EGE ÜNİVERSİTESİ
EGE MYO

MEKATRONİK PROGRAMI

PROGRAMLANABİLİR DENETLEYİCİLER

PLC PROGRAMLAMA DİLLERİ

TWIDO DİLLERİNİN TANITIMI

- LADDER DİLİ
- KOMUT LİST DİLİ
- GRAFCET DİLİ

- **Genel Bakış** Bu kısımda, Twido programlanabilir denetleyiciler için kontrol programları yaratmak için gerekli olan Ladder, List ve Grafcet programlama dillerinin kullanımı hakkında açıklamalar sunulmuştur.

LADDER DILI

Genel Bakış Bu bölümde, Ladder dili ile programlama anlatılmıştır.

Ladder Diyagramlarına Giriş

- **Açıklama** Ladder diyagramları, röle kontrol devrelerini temsil eden röle mantık diyagramlarına benzer. Bu ikisi arasındaki ana farklar, Ladder programlamanın, röle mantık diyagramlarında bulunmayan aşağıdaki özelikleridir:

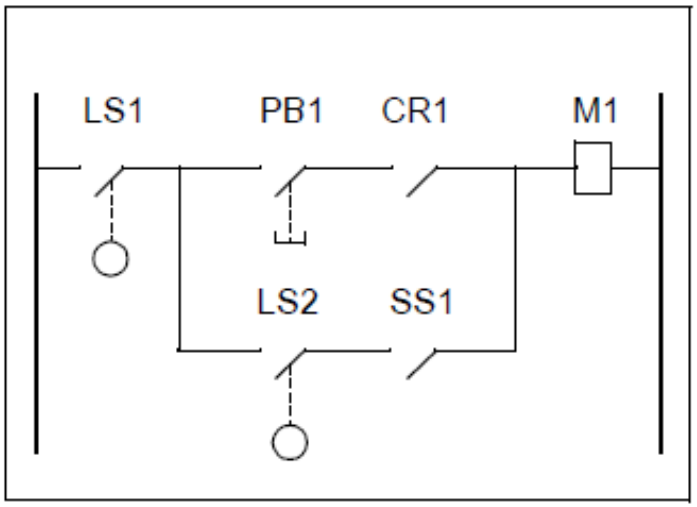
➢ Tüm girişler, kontak sembolleri () tarafından temsil edilir.

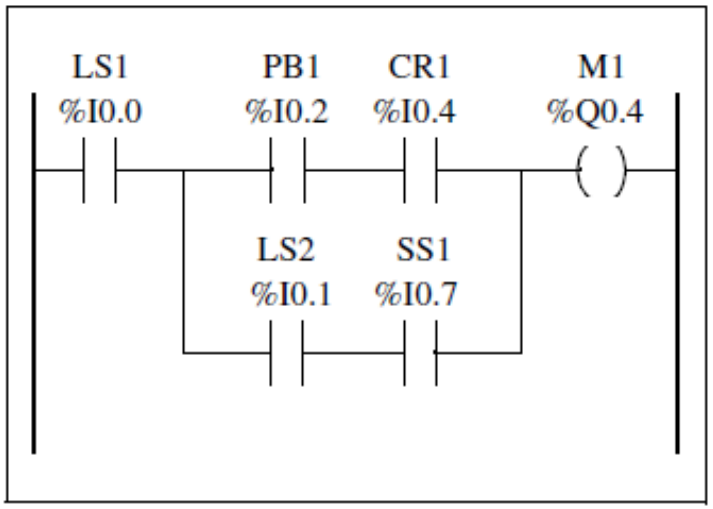
➢ Tüm çıkışlar, bobin sembolleri () tarafından temsil edilir.

➢ Nümerik işlemler, grafiksel Ladder komut setinde dahil edilmiştir.

Röle Devrelerinin Ladder Eşdeğerleri

- Aşağıdaki resimde, bir röle devresinin basitleştirilmiş bir bağlantı diyagramı ve eşdeğer Ladder diyagramı gösterilmiştir





- Röle mantık devresi Merdiven diyagramı

- Yukarıdaki örnekte, röle mantık diyagramındaki bir anahtarlama cihazı ile ilişkilendirilmiş tüm girişlerin, Ladder diyagramında kontaklar olarak gösterildiğine dikkat ediniz.
- Röle mantık diyagramındaki M1 çıkış bobini, Ladder diyagramında bir çıkış bobin sembolü ile temsil edilmiştir. Ladder diyagramındaki her bir kontak/bobin sembolünün üzerinde görünen adres numaraları, kontrolöre gelen harici giriş /çıkış bağlantılarının yerlerine yapılan referanslardır.

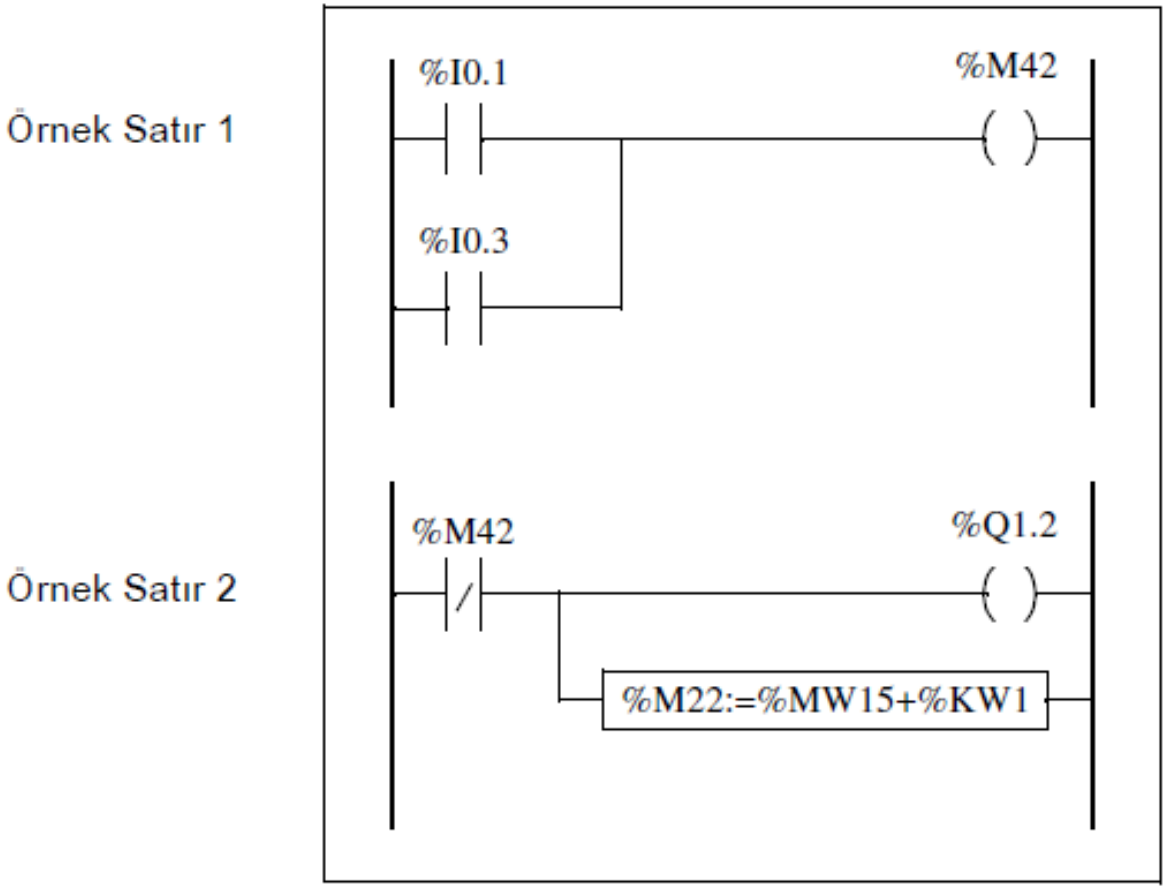
- **Ladder Satırları** Ladder dilinde yazılan bir program, iki dikey potansiyel çubuğu arasına çizilmiş grafiksel komut setleri olan satırlardan (rung) oluşur. Satırlar, denetleyici tarafından sıra ile icra edilir.

- Grafiksel komut seti, aşağıdaki fonksiyonları temsil eder:
- Denetleyici giriş/çıkışları (basma butonlar, sensörler, röleler, pilot ışıklar, ...)
- Denetleyici fonksiyonları (zamanlayıcılar, sayıcılar, ...)
- Aritmetik ve lojik işlemler (toplama, bölme, AND, XOR, ...)
- Karşılaştırma operatörleri ve diğer nümerik işlemler (A<B, A=B, shift, rotate, ...)
- Denetleyicideki dahili değişkenler (bitler, sözcükler, ...)

- Bu grafiksel komutlar, son olarak bir veya birkaç çıkışa ve/veya aksiyona giden düşey ve yatay bağlantılar ile yerleştirilir. Bir satır, bağlantılı komutların bir grubundan daha fazlasını destekleyemez.

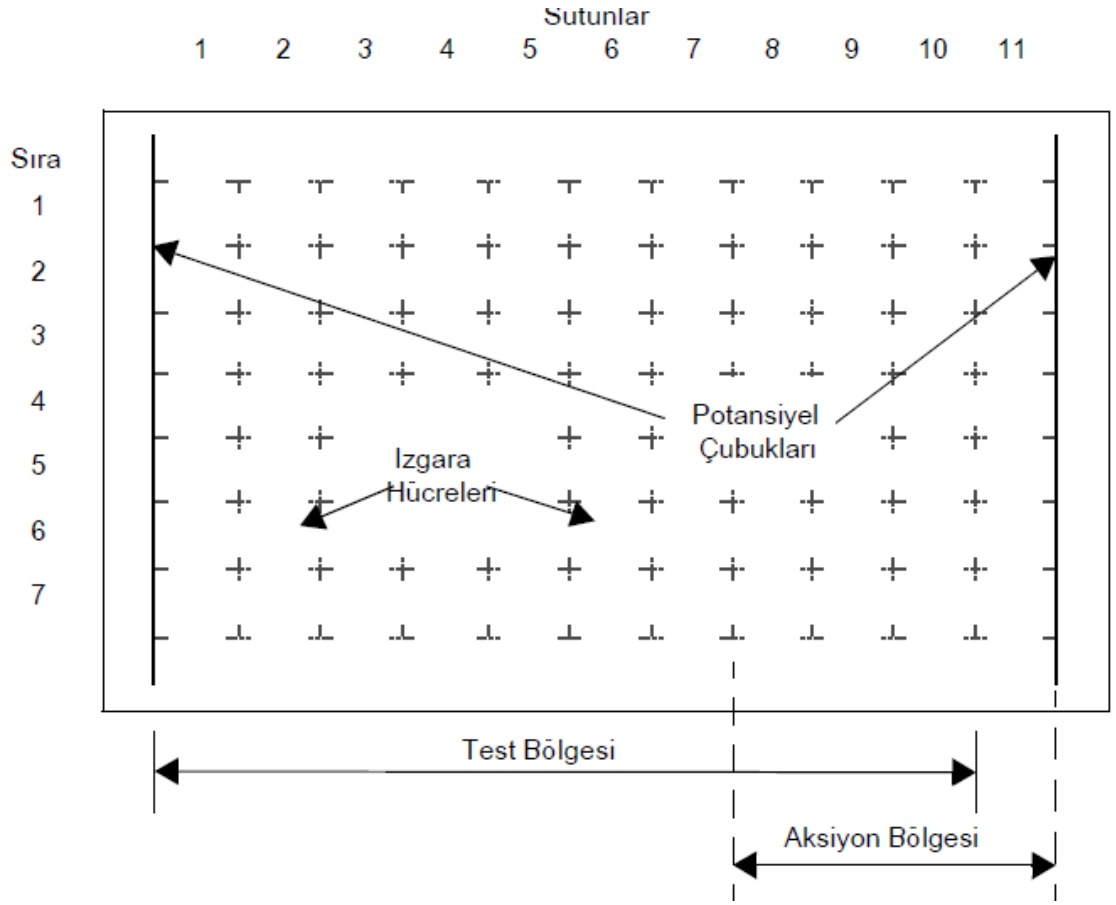
Ladder Satırları Örneği

- Aşağıdaki diyagram, iki satırdan oluşan bir Ladder program örneğidir.



Ladder Diyagramları için Programlama İlkeleri

- **Programlama Izgarası (Grid)**
- Her bir Ladder satırı (rung), aşağıdaki resimde gösterildiği gibi, iki bölgeden oluşan, 7 sıra 11 sütun'luk bir ızgaradan (grid) ibarettir.



- **Izgara Bölgeleri** Ladder diyagram programlama ızgarası, iki bölgeye ayrılmıştır:

➢Test Bölgesi

- Aksiyonları gerçekleştirmek için test edilen koşulları içerir. 1-10 arası sütunlardan oluşur ve kontaklar, fonksiyon blokları ve karşılaştırma blokları içerir.

➢Aksiyon Bölgesi

- Çıkışı veya Test Bölgesindeki koşulların test sonuçlarına göre gerçekleştirilecek olan işlemi içerir. 8-11 arası sütunlardan oluşur ve bobinler ve operasyon blokları içerir.

Izgarada Komutları Girmek

- Bir Ladder satırı, ızgaranın sol-üst köşesindeki ilk hücrede başlayan bir 7x11 programlama ızgarası sağlar. Programlama, buyrukların ızgara hücrelerine girilmesinden ibarettir. Test buyrukları, karşılaştırmalar ve fonksiyonlar test bölgesine girilir ve sola hizalanmıştır. Test mantığı; bobinler, nümerik işlemler ve program akış kontrolü buyruklarının girildiği aksiyon bölgesine süreklilik sağlar ve sağa hizalanmıştır.
- Satır, ızgara içinde yukarıdan aşağıya ve soldan sağa doğru çözülür ve icra edilir (testler yapılır ve çıkışlar atanır).

- **Satır Başlıkları** Satıra ek olarak, satırın tam üzerinde bir satır başlığı (rung header) görünür. Satırın mantık amacını belgelemek için, satır başlığı kullanınız. Satır başlığı, aşağıdaki bilgileri içerebilir:

- Satır numarası
- Etiketler (Labels) (%Li)
- Altyordam (Subroutine) deklarasyonları (SRi:)
- Satır teması (title)
- Satır açıklamaları (comments)

Ladder Diyagram Blokları

- **Açıklama** Ladder diyagramları, program akışını ve fonksiyonları temsil eden bloklardan oluşur.
- Bu bloklar aşağıda verilmiştir:

➢Kontaklar

➢Bobinler

➢Program akış komutları

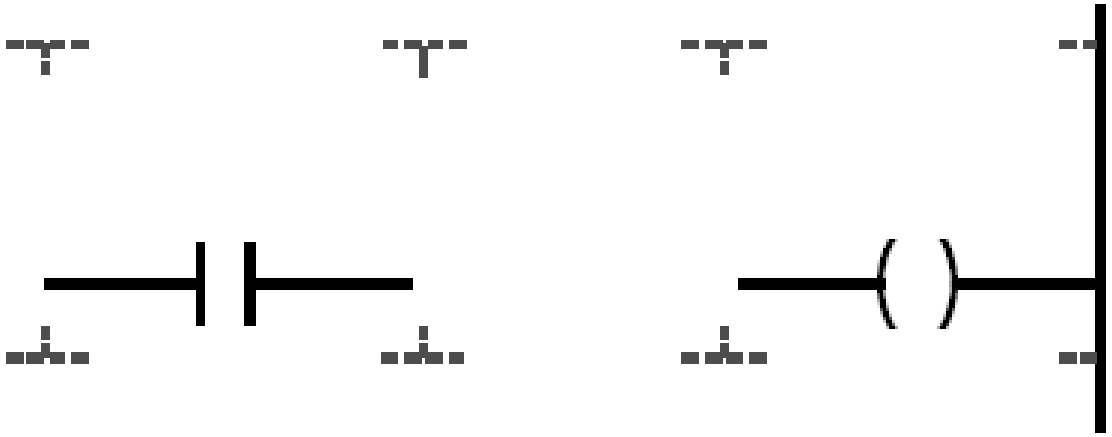
➢Fonksiyon blokları

➢Karşılaştırma blokları

➢İşlem blokları

Kontaklar, Bobinler ve Program Akışı

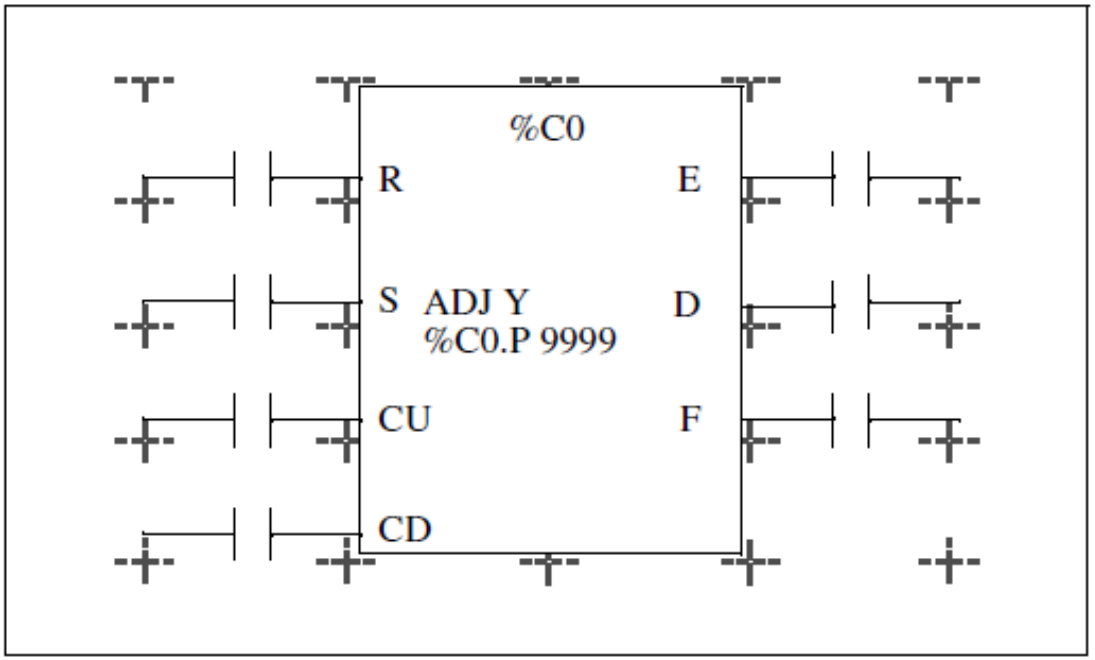
- Kontaklar, bobinler ve program akış (jump ve call) buyrukları, ladder programlama ızgarasında bir tek hücre işgal eder. Fonksiyon blokları, karşılaştırma blokları ve işlem blokları birden çok hücre işgal eder.
- Aşağıda, bir kontak ve bir bobin örneği verilmiştir.



Fonksiyon Blokları

- Fonksiyon blokları, programlama ızgarasının test bölgesine yerleştirilir. Blok, ilk sırada (row) görünmelidir; fonksiyon bloğunun üstünde ve altında, herhangi bir ladder buyruğu veya süreklilik çizgisi görünmemelidir. Ladder test buyrukları, fonksiyon bloğunun giriş tarafına gider ve test komutları ve/veya aksiyon buyrukları, bloğun çıkış tarafına doğru yer alır.
- Fonksiyon blokları, dikey olarak yönlendirilmiştir ve programlama ızgarasının iki sütuna dört sıralık bir kısmını işgal eder.

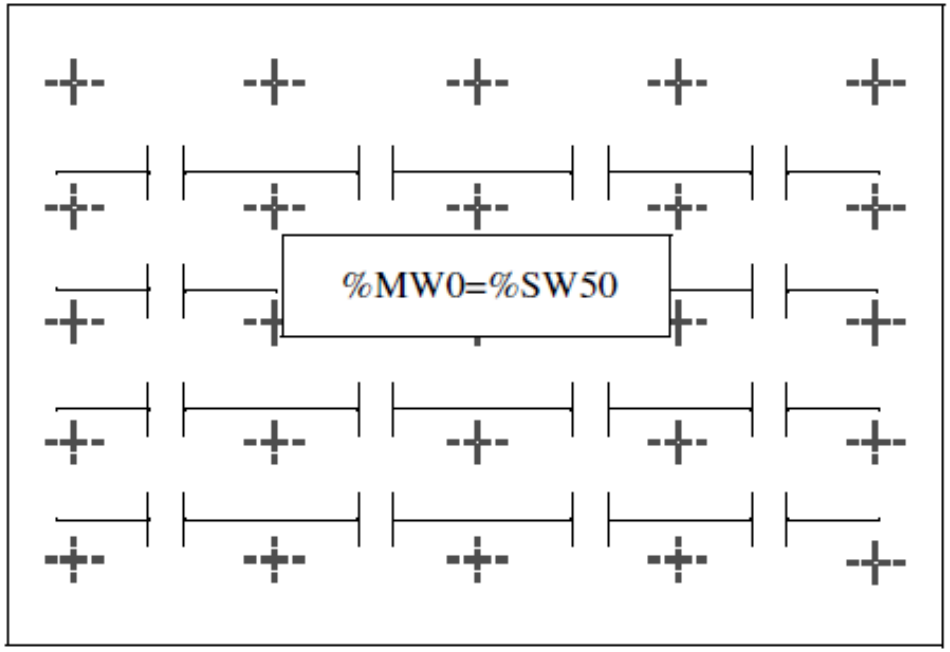
- Aşağıda, bir sayıcı fonksiyon bloğu örneği verilmiştir.



Karşılaştırma Blokları

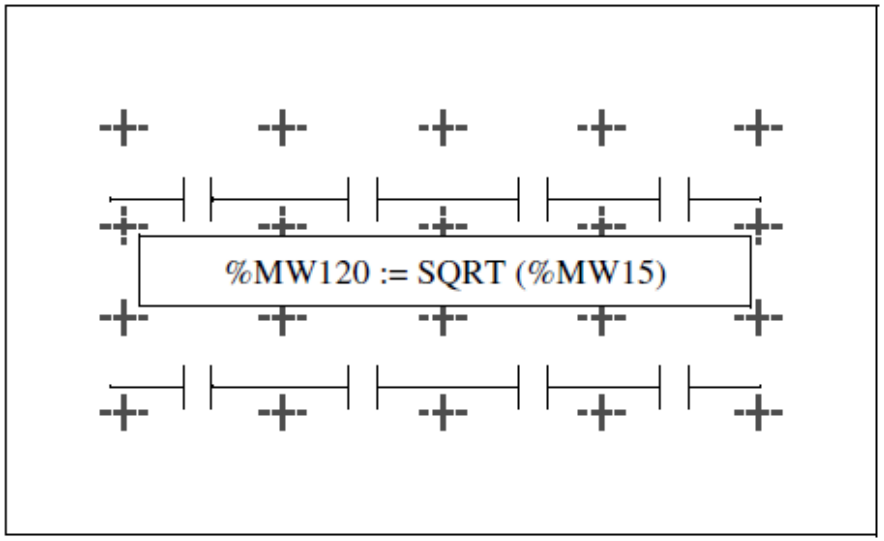
- Karşılaştırma blokları, programlama ızgarasının test bölgesine yerleştirilir. Blok, buyruğunun tüm uzunluğu test bölgesi içinde kaldığı sürece, test bölgesi içinde herhangi bir sıra veya satırda görünebilir.
- Karşılaştırma blokları, yatay olarak yönlendirilmiştir ve programlama ızgarasının iki sütuna bir sıralık bir kısmını işgal eder.

- Aşağıda, bir karşılaştırma bloğu örneği verilmiştir.



İşlem Blokları

- İşlem blokları, programlama ızgarasının aksiyon bölgesine yerleştirilir. Blok, aksiyon bölgesi içinde herhangi bir sırada görünebilir. Buyruk sağa-hizalıdır; sağda görünür ve son sütunda sonlanır.
- İşlem blokları, yatay olarak yönlendirilmiştir ve programlama ızgarasının dört sütuna bir sıralık bir kısmını işgal eder.
- Aşağıda, bir işlem bloğu örneği verilmiştir.

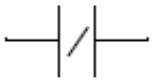
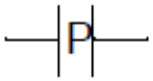
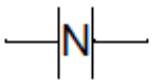


Ladder Dili Grafik Elemanları

- **Açıklama** Ladder diyagramlarındaki komutlar, grafik elemanlardan oluşur. Bu kısımda, Ladder komutlarında kullanılan grafik elemanlar listelenmiş ve tanımlanmıştır. Ladder programlarında, bu elemanların kullanımı hakkında ayrıntılı bilgi için, bkz. PLC Kurs Notu.

Kontaklar

- Kontak grafik elemanları, test bölgesinde programlanır ve bir hücrelik (bir sütuna bir sıralık) yer tutar.

Ad	Grafik eleman	Komut	Fonksiyon
Normalde açık kontak		LD	Kontrol eden bit nesnesi, 1 durumunda iken geçen kontak.
Normalde kapalı kontak		LDN	Kontrol eden bit nesnesi, 0 durumunda iken geçen kontak.
Bir yükselen kenarı saptamak için kontak		LDR	Yükselen kenar: Kontrol eden bit nesnesinin 0'dan 1'e değişiminin saptanması.
Bir düşen kenarı saptamak için kontak		LDF	Düşen kenar: Kontrol eden bit nesnesinin 1'den 0'a değişiminin saptanması.

Bağlantı Elemanları

- Grafik bağlantı elemanları, test ve aksiyon grafik elemanlarını bağlamak için kullanılır.

Ad	**Grafik eleman**	**Fonksiyonlar**
Yatay konnektör		Test ve aksiyon grafik elemanlarını, iki potansiyel çubuk arasında seri bağlar.
Düşey konnektör		Test ve aksiyon grafik elemanlarını, paralel bağlar (bir düşey bağlantı).

Bobinler

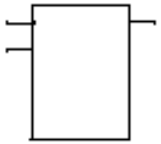
- Bobin grafik elemanları, aksiyon bölgesinde programla-nır ve bir hücrelik (bir sütuna bir sıra) yer tutar.

Ad	Grafik eleman	Komut	Fonksiyonlar
Direkt bobin		ST	İlişkilendirilmiş bit nesnesi, test bölgesi sonucunun değerini alır.
Evrik bobin		STN	İlişkilendirilmiş bit nesnesi, test bölgesi sonucunun evrik değerini alır.
Set bobin		S	İlişkilendirilmiş bit nesnesi, test bölgesi sonucu 1 olduğunda, 1'e set edilir.
Reset bobin		R	İlişkilendirilmiş bit nesnesi, test bölgesi sonucu 1 olduğunda, 0'a set edilir.

Jump veya Altrutin çağrısı	->>%Li ->>%SRi	JMP SR	Etiketlenmiş bir komuta bağlan, akış-yukarı veya akış-aşağı.
Geçiş koşulu bobini	—(#)—		Grafcet dilinde sağlanmıştır; geçişler ile ilişkilendirilmiş geçiş koşullarının programlanması, bir sonraki adıma geçişe neden olduğunda kullanılır.
Bir altrutinden geri dön (return)	<RET>	RET	Ana programa geri dönmek için, altrutinlerin bitimine yerleştirilir.
Programı durdur	<END>	END	Programın bitişini belirler.

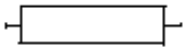
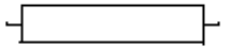
Fonksiyon Blokları

- Fonksiyon bloklarının grafik elemanları, test bölgesinde programlanır ve sekiz hücrelik (iki sütuna dört sıra) yere ihtiyaç duyar. İstisna olarak, çok hızlı sayıcılar, on hücrelik (iki sütuna beş sıra) yer kaplar.

Ad	Grafik eleman	Fonksiyon
Zamanlayıcılar, sayıcılar, register'lar, vb.		Fonksiyon bloklarının her biri, diğer grafik elemanlara bağlantılara olanak sağlayan giriş ve çıkış'ları kullanır. Not: Fonksiyon bloklarının çıkışları, birbirlerine bağlanamaz (düşey kısa devreler).

İşlem ve Karşılaştırma Blokları

- Karşılaştırma blokları, test bölgesinde programlanır ve işlem blokları, aksiyon bölgesinde programlanır.

Ad	Grafik eleman	Fonksiyon
Karşılaştırma bloğu		İki operandı karşılaştırır, sonuç kontrol edildiğinde çıkış 1'e değişir. Boyut: iki sütuna bir sıra
İşlem bloğu		Aritmetik ve lojik işlemleri gerçekleştirir. Boyut: dört sütuna bir sıra

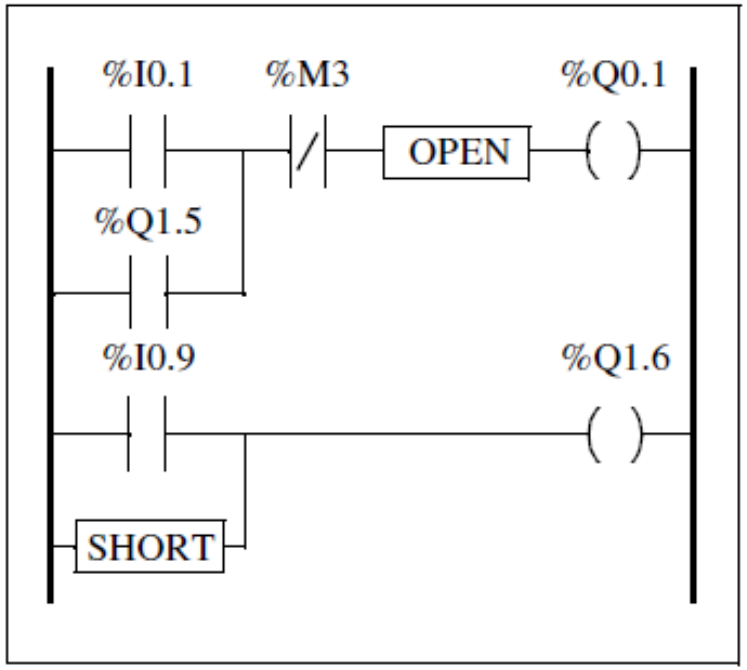
Özel Ladder Komutları OPEN ve SHORT

- **Açıklama** OPEN ve SHORT buyrukları, Ladder programlarında hata bulma (debugging) ve arıza giderme (troubleshooting) için kullanışlı bir yöntem sağlar. Bu özel komutlar, bir satırın sürekliliğini ya kısa devre ederek (short) ya da açık devre yaparak (open), bir satırın mantığını, aşağıdaki tabloda açıklandığı gibi değiştirir.

Komut	Tanım	List Komutu
OPEN	En son lojik işlemin sonucunu dikkate almaksızın, bir ladder satırının sürekliliğini bozar.	`AND 0`
SHORT	En son lojik işlemin sonucunu dikkate almaksızın, satırın sürekliliğini sağlar.	`OR 1`

- List programlamada, OR ve AND komutlarından sonra kullanılan, sırasıyla 1 ve 0 değerleri, OPEN ve SHORT buyruklarını yaratır.

- **Örnekler** Aşağıda, OPEN ve SHORT buyruklarını kullanan örneklere yer verilmiştir.




```
LD      %I0.1
OR      %Q1.5
ANDN    %M3
AND     0
ST      %Q0.1
LD      %I0.9
OR      1
ST      %Q1.6
```

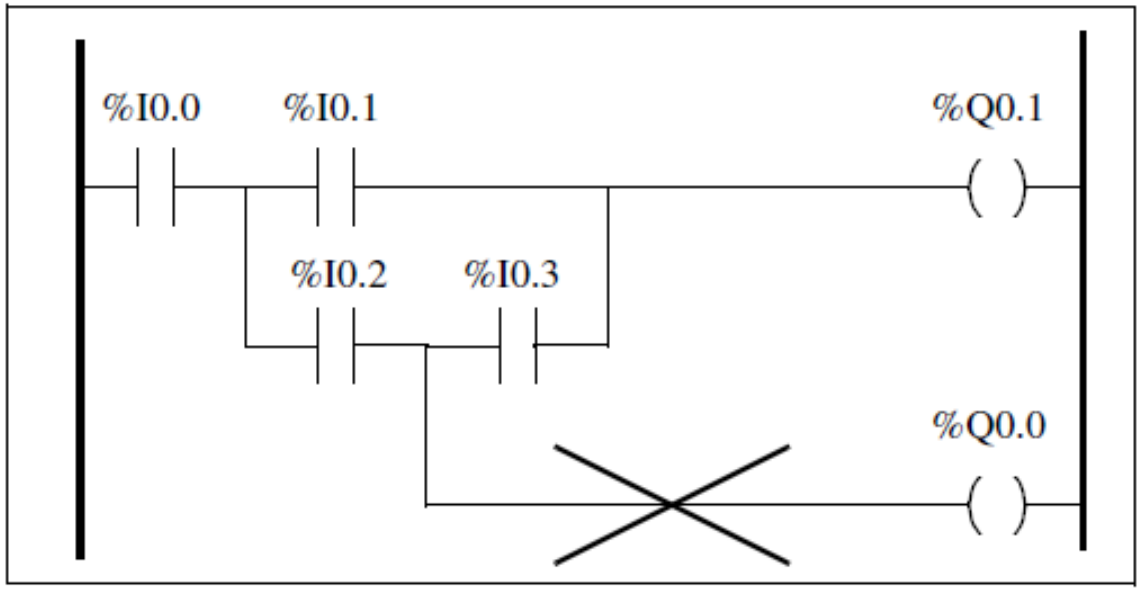
Programlama Önerisi

- **Program Atlamaları**
- Program atlamalarını (jumps), tarama süresini arttırabilen uzun döngülerden sakınmak için, dikkatli bir şekilde kullanınız. Akış-yukarı'da bulunan komutlara atlamalardan sakınınız (Bir akış-yukarı buyruk satırı, bir program içinde bir atlamadan önce görünürken, bir akış-aşağı buyruk satırı ise, sonra görünür).
- **Çıkışları Programlamak**
- Bir çıkış biti veya dahili bit, bir program içerisinde yalnızca bir defa kontrol edilebilir. Çıkış bitleri söz konusu ise, çıkışlar güncellendiğinde, yalnızca en son taranan bit hesaba katılır.

- **Doğrudan-Bağlı Acil Durum Stop Sensörlerini Kullanmak**
- Doğrudan acil durum durmaları (stop) için kullanılan sensörler, denetleyici tarafından işlenmemelidir. Bunlar, karşılık gelen çıkışlara doğrudan bağlanmalıdır.
- **Enerji Gelmesini Ele Almak**
- Enerji yeniden gelmesini, bir manuel çalışmada koşullu yapınız çünkü tesisatın otomatik bir yeniden başlatma yapması, beklenmeyen ekipman çalışmasına neden olabilir (sistem bitleri %S0, %S1 ve %S9'u kullanınız.).
- **Zaman ve Takvim Bloğu Yönetimi**
- Takvim bloğu hatalarını işaret eden sistem biti %S51'in durumu, kontrol edilmelidir.
- **Sözdizimi ve Hata Denetimi**
- Bir program girildiğinde, TwidoSoft; buyrukların, işlenenlerin ve bunların birbirleriyle ilişkisinin sözdizimini denetler.

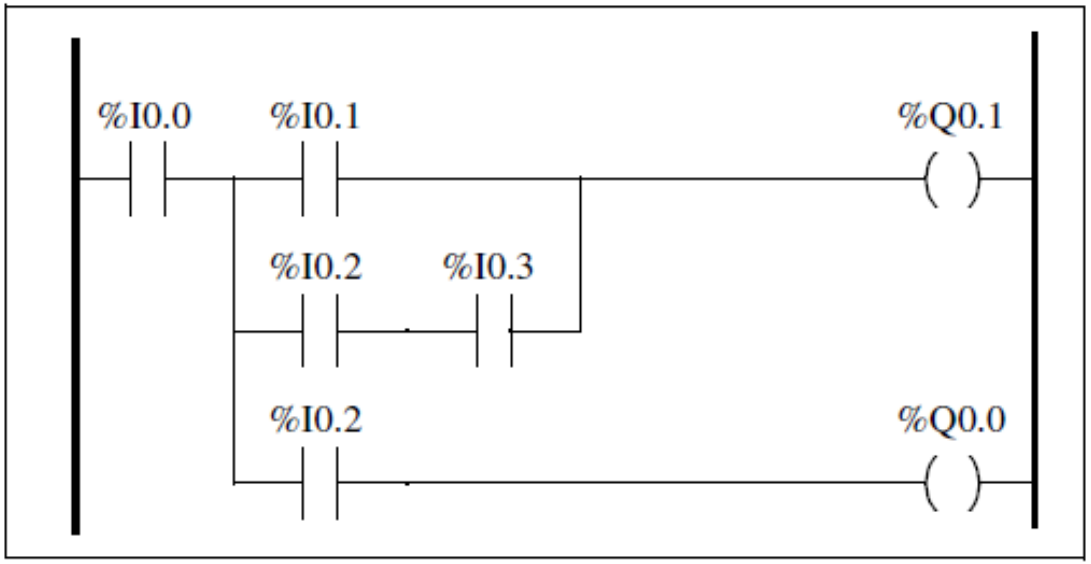
Parantezlerin Kullanımı Hakkında Ek Notlar

- Atama komutları, parantezler içine yerleştirilmemelidir:



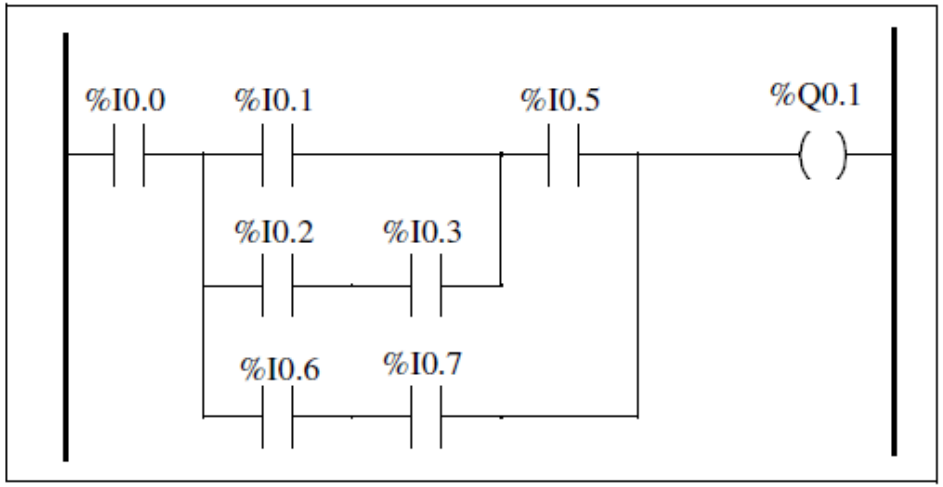

```
LD      %I0.0
AND     %I0.1
OR(     %I0.2
ST      %Q0.0
AND     %I0.3
)
ST      %Q0.1
```

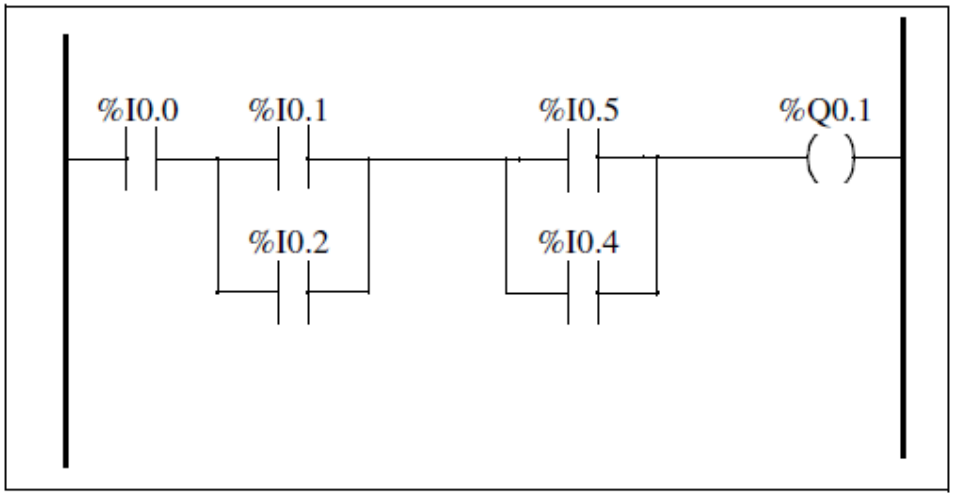
- Aynı fonksiyonu gerçekleştirmek için programlama aşağıdaki gibi olmalıdır:



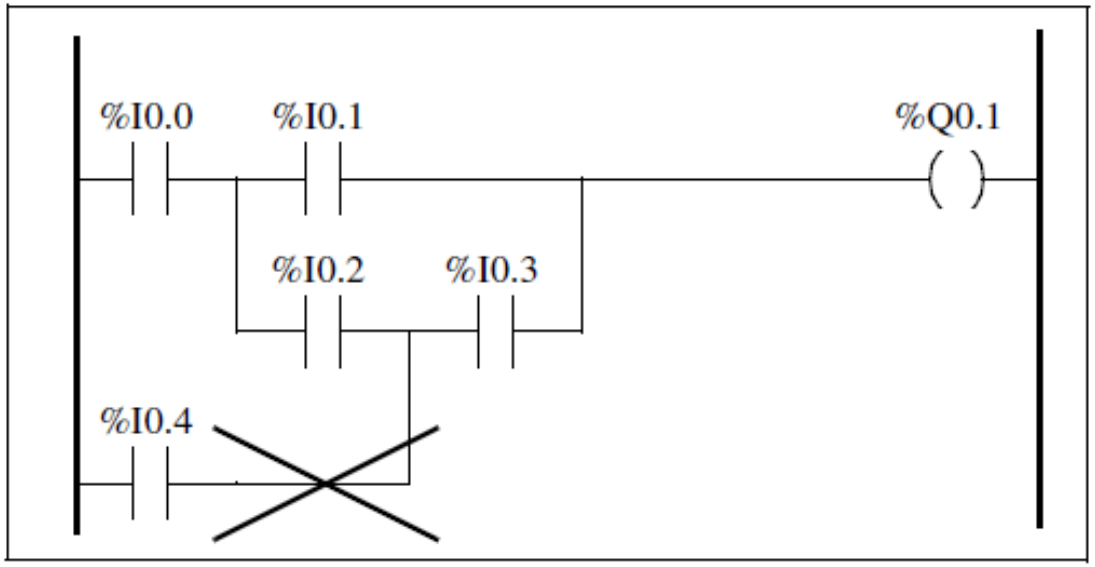

```
LD      %I0.0
MPS
AND(    %I0.1
OR(     %I0.2
AND     %I0.3
)
)
ST      %Q0.1
MPP
AND     %I0.2
ST      %Q0.0
```

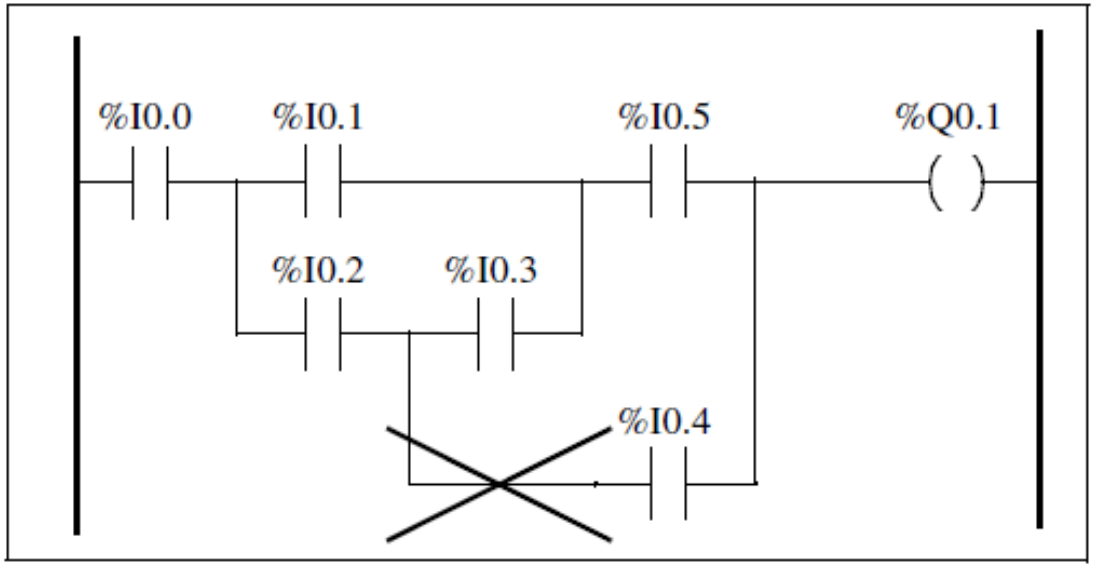
- Birkaç kontak paralel olarak yerleştirilmişse, bunlar ya bir diğeri ile içiçe koyulmalı ya da birbirlerinden tamamen ilişkisiz hale getirilmelidir:



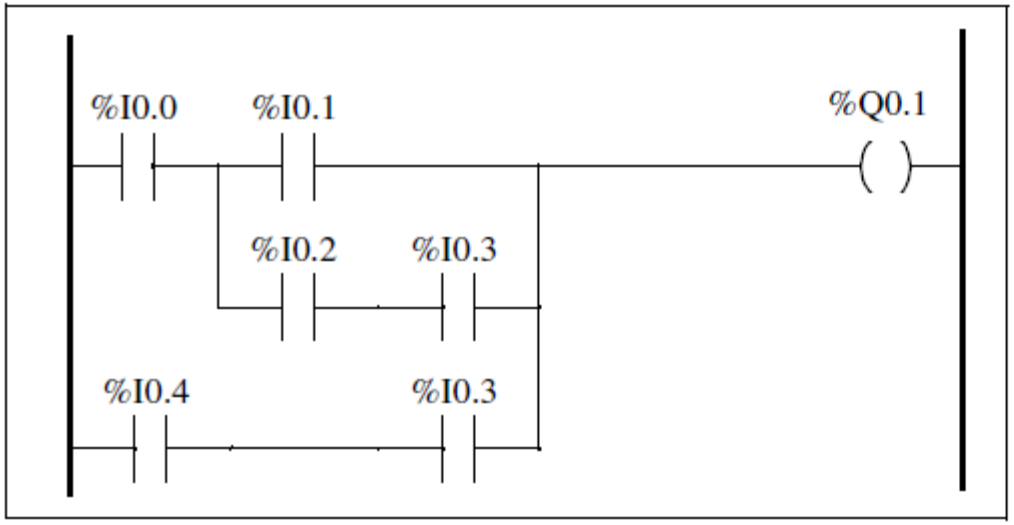


- Aşağıdaki şemalar, programlanamaz:

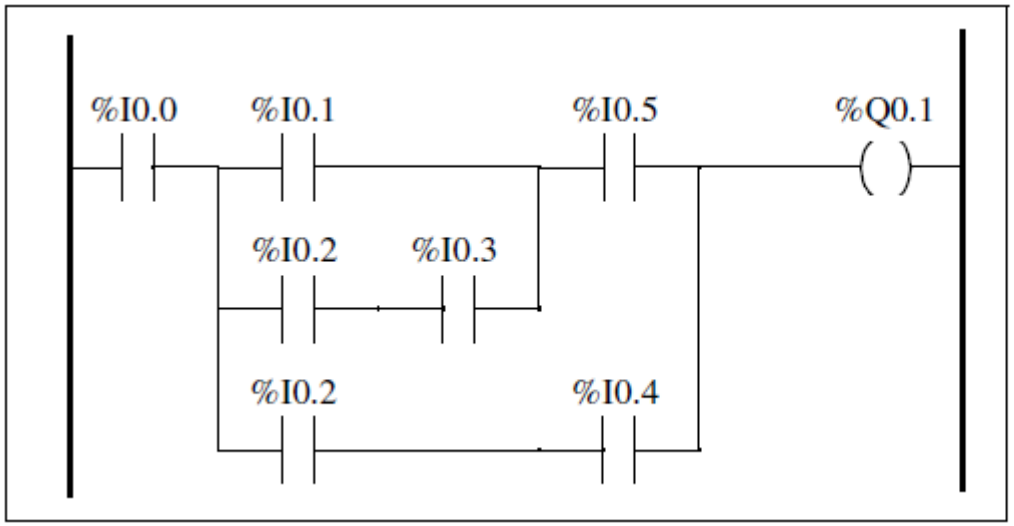




- Bir önceki sayfadaki şemalar, icra edilmek isteniyorsa, aşağıdaki eşdeğer şemalarda görülen değişiklikler yapılmalıdır:




```
LD      %I0.0
AND(    %I0.1
OR(     %I0.2
AND     %I0.3
)
)
OR(     %I0.4
AND     %I0.3
)
ST      %Q0.1
```




```
LD      %I0.0
AND(    %I0.1
OR(     %I0.2
AND     %I0.3
)
AND     %I0.5
OR(     %I0.2
AND     %I0.4
)
)
ST      %Q0.1
```

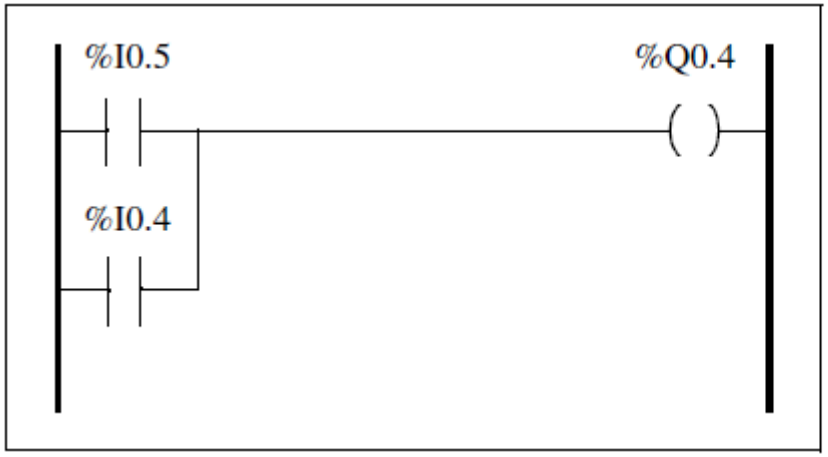
Ladder/List Dönüştürülebilirliği

- **Açıklama** Program dönüştürülebilirliği, uygulama programlarının Ladder'dan List'e ve List'ten Ladder'a geri dönüştürülmesini sağlayan bir TwidoSoft programlama yazılımı özelliğidir.
- **Dönüştürülebilirliği Anlamak**
- Program dönüştürülebilirlik özelliğini anlamanın anahtarı, bir Ladder satırı ve ilgili buyruk List sırasının ilişkisini incelemektir:

➢ **Ladder satırı**: Bir mantık ifadeyi meydana getiren bir Ladder buyrukları topluluğudur.

➢ **List sırası (sequence)**: Ladder buyruklarına karşılık gelen ve aynı mantık ifadeyi temsil eden bir List programlama buyrukları topluluğudur.

- Aşağıdaki resimde, bir Ladder satırı ve bunun, bir sıra halinde List buyrukları olarak ifade edilmiş eşdeğer programı gösterilmiştir.




```
LD    %I0.5
OR    %I0.4
ST    %Q0.4
```

- Bir uygulama programı, programın Ladder dilinde ya da List dilinde yazıldığına bakılmaksızın, List buyrukları halinde dahili olarak saklanır. TwidoSoft, iki dil arasındaki program yapısı benzerliğinden yararlanır ve seçilmiş kullanıcı tercihine bağlı olarak programın bu dahili List imajını, List veya Ladder görüntüleyicilerde veya editörlerde, ya bir List programı (temel form) ya da grafiksel olarak bir Ladder diyagramı şeklinde görüntülemek için kullanır.

Dönüştürülebilirliği Sağlamak

- Ladder'da yaratılmış bir program, her zaman List'e dönüştürülebilir ama bazı List mantığı, Ladder'a dönüştürülemeyebilir.

Ladder/List Dönüştürülebilirliğinin Ana Noktaları

- **Dönüştürülebilirlik için Gerekli Komutlar**
- List dilindeki bir dönüştürülebilir fonksiyon bloğu yapısı, aşağıdaki buyrukların kullanılmasını gerektirir:
  - **BLK**, blok başlangıcını işaret eder; başlangıç satırını ve bloğa giriş kısmının başlangıcını tanımlar.
  - **OUT_BLK**, bloğun çıkış kısmının başlangıcını işaretler.
  - **END_BLK**, bloğun ve satırın sonunu işaretler.
- Dönüştürülebilir fonksiyon bloğu buyruklarının kullanımı, düzgün çalışan bir List programı için zorunlu değildir. Dönüştürülebilir olmayan bazı buyruklar için, List'te programlama mümkündür.

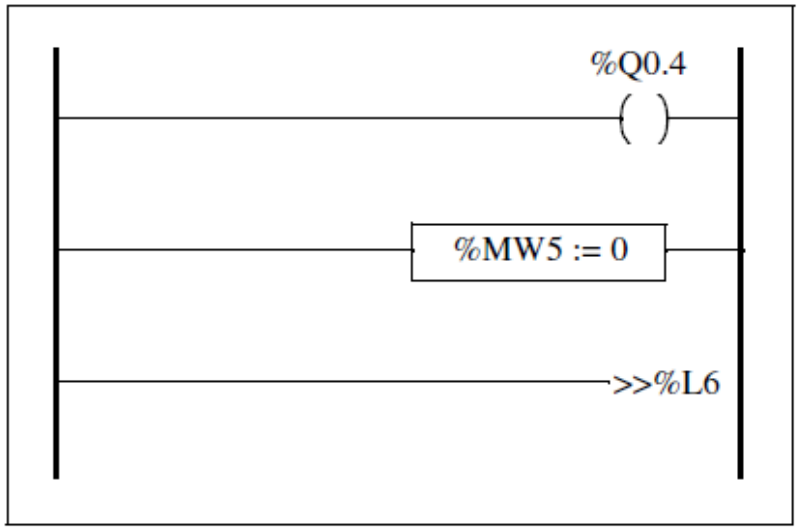
Sakınılacak, Eşdeğeri Olmayan Komutlar

- Ladder diyagramlarında eşdeğerleri bulunmayan belirli bazı List buyruklarını veya buyrukların ve işlenenlerin belirli bazı birleşimlerini kullanmaktan kaçınınız.
- Örneğin, N komutunun (Boole akümülatördeki değeri evirir) Ladder eşdeğeri bulunmamaktadır.
- Aşağıdaki tablo, Ladder'a dönüşmeyen bütün List programlama buyruklarını göstermektedir:

List Komutu	Operand	Tanım
JMPCN	%Li	Jump Conditional Not
N	yok	Negation (Not)
ENDCN	yok	End Conditional Not

Koşulsuz Satırlar

- Koşulsuz satırları programlamak da, List'ten Ladder'a dönüştürülebilirliği sağlamak için List programlama ana noktalarını izlemeyi gerektirir. Koşulsuz satırların testleri veya koşulları yoktur, çıkışlar veya aksiyon komutları daima enerjilendirilir veya icra edilir.
- Aşağıdaki diyagram, koşulsuz satır örneklerini ve eşdeğer List sırasını (sequence) sağlar.




```
LD      1
ST      %Q0.4
LD      1
[%MW5 := 0]
JMP     %L6
```

- JMP komutu haricinde, her bir koşulsuz List sırasının bir load komutu (ve bunu izleyen 1 ile birlikte) ile başladığına dikkat ediniz. Bu birleşim, programın her taramasında Boole akümülatör değerini 1'e set eder ve böylece, bobini (store komutu) 1'e set eder ve %MW5'i 0'a set eder. istisna olan koşulsuz jump List komutu (JMP %L6), akümülatör değerine bakılmaksızın icra edilir ve akümülatörün 1'e set edilmesine ihtiyaç duymaz.

Ladder List Satırları

- Eğer tam olarak dönüştürülebilir olmayan bir List programı dönüştürülürse, dönüştürülebilir kısımlar, Ladder görüntüsünde gösterilir ve dönüştürülemez kısımlar, Ladder List satırlarında gösterilir.
- Bir Ladder List satırı, aynı küçük bir List editörü gibi çalışır, kullanıcıya bir Ladder programının dönüştürülemez kısımlarını görme ve değiştirme olanağı sunar.

Program Dokümantasyonu

- **Programı Belgelemek**
- Programınızı, List ve Ladder editörlerini kullanarak açıklamalar (comments) girmek suretiyle belgeleyebilirsiniz:

➢ Programınızı, List Satır Açıklamaları ile belgelemek için List Editörü'nü kullanınız. Bu açıklamalar, programlama komutları ile aynı satırda bulunabileceği gibi, kendi müstakil satırlarında da bulunabilir.

➢ Programınızı, satırların direkt üzerinde bulunan satır başlıklarını (rung header) kullanarak belgelemek için Ladder Editörü'nü kullanınız.

- TwidoSoft programlama yazılımı, bu açıklamaları dönüştürülebilirlik için kullanır. Bir programı List'ten Ladder'a dönüştürürken, TwidoSoft, List açıklamalarının bazılarını bir satır başlığı oluşturmak için kullanır ve List sıraları arasına yerleştirilen açıklamalar ise satır başlıkları için kullanılır.

List Satır Açıklamaları Örneği

- ---- (* THIS IS THE TITLE OF THE HEADER FOR RUNG 0 *)
- ---- (* THIS IS THE FIRST HEADER COMMENT FOR RUNG 0 *)
- ---- (* THIS IS THE SECOND HEADER COMMENT FOR RUNG 0 *)
- 0 LD % I0. 0 (* THIS IS A LINE COMMENT *)
- 1 OR %I0. 1 (* A LINE COMMENT IS IGNORED WHEN REVERSING TO LADDER *)
- 2 ANDN %M10
- 3 ST %M101
- ---- (* THIS IS THE HEADER FOR RUNG 1 *)
- ---- (* THIS RUNG CONTAINS A LABEL *)
- ---- (* THIS IS THE SECOND HEADER COMMENT FOR RUNG 1 *)
- ---- (* THIS IS THE THIRD HEADER COMMENT FOR RUNG 1 *)
- ---- (* THIS IS THE FOURTH HEADER COMMENT FOR RUNG 1 *)
- 4 % L5:
- 5 LD %M101
- 6 [%MW20 := %KW2 * 16]
- ---- (* THIS RUNG ONLY CONTAINS A HEADER TITLE *)
- 7 LD %Q0. 5
- 8 OR %I0. 3
- 9 ORR I0. 13
- 10 ST %Q0.5