisi *counter*. Jadi dapat dikatakan bahwa *input* R(*Reset*) termasuk dalam kategori *one shot*.

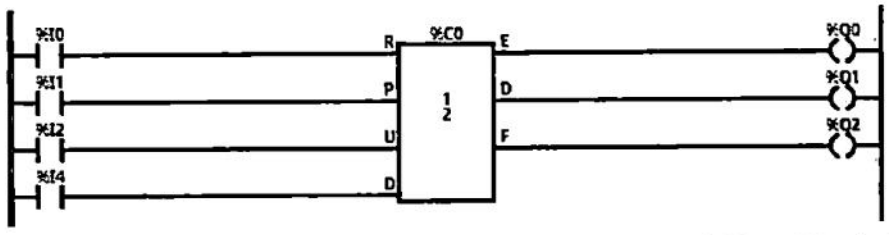
4.6.3 *UP COUNT*

Up count merupakan penghitungan normal, yaitu dari 0 ke angka yang nilainya lebih besar. Sebuah *counter* dalam Classic Ladder memiliki *input* U(*Up Count*). *Input* ini berfungsi untuk memberikan pulsa *rising edge* yang menambah 1 nilai ke nilai aktual *counter* setiap kali diaktifkan. Gambar berikut merupakan simulasi dari penggunaan *input* U(*Up Count*) pada sebuah *counter* dalam Classic Ladder.



Gambar 4.62. *Counter* dengan *Input* U(*Up Count*) Aktif

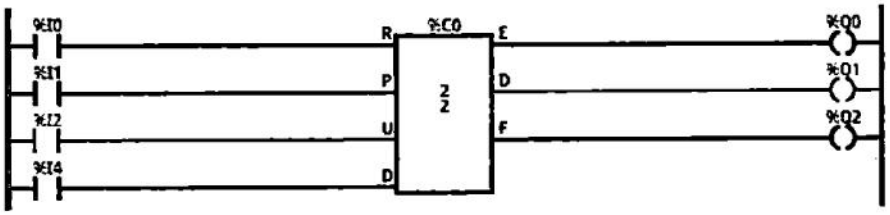
Pada gambar tersebut dapat dilihat bahwa ketika *input* U aktif, nilai aktual dari *counter* tersebut mengalami kenaikan 1 angka. Langkah simulasi selanjutnya terdapat pada gambar berikut.



Gambar 4.63. *Counter* dengan *Input* U(*Up Count*) Dinonaktifkan Kembali

Pada gambar tersebut terlihat bahwa setelah *input* U kembali dinonaktifkan, tidak terjadi perubahan pada nilai aktual *counter*. Hal ini berarti

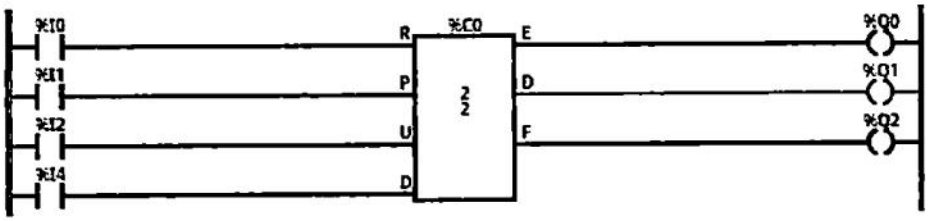
bahwa pada *input* tersebut berlaku *one shot*. Langkah selanjutnya penulis mengaktifkan *input* tersebut untuk kali kedua.



Gambar 4.64. *Counter* dengan *Input* U(*Up Count*) Diaktifkan Kedua kalinya

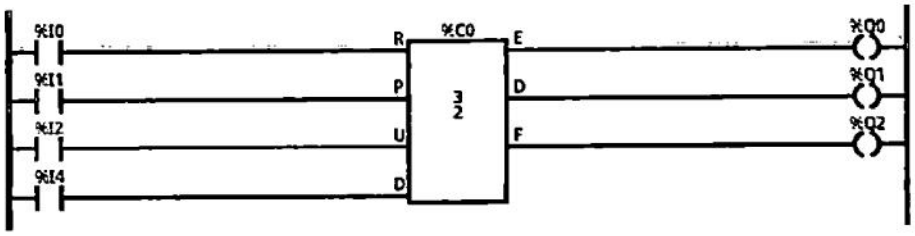
Gambar tersebut memperlihatkan kondisi *output* D(*Done*) yang aktif karena dipicu oleh aktivasi kedua dari *input* U. Alasan utama terjadinya hal ini adalah nilai aktual *counter* tersebut yang naik ke angka 2. Hal ini berarti bahwa nilai aktual *counter* sama dengan nilai *preset*-nya. Dengan kata lain penghitungan *counter* tersebut telah dinyatakan memenuhi syarat selesai.

Adapun pada gambar berikut, terjadi hal yang sama dengan tahap sebelumnya. Ketika *input* U dinonaktifkan kembali, kondisi *counter* tetap.



Gambar 4.65. *Counter* Pasca Penghitungan Selesai dengan *Input* U(*Up Count*)

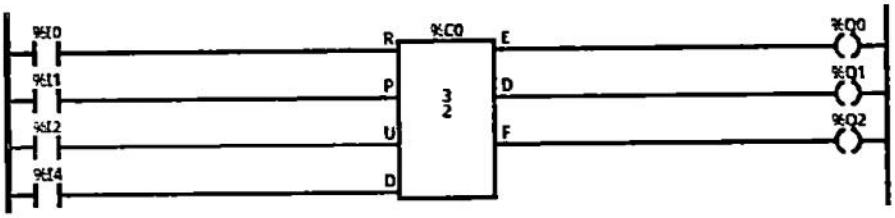
Namun, ketika *input* U kembali diaktifkan dengan kontak %I2, *output* F kembali tidak aktif.



Gambar 4.66. Kondisi *Counter* Ketika *Input* U Diaktifkan Kembali

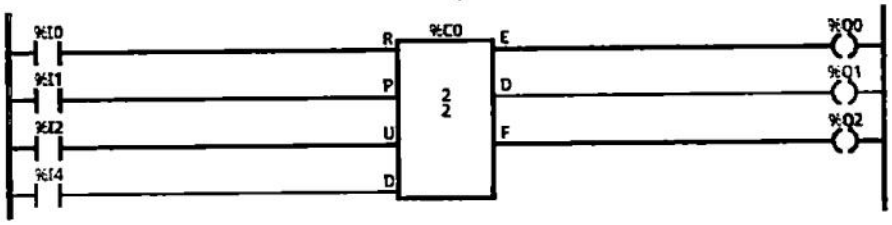
4.6.4 *DOWN COUNT*

Down count merupakan penghitungan mundur ke nilai 0 dari nilai yang lebih besar. Hal ini juga dapat dikatakan sebagai *falling edge*. Gambar-gambar simulasi berikut menjelaskan tentang penggunaan *input* D(*Down Count*) pada sebuah *counter* dalam Classic Ladder. Kondisi awal nilai aktual *counter* yang penulis gunakan di sini adalah 3, sedangkan nilai *preset* tetap 2.



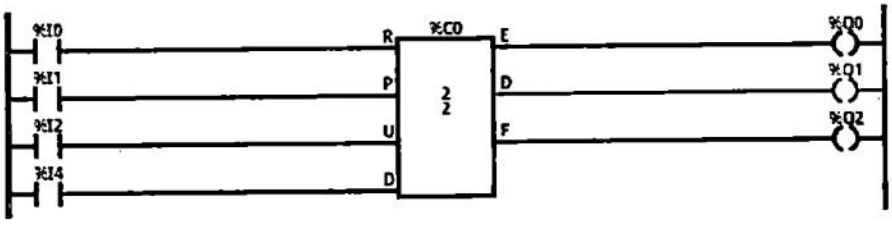
Gambar 4.67. Kondisi Awal Sebelum *Input* D(*Down Count*) Aktif

Pada tahap selanjutnya, *input* D(*Down Count*) diaktifkan. Berikut gambar dari pengaktifan *input* D tersebut.



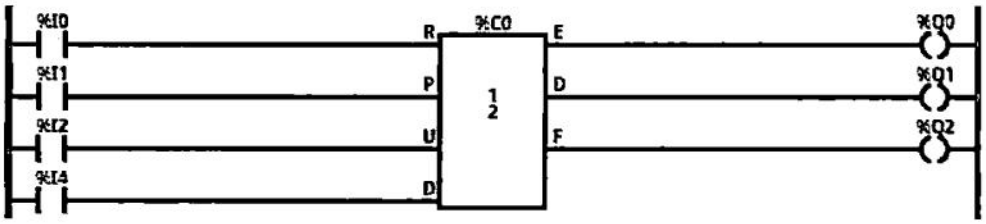
Gambar 4.68. *Counter* dengan *Input* D(*Down Count*) Aktif

Dapat dilihat bahwa ketika *input* D(*Down Count*) diaktifkan, nilai aktual dari *counter* tersebut berkurang 1 angka menjadi 2. Adapun *output* D(*Done*) aktif karena nilai aktual *counter* sama dengan nilai *preset*-nya.



Gambar 4.69. *Counter* dengan *Input* D(*Down Count*) Dinonaktifkan Kembali

Dari gambar tersebut dapat terlihat bahwa *input* D(*Down Count*) juga termasuk *input* jenis *one shot*. Jika nilai *counter* diturunkan dengan mengaktifkan *input* U kembali menggunakan kontak %I4, maka *output* D dan *coil* Q1 kembali tidak aktif atau bernilai salah.

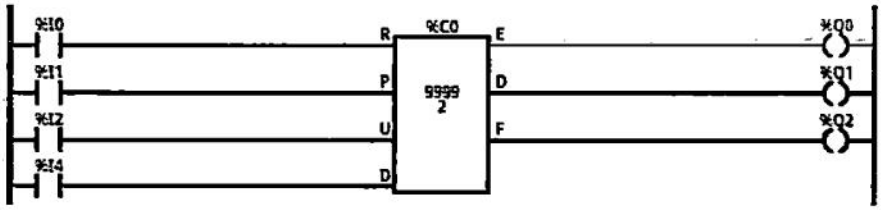


Gambar 4.70. Kondisi *Counter* Setelah *Input* D Diaktifkan Kembali

Dalam penggunaan *counter* terdapat dua fenomena khusus yang dinamakan *underflow* dan *overflow*. Kedua hal tersebut secara teknis diberlakukan pada simulasi berikut.

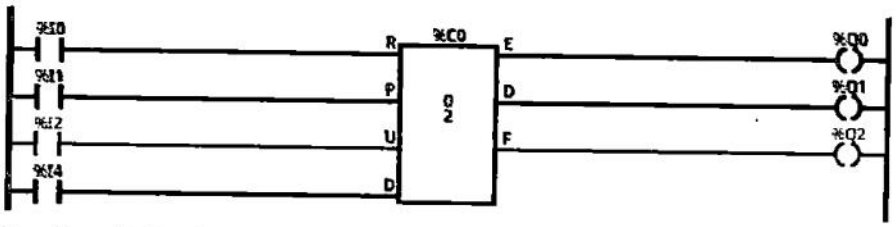
4.6.5 *UP COUNT - OVERFLOW*

Overflow merupakan kondisi di mana nilai aktual *counter* beralih/berubah dari 9999 ke 0 atau dari nilai maksimum ke nilai minimum. Secara sederhana fenomena ini dapat dikatakan sebagai "lubernya" nilai *counter* karena nilainya naik melebihi angka maksimumnya. Gambar berikut menunjukkan kondisi awal sebelum terjadinya *overflow*, di mana nilai aktual *counter* adalah 9999 atau maksimum.



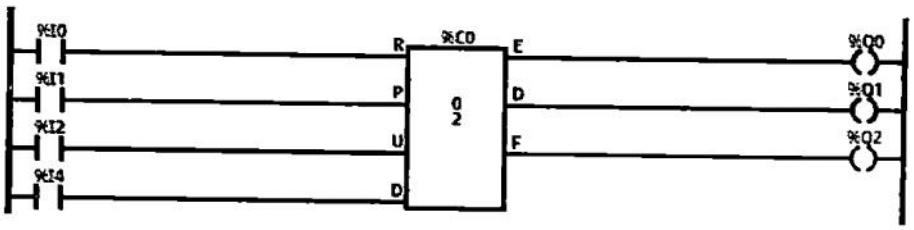
Gambar 4.71. *Counter* dengan Nilai Aktual Maksimum

Pada tahap selanjutnya dilakukan pengaktifan *input* U(*Up Count*) yang dipicu oleh kontak %I2. Dengan demikian kondisi *ladder* menjadi sebagai berikut.



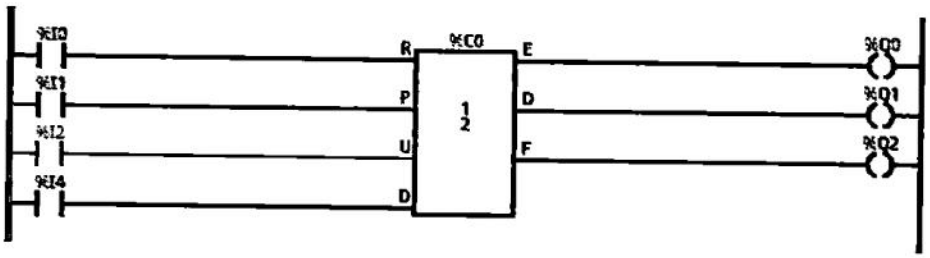
Gambar 4.72. *Counter* dengan *Input* U(*Up Count*) dan *Output* F(*Overflow*) Aktif

Pada gambar tersebut dapat dilihat bahwa *input* U(*Up Count*) yang aktif menyebabkan nilai aktual *counter* berubah naik ke angka 0 dari 9999. Hal ini menyebabkan *output* F(*Overflow*) pada *counter* tersebut aktif. Pada tahap selanjutnya, *input* U(*Up Count*) kembali dinonaktifkan.



Gambar 4.73. *Counter* dengan *Input* U(*Up Count*) Dinonaktifkan Kembali dan *Output* F(*Overflow*) Tetap Aktif

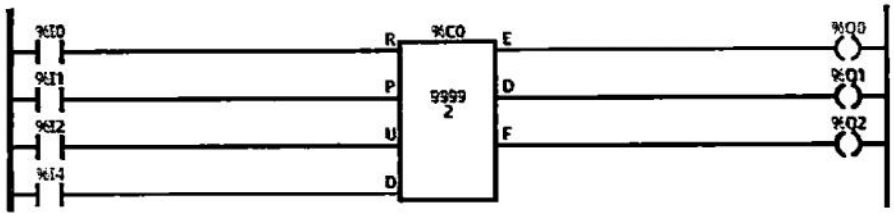
Setelah *overflow* tersebut, *input* U kembali diaktifkan dengan kontak %I2. Akibatnya, *output* E dan *coil* %Q2 kembali bernilai salah atau tidak aktif.



Gambar 4.74. *Counter* dengan *Input* U Aktif dan *Output* F Tidak Aktif

4.6.6 *DOWN COUNT - UNDERFLOW*

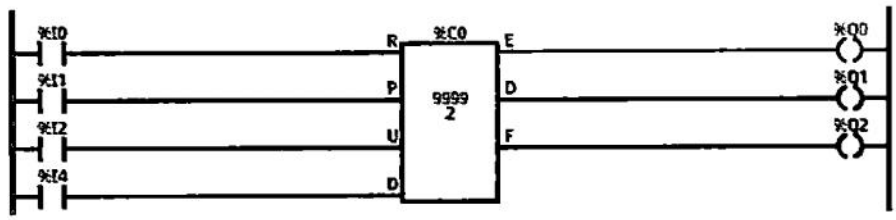
Underflow adalah kondisi di mana nilai aktual *counter* turun dari nilai minimumnya. Secara teknis nilainya akan beralih dari nilai minimum ke nilai maksimum atau dari 0 ke 9999. Gambar-gambar berikut menjelaskan tentang simulasi fenomena *underflow* pada sebuah *counter* dalam Classic Ladder.



Gambar 4.75. *Counter* dengan *Input* D(*Down Count*) Aktif dan *Output* E(*Underflow*) Aktif

Gambar tersebut menunjukkan konsidi saat *input* D(*Down Count*) diaktifkan. Kondisi awal dari *counter* tersebut adalah bernilai aktual 0. Dengan pengaktifan *input* D tersebut, nilai aktual *counter* tersebut beralih dari 0 ke 9999. Oleh karena itu *output* E(*Underflow*) aktif.

Pada tahap selanjutnya, *input* D(*Down Count*) kembali dinonaktifkan. Dapat dilihat pada gambar berikut bahwa kondisi yang *output* E(*Underflow*) akan tetap pada kondisi terakhirnya.



Gambar 4.76. *Counter* dengan *Input* D(*Down Count*) Dinonaktifkan Kembali dan *Output* E(*Underflow*) Tetap

Jika setelah itu *input* D kembali diaktifkan dengan kontak %I4, maka *output* E dan *coil* %Q0 kembali bernilai salah atau tidak aktif.