PROGRAM DENETİM İŞLEMLERİ

Denetim işleminin olmadığı bir program yapısında, birinci komuttan son komuta kadar olan bütün komutlar sırasıyla işlenmektedir (Lineer programlama). Programın tamamı OB1 (MAIN) içerisinde yazılır.

Program denetimi, belirli şartlar oluştuğunda program parçaları ya da komutların yürütülmesi işleminin düzenlenmesidir. Denetimli program yapıları, alt program ya da atlama komutları ile gerçekleştirilmektedir.

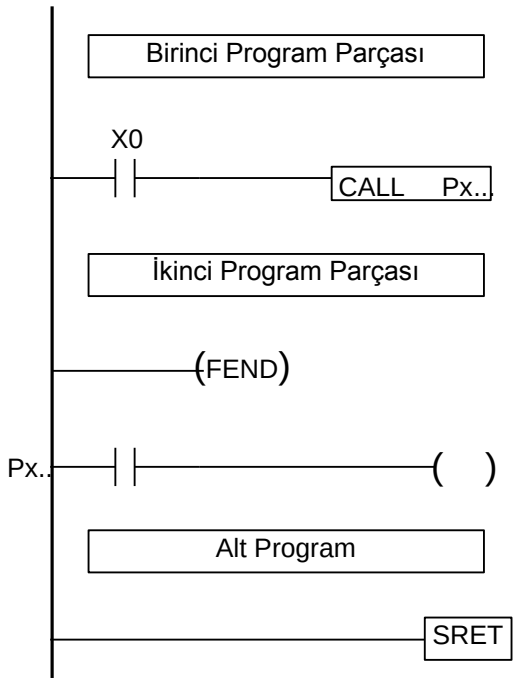
Alt Program Kullanarak Programlama

Alt programlar, ana programın son komutu olan FEND komutundan sonra yazılan ve

Pxx satırından başlayıp

SRET komutu ile son bulan program parçalarıdır.

Bu komut, yığıt tepe değeri 1 iken etkin olur. Alt programlar **CALL Pxx** komutu ile çağrılır. İşlem sırası bu komuta geldiğinde yığıt birinci seviyesindeki değer 0 ise komut işletilmez, bir sonraki komuta geçilir.



Yukarıdaki programda birinci program parçası işletildikten sonra, eğer

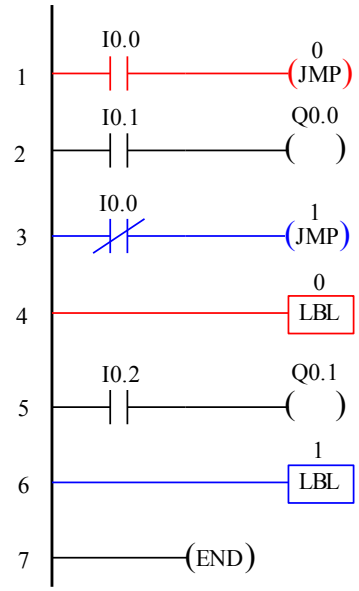
X0= 0 ise yalnız ikinci program parçası

X0= 1 ise önce alt program daha sonra ikinci program parçası işletilir.

Atlama Komutları ile Programlama

Bir programdaki komutların yürütülme sırası, **JMP** atlama komutu ve **LBL** etiket komutu çifti kullanılarak değiştirilir.

JMP komutu yığıt tepe değeri 1 olduğunda etkin olan bir komuttur ve bu komut işlendiğinde **LBL** komutuna kadar yazılmış olan komutlar atlanarak **LBL** komutunu izleyen komut yürütülür.



Yukarıdaki örnek programda

I0.0= 0 iken yığıt tepesindeki değer 0 olacağından JMP 0 komutu işletilmez, bir sonraki komuta geçilir. Bu durumda 2, 3, 6 ve 7 nolu satırlar işleme girer.

I0.0= 1 iken yığıt tepesindeki değer 1 olacağından JMP 0 komutu işletilir. Bu durumda 2 ve 3 nolu satırlar atlanır, 4, 5, 6 ve 7 nolu basamaklar işletilir.

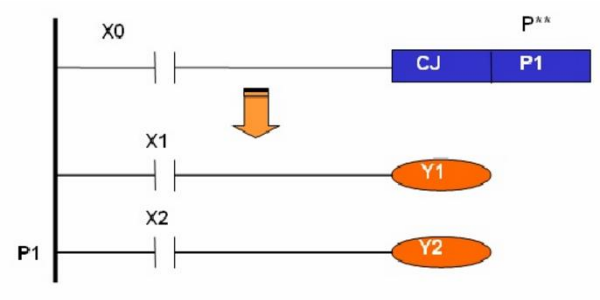
İşleme girmeyen basamaklardaki hesaplanan değerler son durumlarını korurlar.

DELTA PLC

CJ Komutu: (Conditional Jump- Koşullu Atlama)

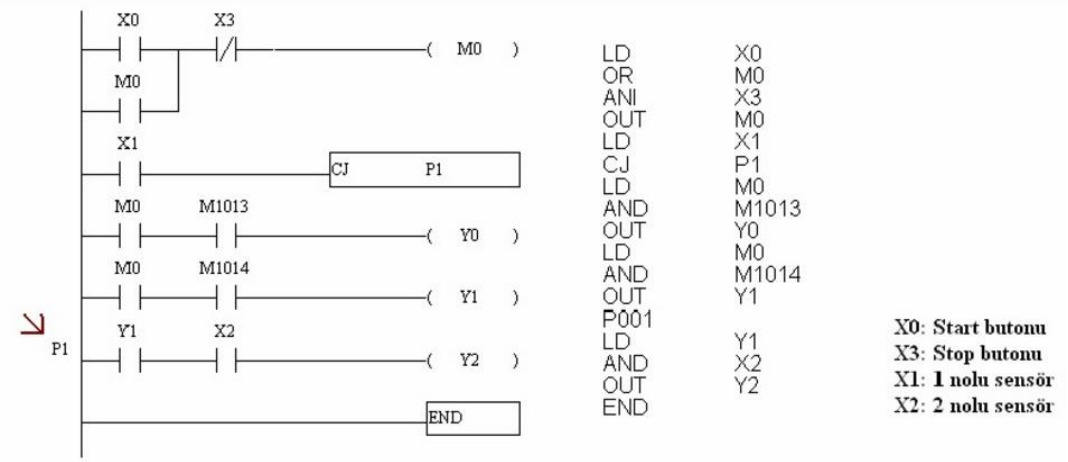
Jump komutu aktif olduğu zaman programı komutta (P..) ile belirtilen yerden (Pointer numarasından) itibaren çalıştırır. Komut pasif olduğu zaman program normal olarak Jump komutundan bir sonraki satırdan itibaren çalışmaya devam eder.

Şekildeki örnek gösterimde X0 girişi aktif olduğu zaman program adresi, (P1) noktasına atlar ve çalışmasına P1'den devam eder. Aradaki adresleri atlar. Eğer bu aradaki adreslerde bir TIMER varsa; TIMER saymayı durduracaktır. X0 girişi pasif olduğu zaman program adres 0'dan itibaren işleyecek CJ komutu aktivitesini yitirecek ve TIMER varsa saymaya devam edecektir



Örnek:

Start butonu ile başlayan ve stop butonu ile durdurulabilen bir çalışmada 1 çıkış sürekli olarak 0,5 saniye aktif ve 0,5 saniye pasif olacak şekilde çalışmakta, başka bir çıkışta 30 saniye aktif ve 30 saniye pasif olacak şekilde çalışmakta, üçüncü bir çıkış da eğer 2 nolu sensör aktifse ikinci çıkışla birlikte aktif ve pasif olmaktadır. Eğer 1nolu sensör aktifse diğer çalışma şekilleri atlanıp sadece 3. çıkışın olduğu kısım çalışması istenmektedir. 3. çıkışın durumu yine 2 nolu sensörün aktifliği ve 2 nolu çıkışın durumuna bağlı olarak değişmektedir. 1 nolu sensörün pasif olduğu durumlarda yine ilk çalışma şekli geçerli olmaya başlamıştır.



M1013 kontağı 1 sn lik bir kontaktır. 0.5 sn aktif 0.5 sn pasif

M1014 kontağı dk sn lik bir kontaktır. 30 sn aktif 30 sn pasif

CALL Komutu:

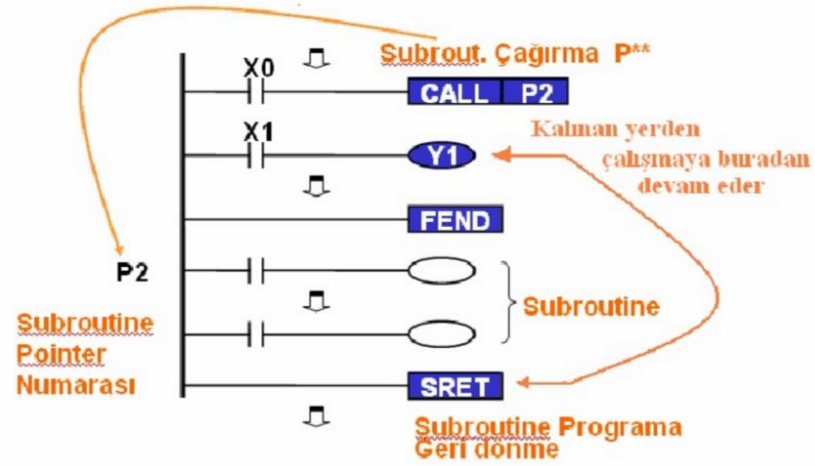
Pointer (Pxx) ile belirlenen adresteki alt programı (subroutine) çalıştıran komuttur.

SRET Komutu:

Alt program çalışması bittikten sonra CALL komutunu takip eden satırdan başlayarak normal programın devamının çalışmasını gerçekleştiren komuttur.

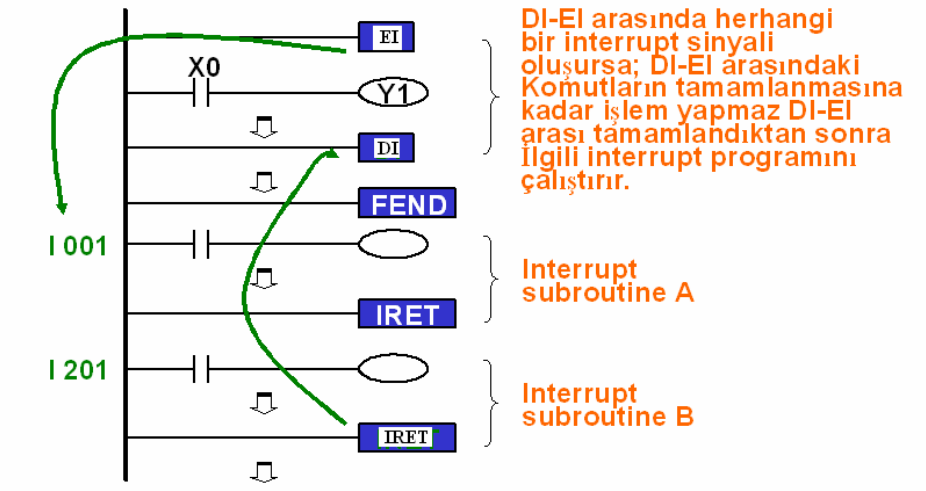
Bir ana program pozisyonundaki programın işleyişinde alt programlar gerekli olduğunda CALL komutu ile çağırılıp alt programının işleyişi tamamlandığı SRET (subroutine return) komutu ile belirlenip tekrar ana programın kalınan yerine geriye dönülmesi işlemi programcılığı bir ölçüde kolaylaştırır.

Şekilde verilen örnek gösterimde X0 girişi aktif olduğunda P2 pointer numarası ile belirtilen adresteki alt program çalışmaya başlar (Bu adresin FEND komutundan sonra olduğunu dikkat ediniz.). SRET komutuna kadar alt program bulunan çalışma tamamlanır ve X1 girişin olduğu satıra geri dönülür. Ana program işleyişi buradan devam eder. En fazla 5 adet alt program yazılabilir.



INTERRUPT KOMUTLARI (EI, DI, IRET):

Kesme (interrupt) işlemi program akışını değiştiren komutlardan biridir. CALL komutunda olduğu gibi bir alt programın çağırılmasını sağlar. Ancak kesme (interrupt) işleminde durum biraz farklıdır. Program taraması devam ederken CPU' ya ani bir cevap gerektiğinde program kesilir ve servis programı çalıştırıldıktan sonra geri dönülerek kesilen yerden programın taraması devam eder.



Şekildeki örnekte "EI" komutuyla (enable interrupt) kesme işlemin izin verilmiştir. Bu anda X0 giriş rölesi aktif ise "I001" servis programı çalışır. "IRET" komutundan (interrupt return) ana programa geri dönülür. Dönülen yerdeki komut interrupt iznini ortadan kaldıran "DI" komutudur (disable interrupt).

"I201" alt programının hangi "EI" ile çağırıldığı örnekte görülmemektedir. Bu şekil sadece alt programların "FEND" komutundan sonra olduğunu göstermek amaçlıdır. Interrupt nedenine göre alt program "pointer"i değişir. İncelenen SS serisinde;

- Harici interrupt işleminde (external interruptions) kesme nedeni olarak X0 girişi kullanıldığında "I001, X1 girişinde kullanılacak olursa "I101", X2 için "I201", X3 için"I301" alt program pointer adresleri kullanılır.
- Zamana bağlı interruptlar için "I6xx" kullanılır. "xx" yerine 10 mili saniye ile 99 mili saniye arasında bir değer yazılabilir.
- Haberleşmeye bağlı interrupt işlemi için alt program başlangıç etiketi olarak "I150" kullanılır.

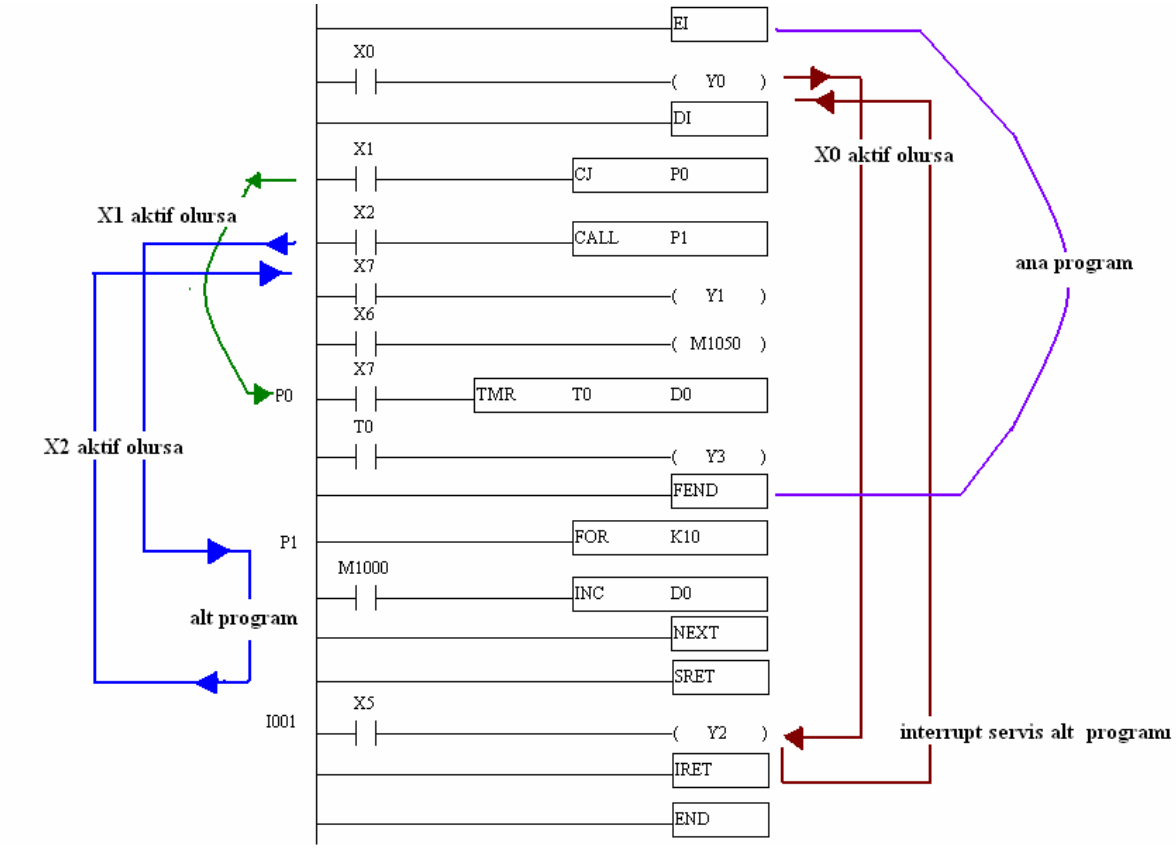
Daha üst versiyon PLC' de daha fazla kesme nedeni ve keme noktası bulunmaktadır. Bunlara en iyi örnek hızlı sayıcılara bağlı olan kesme işlemleridir. Aynı anda birden fazla interrupt nedeni ortaya çıktığında öncelik sırasına göre interrupt servis programları işleme alınır. İnterrupt işlemi için M1050 ile M1053 arasında özel yardımcı röleler bulunmaktadır Bu yardımcı rölelerin aktif olması, kesme işleminin aktivitesini sonlandırır.

Açıklama örneği:

Şekilde gerçekleşen çalışmadaki önemli noktalar şöyle açıklanabilir;

- Ana program "FEND" komutundan dönen kısımdır.

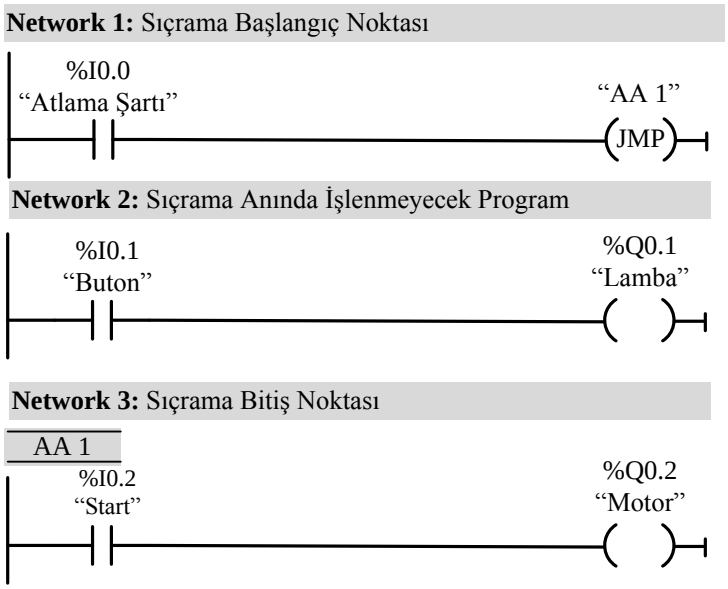
- Eğer X0 aktif olursa external (harici) interrupt işlemi için izin verildiği için “I001” etiketi ile belirlenen alt program çalışır ve “ IRET” komutundan sonra programın kesilen yerine geri döner.
- “DI” komutundan sonraki programın işleyişi sırasında X0 ‘ rın aktifliği servis programın çalışmasını engeller
- X6 girişi aktif ise interrupt işlemi için çalışma gerçekleşmez. M1050 rölesi interrupt aktivitesini ortadan kaldırdığı için X0 giriş aktifliği bir işe yaramaz.
- X1 girişi aktif olursa “P0” etiketli pointerin olduğu program satırına gidilir.
- X2 girişi aktif olursa “P1” etiketli alt program çalışır. Alt program işledikten sonra, SRET komutundan ana programda “CALL” komutunun alt satırına geri dönülür.
- Alt program bir FOR-NEXT döngüsüdür ve her çalıştığında D0 içeriği 10 kez artar. Artış miktarı X0 girişinin aktiflik süresine göre değişir.



S7 1200 PROGRAM DENETİM KOMUTLARI

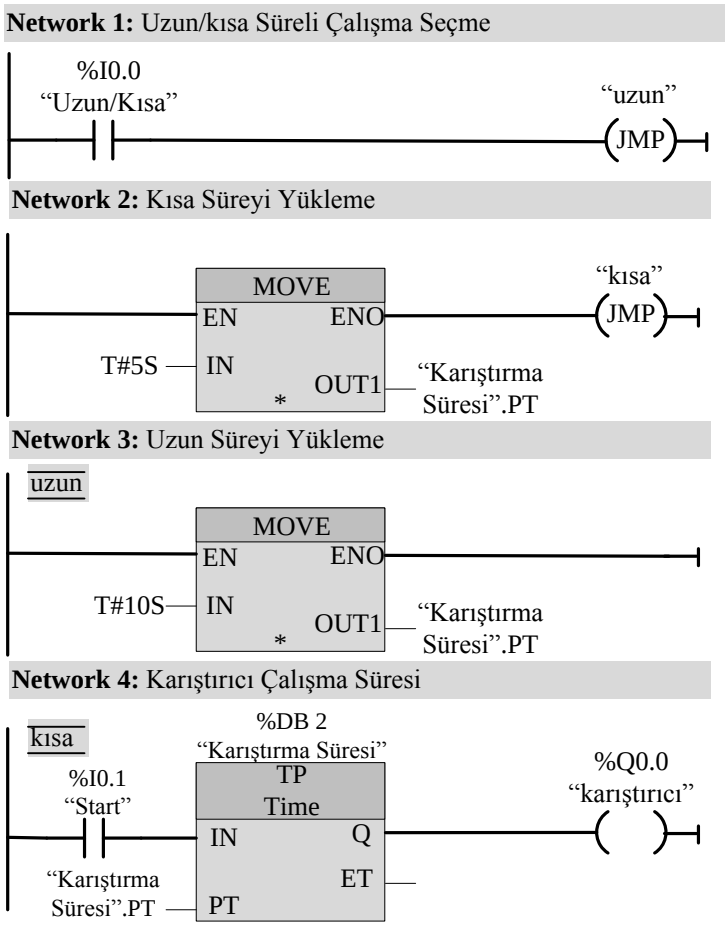
LAD	FBD	
Label_name —(JMP)—	Label_name JMP	**JMP** komutu aktif olduğunda (bu komut işlendiğinde) LBL komutuna kadar yazılmış olan komutlar atlanarak LBL komutunu izleyen komut yürütülür.
Label_name —(JMPN)—	Label_name JMPN	**JMPN** komutu aktif değil ise (Lojik 0 ise) LBL komutuna kadar yazılmış olan komutlar atlanarak LBL komutunu izleyen komut yürütülür.
Label_name	Label_name	JMP ve JMPN ile gönderilmek istenen etiket

Örnek: Aşağıdaki programda I0.0 lojik 0 ise Network 2 içerisindeki program çalışır. I0.0 lojik 1 olduğunda program akışı Network 3'e atlayacak ve Network 2 içerisindeki program parçası çalıştırılmayacaktır. Network 1 ve Network 3 her şartta çalışan programlardır.



Örnek: Bir karıştırıcının farklı iki zamanda çalıştırılması istenmektedir. "Uzun/Kısa" seçme anahtarı uyarılı olduğunda 10 sn çalışacak uyarılı olmadığında 5 sn çalışacaktır.

Çözüm: Karıştırıcı zamanlayıcısı ile ilişkilendirilen data bloğun PT (Preset Time) girişine istenen sürenin yazılması yolu izlenmiştir. Bu amaçla iki adet MOVE komutu kullanılmıştır. Uzun/Kısa girişi (I0.0) uyarılı ise PT girişine 10 sn, uyarılı değil ise 5 sn zaman yüklenmiştir. Start (I0.1) butonuna basıldığında PT girişindeki süre kadar karıştırıcının çalışması sağlanmıştır.



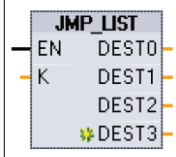
Sıçrama Listesi Oluşturma: JMP_LIST komutu

Bir Byte'lık alan (K atlama dağıtıcısı) içerisinde tanımlanan 0-99 arasında etikete şartlı sıçramayı sağlayan fonksiyondur.

Atlama dağıtıcısı sayı değeri 0 ise DEST0 da tanımlanan etikete, sayı değeri 1 ise DEST1 de tanımlanan etikete, sayı değeri 2 ise DEST2 de tanımlanan etikete vb. sıçrar.

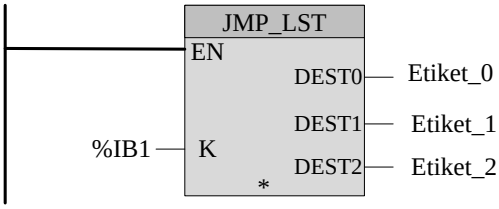
K girişindeki atlama dağıtıcısı sayı değeri hedef sayısından büyük ise hiçbir etikete sıçramaz JMP_LIST fonksiyonunun bir alt satırından devam eder.

LAD / FBD	SCL	

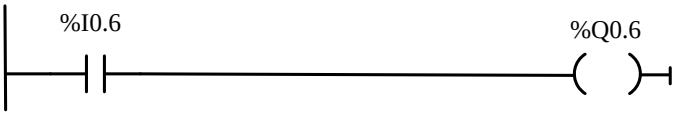
	`CASE k OF`
	`  0: GOTO dest0;`
	`  1: GOTO dest1;`
	`  2: GOTO dest2;`
	`  [n: GOTO destn;]`
	`END_CASE;`
		K ya bağlı olarak programın ilgili etiketlere yönlendirilmesini sağlar. Program ilgili etiketin altından yürütülmeye devam eder.

Örnek: Aşağıdaki programda IB1 den girilen sayı değeri 0 ise Etiket_0, 1 ise Etiket_1, 2 ise Etiket_2' ye sıçrar. Eğer IB1 den girilen sayı 2 den büyük ise Network 2 den itibaren programı çalıştırır. Eğer program sıçranılan noktadan aşağıdaki bütün Networkları çalıştırmaması isteniyor ise RET komutu ile program sonlandırılabilir. Örneğin sayı değeri 1 olduğunda Network 4'e sıçranacak ve Network 5 de işlendikten sonra çevrim başına dönülecektir.

Network 1: Sıçrama Dağıtıcısı

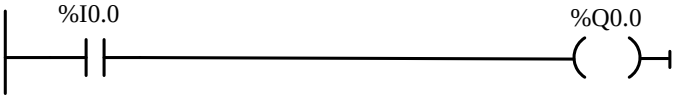


Network 2: Sıçrama sayısının etiket sayısından büyük olması durumunda sıçrama hedefi



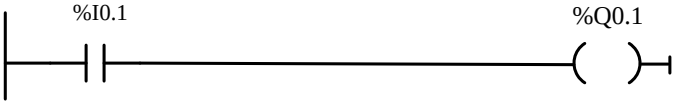
Network 3: Sıçrama sayısının 0 olması durumunda sıçrama hedefi

Etiket_0

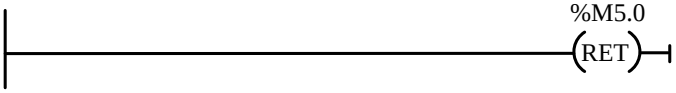


Network 4: Sıçrama sayısının 1 olması durumunda sıçrama hedefi

Etiket_1

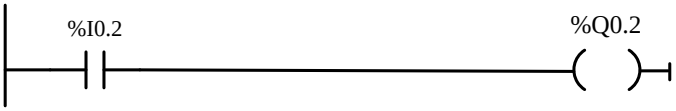


Network 5:



Network 6: Sıçrama sayısının 2 olması durumunda sıçrama hedefi

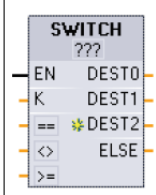
Etiket_2



Karşılaştırma sonucuna göre sıçrama SWITCH komutu

Programın çeşitli kısımlarının çalıştırılması için program atlama komutu JMP gibi davranır. Karşılaştırma sonucuna göre sıçrama (jmp) etiketi tanımlama fonksiyonudur.

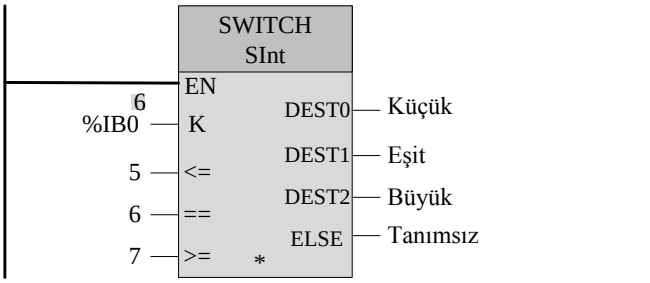
Fonksiyon üzerindeki * sembolüne tıklayıp her giriş şartı eklendiğinde yeni bir çıkış etiketi otomatik olarak eklenmektedir.

LAD / FBD	SCL
	Not available K girişine bağlanan bellek alanının aldığı değer onun altında tanımlanan karşılaştırma fonksiyonlarındaki değer ile kıyaslanır. Hangi sıradaki sonuç doğru ise o numaralı etikete atlanır. Değer belirlenen kıyaslamalara uymuyor ise ELSE etiketine atlanır.

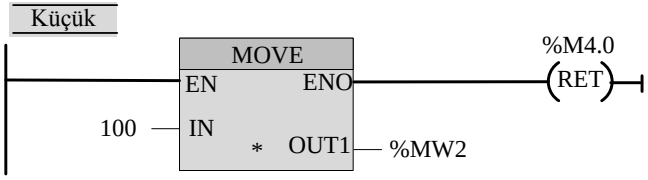
Örnek: Aşağıdaki programda SInt (Short Integer 8 Bit) olarak tanımlanan veri tipine uygun olarak K girişine uygulanan sayı değeri 5 den küçük ise DEST0 yani Küçük etiketine, 6 ya eşit ise DEST1 yani Eşit etiketine, 7 den büyük ise DEST2 yani Büyük etiketine sıçranacaktır. Okunan değer tanımlı karşılaştırmalara uymuyor ise ELSE yani Tanımsız etiketine sıçrar.

Bu programda IB 0<5 ise MW 2 ye 100, IB 0==6 ise MW 2'ye 50, IB 0>7 ise MW 2'ye 20, bunların dışında ise MW 2'ye 0 değeri gönderilir.

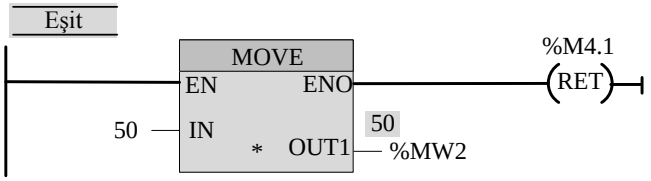
Network 1: Karşılaştırma sonucuna göre sıçrama



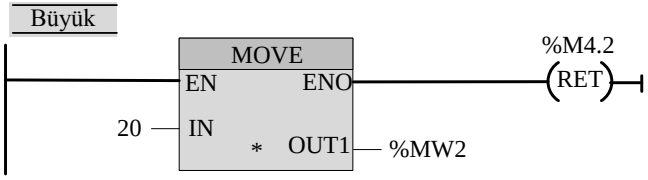
Network 2: Okunan değer küçük



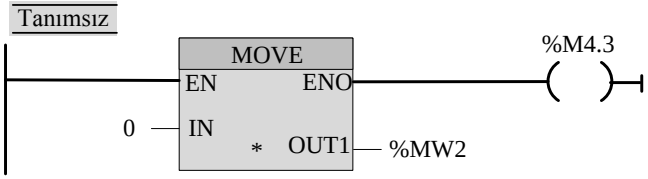
Network 3: Okunan değer eşit



Network 4: Okunan değer büyük



Network 5: Tanımsız değer



RET komutu

RET komutlarının tamamı programın lineer çalışmasını kesip, çevrim başına dönülmesini sağlar.

LAD	FBD	SCL	
"Return_Value" —(RET)—	"Return_Value" RET	RETURN;	Mevcut program bloğunun yürütülmesini sonlandırır.

Örnek: Bir elektrik motoruna bir seçici anahtar kullanarak, iki farklı şekilde kumanda edilmesi istenmektedir. S seçici anahtarı kapalı iken, motor A ve B gibi iki farklı noktadan çalıştırılıp durdurulabilmeli, S anahtarı açık iken sadece A noktasından çalıştırılıp durdurulabilmelidir.

Çözüm: Atlama komutları kullanarak bu kumanda devresinin çözülmesi için iki ayrı program yazılmalıdır. S anahtarının durumuna göre bu programlardan biri çalışır.

Q0.0 → Motor çıkışı

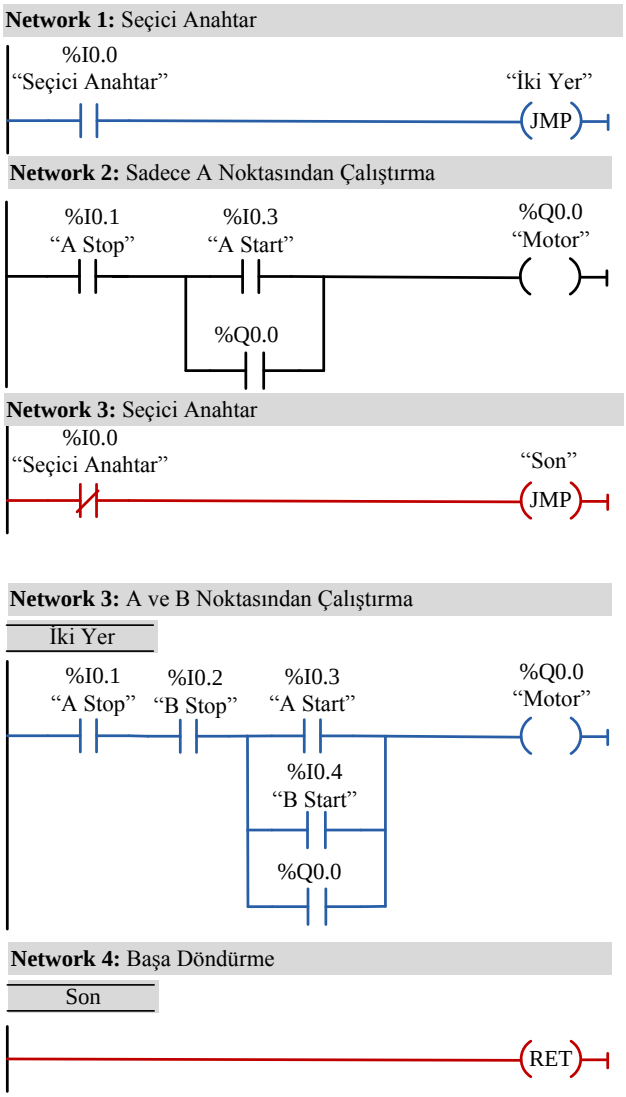
I0.0 → S seçici anahtarı

I0.1 → A noktası stop butonu

I0.2 → B noktası stop butonu

I0.3 → A noktası start butonu

I0.4 → B noktası start butonu



Örnek: Bir elektrik motoruna bir seçici anahtar kullanarak, iki farklı şekilde kumanda edilmesi istenmektedir. S seçici anahtarı kapalı iken, motor A ve B gibi iki farklı noktadan çalıştırılıp durdurulabilmeli, S anahtarı açık iken sadece A noktasından çalıştırılıp durdurulabilmelidir.

Çözüm: Atlama komutları kullanarak bu kumanda devresinin çözülmesi için iki ayrı program yazılmalıdır. S anahtarının durumuna göre bu programlardan biri çalışır.

Y0 → Motor çıkışı

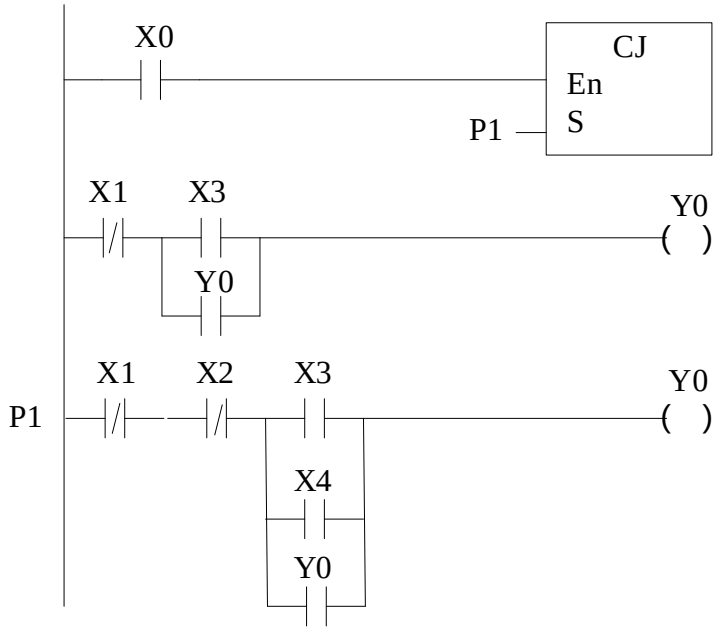
X0 → S seçici anahtarı

X1 → A noktası stop butonu

X2 → B noktası stop butonu

X3 → A noktası start butonu

X4 → B noktası start butonu




```
LD  X0
CJ  P1
LDI X1
LD  X3
OR  Y0
ANB
OUT  Y0
P001
LDI  X1
ANI  X2
LD  X3
OR  X4
OR Y0
ANB
OUT Y0
```

Alt Program Kullanarak Programlama

Örnek: Bir elektrik motoruna bir seçici anahtar kullanarak, iki farklı şekilde kumanda edilmesi istenmektedir. S seçici anahtarı kapalı iken, motor A ve B gibi iki farklı noktadan çalıştırılıp durdurulabilmeli, S anahtarı açık iken sadece A noktasından çalıştırılıp durdurulabilmelidir.

Y0 → Motor çıkışı

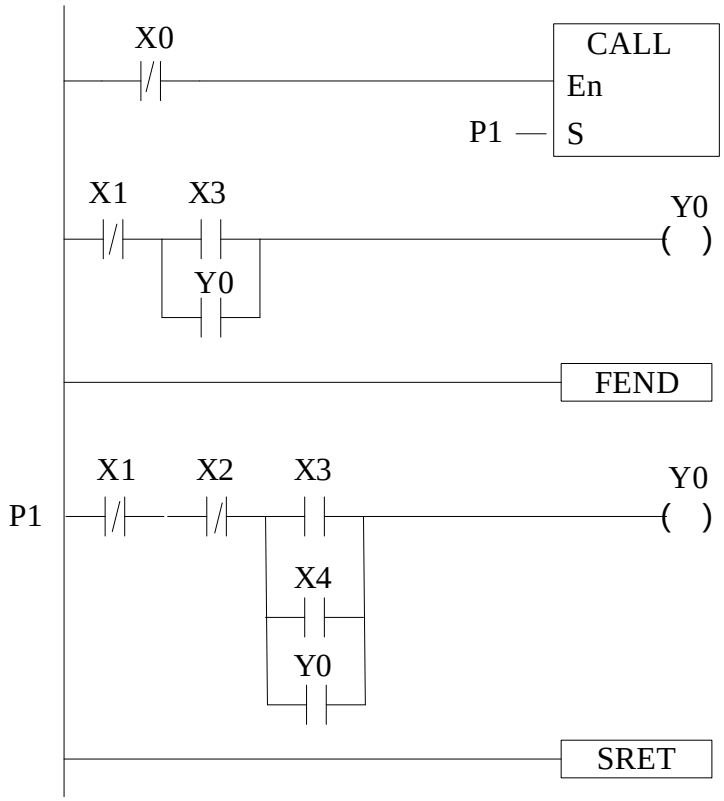
X0 → S seçici anahtarı

X1 → A noktası stop butonu

X2 → B noktası stop butonu

X3 → A noktası start butonu

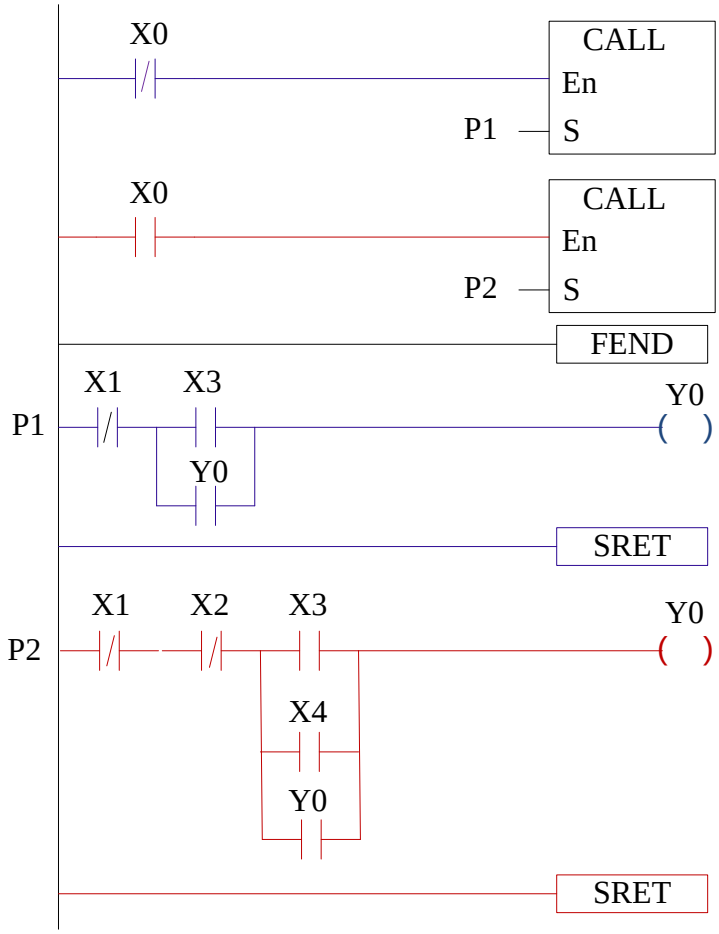
X4 → B noktası start butonu




```
LDI  X0
CALL  P1
LDI  X1
LD  X3
OR  Y0
ANB
OUT  Y0
FEND

P001
LDI  X1
ANI  X2
LD  X3
OR  X4
OR Y0
ANB
OUT Y0
SRET
```

Ya da




```
LDI  X0
CALL  P1
LD X0
CALL P2
FEND

P001
LDI  X1
LD  X3
OR  Y0
ANB
OUT  Y0
SRET

P002
LDI  X1
ANI  X2
LD  X3
OR  X4
OR Y0
ANB
OUT Y0
SRET
```

Bu programda X0 değerine bağlı olarak, iki alt programdan biri işletilir.

X0 = 0 ise CALL P1 komutu etkin olur ve P1 ile başlayan ve SRET komutu ile biten alt program çalışır. Bu durumda CALL P2 komutu etkin olmayacağından P2 ile başlayıp SRET ile biten alt program işletilmez.

X0= 1 ise CALL P2 komutu etkin olur, ve P2 ile başlayıp SRET ile biten alt program işletilir.

Böylece S anahtarının durumuna bağlı olarak, iki ayrı kumanda işlevi gerçeklenmiş olur.